

# ON PASSWORD BASED AUTHENTICATION AND KEY AGREEMENT PROTOCOLS

## A THESIS

*Submitted in partial fulfilment of the  
requirements for the award of the degree  
of*

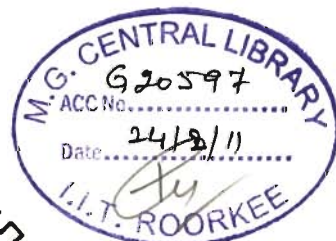
DOCTOR OF PHILOSOPHY

*in*

ELECTRONICS AND COMPUTER ENGINEERING

*by*

**SANDEEP KUMAR SOOD**



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE  
ROORKEE-247 667 (INDIA)

JUNE, 2010

**©INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, ROORKEE, 2010  
ALL RIGHTS RESERVED**



# INDIAN INSTITUTE OF TECHNOLOGY ROORKEE ROORKEE

## CANDIDATE'S DECLARATION

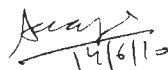
I hereby certify that the work which is being presented in this thesis entitled **ON PASSWORD BASED AUTHENTICATION AND KEY AGREEMENT PROTOCOLS** in partial fulfilment of the requirements for the award of *the Degree of Doctor of Philosophy* and submitted in the Department of Electronics and Computer Engineering of the Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from July 2007 to June 2010 under the supervision of Dr. Anil K. Sarje, Professor and Dr. Kuldip Singh, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

  
(SANDEEP KUMAR SOOD)

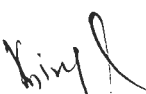
This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

  
(Kuldip Singh)  
Supervisor

  
(Anil K. Sarje)  
Supervisor

Date: 14/6/10

The Ph.D. Viva-Voce Examination of *Mr. Sandeep Kumar Sood*, Research Scholar, has been held on 15.10.2010

  
Signature of Supervisors

  
Signature of External Examiner

## ABSTRACT

---

This thesis is a study of password based authentication and key agreement protocols. Due to simplicity and convenience, password is the most commonly used authentication technique to authenticate users on the web. The main advantage of passwords is that users can memorize them easily without needing any hardware to store them. Efficient password authentication schemes are required to authenticate the legitimacy of remote users over an insecure communication channel. In this thesis, we propose some password based authentication protocols for different types of environments as well as an anti-phishing protocol. Improvements to several static and dynamic identity based authentication protocols have also been suggested.

Password based authentication is used in online web applications and is highly susceptible to phishing attacks. The average user can not distinguish a well designed phishing website from the legitimate site because the phishing site is designed in a manner that imitates visual characteristics of the target organization's website by using similar colors, icons, logos and textual descriptions. Phishing is doing direct damage to the financial industry and is also affecting the expansion of e-commerce. One of the reasons for success of phishing attacks is high rate of password reuse because users tend to use the same password with more and more accounts. Users find it difficult to remember several complex passwords and hence it is difficult to prevent phishing and dictionary attacks. In 2007, Gouda et al. proposed a single password based anti-phishing protocol for Hyper Text Transfer Protocol authentication that allows a user to choose a single password of his choice for multiple online accounts on different web servers. In this thesis, we show that Gouda et al.'s protocol is insecure against offline dictionary attack, denial of service attack and man-in-the-middle attack in presence of an active attacker. Also an improvement to Gouda et al.'s protocol is proposed that can resist offline dictionary attack and denial of service attack. It is however found that Gouda et al.'s protocol is not repairable for man-in-the-middle attack. We propose a new single password based anti-phishing protocol that resolves aforementioned problems and is secure against different types of attacks. In this protocol, the client machine's browser generates a dynamic identity and a dynamic password for each new login request to the server. The dynamic identity and dynamic

password generated for a client are different in different sessions of the Secure Socket Layer (SSL) protocol.

Password based authentication protocols are susceptible to dictionary attacks by means of automated programs because most of the user chosen passwords are limited to the user's personal domain. We propose a cookie based and an inverse cookie based virtual password authentication protocols. In cookie based virtual password authentication protocol, the web server stores a cookie on the user's computer if the user has successfully authenticated himself to the web server from that computer. On the other hand, in an inverse cookie based virtual password authentication protocol, the web server stores cookie on the user's computer when he has not submitted correct identity and password for his authentication to the web server. In both these protocols, the computational effort required from the attacker during login on to the web server increases exponentially with each login failure. The concept of trust has been used so that the legitimate client can easily authenticate himself to the web server from any computer irrespective of whether that computer contains cookie or not. The client generated virtual password for a user is different in each new session of SSL protocol. These concepts combine traditional password authentication with a challenge that is easy to answer by a legitimate client but the computation cost of authentication for an attacker increases with each login failure. Therefore, even automated programs can not launch online dictionary attacks on these proposed protocols. These protocols removes some of the deficiencies of previously suggested password based authentication protocols and are shown to be secure against different types of attacks that can be launched by the attacker.

Most of the password based authentication protocols rely on a single authentication server for user's authentication. The user's password verification information stored on the single server is a main point of susceptibility and remains an attractive target for the attacker. We present Single Sign-On (SSO) password based two-server authentication protocol that issues a ticket to the user for a specific time period. The user can use this ticket to generate dynamic ticket information to login on to the authentication server. Ticket issued for one authentication server can be used for user authentication on another authentication server that is under the control of the same control server. Our protocol uses two-server paradigm by imposing different levels of trust upon the two servers so that password verification information is distributed between two servers (an authentication

server and a control server). Therefore, the proposed protocol is more resistant to dictionary attacks as compared to other existing single-server password based authentication protocols. The proposed protocol is efficient and practical for its implementation because it does not use public key that causes computation and communication burden in a resource constrained environment.

Next, a brief review of some static identity based smart card authentication protocols is presented. Cryptanalysis of these protocols is carried out for different types of attacks and improved protocols are proposed. The comparison of the cost and functionality of the proposed protocols with the other related protocols is also done. Yoon et al. (2005) proposed a remote user authentication scheme which is an improvement on Hwang et al.'s scheme. However, we found that Yoon et al.'s scheme can easily reveal a user's password and is susceptible to impersonation attack using stolen smart card. This scheme is also found to be susceptible to parallel session attack and man-in-the-middle attack. We propose a remote user authentication scheme that resolves the aforementioned problems, while keeping the merits of Yoon et al.'s scheme. We also analyze the smart card based authentication protocols of Kim and Chung (2009), Xu et al. (2009) and Liu et al. (2008) and show that they are susceptible to different types of attacks. We propose improvements of these protocols to overcome their weaknesses.

Next, this thesis investigates some smart card authentication protocols for different attack scenarios and improved dynamic identity based smart card authentication protocols are proposed. In 2004, Das et al. proposed a dynamic identity based remote user authentication protocol. Many researchers demonstrated that Das et al.'s protocol is susceptible to several types of attacks. In 2005, Liao et al. published an improved protocol and claimed that this improved scheme withstands password guessing attack and insider attack. However, we found that Liao et al.'s protocol is also susceptible to malicious user attack, impersonation attack, stolen smart card attack and offline password guessing attack. Moreover, Liao et al.'s protocol does not maintain the user's anonymity and its password change phase is insecure. We present a secure dynamic identity based authentication protocol using smart cards to resolve the aforementioned problems, while keeping the merits of different dynamic identity based authentication protocols. We also analyze the smart card based authentication protocols of Liou et al. (2006), Wang et al.

(2009), Lee et al. (2005) and Hsiang and Shih (2009) and show that they are susceptible to different types of attacks. We propose improvements of these protocols to overcome their weaknesses. The security analysis of the proposed improved protocols is presented. The comparison of the cost and functionality of the proposed protocols with the other related protocols is also done.

In e-commerce, the number of servers providing the services to the user is usually more than one and hence secure authentication protocols for multi-server environment are required. Moreover, the multi-server architecture based authentication protocols make it difficult for the attacker to find out any significant authentication information related to the legitimate users. In 2009, Liao and Wang proposed a dynamic identity based remote user authentication protocol for multi-server environment. But we found that Liao and Wang's protocol is susceptible to malicious server attack and malicious user attack. We propose a dynamic identity based authentication protocol for multi-server architecture using smart card that resolves the aforementioned security flaws, while keeping the merits of Liao and Wang's protocol. In 2009, Hsiang and Shih improved Liao and Wang's dynamic identity based remote user authentication protocol for multi-server environment. However, we show that Hsiang and Shih's protocol is susceptible to replay attack, impersonation attack and stolen smart card attack. Moreover, the password change phase of Hsiang and Shih's protocol is incorrect. We present a dynamic identity based authentication protocol for multi-server architecture using smart card that resolves the aforementioned flaws, while keeping the merits of Hsiang and Shih's protocol. The proposed protocols use two-server paradigm by imposing different levels of trust upon the two servers and the user's authentication functionality is distributed between these two servers known as the service provider server and the control server.

In our proposed single password based anti-phishing protocol, client can use a single password for different online accounts and that password can not be detected by any of the malicious server or the attacker. This protocol is equally secure for security ignorant users, who are not very conversant with the browser's security indicators. The protocol does not allow the server to know the client's password at any time. This protocol can be easily integrated into different types of services such as banking and enterprise applications. The proposed cookies based and inverse cookie based virtual password authentication protocols are very effective to thwart online dictionary attacks because the

computation cost of login on to the web server increases exponentially with each login failure for an attacker. The legitimate client can easily authenticate himself to the web server from any computer irrespective of whether that computer contains cookie or not. SSO authentication is time efficient because it allows the user to enter his identity and password once within a specific time period to login on to multiple hosts and applications within an organization. Most of the existing SSO password based authentication protocols are designed for single-server environment. We proposed an efficient SSO password based two-server architecture in which the user has to login once to get a valid ticket. Smart card based password authentication is one of the most convenient ways to provide multi-factor authentication by acquiring the smart card and knowing the correct identity and correct password for the communication between a client and a server. Improvements to several static and dynamic identity based authentication protocols have also been suggested. User's privacy is an important issue in e-commerce applications. The proposed dynamic identity based authentication protocols aim to provide the privacy to the user's identity so that users are anonymous in communication channel. Also the concept of two-tier authentication for the client makes it difficult for an attacker to guess out the information pertaining to password and ticket. Confidence of clients in e-commerce and other online transactions can be enhanced by negating phishing, dictionary and other possible attacks. The work presented in this thesis is a step toward making e-commerce transactions more reliable and secure.



## ACKNOWLEDGEMENTS

---

In writing acknowledgements for my Ph. D. thesis, both words and space fall short to express gratitude for a number of individuals/organizations. I would like to pen down some of them through this opportunity. This research work is an out come of moral support from many individuals/organizations directly or indirectly involved with me during my research work at Indian Institute of Technology, Roorkee.

From the deep of my heart, the acknowledgements with warm regards and gratitude goes to my thesis supervisors Dr. Anil K. Sarje, Professor and Dr. Kuldip Singh, Professor, Department of Electronics & Computer Engineering, Indian Institute of Technology, Roorkee for their proficient and enthusiastic guidance, useful criticism, encouragement and immense support throughout the research work for Ph.D. degree and shape out this thesis in the present form. I sincerely appreciate their pronounced humanistic and warm personal approach, which has given me strength and inspiration to carry out this research work smoothly. Working under them has been a wonderful experience, which has provided me with a deep insight in the world of research. I humbly acknowledge a lifetime's gratitude to them.

I am thankful with humble submission to members of my 'Students Research Committee' for inspiring and approving this research work.

I take this opportunity to express my sincere thanks to Vice-Chancellor, Guru Nanak Dev University, Amritsar for sponsoring me to pursue the Ph.D. programme under QIP scheme of MHRD, Govt. of India. I express my thanks to the staff of QIP Centre for their co-operation to carryout the official matters during my stay at IIT Roorkee. The financial support provided by MHRD, Govt. of India is sincerely and honestly acknowledged. I wish to thank everybody who has directly or indirectly helped me throughout my research work. I am thankful to my department for providing computing and other facilities for my research work. I am grateful to the whole administrative and technical staff of the department for their co-operations and help. I am very grateful to Indian Institute of Technology Roorkee for providing access to journals related to my research work.

I acknowledge the blessings of my parents Shri Sushil Kumar Sood and Smt Raj Kumari Sood for encouragement and moral support rendered to me throughout my life. I sincerely acknowledge the moral support and encouragement from my younger brother, Dr. Pankaj Sood along with his family for the sign of love and affection towards me. I also acknowledge the kind blessing and support from my sister Mrs. Alka Sood and her husband Mr. Vikul Sood. My heartily gratitude to Mr. Rajiv Sood and his family, Mr. Sanjiv Sood and his family, Mr. Nilesh Sood and his family and my father in law Shri Ram Kumar Sood and mother in law Smt Shashi Sood.

I am especially indebted to my wife Mrs. Vaishali Sood, who remains critical proof reader of my research papers. Her contribution to this work is unexplainable in words. Cheerfulness and tolerance capacity of my daughter Kashvi, who missed many precious moments of fatherly love and care, is really admirable.

Above all, I express my deepest regards and gratitude to “All Mighty: God” whose divine light and warmth showered upon me the perseverance, inspiration, faith and enough strength to keep the momentum of work high even at tough moments of research work.

I dedicate this research work to my parents, wife and my loving daughter.

**(SANDEEP KUMAR SOOD)**

# CONTENTS

---

	<b>Page</b>
<b>CANDIDATE'S DECLARATION</b>	<b>i</b>
<b>ABSTRACT</b>	<b>iii</b>
<b>ACKNOWLEDGEMENTS</b>	<b>ix</b>
<b>CONTENTS</b>	<b>xi</b>
<b>LIST OF FIGURES</b>	<b>xvii</b>
<b>LIST OF TABLES</b>	<b>xix</b>
<b>ABBREVIATIONS</b>	<b>xxiii</b>
<b>CHAPTER- 1 INTRODUCTION AND LITERATURE REVIEW</b>	<b>1</b>
1.1 The Importance of Authentication	1
1.2 Motivation	3
1.3 Password based Authentication Protocols	6
1.4 Anti-Phishing Protocols	11
1.5 Smart Card based Authentication Protocols	16
1.6 Research Problems	22
1.7 Organization of the Thesis	25
1.8 Conclusion	26
<b>CHAPTER-2 DYNAMIC IDENTITY BASED SINGLE PASSWORD ANTI-PHISHING PROTOCOL</b>	<b>29</b>
2.1 Introduction	29
2.2 Review of Gouda et al.'s Protocol	30
2.3 Cryptanalysis of Gouda et al.'s Protocol	32
2.4 Improved Gouda et al.'s Protocol	33
2.5 Proposed Single Password Anti-Phishing Protocol	35
2.5.1 Registration phase	36
2.5.2 Login phase	36
2.5.3 Authentication phase	37
2.5.4 Password change phase	38

2.6	Security Analysis	38
2.7	Cost and Functionality Analysis	42
2.8	Conclusion	43

**CHAPTER-3 COOKIE AND INVERSE COOKIE BASED VIRTUAL PASSWORD AUTHENTICATION PROTOCOLS 45**

3.1	Introduction	45
3.2	Proposed Cookie based Virtual Password Authentication Protocol	46
3.2.1	Protocol 1	48
	Registration phase	48
	Login phase	49
	Authentication phase	50
	Password change phase	52
3.2.2	Protocol 2	52
	Registration phase	52
	Login phase	53
	Authentication phase	53
	Password change phase	54
3.3	Security Analysis	55
3.4	Proposed Inverse Cookie based Virtual Password Authentication Protocol	58
3.4.1	Protocol 1	59
	Registration phase	59
	Login phase	60
	Authentication phase	60
	Password change phase	61
3.4.2	Protocol 2	61
	Registration phase	61
	Login phase	61
	Authentication phase	61
	Password change phase	64
3.5	Security Analysis	64
3.6	Conclusion	65

**CHAPTER-4 SSO PASSWORD BASED TWO-SERVER AUTHENTICATION  
PROTOCOL 67**

4.1	Introduction	67
4.2	Proposed SSO Authentication Protocol	69
4.2.1	Registration phase	71
4.2.2	Login and ticket request phase	72
4.2.3	Authentication phase	74
4.2.3.1	Ticket based re-authentication phase	74
4.2.4	Password change phase	76
4.3	Security Analysis	78
4.4	Cost and Functionality Analysis	83
4.5	Conclusion	84

**CHAPTER-5 STATIC IDENTITY BASED SMART CARD AUTHENTICATION  
PROTOCOLS 87**

5.1	Introduction	87
5.2	Review of Yoon et al.'s Scheme	87
5.2.1	Cryptanalysis of Yoon et al.'s scheme	89
5.2.2	Proposed protocol	91
5.2.3	Security analysis	93
5.2.4	Cost and functionality analysis	96
5.3	Review of Kim and Chung's Scheme	98
5.3.1	Cryptanalysis of Kim and Chung's scheme	100
5.3.2	Proposed protocol	101
5.3.3	Security analysis	104
5.3.4	Cost and functionality analysis	107
5.4	Review of Xu et al.'s Scheme	108
5.4.1	Cryptanalysis of Xu et al.'s scheme	110
5.4.2	Proposed protocol	110
5.4.3	Security analysis	112
5.4.4	Cost and functionality analysis	115

5.5	Review of Liu et al.'s Scheme	116
5.5.1	Cryptanalysis of Liu et al.'s scheme	118
5.5.2	Proposed protocol	119
5.5.3	Security analysis	122
5.5.4	Cost and functionality analysis	124
5.6	Conclusion	126

**CHAPTER-6 DYNAMIC IDENTITY BASED SMART CARD AUTHENTICATION PROTOCOLS 127**

6.1	Introduction	127
6.2	Review of Liao et al.'s Scheme	127
6.2.1	Cryptanalysis of Liao et al.'s scheme	129
6.2.2	Proposed protocol	133
6.2.3	Security analysis	135
6.2.4	Cost and functionality analysis	138
6.3	Review of Liou et al.'s Scheme	139
6.3.1	Cryptanalysis of Liou et al.'s scheme	141
6.3.2	Proposed protocol	145
6.3.3	Security analysis	147
6.3.4	Cost and functionality analysis	150
6.4	Review of Wang et al.'s Scheme	151
6.4.1	Cryptanalysis of Wang et al.'s scheme	153
6.4.2	Proposed protocol	156
6.4.3	Security analysis	158
6.4.4	Cost and functionality analysis	161
6.5	Review of Lee et al.'s Scheme	162
6.5.1	Cryptanalysis of Lee et al.'s scheme	163
6.5.2	Proposed protocol	165
6.5.3	Security analysis	167
6.5.4	Cost and functionality analysis	170

6.6	Review of Hsiang and Shih's Scheme	172
6.6.1	Cryptanalysis of Hsiang and Shih's scheme	174
6.6.2	Proposed protocol	175
6.6.3	Security analysis	177
6.6.4	Cost and functionality analysis	180
6.7	Conclusion	181
<b>CHAPTER-7</b>	<b>DYNAMIC IDENTITY BASED AUTHENTICATION PROTOCOLS FOR MULTI-SERVER ARCHITECTURE</b>	<b>183</b>
7.1	Introduction	183
7.2	Review of Liao and Wang Protocol	184
7.2.1	Cryptanalysis of Liao and Wang's protocol	186
7.2.2	Proposed protocol	188
7.2.3	Security analysis	192
7.2.4	Cost and functionality analysis	196
7.3	Review of Hsiang and Shih Protocol	197
7.3.1	Cryptanalysis of Hsiang and Shih's protocol	201
7.3.2	Proposed protocol	204
7.3.3	Security analysis	208
7.3.4	Cost and functionality analysis	212
7.4	Conclusion	214
<b>CHAPTER-8</b>	<b>CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK</b>	<b>215</b>
8.1	Contributions of the Thesis	215
8.2	Recommendations and Future Scope of Work	222
	<b>REFERENCES</b>	<b>223</b>
	<b>AUTHOR'S PUBLICATIONS</b>	<b>241</b>

## LIST OF FIGURES

<b>Fig. No.</b>	<b>Caption</b>	<b>Page</b>
1.1	Reverse turing test	7
2.1	Gouda et al.'s protocol	31
2.2	Improved Gouda et al.'s protocol	34
2.3	Dynamic identity based single password anti-phishing protocol	37
3.1	Protocol 1: (Case 1) Virtual password authentication protocol with cookie	49
3.2	Protocol 1: (Case 2) Virtual password authentication protocol with cookie	51
3.3	Protocol 2: Virtual password authentication protocol without cookie	54
3.4	Protocol 1: Virtual password authentication protocol without cookie	59
3.5	Protocol 2: (Case 1) Virtual password authentication protocol with cookie	62
3.6	Protocol 2: (Case 2) Virtual password authentication protocol with cookie	64
4.1	SSO password based two-server authentication protocol	72
4.2	SSO password based two-server authentication protocol (Ticket based re-authentication phase)	76
4.3	Authentication server's password change protocol	77
4.4	Control server's password change protocol	78
5.1	Yoon et al.'s scheme	88
5.2	Proposed improvement in Yoon et al.'s scheme	92
5.3	Kim and Chung's scheme	98
5.4	Proposed improvement in Kim and Chung's scheme	102
5.5	Xu et al.'s scheme	109



<b>Fig. No.</b>	<b>Caption</b>	<b>Page</b>
5.6	Proposed improvement in Xu et al.'s scheme	111
5.7	Proposed improvement in Liu et al.'s scheme	120
6.1	Liao et al.'s scheme	128
6.2	Proposed improvement in Liao et al.'s scheme	134
6.3	Liou et al.'s scheme	140
6.4	Proposed improvement in Liou et al.'s scheme	145
6.5	Wang et al.'s scheme	152
6.6	Proposed improvement in Wang et al.'s scheme	157
6.7	Lee et al.'s scheme	162
6.8	Proposed improvement in Lee et al.'s scheme	166
6.9	Hsiang and Shih's scheme	172
6.10	Proposed improvement in Hsiang and Shih's scheme	175
7.1	Liao and Wang's dynamic identity based multi-server authentication protocol	185
7.2	Proposed improvement in Liao and Wang's authentication protocol	190
7.3	Hsiang and Shih's dynamic identity based multi-server authentication protocol	199
7.4	Proposed improvement in Hsiang and Shih's authentication protocol	206

## LIST OF TABLES

Table No.	Caption	Page
1.1	Domain, country domain and phishing count	14
1.2	Cost and functionality comparison among different anti-phishing protocols	15
1.3	Phishing attacks and their countermeasures	15
1.4	Organization based phishing sites	15
1.5	Comparison among related smart card based authentication schemes	20
2.1	Notations	30
2.2	Notations	35
2.3	Cost and functionality comparison among different anti-phishing protocols	43
3.1	Notations	47
4.1	Notations	70
4.2	Cost and functionality comparison among related two-server authentication schemes	84
5.1	Cost comparison among related smart card based authentication schemes	97
5.2	Functionality comparison among related smart card based authentication schemes	97
5.3	Cost comparison among related smart card based authentication schemes	107
5.4	Functionality comparison among related smart card based authentication schemes	108
5.5	Cost comparison among related smart card based authentication schemes	116

<b>Table No.</b>	<b>Caption</b>	<b>Page</b>
5.6	Functionality comparison among related smart card based authentication schemes	116
5.7	Cost comparison among related smart card based authentication schemes	125
5.8	Functionality comparison among related smart card based authentication schemes	126
6.1	Cost comparison among related smart card based authentication schemes	139
6.2	Functionality comparison among related smart card based authentication schemes	139
6.3	Cost comparison among related smart card based authentication schemes	150
6.4	Functionality comparison among related smart card based authentication schemes	151
6.5	Cost comparison among related smart card based authentication schemes	161
6.6	Functionality comparison among related smart card based authentication schemes	162
6.7	Cost comparison among related smart card based authentication schemes	171
6.8	Functionality comparison among related smart card based authentication schemes	171
6.9	Cost comparison among related smart card based authentication schemes	180
6.10	Functionality comparison among related smart card based authentication schemes	181
7.1	Notations	184
7.2	Notations	188

<b>Table No.</b>	<b>Caption</b>	<b>Page</b>
7.3	Cost comparison among related smart card based multi-server authentication schemes	196
7.4	Functionality comparison among related smart card based multi-server authentication schemes	197
7.5	Notations	198
7.6	Cost comparison among related smart card based multi-server authentication schemes	213
7.7	Functionality comparison among related smart card based multi-server authentication schemes	214

## ABBREVIATIONS

---

SSL/TLS	Secure Socket Layer/Transport Layer Security
HTTP	Hyper Text Transfer Protocol
URL	Uniform Resource Locator
IT	Information Technology
RTT	Reverse Turing Test
3PAKE	Three Party Authentication Key Exchange
SSO	Single Sign On
OTP	One Time Password
AOL	America On Line
IP Address	Internet Protocol Address
GUI	Graphical User Interface
IDS	Intrusion Detection System
S/KEY	Secure Key
S/MIME	Secure/Multipurpose Internet Mail Extensions
RFC	Request For Comments
DNS	Domain Name System
SRD	Synchronized Random Dynamic
DSS	Dynamic Security Skin
ADSI	Automatic Detecting Security Indicator
LSOP	Locked Same Origin Policy
ECS	Encrypted Cookie Scheme
SSPA	Simple Strong Password Authentication
OSPA	Optimal Strong Password Authentication
RSA	Rivest Shamir Adleman
E-Commerce	Electronic Commerce
SMS	Short Message Service
SPP	Single Password Protocol
MD ( )	Message Digest (One Way Hash Function)
H ( )	One Way Hash Function

SS	Secure Socket Layer Session Key
PK	Public Key of Server
SK	Private Key of Server
$S_k$	Session Key between User and Server
CK	Cookie
MITM	Man-in-the-middle
EXP_TIME	Expiration Time
MIN_TRUST	Minimum Trust
MAX_TRUST	Maximum Trust
CUR_TRUST	Current Trust
TRUST_BITS	Trust Bits Value
$ID_i$	Identity of User $U_i$
$P_i$	Password of User $U_i$
XOR	Bit-wise Exclusive OR Operation
	Concatenation Operation
PUID	Passport Unique Identity
AS	Authentication Server
CS	Control Server
$T_H$	Time Complexity for Hash Function
$T_X$	Time Complexity for XOR Operation
$T_S$	Time Complexity for Symmetric Encryption
$T_P$	Time Complexity for Public Key Operation
$T_R$	Time Complexity for Pseudo Random Function
$T_E$	Time Complexity for Exponential Operation
KIC	Key Information Center
RC	Registration Center

## INTRODUCTION AND LITERATURE REVIEW

---

### 1.1 THE IMPORTANCE OF AUTHENTICATION

Authentication is reliably identifying an entity. It is the most important defence in the security of a system. The active hackers, dictionary attacks, phishing scams and other malicious threats have brought great challenges and potential threats to online transactions [102][127]. Authentication is essential because the numbers of online transactions are increasing exponentially on the web [43]. The most common verification technique is to check whether claimant possesses some information or characteristics that a genuine entity should possess. Authentication process gets complicated when text, visual or audio clues are not available to verify the identity. Authentication protocols are capable of simply authenticating the user to the connecting party and vice-versa [152]. The current technologies used in authentication are password, smart card, passphrase, biometrics, public key cryptography, zero knowledge proof, digital signature, SSL/TLS (Secure Socket Layer/ Transport Layer Security), IPsec (IP Security) and secure shell [18][34][131][132]. The selection of an environment appropriate authentication method is one of most crucial decisions in designing secure systems.

Password is the most commonly used technique for user authentication due to its simplicity and convenience. The main advantage of passwords is that users can memorize them easily without needing any hardware to store them. Each user requires an identifier and a password to access the resources. The user will get service after being verified by the remote server. The password based authentication schemes are susceptible to different types of attacks in insecure communication channels like Internet. Therefore, efficient password authentication schemes are required to authenticate the legitimacy of remote users over an insecure communication channel. Researchers have engineered several acceptable secure password based authentication protocols that can resist different attacks such as dictionary, phishing, denial of service, impersonation, man-in-the-middle, server spoofing, replay, stolen smart card, stolen verifier and parallel session attacks [41]. Applications based on password authentication protocols include remote login systems, Personal Digital Assistant (PDA) and database management systems.

The password based authentication is highly susceptible to phishing attacks by exploiting the visual resemblance of domain names to allure the victims (e.g. [www.eway.com](http://www.eway.com) instead of actual [www.ebay.com](http://www.ebay.com)). Phishing is a website spoofing technique that cheats the user by redirecting the user's confidential information to a web server which is under the control of the attacker. The attacker can use this captured information to make an illegal economic profit in commercial transactions [60]. In phishing attack, the attacker sends a large number of spoofed e-mails to random Internet users that appear to be coming from a legitimate business organization such as a bank. The e-mail requests the recipient to update his personal information and may also warn that failure to reply the request will result in closure of his online banking account. The unsuspecting victim may furnish his personal account details to a phishing website that is under the control of the attacker. The average user can not distinguish a well designed phishing website from the legitimate site because the phishing site replicates visual characteristics of the target organization's website by using similar colors, icons, logos and textual descriptions [7]. Phishing attacks are increasing despite the use of preventive measures like browser based security indicators, integration of blacklist into the web browsers, e-mail filters and content analysis [137]. The phishing attacks are becoming more and more sophisticated and therefore require strong countermeasures [89]. It is important to detect the phishing sites early because most of them are short-lived and cause the damage in the short time span between appearing online and vanishing. Phishing is doing direct damage to the financial industry and is also affecting the expansion of e-commerce.

On the other hand, smart card based password authentication technology has been widely deployed in various kinds of authentication applications such as online banking, remote host login, access control of restricted vaults, activation of security devices and many e-commerce applications due to their low cost, portability, efficiency and the cryptographic properties. Smart card stores some sensitive data corresponding to the user that assist in user authentication.

This chapter is organized as follows. In Section 1.2, the motivation for this research work is given. Literature review is presented in next three sections. Section 1.3 surveys password based authentication protocols followed by anti-phishing protocols (Section 1.4) and smart card based authentication protocols (Section 1.5). In Section 1.6, research gaps have been identified. In Section 1.7, organization of the thesis has been shown and Section 1.8 concludes the chapter.



## 1.2 MOTIVATION

A computer can authenticate human users using passwords, smart cards or biometric devices such as retinal scanner, finger print analyzer and voice recognition system. The most common method for two communicating parties to authenticate each other is using a previously shared password. Using this shared secret, communicating parties may negotiate a session key to encrypt the further communication messages between themselves. An insider or a person close to the user has the maximum ability to steal the user's password because most of the users chosen passwords are limited to the user's personal domain. Therefore, password based authentication protocols are vulnerable to dictionary attacks.

One common practice adopted by the users is to select a single strong password and use it for many accounts instead of choosing a unique password for each account [45]. Here an attacker can take the opportunity to steal the password from one of the user's account and guess it on other accounts of the same user. Password can be stolen from less secure site and reused to compromise the secure site. There are many possible avenues for stealing the passwords that include insider attack, exploiting weak secure site, key logging on public terminal and website spoofing. Hacking and identity theft are the two main concerns in password based authentication protocols.

Online services allow the user to access information ubiquitously and are useful for service providers because they reduce operational costs involved in offering a service to the customer [102]. Three party authentication protocols are used in various applications such as use of trusted server to assist in transactions between buyer and seller in e-commerce. In 2002, online dictionary attacks were launched on ebay accounts. The most common countermeasures to online dictionary attacks are suggested by Pinkas and Sander [115]. Several techniques are available to withstand various attacks but still most of existing password based authentication protocols are vulnerable to different types of attacks such as dictionary and phishing and hence not able to serve as an ideal password authentication protocol. Phishing attacks are also increasing significantly in online transactions. Information Technology (IT) companies such as Microsoft, Google, America On Line (AOL) and Opera have recently started announcing browser integrated blacklist based anti-phishing solutions. A solution is required to list out the new phishing sites in blacklist database quickly otherwise they will do the damage before being included in the

blacklist database. Researchers are putting efforts in developing better password based authentication protocols that should achieve required goals and satisfy security requirements to withstand all possible attack scenarios.

The security analysis of password based authentication protocols found various security weaknesses shortly after they were proposed. These flaws may range from its design to implementation. The different attacks on security protocols are impersonation, dictionary, replay, denial of service, man-in-the-middle, eavesdropping, brute force, leak of verifier, message modification or insertion, stolen smart card, phishing, pharming and other different possible attacks relevant to that specific protocol. Good design criteria of the protocol make it difficult for an attacker to launch attack on the protocol. An attacker can impersonate as a legitimate user by stealing the user's identity and password stored in clear text from the password table on the remote server. Hashed or encrypted passwords can solve this problem. Researchers are working on the solutions where password is not directly stored on the server.

There is no common set of desirable security properties that has been widely adopted for the development of authentication protocols. Password reuse rate increases because people accumulate more accounts with the same password. Researchers have conducted empirical studies on password use and concluded that people tend to pick passwords which represent themselves. The personalized passwords such as phone number, vehicle number, pet's name or a social security number can be cracked by giving a large enough dictionary tries. Gaw et al. [37] give tips and rules for creating strong passwords: use of both uppercase and lowercase letters, at least six characters, avoid common literary names, mix up two or more separate words, create an acronym from an uncommon phrase, avoid passwords that contain login identity, use of numbers, dropping of letters from a familiar phrase, deliberate misspelling and use of punctuation in the password. The average user finds it difficult to remember complex passwords. Moreover, most of the users lack motivation and do not understand the need of password security policies. An ideal password authentication scheme should not store verification table directly on the server, allows the user to change password freely, not revealing password to the server, password transmission should not be in clear text, appropriate password for memorization, unauthorized login can be detected quickly and the scheme should be secure even if the secret key of the server is leaked out or stolen.

Cookie technology is the most innovative feature that made the web stateful. A number of the web applications built on the top of Hyper Text Transfer Protocol (HTTP) need to be stateful and require cookies to maintain the user's state. The web server creates a cookie that contains the state information of a client and stores it on the client computer from where the request is originated. It helps the web server to keep track of the user's movement and his behavior on the visited web server. Therefore, the web server can obtain significant information about the long term habits of their clients. Cookies from the client's computer can be stolen by the attacker through plug-ins or other means but the efficient and effective use of cookies help in authenticating the client's computer.

Smart cards are used in a number of applications on the web. If the smart card is stolen by an attacker, he can extract the information stored in it using different techniques such as by monitoring their power consumption [67][98]. Some other reverse engineering techniques are also available for extracting information from smart cards. A good password authentication scheme should provide protection against different types of attacks such as dictionary, phishing, man-in-the-middle, denial of service, impersonation, server spoofing, replay, stolen smart card, stolen verifier, parallel session and other feasible attacks [143]. Most of the existing password based smart card authentication protocols are vulnerable to impersonation and dictionary attacks and hence are not able to serve as ideal password authentication schemes. Researchers have suggested a number of static identity based smart card authentication protocols such as [65][91][150][157][170]. The static identity leaks out partial information about the user's authentication messages to the attacker. The user may change his password but can not change his identity in password based smart card authentication protocols. The concept of dynamic identity helps in maintaining the anonymity [164] of the user in insecure communication channels like Internet. Security of messages in online transaction in insecure communication channel can be managed with SSL protocol but it is computational intensive. Password information stored on single server is vulnerable to dictionary attack. The concept of multi-server model removes this main point of vulnerability. The effective and efficient dynamic identity based multi-server authentication schemes using smart cards are required for the user's authentication in commercial transactions.

### 1.3 PASSWORD BASED AUTHENTICATION PROTOCOLS

Password is the most common technique to authenticate the users on the web. Short and easily memorable passwords are susceptible to attacks on insecure communication channels like the Internet. On the other hand, complex passwords are difficult to memorize and might get lost or stolen if the user write them down and hence defeat the purpose of constructing secure password based authentication schemes. An attacker can eavesdrop on the communication channel and record the transcript of a successful run of the protocol. Then he can launch offline dictionary attack to generate a response that matches the recorded one. Password based authentication schemes are also subject to man-in-the-middle, insider and other feasible attacks.

In 1992, Bellare and Merritt [12] made the first attempt to develop a password based Augmented Encrypted Key Exchange (A-EKE) protocol, which stores passwords using a one-way hash function so as to defend it against offline dictionary attack. It makes difficult for an attacker to compute the password even if he compromises the hashed password file from the server. In 2000, Bellare et al. [11] suggested authenticated key exchange protocol that is secure against dictionary attack. Another concept of delayed response from the server prevent the attacker to check large number of passwords for a user identity in the same specified time and hence prevents online dictionary attack [17]. An attacker can try several login attempts in parallel on the server to work around the delayed response approach. A different concept of account locking [17] prevents an attacker from trying many passwords for a particular user identity because accounts get locked after a certain number of unsuccessful login attempts. Here an attacker can mount a denial of service attack by choosing a valid user identity and trying several passwords until the account gets locked. That causes a lot of inconvenience to the owners of the locked accounts. Moreover, setting up customer service to handle the user calls regarding locked accounts would not be cost effective. One more concept of extra computation enforces the user to perform some nontrivial computation and send proof of it during login process. This computation would be negligible for a single login attempt and the cost of computation increases with each login failure [17].

In 2002, Pinkas and Sander [115] suggested Reverse Turing Tests (RTT) for authentication so that human user can easily pass out the test but it is very difficult for the automated program to pass out the test as shown in Figure 1.1. The examples of RTT are colored images with distorted text that appears on different websites such as Yahoo,

Hotmail and Paypal, while the user authenticates itself to these websites. Pinkas and Sander assumed that the users login from limited set of computers containing activated cookies. The user is asked to pass RTT during login from a new computer or after entering a wrong password from a trusted computer. Stubblebine and Oorschot [136] observed that RTT based protocols are vulnerable to RTT relay attack. Suppose an attacker wants to perform an online dictionary attack on the ebay.com and hence needs correct responses to RTT produced by ebay.com. An attacker have to host or hack a high volume website such as google.com and install attack software, which initiates a fraudulent attempt to login at ebay.com, when a visitor accesses google.com. The RTT challenge produced by ebay.com is redirected to the user trying to view the google.com page. To counter these kinds of RTT relay attacks, Stubblebine and Oorschot [136] developed a protocol based on the user's login history and suggesting modifications to Pinkas and Sander's RTT based protocol so that only trustworthy machines are used to store cookies.



**Figure 1.1: Reverse turing test**

Two most common solutions for password management are: either let the users choose their own passwords and store them somewhere safe or the server assign fixed passwords for each site or service that can be computed by the client, whenever needed. Most of the authentication schemes use the first approach for passwords management and PwdHash [120] uses the second approach. In 1998, Kelsey et al. [64] proposed a scheme in which a new password is computed by repeatedly iterating a hash function on the original master password. Therefore, if the scheme is parameterized to use  $k$  iterations, an attacker needs to compute  $k$  hash functions for each guess. In 1996, Manber [97] and in 1997, Abadi et al. [1] use the technique in which password is concatenated with a random value known as a “password supplement” before it is hashed. To check a password guess, an attacker needs to perform a hash for each possible supplement corresponding to each guessed password until the complete dictionary is searched. In 2005, Halderman et al. [45] proposed a convenient method for securely managing Passwords. This approach made the

guessing of passwords more time intensive so as to raise the cost of a brute force attack. The main concern in password authentication protocols is to prevent dictionary attack.

A large number of cookies based authentication solutions were proposed for password management such as Microsoft's Passport initiative (Window live ID) [100]. Cookies are obscure to the users and are completely controlled by the web server. Therefore, cookies are good choice for a Single Sign On (SSO) solution. In 1999, Samar [123] suggested SSO using HTTP cookies for web based environment. He suggested three approaches namely centralized cookie server, decentralized cookie server and centralized login server to provide SSO for web applications. The client can choose any of the three SSO solutions depending upon the requirements of web application in terms of deployability, performance and management.

In 2000, Park and Sandhu [112] suggested address based (IP\_Cookie), password based (Pswd\_Cookie) and digital signature (Sign\_Cookie) based secure cookies for the user authentication. They suggested different set of inter dependent cookies such as name cookie, life cookie, password cookie and seal cookie. The role server issues one or more cookies by storing it on the client's computer. As the client connects to the web server, the relevant cookies are transmitted to the web server. Any of the web servers that accept these cookies verifies the cookie and provides the access of resources depending upon the role of the cookie. These secure cookies are used for user authentication especially in e-commerce transactions on the web.

In 2001, Fu et al. [35] designed a secure cookie based client authentication framework in conjunction with Secure Socket Layer (SSL) protocol based on informal survey of commercial protocols. They claimed that their protocol is secure against different types of attacks launched by the attacker. In 2005, Liu et al. [90] analyzed and found that Fu et al.'s protocol is susceptible to cookie replay, volume attacks and does not provide high-level confidentiality. Therefore, they proposed a cookie based authentication protocol that provides confidentiality, integrity and protection from replay attack. Their scheme does not involve any database lookup or public key cryptography. It also does not require changes in Internet cookie specification and can be easily deployed on an existing web server.

In 2002, Xu et al. [156] presented a cookie based authentication protocol in which the server stores credit card information of each client in their respective cookie. They

exploited the concept of secure distributed storage by storing some sensitive information in the HTTP cookie in encrypted form. The web server stores the One Time Pad keys in its local database and encrypt/decrypt the cookies using these keys. This protocol can not handle multiple simultaneous requests with the same cookie. Moreover, the server has the overhead of encryption and decryption for verifying each cookie and also has to do database lookups.

In 2005, Blundo et al. [15] proposed an encrypted cookie based web authentication protocol. The main weakness of this cookie based protocol is that the server has to do database lookups for verifying each received cookie. In 2005, Wang et al. [147] presented cookies based password authentication protocol that uses cryptographic puzzles to prevent online dictionary attack. Their scheme increases the computational burden for an attacker and imposing negligible load on the legitimate clients as well as on the authentication server.

In 2005, Imamoto and Sakurai [54][55] gave a very good concept of one-time user's identity based on Diffie-Hellman key exchange that can be used only once. A different identity corresponding to the same user is generated for each new session. The key idea is to prevent an attacker from detecting the user's identity from even an insecure communication channel like the Internet. One-time identity has the advantage that the attacker can not identify who is communicating even when he eavesdrops on the communication channel. Moreover, one-time identity can be used only once and the attacker can not compute the real identity from one-time identity of the user. The concept of dynamic identity provides two-factor authentication using its real identity and the password. In 2006, Goyal et al. [40] proposed an authentication protocol that prevents online dictionary attack in an effective manner. This protocol uses challenge response mechanism and one-way hash function [108] to thwart online dictionary attack. The legitimate user can easily login on to the web server and the computational efforts increases for the attacker trying thousands of authentication requests in an attempt to launch online dictionary attack.

Authentication between two communicating parties can be achieved using a common session key for that session, which is derived from the password shared between the clients [163]. Three Party Authentication Key Exchange (3PAKE) protocol can solve key management problem using a trusted server that shares a secret with every client and

shoulders the responsibility to achieve mutual authentication as well as a session key agreement between the clients [110]. In 1995, Steiner et al. [135] extended the two party authentication protocol into a three party encrypted key exchange protocol in which each client shares an easy to remember password with a trusted server and hence the server is used to authenticate the communicating parties. The most common attack on 3PAKE protocols is dictionary attack [109]. In 1995, Ding and Horster [30] classified dictionary attack into three classes as offline dictionary attack, undetectable online dictionary attack and detectable online dictionary attack. Ding and Horster [30] showed that the protocol proposed by Steiner et al. [135] is vulnerable to undetectable online dictionary attack.

3PAKE protocols without server public key such as Kerberos [107] and KryptoKnight [103] are vulnerable to dictionary attack. Hence, a number of secure three party key exchange protocols with server's public key were proposed [38][84][139] so that they can resist dictionary attack. In 2000, Lin et al. [84] pointed out that Steiner et al.'s protocol also suffers from offline dictionary attack and presented an improved version based on the server's public key. The complex computation of public key cryptosystems causes heavy computational overhead and also certificates are needed to verify the server's public key to avoid impersonation attack. Therefore, public key based authentication schemes are not suitable for applications having limited computational resources. To overcome this drawback, schemes were proposed by Bird et al. [14], Chang and Chang [20], Jaung [59], Lee et al. [75], Lin et al. [86], Neuman and Ts'o [107] and Steiner et al. [135] that do not involve the server's public key. Some of these 3PAKE protocols without server's public key resist both offline and undetectable online dictionary attacks but more number of rounds are needed in these protocols.

The two important efficiency metrics of a protocol are the number of steps and the number of rounds. In 2004, Lee et al. [75] proposed an enhanced three party encrypted key exchange protocol without the server's public key with less number of rounds. In 2005, Abdalla and Pointcheval [2] proposed a password based encrypted key exchange protocol in which both clients share a human memorable password with a trusted server. This password is used to agree on a unique secure session key for each new session. They claimed that their protocol can resist various known attacks without requiring the server's public key. In 2006, Wang and Hu [148] found that Abdalla and Pointcheval's [2] protocol is vulnerable to man-in-the-middle attack and undetectable online dictionary attack. In



2007, Lu and Cao [92] proposed a three party password encrypted key exchange authentication protocol based on [2]. However in 2008, Guo et al. [44] pointed out that Lu and Cao's [92] protocol is completely insecure against the man-in-the-middle and undetectable online dictionary attacks. Then, they presented an improved version of this protocol free from these flaws. In 2008, Phan et al. [113] found an unknown key share and an undetectable online dictionary attacks on Chung and Ku's [27] protocol. In 2008, Chen et al. [24] proposed a round and computational efficient three party authentication key exchange protocol. In 2000, Bhattacharyya and Nandi [13] proposed a password based encrypted key exchange protocol based on linear and non-linear groups in which both clients share a human memorable password with a trusted server. In 2007, Sivalingam et al. [105] addresses the cryptographic key exchange problem in a wireless network. Their technique makes use of ranging information between nodes to establish a shared secret key between security principals. This key exchange technique is based on the notion of symmetric ranging combined with Merkle's puzzles. In 2008, Moona et al. [36] proposed a solution which uses the personal mobile devices held by the user to interact with the service outlets. In 2008, Lei et al. [79] proposed a virtual password concept based on the randomized linear functions involving human computing to secure the user's password in online transaction. They analyzed that their scheme defends phishing, key logger and shoulder surfing attacks.

#### **1.4 ANTI-PHISHING PROTOCOLS**

To mitigate the risk of phishing attacks, defense mechanisms have been deployed at both the client and the server sides. These solutions include the digitally signed e-mail (S/MIME), anti-phishing plug-ins for browsers like SpoofGuard [134], Kirda and Kruegel's measures [66], blacklist integration into Internet Explorer browser [114], Google safe browsing [39] and Mozilla phishing protection [104].

In 1999, RFC 2617 [119] proposed a Digest Access Authentication scheme that uses a password digest to authenticate a user. In 2004, Herzberg and Gbara [47] constructed a TrustBar that associates the logo with the public key certificate of the visited site. In 2004, SecurID [121] scheme was suggested that uses one-time password for authentication and has been deployed in a number of financial organizations. In 2005, PwdHash [120] scheme was suggested that authenticates a user with one-way hash [116] on  $\langle \text{password}, \text{domain name} \rangle$  instead of the password only so as to defeat the visual

similarity of the domain name. This technique creates a domain specific password that becomes useless if it is submitted to another domain. However, PwdHash is susceptible to offline dictionary attack and ineffective against pharming or DNS spoofing attack where the attacker presents correct domain name to the browser but redirects the user's request to its own server. In 2005, Synchronized Random Dynamic (SRD) [165] scheme was suggested that is having an internal reference window, whose color changes randomly and sets up the boundary of the browser window with different colors according to certain rules. This scheme is impractical for hand held devices and is also ineffective if the attacker creates a bogus reference window to overlap the original reference window. In 2005, Dhamija and Tygar [29] proposed a technique that uses Dynamic Security Skin (DSS) on the user's browser. It creates a dedicated window containing a specific image shared between the user and the server for inputting user name and password so as to defeat a bogus window. In 2005, SpoofGuard [134] technique was suggested that examines the domain name, images and links on the web pages and raises an alarm to the users if the site has a high probability of being a phishing site. In 2005, Adelsbach et al. [3] combines different concepts of an adaptive web browser toolbar that summarizes all relevant information and allows user to get required crucial information at a glance. This toolbar is a local component of user's system on client side and hence a remote attacker can not access it by means of active web languages. The main disadvantage of this toolbar scheme is that the user has to recognize his personal image at each login.

In 2006, Wu et al. [155] found that 13-54 % of the users visited phishing websites, despite the warnings from anti-phishing toolbars. Several browser toolbars like SpoofGuard [134] and TrustBar [47] have been proposed to find a pattern in phishing websites and alerts the user if the given site matches the pattern. In 2006, Juels et al. [62] suggested the use of cache cookies for the user identification and authentication that uses the browser cache files to identify the browser. These cookies are easy to deploy because it does not require installation of any software on the client side. Then they extended the concept to active cookie scheme, which stores the user's identification and a fixed IP address of the server. During the client's visits to the server, the server will redirect the client request to the fixed IP address so as to defeat phishing and pharming attacks. SiteKey [121] has been deployed by the bank of America [9] and Yahoo's sign-in seal [158] to prevent a phishing attack. Initially, it recognizes the client's browser by a previously installed cookie and then requests the user to enter his user name. After

successful authentication, a user specific image is displayed on the browser. Finally, the user submits his password after recognizing the image displayed on the browser to authenticate itself. In 2006, Automatic Detecting Security Indicator (ADSI) [117] was proposed as an enhancement of toolbar scheme that generates a random picture and embeds it in to the existing web browser. It can be triggered by any security related event occurred on browser and then performs automatic checking on current active security status. In case mismatch in embedded images is detected, an alarm goes off to alert the users. ADSI can not prevent man-in-the-middle and phishing attacks with self sign certificate.

In 2007, Ludl et al. [93] analyzed legitimate and phishing websites to define the metrics that can be used to detect a phishing site. In 2007, Microsoft deployed Sender ID [99] and Yahoo deployed DomainKeys [158] protocols to detect the phishing e-mails. In 2007, Karlof et al. [63] proposed the cookies based Locked Same Origin Policy (LSOP) that enforces access control for the SSL web objects based on the server's public key. In 2007, Microsoft [114] integrated the blacklisted phishing domains in to the web browser so that browser refuses to visit these phishing websites. In 2007, Google Safe Browsing [39] uses a blacklist of phishing URLs to find out a phishing site. This technique can not recognize those phishing sites which are not present in the blacklist maintained by the server. This approach can prevent phishing attack if the fraudulent sites are discovered and listed quickly. A study carried out by the Microsoft [114] in 2007 reported that the Microsoft's blacklist is superior to the Google's blacklist. Another study initiated by the Mozilla [39] drew the opposite conclusion in favor of the Google. In 2007, Zhang et al. [173] performed a similar study that tested the detection rates of different blacklist based anti-phishing solutions. Their dataset includes 100 phishing URLs collected over a period of three days in November 2006. They analyzed ten toolbars experimentally and reported that the only toolbar consistently identifying more than 90 % of phishing URLs also classified 42 % of legitimate URLs incorrectly as phishing. VeriSign [146] is providing a commercial anti-phishing service. The company is crawling millions of web pages to spot out clones to identify phishing websites. In 2007, Adida [4] suggested a FragToken scheme that uses the URL portion as an authenticator and accordingly change response for authentication. FragToken is only useful in low security environment like blog because it is vulnerable to man-in-the-middle attack.

In 2007, Gouda et al. [42] proposed an anti-phishing single password protocol that allows the user to choose a single password of his choice for multiple online accounts on the web. In 2008, Yongdong et al. [167] proposed SSO anti-phishing technique based on encrypted cookie that defeats phishing and pharming attacks. They mentioned different reasons for web spoofing like self signed certificates or insertion of a spoofed image representing security indicator where one does not exist. Most of the users can not distinguish the spoofed browser’s security indicators from actual security indicators such as public key certificate, URL bar and locked icon. It encrypts the sensitive data with the server’s public key and stores this cookie on the user’s computer. This Encrypted Cookie Scheme (ECS) has advantage that the user can ignore SSL indicator in online transaction procedure.

NetCraft Tool Bar [106] is based on risk rating system. Risk is computed based on the age of domain. This technique uses the database of phishing sites and hence might not recognize new phishing sites successfully. SpoofStick [133] provides basic domain information. It will show that you are on paypal.com when you are on paypal site or will display you are on IP address of spoofed site. It is not efficient against spoofed sites opened in multiple frames. McAfee SiteAdvisor [96] protects the users from spyware and ad-ware attacks. It uses the crawler to create a large database of malware and test results on them to provide rating for a site. This technique will not be able to find new phishing sites. The ebay Tool Bar [31] solution is based on “Account Guard” that changes color if the user is on a spoofed site and is specifically designed for ebay and paypal websites. Table 1.1 gives the domains, country domains and phishing count. Table 1.2 gives the cost and functionality comparison among recent anti-phishing protocols [167].

**Table 1.1: Domain, country domain and phishing count [7]**

Domain	Phishing count	Country domain	Phishing count
.com	12275	.in	252
.biz	353	.us	334
.net	2305	.uk	1584
.org	1425	.hk	2278

**Table 1.2: Cost and functionality comparison among different anti-phishing protocols**

Web based password protocols	Need of checking browser indicators	Need of checking URL	Need of checking GUI	Need of installing additional software	Security ignorant users	Dictionary attack
SSL [34]	Y	N	N	N	N	Y
Digest Access [119]	Y	Y	N	N	N	Y
PwdHash [120]	Y	N	N	Y	N	Y
SRD [165]	Y	N	Y	Y	N	Y
DSS [29]	Y	Y	Y	Y	N	Y
SpoofGuard [134]	Y	Y	N	Y	N	Y
LSOP [63]	Y	Y	Y	Y	N	Y
Cache cookies [62]	Y	N	Y	N	N	Y
SPP [42]	Y	Y	N	Y	N	Y

\* Y implies yes and N means no

**Table 1.3: Phishing attacks and their countermeasures [7]**

Attacks	Measures
Malware	Firewall, Anti-virus, Anti-keylogger & IDS
Phishing e-mail	Digitally signed e-mail, Bank e-mail
Bogus web sites	Trusted path browser, Browser indicator, Dynamic security skin
Identity theft	Smart card, Dynamic identity

**Table 1.4: Organization based phishing sites [7]**

Organization	Phishing sites	Success rate (%)
ebay	231	14.8
paypal	211	7.6
Bank of America	28	2
HSBC	7	0
amazon	4	4

Table 1.3 gives the attacks and their countermeasures. Table 1.4 gives the statistics of organization based phishing sites [7].

## 1.5 SMART CARD BASED AUTHENTICATION PROTOCOLS

Smart card based authentication protocols provide multi-factor authentication to the user by acquiring the smart card and knowing the identity and password [124]. An attacker can not impersonate as a legitimate user unless the smart card, identity and the password of the user are compromised. That is why, smart card based authentication schemes with low entropy password are more resistant to dictionary attack.

In 1981, Lamport [72] proposed a password based authentication scheme that authenticates the remote users over an insecure communication channel. Lamport's scheme removes the problems of password table disclosure and communication eavesdropping. Since then, a number of remote user authentication schemes have been proposed to improve security, efficiency and cost. To prevent the password table from being stolen or modified by the attackers, solutions have been proposed such as [58] in which the password table is not required to be kept on the server. In 1995, Haller [46] proposed a secure key (S/KEY) one-time password for an Internet draft RFC 1760. Later on, this S/KEY authentication scheme is found to be flawed for replay, server spoofing and dictionary attacks [101][166]. In 2000, Sandirigama et al. [126] and in 2001, Lin et al. [85] proposed the Simple Strong Password Authentication (SSPA) and Optimal Strong Password Authentication (OSPA) protocol respectively, which turn out to be better than Lamport's scheme in terms of storage utilization, computation time and transmission overhead. However in 2002, Chen and Ku [23] showed that the SSPA and OSPA protocols are vulnerable to stolen verifier attack.

In 1996, Shoup and Rubin [130] proposed an extension of Bellare and Rogaway's model (1995) [10] for three party key distribution protocol based on smart cards that is used to store the long term keys. In 2000, Hwang and Li [52] found that Lamport's scheme [72] is vulnerable to the risk of a modified password table and the cost of protecting and maintaining the password table is large. Therefore, they proposed a cost effective remote user authentication scheme using smart cards that is free from the mentioned risk. Hwang and Li's scheme [52] can withstand replay attack and also authenticate the remote users without maintaining a password table. In 2000, Sun [138] proposed an efficient smart card based remote user authentication scheme to improve the efficiency of Hwang and Li's scheme [52].

In 2002, Chien et al. [25] found that Sun's scheme only achieves unilateral user authentication so that only authentication server authenticates the legitimacy of the remote

user. Chien et al. also proposed a remote user authentication scheme and claimed that their scheme provides mutual authentication, requires no verification table, password selection freedom and uses only few hash operations. In 2004, Hsu [50] demonstrated that Chien et al.'s scheme is susceptible to parallel session attack so that the intruder without knowing the user's password can masquerade as the legitimate user by creating a valid login message from the eavesdropped communication between the authentication server and the user. In 2004 and 2005, Lee et al. [74][77] improved Chien et al.'s scheme by eliminating the parallel session attack. In 2005, Yoon and Yoo [171] found that Lee et al.'s [74][77] schemes are susceptible to masquerading server attack and proposed an improvement on Lee et al.'s schemes. In 2009, Kim and Chung [65] found that Yoon and Yoo's scheme [171] easily reveals a user's password and is susceptible to masquerading user attack, masquerading server attack and stolen verifier attack. Therefore, Kim and Chung proposed a new remote user authentication scheme. They claimed that the proposed scheme resolves all aforementioned security flaws, while keeping the merits of Yoon and Yoo's scheme.

In 2004, Ku and Chen [69] showed that Chien et al.'s scheme [25] is vulnerable to reflection attack, insider attack and is not reparable. Therefore, they proposed an improved remote user authentication scheme using smart cards to preclude the weaknesses of Chien et al.'s scheme. Later, Yoon et al. [168] found that the Ku and Chen's scheme is vulnerable to parallel session attack and insecure for changing the user's password in password change phase. They presented an enhancement to Ku and Chen's remote user authentication scheme. In 2009, Hsiang and Shih [48] showed that Yoon et al.'s scheme [168] is vulnerable to parallel session attack, masquerading attack and password guessing attack using stolen smart card and proposed an improved scheme free from these flaws.

In 2002, Hwang et al. [53] proposed a remote user authentication scheme that does not require any password or verification table on the remote server and legitimate users are free to choose and change their password freely without the help of a remote server. They claimed that their scheme provides effective authentication and requires less computation as compared to other schemes proposed by Wu [153] in 1995, Jan and Chen [58] in 1998, Yang and Shieh [160] in 1999, Hwang and Li [52] in 2000 and Chien et al. [25] in 2002. In 2005, Yoon et al. [170] found that Hwang et al.'s scheme [53] is vulnerable to stolen verifier attack and denial of service attack using the stolen smart card. They proposed an improved scheme to preclude the weaknesses of Hwang et al.'s scheme [53].

A number of smart card based remote user authentication protocols have been proposed due to the convenience and secure computation provided by the smart cards. However, most of these protocols do not protect the user's identities in authentication process. User's anonymity is an important issue in many e-commerce applications. A number of static identity based remote user authentication schemes have been proposed to improve security, efficiency and cost. The static identity leaks out partial information about the user's authentication messages to the attacker. On the other hand, the dynamic identity based authentication schemes provide multi-factor authentication by acquiring the smart card, knowing the identity and password and hence more suitable to e-commerce applications [22]. The dynamic identity is computed from the user specific parameters and is different for the same user in each new session of the protocol. Therefore in 2004, Das et al. [28] proposed a dynamic identity based remote user authentication scheme to authenticate the users that preserves the user's anonymity. Their scheme uses dynamic identity to achieve this purpose and the user's identity is dynamically changed during each new authentication process. The server does not require to keep any verification table and the users can choose and change their passwords without the server's help. Das et al. claimed that their scheme is secure against stolen verifier attack, replay attack, forgery attack, password guessing attack, insider attack and identity theft. However, many researchers demonstrated susceptibility of Das et al.'s scheme to different attacks. In 2005, Chien and Chen [26] pointed out that Das et al.'s scheme fails to preserve the user anonymity effectively because the authentication messages belonging to the same user can be identified. They proposed an authentication scheme and claimed that the proposed scheme preserves the user's anonymity more efficiently. Though their scheme preserves the user's anonymity and secure against various attacks but it is highly computational intensive. In 2005, Liao et al. [81] proposed an improved scheme that enhances the security of Das et al.'s scheme and achieves mutual authentication. In 2006, Yoon and Yoo [172] demonstrated a reflection attack on Liao et al.'s scheme that breaks the mutual authentication. They also proposed an improved dynamic identity based mutual authentication scheme that eliminates the security flaws of Liao et al.'s scheme.

In 2006, Liou et al. [88] suggested a new dynamic identity based remote user authentication scheme using smart cards that achieves mutual authentication. They claimed that their scheme preserves the advantages of Das et al.'s scheme and overcomes the weaknesses of Das et al.'s scheme. In 2008, Shih [129] demonstrated that Liou et al.'s



scheme fails to achieve mutual authentication. In 2009, Wang et al. [151] argued that Das et al.'s scheme [28] is vulnerable to stolen smart card attack [8] and proposed an improved scheme to preclude the weaknesses of Das et al.'s scheme.

In 2005, Lee and Chiu [76] proposed a smart card based password authentication scheme as an improvement of Wu and Chieu's [154] scheme. In 2006, Liao et al. [82] proposed a new authentication scheme and claimed that their scheme satisfies all the security properties specified in their paper. In 2008, Yang et al. [159] showed that the scheme proposed by Liao et al. [82] does not satisfy some of their security properties. They also described an offline password guessing attack on Liao et al.'s scheme [82] to find the client's password once the client's smart card is stolen by the attacker. In 2009, Xu et al. [157] found that Lee et al.'s [77] scheme is susceptible to offline password guessing attack and Lee and Chiu's [76] scheme is vulnerable to forgery attack in case the smart card is lost or stolen by the attacker. Therefore, Xu et al. proposed an improved scheme and claimed that improved scheme eliminates the security flaws of [76][77].

In 1999, Yang and Shieh [160] proposed a timestamp based password authentication scheme using smart cards. In their scheme, users are allowed to choose and change their passwords freely and the remote server does not require to keep the password table or verification table. The verification and authentication data pertaining to user is generated and provided by the user to the server during authentication phase. However, many researchers [19][32][128] demonstrated vulnerability of Yang and Shieh's scheme to forged login attack and other attacks. In 2003, Shen et al. [128] proposed an improved scheme to preclude the weaknesses of Yang and Shieh's scheme [160]. They claimed that their scheme can resist the forged login attack and uses mutual authentication to protect from the forged server attack. In 2005, Yoon et al. [169] demonstrated that the scheme proposed by Shen et al. [128] was still vulnerable to forged login attack. The attacker can intercept the legitimate user's login request message and register the new smart card corresponding to computed identity from intercepted login request messages to carry out the forged login attack. In 2007, Sakurai et al. [56] proposed an efficient authentication scheme for identity based cryptography system that stores protected secret key in the smart card which is made by combination of biometrics and secret key [125]. The user can restore the secret key from protected secret key by presenting his finger print to smart card that has protected secret key and helper data. In 2008, Liu et al. [91] proposed a nonce based mutual authentication scheme using smart cards and claimed that their scheme is free from forged login attacks. In 2009, Sun et al. [140] demonstrated man-in-the-middle attack on Liu et al.'s scheme [91]. In 2009, Lee and Sivalingum [78] proposed a One-Time

Password (OTP) authentication mechanism using smart cards particularly to thin clients in wireless networks. Table 1.5 gives the comparison of related smart card based authentication schemes in terms of important metrics and vulnerabilities to different attack scenarios.

**Table 1.5: Comparison among related smart card based authentication schemes**

Smart Card Based Authentication Protocols	Password Guessing	Verification Table	Mutual Authentication	User's Anonymity	Stolen Verifier Attack	Impersonation Attack
Lamport (1981) [72]	Y	Y	N	N	Y	Y
Hwang and Li (2000) [52]	Y	N	N	N	N	Y
Sun (2000) [138]	Y	N	N	N	N	Y
Chien et al. (2002) [25]	Y	N	Y	N	Y	Y
Das et al. (2004) [28]	Y	N	N	Y	Y	Y
Chein and Chen (2005) [26]	Y	N	N	Y	Y	Y
Liao et al. (2005) [81]	Y	N	Y	Y	N	Y
Yoon and Yoo (2006) [172]	N	N	Y	Y	N	Y
Liou et al. (2006) [88]	Y	N	Y	Y	Y	Y
Wang et al. (2009) [151]	Y	N	Y	N	Y	Y

In 2000, Ford and Kaliski [33] proposed the first multi-server password based authentication protocol that splits a password among multiple servers. This protocol generates a strong secret using password based on the communications exchanges with two or more independent servers. The attacker can not compute the strong secret unless all the servers are compromised. This protocol is highly computational intensive due to the use of public keys by the servers. Moreover, the user requires a prior secure authentication channel with the server. Therefore in 2001, Jablon [57] improved this protocol and proposed a multi-server password authentication protocol in which the servers do not use public keys and the user does not require prior secure communication channels with the servers. In 2000, Lee and Chang [73] proposed an user identification and key distribution protocol for multi-server environment based on the hash function [71] and difficulty of factorization. In 2001, Li et al. [80] proposed a remote password authentication protocol for multi-server environment. This password authentication system is a pattern classification system based on an artificial neural network. The user has to register with registration centre once and then can obtain services from multiple servers without requiring to register individually with each server. The users can choose their passwords freely and the server does not require to keep any verification table. This protocol can withstand the replay attack effectively but it requires intensive communication and

computation efforts. In 2002, Mackenzie et al. [94] proposed a password based multi-server authentication protocol using threshold user authentication and analyzed its security properties with formal methods using random oracle model.

In 2003, Lin et al. [87] proposed a multi-server authentication protocol based on the ElGamal digital signature scheme that uses simple geometric properties of the Euclidean and discrete logarithm problem concept. The server does not require to keep any verification table but the use of public keys makes this protocol computational intensive. In 2003, Raimondo and Gennaro [118] proposed two multi-server password authentication protocols in which the user has to communicate in parallel with all authentication servers. They proved that these protocols are provable secure in the standard model. The attacker has to compromise minimum threshold number of servers to gain any meaningful information regarding the password of the user. These two protocols differ in the way the client interacts with the different servers. In these protocols, the servers are equally exposed to the user as well as to the attacker. In 2003, Brainard et al. [16] proposed a password based two-server authentication protocol in which only one server was exposed to the users. The use of public keys makes this system computationally intensive. Moreover, it uses SSL to establish a session key between a user and the front-end server to provide authentication but it provides only unilateral authentication. In 2005 and 2006, Yang et al. [161][162] extended this two-server protocol in which only a front-end server communicate directly with users and the back-end control server does not interact with users directly. The back-end control server in this protocol also requires public key for its operations. Two-server authentication protocols are proposed to eliminate the main point of vulnerability in the single-server systems. The concept of distributing the password verification information and authentication functionality into two servers requires additional efforts from an attacker to compromise two servers to launch successful offline dictionary attack. The attacker has to compromise both the servers simultaneously to launch offline dictionary attack.

In 2004, Juang [61] proposed a smart card based multi-server authentication protocol using symmetric encryption algorithm without maintaining any verification table on the server. In 2004, Chang and Lee [21] improved Juang's protocol and proposed a smart card based multi-server authentication protocol using symmetric encryption algorithm without any verification table. Their protocol is more efficient than the multi-server authentication protocol of Juang [61]. In 2004, Tsaur et al. [145] proposed a smart

card based multi-server authentication protocol that uses the RSA cryptosystem and Lagrange interpolating polynomial without using any password verification table. This protocol involves high communication and computation costs. In 2006, Mackenzie et al. [95] proposed a password-authenticated key exchange protocol that uses a set of servers with known public keys so that a certain threshold number of servers must participate to authenticate a user. Therefore, the attacker has to compromise the minimum threshold number of servers to launch offline dictionary attack.

In 2007, Hu et al. [51] proposed a password authentication key agreement protocol for multi-server architecture in which user can access multi-server using smart card and one weak password. The client and the server authenticate each other and agree on a common secret session key. The proposed protocol is more efficient and more user friendly than that of Chang and Lee [21] protocol. In 2008, Tripathy and Nandi [141][142] suggested a secure and efficient user identification and key distribution remote user authentication and key establishment scheme for multi-server environment using smart cards based on reversible cellular automata. In 2008, Tsai [144] proposed a multi-server authentication protocol using smart cards based on the nonce and one-way hash functions that does not require to store any verification table on the server and registration center. The proposed authentication protocol is efficient as compared to other such related protocols because it does not use any symmetric and asymmetric encryption algorithm for its implementation. In 2009, Liao and Wang [83] proposed a dynamic identity based remote user authentication protocol using smart cards to achieve user's anonymity. This protocol uses only hash functions to implement a strong authentication for the multi-server environment. It provides a secure method to update the user's password without the help of trusted third party. In their paper, they claimed that suggested protocol can resist various known attacks. However, in 2009, Hsiang and Shih [49] found that Liao and Wang's protocol is susceptible to insider attack, masquerade attack, server spoofing attack, registration center spoofing attack and is not reparable. Furthermore, it fails to provide mutual authentication. To remedy these flaws, Hsiang and Shih proposed an improvement over Liao and Wang's protocol.

## **1.6 RESEARCH PROBLEMS**

The aim of the present research work is to investigate password based authentication protocols and propose efficient and better solutions. The challenges inherent in the

development of password based authentication protocols include:

1. Potential scope of research work contains the important issues identified as the dynamic identity management, multi level password verification and two layers based password concept so that efficient password authentication schemes can be designed which satisfy all the security requirements and achieve the goals of an ideal password authentication scheme. An ideal password authentication scheme should have protection from eavesdropping, denial of service, impersonation, parallel session, password guessing, replay, stolen smart card, stolen verifier, man-in-the-middle, malicious user, malicious server, phishing, pharming and other feasible attacks relevant to that protocol and should achieve mutual authentication.
2. One of the reasons for success of phishing and dictionary attacks is high rate of password reuse because users tend to use the same password with more and more accounts. Users find it difficult to remember several complex passwords and hence it is difficult to prevent phishing and dictionary attacks. One of the thrust and major area of research is to find technical solutions for the online password management without significantly changing the user's behavior.
3. Researchers have proposed different anti-phishing techniques based on the web browser security indicators. The main reason for the success of phishing attack is that users do not constantly notice the presence of a security indicator or find it difficult to understand the meaning of these browser based security indicators. Therefore, the web browser must provide an easy to use interface for the users and minimize the efforts in checking the browser based security indicators. A solution is required in which the user does not have the need of interpreting the browser based security indicators.
4. Researchers have proposed an anti-phishing solution based on integration of blacklist into the web browsers. Therefore, effective techniques must be devised to check whether a web page is legitimate or a phishing page. It is not easy to provide a mechanism to prevent the users from visiting a phishing site. It is important to detect phishing pages early because most of them are short lived and do the damage in time span between appearing online and vanishing.
5. Different solutions to thwart online dictionary attacks in authentication protocols have been suggested. These solutions include Reverse Turing Tests (RTT), single password

to different accounts, virtual password generation, two layers based password verification and password based authentication using multi-server environment. Most of the suggested solutions are vulnerable to dictionary attacks, even the most commonly used RTT is vulnerable to RTT relay attack. More effective and efficient techniques are required to thwart online dictionary attacks.

6. The role of cookies can be enhanced in virtual password authentication protocols to preserve the advantages of basic password authentication and simultaneously increasing the efforts required for online dictionary attacks. The legitimate client can easily authenticate itself to the web server from any computer irrespective of whether that computer contains cookie or not. However, the computational efforts required from the attacker during login on to the web server increases with each login failure. Therefore, even the automated programs can not launch online dictionary attacks on the proposed protocol.
7. Single-Sign-On (SSO) provides an environment in which the client sign in once and are able to access the services offered by different servers under the same administrative control. However, the user's password verification information stored on the single centralized server is a main point of susceptibility and remains an attractive target for the attacker. Therefore, the concept of SSO password based two-server architecture that uses two-server paradigm so that password verification information is distributed between two servers (an authentication server and a control server) is more resistant to dictionary attacks as compared to existing single-server password based SSO authentication protocols.
8. Smart card based password authentication is one of the most convenient ways to provide multi-factor authentication for the user by acquiring the smart card and knowing the identity and password. They are used in financial transactions and therefore require secure authentication protocols with high computational and communication efficiency. The protocol designer should also take memory requirement, number of rounds and time complexity into consideration.
9. A number of static identity based remote user authentication schemes have been proposed to improve security, efficiency and cost. The static identity leaks out partial information about the user's authentication messages to the attacker. On the other hand, the dynamic identity based authentication schemes preserve the user's anonymity. The dynamic identity is computed from the user specific parameters and is

different for the same user in each new session of the protocol. Therefore, the dynamic identity based authentication schemes are more suitable to e-commerce applications.

10. In e-commerce, the number of servers providing the services to the user is usually more than one and hence secure authentication protocols for multi-server environment are required. The concept of multi-server authentication helps to distribute the user's verifier information among different servers. Therefore, the multi-server architecture based authentication protocols make it difficult for the attacker to find out any significant authentication information related to the legitimate users. The issue of remote login authentication with smart card in single server environment has already been solved by a variety of schemes. These conventional single-server password authentication protocols can not be directly applied to multi-server environment because each user needs to remember different sets of identities and passwords. Researchers are working in this direction to develop secure and efficient remote user smart card based authentication protocols for multi-server environment.

## 1.7 ORGANIZATION OF THE THESIS

The present thesis embodies detailed investigation on the contributions made by the author. The thesis is organized into eight chapters and the work included in each chapter is briefly outlined as follows:

After presenting an overview of authentication protocols and a brief literature review in the **present Chapter 1**, the subsequent chapters cover the topics as below.

The **Chapter-2** presents the dynamic identity based single password anti-phishing protocol. The security analysis of this protocol is carried out. The cost and functionality analysis of this protocol with the other related protocols has been presented.

The **Chapter-3** presents a cookie and an inverse cookie based virtual password authentication protocols. The security analysis of these protocols is carried out. The presented protocols are very effective to thwart online dictionary attacks.

The **Chapter-4** presents the SSO password based two-server authentication protocol. The security analysis of this protocol is carried out. The cost and functionality analysis of this protocol with the other related protocols has been presented.

The **Chapter-5** presents improvements to different static identity based smart card

authentication protocols. The security analysis of these protocols is carried out. The cost and functionality analysis of these protocols with the other related protocols has been presented.

The **Chapter-6** presents improvements of different smart card based authentication protocols as different dynamic identity based smart card authentication protocols. The security analysis of these protocols is carried out. The cost and functionality analysis of these protocols with the other related protocols has been presented.

The **Chapter-7** presents the dynamic identity based authentication protocols for multi-server architecture. The security analysis of these protocols is carried out. The cost and functionality analysis of these protocols with the other related protocols has been presented.

The **Chapter-8** highlights the main conclusions and significant contribution of the thesis and states the scope for further research in this area.

## 1.8 CONCLUSION

Corporate network and e-commerce applications require secure and practical remote user authentication solutions [6][149]. Password based authentication protocols are mainly susceptible to dictionary and phishing attacks. Password theft is growing significantly and shaking the confidence of customer in e-commerce. In this chapter, we analyzed currently available password authentication schemes over insecure communication channels. Most of these schemes do not fulfill security requirements and can not resist in different attack scenarios. Cookies are good means to provide weak authentication. SSO authentication is time efficient because it allows the user to enter his identity and password once within specific time period to login on to multiple hosts and applications within an organization. The concept of two-tier authentication for the client makes it difficult for an attacker to guess out the information pertaining to password and ticket. Smart card based password authentication is one of the most convenient ways to provide multi-factor authentication for the communication between a client and a server. User's privacy is an important issue in e-commerce applications. Dynamic identity based authentication schemes aim to provide the privacy to the user's identity so that users are anonymous in communication channel. Transaction authorization method based on out of band channels like SMS messages was introduced by banks to thwart dictionary and phishing attacks but it requires



two separate communication channels [5]. The concept of virtual password authentication protocol changes the password in each login attempt corresponding to the same client. In future, more computation and communication efficient password authentication schemes should be developed which can resist different attacks in a better way. In this chapter, a brief review of the literature on the research topic has been carried out. The scope of the research work has been outlined and the organization of the thesis has been summarized.

# DYNAMIC IDENTITY BASED SINGLE PASSWORD ANTI-PHISHING PROTOCOL

---

## 2.1 INTRODUCTION

Password is the most commonly used technique to authenticate the users on the web. Short and easily memorable passwords are susceptible to attacks on insecure communication channels like the Internet. On the other hand, the users find it difficult to remember long and complex passwords. A common practice adopted by the users is to choose a single strong password and use it for multiple accounts, instead of choosing a unique password for each account [45]. The attacker can learn the password of a user from a less secure site and reuse it to compromise a secure site. The password based authentication schemes are vulnerable to phishing, dictionary, man-in-the-middle and insider attacks. Hacking and identity thefts are the two main concerns in password based authentication protocols.

Phishing is an online identity theft that combines social engineering and website spoofing techniques to cheat the user by redirecting his confidential information to an untrusted destination. The attacker can use this information in online transactions to make an illegal economic profit. In a phishing attack, the attacker sends a large number of spoofed e-mails to random Internet users that appear to be coming from a legitimate business organization such as a bank. The e-mail requests the recipient to update his personal information and also warns that failure to reply the request will result in closure of his online banking account. The victim follows the phishing link provided in the e-mail and is directed to a website that is under the control of the attacker. The average user can not distinguish a well designed phishing website from the legitimate site because the phishing site is prepared in a manner that imitates visual characteristics of the target organization's website by using similar colors, icons, logos and textual descriptions [7]. Password based authentication is highly susceptible to phishing attacks by exploiting the visual resemblance of domain names to allure the victims (e.g. [www.paypai.com](http://www.paypai.com) instead of actual [www.paypal.com](http://www.paypal.com)). Phishing attacks are increasing despite the use of preventive measures like e-mail filters and content analysis. The effectiveness of these anti-spam

techniques depends upon many critical factors such as regular filter training. There is still a possibility that some of the phishing e-mails manage to get through the filters and reach the potential victims. The phishing attacks are becoming more and more sophisticated and therefore require strong countermeasures. It is important to detect the phishing sites early because most of them are short-lived and cause the damage in the short time span between appearing online and vanishing. Phishing is doing direct damage to the financial industry and is also affecting the expansion of e-commerce.

One of the solutions to counter phishing is to render the browsers with security indicators such as use of https in URL bar, locked icon, public key certificate and security warnings. The main reason for the success of phishing attacks is that average users do not constantly notice the presence of security indicators and do not know how to interpret them. A solution is required in which the user does not have the need of interpreting the browser based security indicators. The protocol proposed in this chapter is very effective and suitable to the security ignorant or naive users because it does not require checking of security indicators by the user to thwart phishing and dictionary attacks.

## 2.2 REVIEW OF GOUDA ET AL.’S PROTOCOL

In this section, we describe the Anti-phishing Single Password Protocol (SPP) for HTTP authentication proposed by Gouda et al. [42]. The notations used in this section are listed in Table 2.1 and the protocol is shown in Figure 2.1.

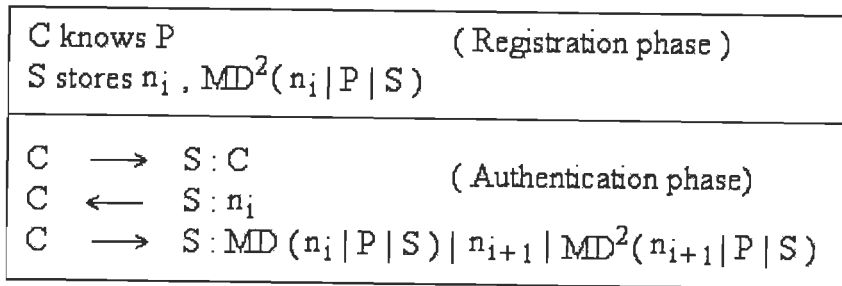
**Table 2.1**  
**Notations**

C	Client identity
S	Server identity
P	Password remembered by client
$n_i$	Random number
MD ( )	One-way hash function
MD <sup>2</sup> ( )	MD ( MD ( ) )
	Concatenation

In this protocol, the client C uses the same password P to register on different web servers. The client C chooses a random challenge  $n_i$  and stores  $n_i$  and the ticket verification information MD<sup>2</sup> ( $n_i | P | S$ ) in the password file of the server S. The client uses the

one-time ticket to authenticate itself to the server as shown in the last step of Figure 2.1. Therefore, the response from a client in each login attempt must be unique. To achieve this uniqueness, random number  $n_i$  involved in the protocol is changed by the client in each run of the protocol.

Whenever the client  $C$  wants to login on to the server  $S$ , the client  $C$  sends his user name  $C$  to the server  $S$  and the server  $S$  sends the challenge  $n_i$  to the client  $C$ . The client  $C$  computes one-time ticket  $MD(n_i | P | S)$ , chooses a new random challenge  $n_{i+1}$  and computes new ticket verification information  $MD^2(n_{i+1} | P | S)$ . Then the client  $C$  sends all of them  $\{MD(n_i | P | S) | n_{i+1} | MD^2(n_{i+1} | P | S)\}$  to the server  $S$ . After receiving this message, the server  $S$  first verifies the received one-time ticket  $MD(n_i | P | S)$  with the stored ticket verification information  $MD^2(n_i | P | S)$ . If the one-time ticket is verified, the client is successfully authenticated to the server. Afterwards, the server  $S$  replaces the stored values of  $n_i$  and  $MD^2(n_i | P | S)$  by  $n_{i+1}$  and  $MD^2(n_{i+1} | P | S)$  in its password file.



**Figure 2.1: Gouda et al.'s protocol**

In this protocol, the client can use the same user name and password on multiple web servers. Gouda et al. claimed that the attacker can not derive the password from the ticket verification information stored on the server. Hence the attacker can not impersonate as a legitimate client to the web servers providing online transaction services. It is assumed that the SPP is running on the top of the SSL protocol [34]. In the SSL protocol, the server authenticates itself to the client using the public key certificate of the server. Then the client generates SSL session key, encrypts it with the public key of the server and sends it to the server. The server decrypts the SSL session key using its private key. Then all the subsequent messages between the client and the server in the SPP are encrypted with the SSL session key.

## 2.3 CRYPTANALYSIS OF GOUDA ET AL.'S PROTOCOL

It is claimed in [42] that SPP resists various known attacks. However, this protocol is found to be flawed as it is susceptible to offline dictionary attack and denial of service attack in the presence of a malicious server. This protocol is also found to be susceptible to man-in-the-middle attack.

### 2.3.1 Offline dictionary attack

The malicious server  $S'$  sends the challenge  $n_i'$  to the client  $C$  and the client  $C$  sends the ticket verification information  $\{MD(n_i' | P | S') | n_{i'+1} | MD^2(n_{i'+1} | P | S')\}$  to the malicious server  $S'$ . Then the malicious server  $S'$  records the received value of  $MD(n_i' | P | S')$ . Now the malicious server  $S'$  knows the nonce value  $n_i'$ , its own identity  $S'$  and hence can launch offline dictionary attack on the recorded message  $MD(n_i' | P | S')$  to know the password  $P$  of the client  $C$ . Once the password is guessed, it can be verified on the subsequent message  $MD(n_{i'+1} | P | S')$  sent by the client in next run of the protocol. Thus the malicious server  $S'$  can impersonate as the client  $C$  to other legitimate servers after getting the password  $P$  of the client  $C$ .

### 2.3.2 Denial of service attack

After getting the password  $P$  of the legitimate client  $C$ , the malicious server  $S'$  can impersonate as client  $C$ . The malicious server  $S'$  can frame message  $\{MD(n_m | P | S) | n_k | MD^2(n_{m+1} | P | S)\}$  in response to challenge  $n_m$  imposed by the legitimate server  $S$  and sends it to server  $S$ . When the legitimate server  $S$  receives this message, it first verifies the received one-time ticket  $MD(n_m | P | S)$  using the stored ticket verification information  $MD^2(n_m | P | S)$ . This one-time ticket is verified by the legitimate server  $S$  and afterwards it replaces the stored value of  $n_m$  and  $MD^2(n_m | P | S)$  in its password file by  $n_k$  and  $MD^2(n_{m+1} | P | S)$ . In the next login session, the legitimate server  $S$  will present  $n_k$  as a challenge to the legitimate client  $C$ . Since the client  $C$  has not stored the value of  $n_m$ , hence it can not distinguish  $n_k$  from  $n_m$ . Therefore the client  $C$  sends the ticket verification information  $\{MD(n_k | P | S) | n_{k+1} | MD^2(n_{k+1} | P | S)\}$  to the server  $S$ . When the server  $S$  receives this message, it tries to verify the received one-time ticket  $MD(n_k | P | S)$  using the stored ticket verification information  $MD^2(n_{m+1} | P | S)$ . This one-time ticket is not verified by the server  $S$ . Thus authentication will always fail in the current and any

subsequent login attempts by the client C. In this way, the denial of service attack can be launched on the client C by a malicious server.

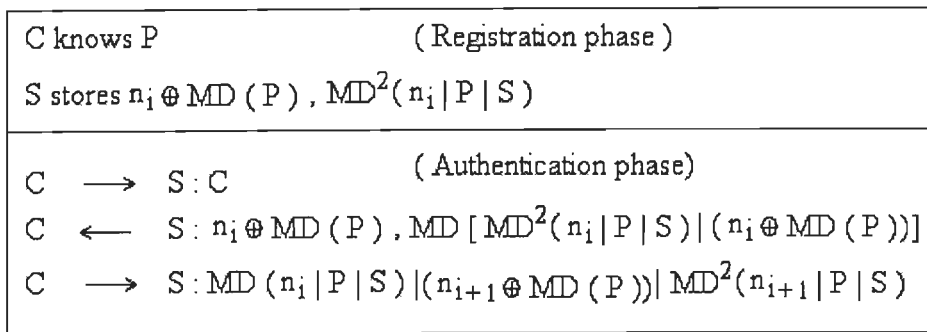
### 2.3.3 Man-in-the-middle attack

Man-in-the-middle attack on the SPP is possible if the attacker carefully overlays a rogue window on top of the trusted or authenticated browser window. The attacker can gather the information  $\{MD(n_i | P | S) | n_{i+1} | MD^2(n_{i+1} | P | S)\}$  sent by the client C to the server S, which the client C has submitted to the rogue window. Similarly, the attacker can gather the information  $\{MD(n_{i+1} | P | S) | n_{i+2} | MD^2(n_{i+2} | P | S)\}$  submitted by the client C for the server S on to the rogue window in the next login attempt. Then the attacker changes this login request information  $\{MD(n_{i+1} | P | S) | n_{i+2} | MD^2(n_{i+2} | P | S)\}$  with the fabricated login request information  $\{MD(n_{i+1} | P | S) | n_{i+1} | MD^2(n_{i+1} | P | S)\}$  and sends it to the server S. The server S verifies the received ticket  $MD(n_{i+1} | P | S)$  with the stored ticket verification information  $MD^2(n_{i+1} | P | S)$  and replaces the stored values of  $n_{i+1}$  and  $MD^2(n_{i+1} | P | S)$  by the same values  $n_{i+1}$  and  $MD^2(n_{i+1} | P | S)$  in its password file. Then the attacker can establish a new SSL connection with the server S because the SSL session key SS is to be generated by the client (in this case the attacker). The server S sends the challenge  $n_{i+1}$  to the attacker and the attacker sends the ticket verification information  $\{MD(n_{i+1} | P | S) | n_{i+1} | MD^2(n_{i+1} | P | S)\}$  to the server S. The server S verifies the received one-time ticket  $MD(n_{i+1} | P | S)$  with the stored ticket verification information  $MD^2(n_{i+1} | P | S)$  and replaces the stored values of  $n_{i+1}$  and  $MD^2(n_{i+1} | P | S)$  by the same values  $n_{i+1}$  and  $MD^2(n_{i+1} | P | S)$  in its password file. Hence the attacker can login on to the server S by masquerading as the legitimate client. The legitimate client can also login on to the server S at any time and hence may not be aware that the attacker had login on to the server S by masquerading as the legitimate client. Therefore, the Gouda et al.'s protocol is susceptible to the man-in-the-middle attack.

## 2.4 IMPROVED GOUDA ET AL.'S PROTOCOL

The simplest way to eliminate the malicious server attack on Gouda et al.'s [42] protocol is to change the value of  $n_i$  stored on the server to  $n_i \oplus MD(P)$  and change the server's response to the client from  $n_i$  to  $\{n_i \oplus MD(P), MD[MD^2(n_i | P | S) | (n_i \oplus MD(P))]\}$ . Then change the client C's response to the server S from  $\{MD(n_i | P | S) | n_{i+1} | MD^2(n_{i+1} |$

$P | S\}$  to  $\{MD(n_i | P | S) | (n_{i+1} \oplus MD(P)) | MD^2(n_{i+1} | P | S)\}$  as shown in Figure 2.2. This improved protocol also runs on the top of SSL protocol [34]. The server authenticates itself to the client using the public key certificate of the server. Then the client generates, encrypts the SSL session key with the public key of the server and sends it to the server. The server decrypts the SSL session key using its private key. Then all the subsequent messages between the client and the server in the improved SPP are encrypted with the SSL session key.



**Figure 2.2: Improved Gouda et al.'s protocol**

The client C does not have to remember the nonce value  $n_i$  because the client can recover the nonce value  $n_i$  from the received message  $n_i \oplus MD(P)$ . The confirmation that the malicious server has not changed the value  $n_i \oplus MD(P)$  can be made by the client C after verifying the received message  $MD [ MD^2(n_i | P | S) | (n_i \oplus MD(P)) ]$ . Now the malicious server has to guess the password P of the client and the nonce value  $n_i$  correctly at the same time. There are many combinations of  $n_i$  having XOR operation with MD(P) that give the required value as  $n_i \oplus MD(P)$ . Moreover, the nonce value  $n_i$  is changed in every run of the protocol that makes guessing more difficult for the malicious server. Hence the malicious server can not launch offline dictionary attack and denial of service attack on the password verifier information of the client C. Man-in-the-middle attack is still possible on the above improved version of the protocol and the Gouda et al.'s protocol is not repairable for this attack. A new single password based protocol is proposed in section 3.5, which is free from these attacks.

## 2.5 PROPOSED SINGLE PASSWORD ANTI-PHISHING PROTOCOL

Microsoft's Passport initiative (Window Live ID) [100] is a cookie based password management system. This service authenticates the user to different websites that are under the control of this centralized system. The main limitations of this approach are that the users have to trust the centralized server and it requires web administration changes on those sites that use this system for its authentication [68].

**Table 2.2**  
**Notations**

C	Client
S	Server
$P_i$	Password remembered by client
$ID_i$	Identity of client
$N_i, N_k$	Random numbers generated by client
$N_s$	Random number generated by server
$H()$	One-way hash function
OTP	One time password of server for each client
PK	Public key of server
SK	Private key of server
SS	Session key of SSL connection
URL	Destination web site
CK	Cookie information available to client
EXP_TIME	Expiration time of cookie
$\oplus$	XOR operation
	Concatenation

In this section, we propose a cookie based new single password anti-phishing protocol that is free from all the attacks considered above. The proposed protocol provides solution for online password management without significantly changing the user's behavior. One of the main reasons for phishing attacks is that the users do not consistently notice the presence of security indicators on the web browser. Sometimes, the users do not know how to interpret these browser based security indicators. The proposed protocol provides single password solution for different online accounts on the web in which the client has no need to check the browser based security indicators. The client machine's browser generates the dynamic identity and dynamic password verifier information using different security indicators and other user specific parameters of the domain name for



each login request to the server. The client can choose single and easily memorable password of his choice for all online accounts on the different web servers. The notations used in this section are listed in Table 2.2. This scheme consists of four phases (i.e. registration, login, authentication and password change) as summarized in Figure 2.3.

### 2.5.1 Registration phase

The client  $C$  chooses a random nonce value  $N_i$ , computes  $N_i \oplus H(P_i)$  and submits his identity  $ID_i$  and password verifier information  $N_i \oplus H(P_i)$  to the server  $S$  for its registration over a secure communication channel. On the other hand, the server  $S$  computes  $N_i \oplus H(P_i) \oplus H(SK | ID_i)$  and stores it with the client's identity  $ID_i$  in its database. Then the server  $S$  computes cookie information  $CK = H((N_i \oplus H(P_i)) | N_s | EXP\_TIME)$ , where  $N_s$  is a random value chosen by the server  $S$  and  $EXP\_TIME$  is the expiration time of the cookie. The server  $S$  chooses the value of  $N_s$  in such a way so that the value of  $CK$  must be unique for each client. Afterwards, the server  $S$  stores cookie  $CK$  and  $EXP\_TIME$  with the client's identity  $ID_i$  in its database and also stores cookie  $CK$  and  $EXP\_TIME$  on the client's machine. Once the cookie is expired, then the server  $S$  chooses a new random value  $N_s'$  and computes fresh cookie  $CK^{new} = H((N_i \oplus H(P_i)) | N_s' | EXP\_TIME')$  for the same client.

### 2.5.2 Login phase

The client  $C$  establishes a connection with the server  $S$  using the SSL protocol so that the server  $S$  authenticates itself to the client  $C$  with its public key certificate. Then the client  $C$  generates a new SSL session key ( $SS$ ), encrypts it using the public key  $PK$  of the server  $S$  as  $(SS)_{PK}$  and sends it to the server  $S$ . The server  $S$  decrypts the SSL session key  $SS$  from  $(SS)_{PK}$  using its private key  $SK$ . Then all the subsequent messages of this protocol are encrypted with the SSL session key ( $SS$ ).

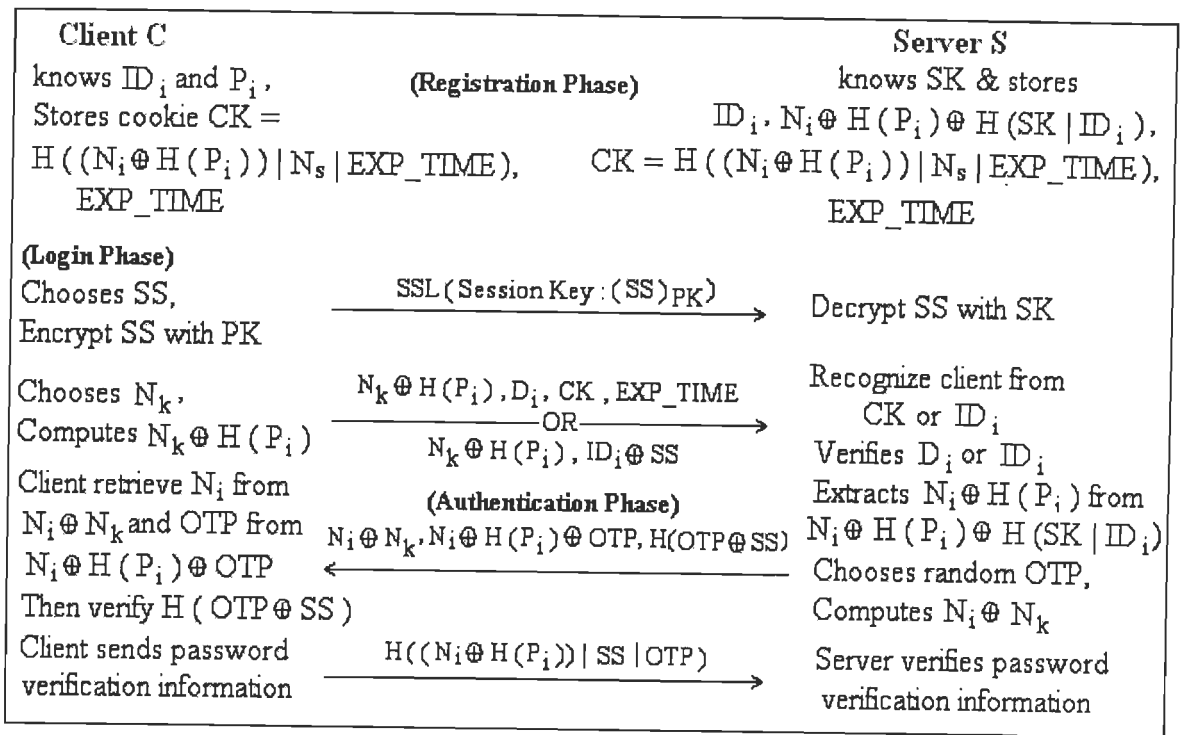
#### Case 1:

If the client's machine contains cookie  $CK$  then the client  $C$  chooses random nonce value  $N_k$ , computes  $N_k \oplus H(P_i)$  and dynamic identity  $D_i = H(ID_i | URL | PK | SS | CK)$ . The client  $C$  submits  $N_k \oplus H(P_i)$ ,  $D_i$ ,  $CK$  and  $EXP\_TIME$  to the server  $S$ . The server  $S$  recognizes the client  $C$  from the received cookie  $CK$  and verifies the received  $EXP\_TIME$  with extracted value of  $EXP\_TIME$  corresponding to cookie  $CK$  from its database to

check the validity of the cookie. The server S uses  $ID_i$ , URL, PK, SS and CK to compute the dynamic identity  $D_i' = H (ID_i | URL | PK | SS | CK)$  and verifies it with the received value of  $D_i$ . If both values are equal, the server S proceeds to the next step. Otherwise, the login request from the client C is rejected.

**Case 2:**

If the client’s machine does not contain cookie CK or cookie is expired then the client C chooses random nonce value  $N_k$ , computes  $N_k \oplus H (P_i)$  and  $ID_i \oplus SS$ . The client C submits  $N_k \oplus H (P_i)$ ,  $ID_i \oplus SS$  to the server S. Then the server S extracts  $ID_i$  from  $ID_i \oplus SS$  and recognizes the client C from its identity  $ID_i$ .



**Figure 2.3: Dynamic identity based single password anti-phishing protocol**

**2.5.3 Authentication phase**

The server S extracts  $N_i \oplus H (P_i)$  from  $N_i \oplus H (P_i) \oplus H (SK | ID_i)$  because the server S knows the values of SK and  $ID_i$ . Then the server S chooses a random value of OTP and computes  $N_i \oplus N_k = N_i \oplus H (P_i) \oplus N_k \oplus H (P_i)$ ,  $N_i \oplus H (P_i) \oplus OTP$ ,  $H (OTP \oplus SS)$  and sends all of them to the client C. The client C extracts  $N_i$  from  $N_i \oplus N_k$  because the client C knows the random nonce value  $N_k$ . The client C retrieves OTP from  $N_i \oplus H (P_i) \oplus OTP$

because now the client  $C$  knows the password  $P_i$  and the nonce value  $N_i$ . Afterwards, the client  $C$  verifies the retrieved OTP value with the received message  $H(OTP \oplus SS)$  to validate that the messages are sent by the legitimate server  $S$  and not tampered during transmission. Once it is verified, the client  $C$  sends  $H((N_i \oplus H(P_i)) | SS | OTP)$  as dynamic password verifier information to the server  $S$ . Then the server  $S$  computes  $H((N_i \oplus H(P_i)) | SS | OTP)$  and verifies it with the received value of  $H((N_i \oplus H(P_i)) | SS | OTP)$ . This equivalency authenticates the legitimacy of the client  $C$  and the server  $S$  and the login request is accepted else the connection is interrupted. Hence the mutual authentication between the client and the server is achieved as shown in Figure 2.3. If the client's machine does not contain the cookie or cookie is expired, then the server stores the cookie  $CK$  on the client's machine once the client and the server authenticate each other using this protocol. The dynamic identity and dynamic password information in the proposed protocol avoids man-in-the-middle, replay and other such kinds of attacks.

#### 2.5.4 Password change phase

The client  $C$  has to authenticate itself to all web servers on which the client has online accounts using the protocol shown in Figure 2.3. Once the mutual authentication between the client and the server is achieved, the client chooses a new password  $P_i^{new}$  and a new random nonce value  $N_i^{new}$ . Afterwards, the client  $C$  computes  $N_i^{new} \oplus H(P_i^{new})$  and sends it to the server  $S$  through a secure communication channel. The server  $S$  updates the password verifier information  $N_i \oplus H(P_i) \oplus H(SK | ID_i)$  with  $N_i^{new} \oplus H(P_i^{new}) \oplus H(SK | ID_i)$  in its database for the client's identity  $ID_i$ . The client  $C$  repeats the same procedure to change its password for all online accounts on the other web servers.

## 2.6 SECURITY ANALYSIS

Security of messages in online transaction inside the communication channel is managed with SSL protocol. Browsers are rendered with security indicators such as https in URL bar containing the target domain name and a closed lock in status bar to indicate the use of SSL protocol to provide trustworthy interface for the authenticity of the website. The user can click on the closed lock to check the public key certificate of the server. A good password authentication scheme should provide protection from man-in-the-middle,

eavesdropping, replay, malicious server, password guessing, denial of service, phishing, pharming, stolen verifier attacks and should achieve mutual authentication. Man-in-the-middle (MITM) attacks pose a serious threat to SSL/TLS based e-commerce applications [111]. Brute force attack on the session key of SSL protocol is also possible [34]. In the following, we analyze the security of our protocol against various types of attacks.

1. **Man-in-the-middle attack:** Man-in-the-middle attack on the proposed protocol may be possible if the attacker carefully overlays a rogue window on top of a trusted or authenticated browser window. The attacker can gather login information  $N_k \oplus H(P_i)$ ,  $D_i$ ,  $CK$ ,  $EXP\_TIME$  or  $N_k \oplus H(P_i)$ ,  $ID_i \oplus SS$  and dynamic password  $H((N_i \oplus H(P_i)) \mid SS \mid OTP)$  by carefully overlaying the rogue window on top of the authenticated browser window. Then the attacker can establish a new SSL connection with the server  $S$  because the SSL session key  $SS$  is to be generated by the client (in this case the attacker). The attacker can not replay login information  $N_k \oplus H(P_i)$ ,  $D_i$ ,  $CK$ ,  $EXP\_TIME$  or  $N_k \oplus H(P_i)$ ,  $ID_i \oplus SS$  to the server  $S$  because SSL session key  $SS$  is different in each new SSL session. Moreover, the server  $S$  chooses new random value of  $OTP$  and sends  $N_i \oplus N_k$ ,  $N_i \oplus H(P_i) \oplus OTP$  and  $H(OTP \oplus SS)$  to the attacker. Now the attacker can not compute the dynamic password  $H((N_i \oplus H(P_i)) \mid SS \mid OTP)$  because it depends upon the new value of one-time password  $OTP$ , sent by the server. The attacker requires to know  $N_i \oplus H(P_i)$ ,  $SS$  and  $OTP$  to generate dynamic password. Therefore, the proposed protocol is secure against man-in-the-middle attack.
2. **Eavesdropping attack:** In this type of attack, the attacker first listens to all the communication between the client and the server and then tries to find out the client's password  $P_i$ . The client's browser uses SSL session key  $SS$  for the generation of dynamic identity  $H(ID_i \mid URL \mid PK \mid SS \mid CK)$  or  $ID_i \oplus SS$  of the client, which will be different for each new SSL session. Also, the eavesdropper can not compute the client's password  $P_i$  from  $N_i \oplus H(P_i) \oplus OTP$  and  $H((N_i \oplus H(P_i)) \mid SS \mid OTP)$ . Moreover, all the communication between the client and the server is encrypted with a SSL session key. Therefore, the proposed protocol is secure against eavesdropping attack.

3. **Replay attack:** In this type of attack, the attacker first listens to the communication between the client and the server, then tries to imitate the user to login on to the server by resending the captured messages. Replaying a message of one SSL session into another SSL session is useless because each SSL session generates a different dynamic identity  $H(\text{ID}_i \mid \text{URL} \mid \text{PK} \mid \text{SS} \mid \text{CK})$  or  $\text{ID}_i \oplus \text{SS}$  for the same client because the session key  $\text{SS}$  is different for each new SSL session. Also, the messages  $\text{N}_i \oplus H(\text{P}_i) \oplus \text{OTP}$ ,  $H(\text{OTP} \oplus \text{SS})$  and  $H((\text{N}_i \oplus H(\text{P}_i)) \mid \text{SS} \mid \text{OTP})$  are session key or OTP dependent and hence can not be replayed successfully in any other SSL session. The messages are also encrypted with different SSL session keys in the different runs of the protocol. Therefore, the proposed protocol is secure against message replay attack.
4. **Malicious server attack:** The attacker sets up a malicious server and entices the users to register with the server so that the password of the client can be compromised. In the proposed protocol, the server can not compute the client's password  $\text{P}_i$  from the stored password verifier information  $\text{N}_i \oplus H(\text{P}_i) \oplus H(\text{SK} \mid \text{ID}_i)$  or the received dynamic password  $H((\text{N}_i \oplus H(\text{P}_i)) \mid \text{SS} \mid \text{OTP})$ . Therefore, the proposed protocol is secure against malicious server attack. In Gouda et al.'s protocol, a malicious server  $S$  can launch offline dictionary attack on the message  $\text{MD}(n_i \mid \text{P} \mid \text{S})$  to know the password  $\text{P}$  of the client as discussed in section 2.3.1.
5. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess the user's password from the recorded messages. The attacker obtains some password verification information such as  $\text{N}_i \oplus H(\text{P}_i) \oplus H(\text{SK} \mid \text{ID}_i)$ ,  $\text{N}_i \oplus H(\text{P}_i) \oplus \text{OTP}$  and  $H((\text{N}_i \oplus H(\text{P}_i)) \mid \text{SS} \mid \text{OTP})$ . The attacker can not compute  $\text{P}_i$  from these recorded messages. Moreover, there are many combination of  $H(\text{P}_i)$  having XOR operation with  $\text{N}_i$  that give the required value as  $\text{N}_i \oplus H(\text{P}_i)$ . Therefore, the proposed protocol is secure against offline dictionary attack. In Gouda et al.'s protocol, a malicious server  $S$  knows the nonce value  $n_i$ , its own identity  $S$  and has to guess only the password  $\text{P}$  to launch offline dictionary attack on the message  $\text{MD}(n_i \mid \text{P} \mid \text{S})$  to know the password  $\text{P}$  of the client.
6. **Denial of service attack:** In a specific type of denial of service attack, the server is cheated by the attacker to update the password verifier information with some false password verification information so that the legitimate user can not login successfully in subsequent login request to the server. The proposed protocol does not update the

password verifier information  $N_i \oplus H(P_i) \oplus H(SK | ID_i)$  on the server after each successful login attempt by the client C. Moreover, the client C can change his password after the client and the server authenticate each other using the protocol shown in Figure 2.3. Therefore, the proposed protocol is secure against the user specific denial of service attack. In Gouda et al.'s protocol, if the malicious server S' login successfully on the other legitimate server S by masquerading as the legitimate client then the malicious server S' can change the password verifier information to some random value in the database of server S so that the legitimate client can not login on to the server S in subsequent login attempt.

7. **Phishing attack:** In this type of attack, the attacker sends spoofed e-mails to different users from a website that is under the control of the attacker. Victim enters his valid login credentials into the fraudulent website that allows the attacker to transfer funds from the victim's account or cause other damages. The proposed protocol generates a new dynamic identity  $H(ID_i | URL | PK | SS | CK)$  or  $ID_i \oplus SS$  for the client in each new SSL session. The fraudulent server can ignore dynamic identity but can not produce valid credentials  $N_i \oplus N_k$ ,  $N_i \oplus H(P_i) \oplus OTP$  and  $H(OTP \oplus SS)$  meant for the client, because it does not have any such credentials. Therefore, the proposed protocol is secure against phishing attack.
8. **Pharming attack:** Pharming is a technique that fools the user by connecting his machine to a fake website even when the user submits correct domain name in to the web browser. This technique exploits vulnerabilities in the DNS servers to distribute the fake address information by DNS spoofing attack. Like phishing attacks, the attacker sets up a capture site to collect identity and password verifier information. The attacker can cause the DNS caching server to return false information and direct the user to a malicious site. Malicious site can not impersonate as valid server because it can not generate valid credentials  $N_i \oplus N_k$ ,  $N_i \oplus H(P_i) \oplus OTP$  and  $H(OTP \oplus SS)$  meant for the client, which are unique for each session. Therefore, the attacker can not launch pharming attack on the proposed protocol.
9. **Online dictionary attack:** In this type of attack, the attacker pretends to be legitimate client and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to generate dynamic identity  $H(ID_i | URL | PK | SS | CK)$  or  $ID_i \oplus SS$  corresponding to that client, which is

different for each new SSL session. Moreover, the attacker has to know the values of  $N_i$ ,  $P_i$ ,  $SS$  and  $OTP$  to generate the dynamic password  $H((N_i \oplus H(P_i)) | SS | OTP)$ . Therefore, the attacker can not launch online dictionary attack on the proposed protocol.

**10. Leak of verifier attack:** In this type of attack, the attacker may be able to steal verification table from the server. In case the password verifier information  $N_i \oplus H(P_i) \oplus H(SK | ID_i)$  is stolen by breaking into the server's database, the attacker does not have sufficient information to calculate the user's password  $P_i$  because the attacker has to guess  $N_i$ ,  $P_i$ ,  $SK$  and  $ID_i$  correctly. Therefore, the proposed protocol is secure against leak of verifier attack.

**11. Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the client's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can result in authentication failure between the client and the server but can not allow the attacker to gain any information about the client's password or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

**12. Brute force attack:** To launch brute force attack, the attacker first obtains some password verification information such as  $N_i \oplus H(P_i) \oplus H(SK | ID_i)$ ,  $N_i \oplus H(P_i) \oplus OTP$  and  $H((N_i \oplus H(P_i)) | SS | OTP)$ . Even after recording these messages, it is not possible to find the password  $P_i$  using the brute force attack. Moreover, there are many combination of  $H(P_i)$  having XOR operation with  $N_i$  that gives the required value as  $H(P) \oplus N_i$ . Therefore, the proposed protocol is secure against brute force attack.

## 2.7 COST AND FUNCTIONALTY ANALYSIS

The proposed protocol is practical and efficient because only one-way hash functions and XOR operations are used in its implementation. The cost and functionality comparison of the proposed protocol with other recent anti-phishing protocols is summarized in Table 2.3.

Implementation of the proposed protocol requires installation of a software patch on the client's browser. If this software patch can be made permanent feature of the web

browser then there is no need to install software patch separately. Many users have trouble in interpreting browser based security indicators and clues such as URL bar, locked icon, public key certificate and security warnings. Our proposed protocol is very effective and suitable to security ignorant or naive users because it does not require the checking of security indicators by the client to thwart phishing and dictionary attacks. The proposed protocol is highly secure as compared to the other anti-phishing protocols.

**Table 2.3**  
**Cost and functionality comparison among different anti-phishing protocols**

Web based password protocols	Need of checking browser indicators	Need of checking URL	Need of checking GUI	Need of installing additional software	Security ignorant users	Dictionary attack
SSL [34]	Y	N	N	N	N	Y
Digest Access [119]	Y	Y	N	N	N	Y
PwdHash [120]	Y	N	N	Y	N	Y
SRD [165]	Y	N	Y	Y	N	Y
DSS [29]	Y	Y	Y	Y	N	Y
SpoofGuard [134]	Y	Y	N	Y	N	Y
LSOP [63]	Y	Y	Y	Y	N	Y
Cache cookies [62]	Y	N	Y	N	N	Y
SPP [42]	Y	Y	N	Y	N	Y
Proposed Protocol	N	N	N	Y	Y	N

\* Y means Yes and N means No

## 2.8 CONCLUSION

Instances of phishing attacks are rapidly growing in number. This is sufficient to shake the confidence of the customers in e-commerce. Naive users find it difficult to understand the security indicators of the web browser. Authenticating the user on the web is an essential primitive and is target of various attacks. In this chapter, we presented a cryptanalysis of Gouda et al.'s protocol and showed that their protocol is vulnerable to offline dictionary attack, denial of service attack and man-in-the-middle attack in the presence of an active attacker. Even the improved Gouda et al.'s protocol can not resist man-in-the-middle attack. We have specified and analyzed a dynamic identity and a dynamic password based single password anti-phishing authentication protocol. The client can use a single



password for different online accounts and that password can not be detected by any of the malicious servers or the attacker. The protocol is equally secure for security ignorant users, who are not very conversant with the browser's security indicators. The presented protocol is a step towards bridging the gap between skilled and unskilled users in online transactions. Our proposed protocol does not allow the server to know the client's password at any time. The proposed protocol will be helpful to the naive users in detecting the phishing website quickly. This protocol can be easily integrated into different types of services such as banking and enterprise applications.

**COOKIE AND INVERSE COOKIE BASED VIRTUAL  
PASSWORD AUTHENTICATION PROTOCOLS**

---

**3.1 INTRODUCTION**

Hyper Text Transfer Protocol (HTTP) that provides interaction between the web browser and the web server is stateless because the HTTP server treats each request independent of any previous request from the same client. The HTTP server does not maintain the correlation of the user visits from the same browser between successive sessions. The users are always strange to the web server if the web server does not maintain the state and continuity of the user [112]. Statelessness on the web makes it difficult to carry out online financial transactions in e-commerce. The merchant web server can not remember users on the web server without a state mechanism. Therefore, the web server uses cookies to maintain the state and connection of the user with the web server. Cookie technology is the most innovative feature that made the web stateful. A number of the web applications built on the top of HTTP need to be stateful and require cookies to maintain the user's state.

The web server creates a cookie that contains the state information of a client and stores it on the client computer from where the request is originated. The web server uses cookies to authenticate HTTP requests from the same client and to maintain persistent client state. Cookie enabled server can maintain information related to the client that can be used by the server during subsequent login requests from the same client. The client's browser attaches the cookie with each subsequent request made by the client to the same web server. The web server retrieves the user's information from this cookie. The default parameters of HTTP cookie are cookie name, value, expiration date, URL path for which the cookie is valid, domain name and a flag to indicate whether the cookie had been sent using the SSL protocol. Secure cookies are required so that they can not be forged and all of their contents are not readable [90][112]. These secure cookies use different cryptographic primitives such as message digest, message authentication code, digital signature and encryption.

Cookies strengthen the connection between a legitimate client and a genuine web server across the web. It helps the web server to keep track of the user's movement and his behavior on the visited web server. Therefore, a web server can obtain significant information about long term habits of its clients. There is no notification mechanism to alert the users when the cookies are being placed on their computer. The users are not aware of what information about them is being stored in the cookies. Cookies can persist for many years, for example google search engine routinely sets an expiration date in the year 2038 for its cookies. Third party cookies can be used by online business organizations to create detailed records on the user's web browsing habits. Cookies can be used in conjunction with passwords to provide different levels of authentication to users.

Password is the most commonly used authentication technique to authenticate users on the web. Short and easily memorable passwords are susceptible to different types of attacks such as dictionary, phishing, stolen verifier, man-in-the-middle and insider attacks. On the other hand, the users find it difficult to memorize long and complex passwords. The concept of virtual password helps to defend the password authentication protocols from different types of attacks. Virtual password is a dynamic password that will be different for each new session between the same client and the server. The virtual password involves some computation on the client side to generate different passwords corresponding to the same user in different login sessions based on a single password shared between the client and the server [79].

Online dictionary attacks are one of the major concerns in password based authentication protocols. The solutions are required in which it is not possible for the attacker to launch online dictionary attacks on password based authentication protocols. The aim of this chapter is to develop virtual password based authentication solutions using cookies for user authentication. The protocols proposed in this chapter are very effective and suitable for business organizations such as online banks and online credit card organizations.

## **3.2 PROPOSED COOKIE BASED VIRTUAL PASSWORD AUTHENTICATION PROTOCOL**

A HTTP cookie contains information related to the user such as user name, domain name and token for authentication. It is designed and created by the web server and stored on the

user's computer to keep track of the client state. The cookie is transferred back from the client's computer to the web server in succeeding login request by the client. The cookies are server controlled and hence the design and contents of a cookie are decided by the web server without requiring any infrastructural changes on the client side. The web server decides various fields required in the cookie depending upon the information that the web server wants to keep related to their clients.

The proposed protocols provide cookie based virtual password authentication for online password management. An attacker can not launch online dictionary attack because the complexity of computation on the client side increases with each login failure so that it is very difficult for an attacker to impersonate as a legitimate user. The proposed protocols run on top of the SSL protocol [34] and comprise four phases as follows. The notations used in this section are listed in Table 3.1.

**Table 3.1**  
**Notations**

$U_i$	$i$ th user
$S$	Server
$ID_i$	Unique identification of user $U_i$
$P_i$	Password of user $U_i$
URL	Destination web site
OTP	One time password of server for each user
$H()$	One-way hash function
MAX_TRUST	Maximum trust assigned to user $U_i$
MIN_TRUST	Minimum trust assigned to user $U_i$
CUR_TRUST	Current trust value of user $U_i$
SK	Private key of server
PK	Public key of server
SS	Session key of SSL protocol
$(SS)_{PK}$	Session key encrypted with server's public key
$\oplus$	XOR operation
	Concatenation

We present two authentication protocols. Each protocol has four phases. These two protocols have same registration phase and password change phase and they differ in login phase and authentication phase. Protocol 1 makes use of cookies for the user's authentication whereas Protocol 2 does not use cookies. The user  $U_i$  has to follow the Protocol 1 if the user  $U_i$ 's computer contains cookie else the user  $U_i$  has to follow Protocol 2.

### 3.2.1 Protocol 1

This protocol is shown in Figure 3.1 and Figure 3.2 and its various phases are described below.

#### 1. Registration phase

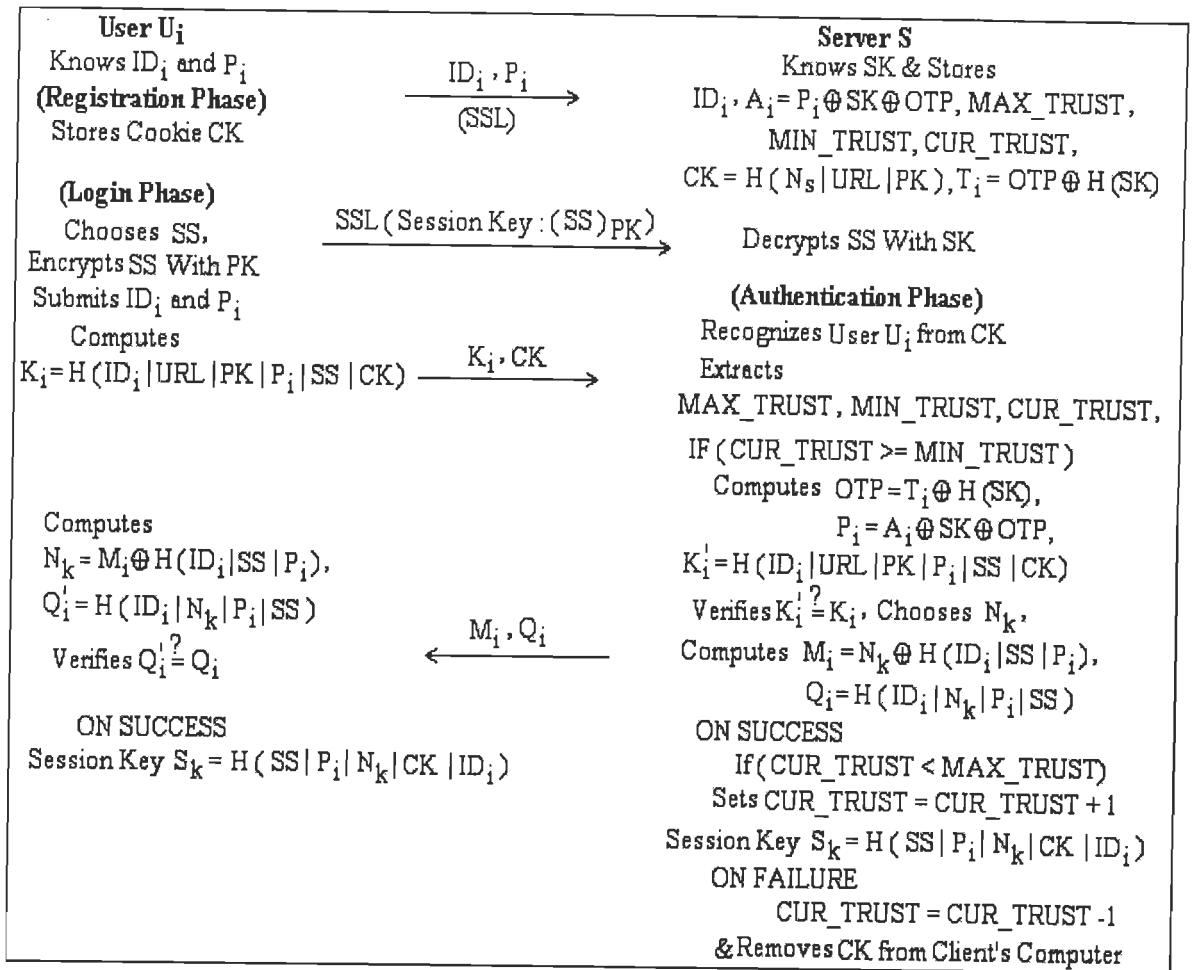
A new user has to register with the web server S to become a legitimate client C. The user  $U_i$  submits its identity  $ID_i$  and password  $P_i$  to the web server S over a secure communication channel established using SSL protocol.

Step 1:  $U_i \rightarrow S: ID_i, P_i$

In this chapter, the concept of trust (MAX\_TRUST, MIN\_TRUST and CUR\_TRUST) has been used and is defined as “to have belief or confidence in the honesty, goodness, skill or safety of the legitimate user”. The web server S chooses a random one time password OTP for each user and stores  $ID_i$ ,  $A_i = P_i \oplus SK \oplus OTP$ , MAX\_TRUST, MIN\_TRUST and CUR\_TRUST in its database. The web server S can assign random trust values to different clients depending upon its trust management policies. The web server S can decide the fixed MAX\_TRUST value that represents the maximum trust, fixed MIN\_TRUST value that represents the minimum trust and variable CUR\_TRUST value that represents the current trust value assigned to the user  $U_i$ . Initially, the web server S sets CUR\_TRUST value equal to MIN\_TRUST value. Suppose the web server S decides MIN\_TRUST to be 0, MAX\_TRUST to be 50 and hence initial CUR\_TRUST value will be 0. The CUR\_TRUST value stored in the database of web server S is incremented by one after each successful login attempt by the user  $U_i$  on the web server S and decremented by one on login failure. Once the CUR\_TRUST value stored on the web server becomes equal to MAX\_TRUST, it is not incremented further even after successful login by the user  $U_i$ . After successive login failures, the CUR\_TRUST value may become less than MIN\_TRUST value.

The web server S chooses a random value  $N_s$ , computes  $CK = H(N_s | URL | PK)$  and  $T_i = OTP \oplus H(SK)$ . The web server S is required to choose the value of  $N_s$  in such a way that the value of CK is unique for each client. The web server S stores CK and  $T_i$  corresponding to the user  $U_i$ 's identity  $ID_i$  in its database and stores CK as cookie information on the client's computer when the user  $U_i$  authenticates itself successfully to the web server S.

Step 2:  $S \rightarrow U_i: CK$



**Figure 3.1: Protocol 1: (Case 1) Virtual password authentication protocol with cookie**

## 2. Login phase

The user  $U_i$  establishes a connection with the web server  $S$  using the SSL protocol. In the SSL protocol, the web server  $S$  authenticates itself to user  $U_i$  with its public key certificate. Then the user  $U_i$  generates a new SSL session key ( $SS$ ), encrypts it using the public key  $PK$  of the web server  $S$  as  $(SS)_{PK}$  and sends it to the web server  $S$ . The web server  $S$  decrypts the SSL session key  $SS$  from  $(SS)_{PK}$  using its private key  $SK$ . Then all the subsequent messages of this protocol are transmitted in insecure communication channel like Internet without using SSL protocol.

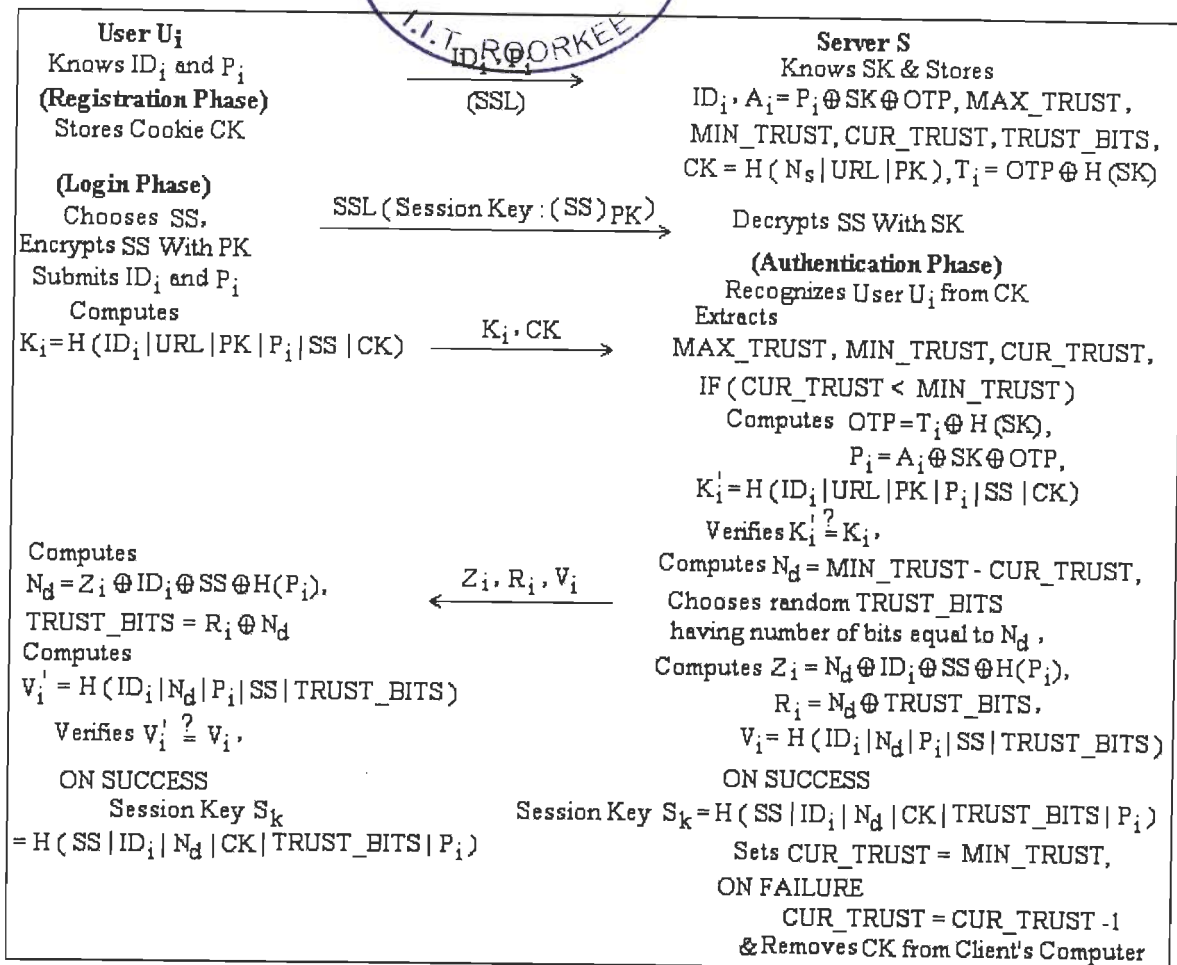
The user  $U_i$  submits his identity  $ID_i$  and password  $P_i$  to the web browser. If the user  $U_i$ 's computer contains cookie  $CK$  then the user  $U_i$ 's web browser computes dynamic identity and password verifier information  $K_i = H(ID_i | URL | PK | P_i | SS | CK)$  and submits  $K_i$  and  $CK$  to the web server  $S$  as shown in Figure 3.1.

### 3. Authentication phase

The web server  $S$  recognizes the user  $U_i$  from the received cookie  $CK$  and extracts  $MAX\_TRUST$ ,  $MIN\_TRUST$  and  $CUR\_TRUST$  corresponding to cookie  $CK$  from its database.

#### Case 1:

If  $CUR\_TRUST$  value is more than or equal to  $MIN\_TRUST$  value then the web server  $S$  computes  $OTP$  as  $OTP = T_i \oplus H(SK)$  because the web server  $S$  knows its private key  $SK$ . Then the web server  $S$  computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and computes the dynamic identity and password verifier information  $K_i' = H(ID_i | URL | PK | P_i | SS | CK)$  and verifies the computed value of  $K_i'$  with the received value of  $K_i$ . If both values are equal, the web server  $S$  proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. Then the web server  $S$  chooses a random value of  $N_k$ , computes  $M_i = N_k \oplus H(ID_i | SS | P_i)$ ,  $Q_i = H(ID_i | N_k | P_i | SS)$  and sends  $M_i$  and  $Q_i$  to the web browser of user  $U_i$ . The web browser of user  $U_i$  computes  $N_k = M_i \oplus H(ID_i | SS | P_i)$ ,  $Q_i' = H(ID_i | N_k | P_i | SS)$  and verifies the computed value of  $Q_i'$  with the received value of  $Q_i$  to validate that the messages are sent by the legitimate web server  $S$  and not tampered during transmission. This equivalency authenticates the legitimacy of the user  $U_i$  and the web server  $S$  and the login request is accepted else the connection is interrupted. Hence the mutual authentication between the user  $U_i$  and the web server  $S$  is achieved as shown in Figure 3.1. Afterwards, the web server  $S$  checks  $CUR\_TRUST$  value in its database corresponding to the user identity  $ID_i$ . If the  $CUR\_TRUST$  value stored in the database of web server  $S$  is less than  $MAX\_TRUST$  value then the  $CUR\_TRUST$  value is incremented by one ( $CUR\_TRUST = CUR\_TRUST + 1$ ) after successful login attempt by the user  $U_i$  on the web server  $S$ . Finally, the user  $U_i$  and the web server  $S$  agree on the common session key as  $S_k = H(SS | P_i | N_k | CK | ID_i)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the web server  $S$  are  $XOR^{ed}$  with the session key. Therefore, either the user  $U_i$  or the web server  $S$  can retrieve the original message because both of them know the common session key. If the user  $U_i$  fails to authenticate itself to the web server  $S$  then the web server  $S$  decreases the  $CUR\_TRUST$  value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ) and the server  $S$  removes the cookie  $CK$  from the client's computer.



**Figure 3.2: Protocol 1: (Case 2) Virtual password authentication protocol with cookie**

**Case 2:**

If CUR\_TRUST value is less than MIN\_TRUST value then the web server S computes OTP as  $OTP = T_i \oplus H(SK)$  because the web server S knows its private key SK. Then the web server S computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and computes the dynamic identity and password verifier information  $K_i' = H(ID_i | URL | PK | P_i | SS | CK)$  and verifies the computed value of  $K_i'$  with the received value of  $K_i$ . If both values are equal, the web server S proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. Then the web server S computes  $N_d = MIN\_TRUST - CUR\_TRUST$  and chooses random TRUST\_BITS value having bits equal to the value of  $N_d$ . Suppose the value of  $N_d$  is 2 then the number of bits in TRUST\_BITS value will be 2. Then the web server S computes  $Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i), R_i = N_d \oplus TRUST\_BITS, V_i = H(ID_i | N_d | P_i | SS | TRUST\_BITS)$  and sends  $Z_i, R_i$  and  $V_i$  to the web browser of user  $U_i$ . The web browser of legitimate user  $U_i$  can compute the value of  $N_d$  as  $N_d = Z_i \oplus ID_i \oplus SS \oplus H(P_i),$



TRUST\_BITS as  $TRUST\_BITS = R_i \oplus N_d$  and  $V_i' = H(ID_i | N_d | P_i | SS | TRUST\_BITS)$  and verifies the computed value of  $V_i'$  with the received value of  $V_i$ . Hence the mutual authentication between the user  $U_i$  and the web server  $S$  is achieved as shown in Figure 3.2. Finally, the user  $U_i$  and the web server  $S$  agree on the common session key as  $S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the web server  $S$  are  $XOR^{ed}$  with the session key. Therefore, either the user  $U_i$  or the web server  $S$  can retrieve the original message because both of them know the common session key. Then the web server  $S$  resets the  $CUR\_TRUST$  value corresponding to user  $U_i$  equal to  $MIN\_TRUST$  value after successful authentication. If the user  $U_i$  fails to authenticate itself to the web server  $S$  then the web server  $S$  decreases the  $CUR\_TRUST$  value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ) and the server  $S$  removes the cookie  $CK$  from the client's computer. The attacker has to guess the values of  $SS$ ,  $ID_i$ ,  $N_d$ ,  $TRUST\_BITS$  and  $P_i$  to compute the common session key as  $S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i)$ .

#### 4. Password change phase

The legitimate user  $U_i$  authenticates itself to the web server  $S$  using the protocol 1 or protocol 2. Once the mutual authentication between the user  $U_i$  and the web server  $S$  is achieved, the user  $U_i$  submits  $Y_i = SS \oplus P_i \oplus P_i^{new}$  and  $X_i = H(ID_i | P_i | SS | P_i^{new})$  to the web server  $S$ . The web server  $S$  retrieves  $P_i^{new}$  from  $Y_i$  as  $P_i^{new} = Y_i \oplus SS \oplus P_i$ , computes  $X_i^* = H(ID_i | P_i | SS | P_i^{new})$  and verifies the computed value of  $X_i^*$  with the received value of  $X_i$  to validate that the messages are sent by the legitimate user  $U_i$  and not tampered during transmission. Afterwards, the web server  $S$  updates the values of  $A_i = P_i \oplus SK \oplus OTP$  and  $T_i = OTP \oplus H(SK)$  stored in its database with  $A_i^{new} = P_i^{new} \oplus SK \oplus OTP^{new}$  and  $T_i^{new} = OTP^{new} \oplus H(SK)$  and the password gets changed.

### 3.2.2 Protocol 2

This protocol is shown in Figure 3.3 and its various phases are described ahead.

#### 1. Registration phase

The registration phase is same as in Protocol 1. (See Section 3.2.1)

## 2. Login phase

The user  $U_i$  agrees on SSL session key  $SS$  with the web server  $S$  using the SSL protocol as shown in login phase of Protocol 1 in Section 3.2.1. Then all the subsequent messages of this protocol are transmitted in the open without using SSL protocol.

The user  $U_i$  submits his identity  $ID_i$  and password  $P_i$  to the web browser. If the user  $U_i$ 's computer does not contain cookie  $CK$  then the web browser chooses random nonce value  $N_r$ , computes  $B_i = N_r \oplus H(P_i)$ ,  $C_i = ID_i \oplus SS$  and  $D_i = H(ID_i | SS | P_i | N_r)$ . The web browser of user  $U_i$  submits  $B_i$ ,  $C_i$  and  $D_i$  to the web server  $S$  as shown in Figure 3.3.

## 3. Authentication phase

The web server  $S$  computes  $ID_i$  from  $C_i$  as  $ID_i = C_i \oplus SS$  and recognizes the user  $U_i$  from its identity  $ID_i$ . After that, the web server  $S$  computes OTP as  $OTP = T_i \oplus H(SK)$  because the web server  $S$  knows its private key  $SK$ . Then the web server  $S$  computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and  $N_r$  from  $B_i$  as  $N_r = B_i \oplus H(P_i)$ . Afterwards, the web server  $S$  computes  $D_i' = H(ID_i | SS | P_i | N_r)$  and verifies it with the received value of  $D_i$ . If both values are equal, the web server  $S$  proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. The web server  $S$  chooses random nonce value  $N_i$  and computes  $E_i = N_i \oplus H(P_i)$ ,  $N_d = | MIN\_TRUST - CUR\_TRUST |$  and chooses random  $TRUST\_BITS$  value having bits equal to the value of  $N_d$ , where  $| MIN\_TRUST - CUR\_TRUST |$  represents modulus or positive value of the difference between  $MIN\_TRUST$  and  $CUR\_TRUST$ . Then the web server  $S$  computes  $Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i)$ ,  $R_i = N_d \oplus TRUST\_BITS$ ,  $F_i = H(N_i | N_r | ID_i | N_d | P_i | SS | TRUST\_BITS)$  and sends  $E_i$ ,  $Z_i$ ,  $R_i$  and  $F_i$  to the web browser of user  $U_i$ . The web browser of user  $U_i$  can compute  $N_i$  from  $E_i$  as  $N_i = E_i \oplus H(P_i)$ , value of  $N_d$  as  $N_d = Z_i \oplus ID_i \oplus SS \oplus H(P_i)$ ,  $TRUST\_BITS$  as  $TRUST\_BITS = R_i \oplus N_d$  and  $F_i' = H(N_i | N_r | ID_i | N_d | P_i | SS | TRUST\_BITS)$  and verifies the computed value of  $F_i'$  with the received value of  $F_i$  to validate that the messages are sent by the legitimate server  $S$  and not tampered during transmission. Hence the mutual authentication between the user  $U_i$  and the web server  $S$  is achieved as shown in Figure 3.3. Afterwards, the web server  $S$  checks  $CUR\_TRUST$  value in its database corresponding to the user identity  $ID_i$ . If  $CUR\_TRUST$  value stored in its database is more than or equal to  $MIN\_TRUST$  value but less than  $MAX\_TRUST$  value then the web server  $S$  increases the  $CUR\_TRUST$  value by one ( $CUR\_TRUST =$

CUR\_TRUST + 1) after successful authentication. If CUR\_TRUST value stored in its database is less than MIN\_TRUST then the web server resets the CUR\_TRUST value equal to MIN\_TRUST value after successful authentication. After successful authentication, the web server S stores the cookie CK on the client's computer. Then the user  $U_i$  and the web server S agree on the common session key as  $S_k = H(SS | ID_i | N_d | N_r | P_i | N_i | TRUST\_BITS)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the web server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the web server S can retrieve the original message because both of them know the common session key. If the user  $U_i$  fails to authenticate itself to the web server S then the web server S decreases the CUR\_TRUST value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ).

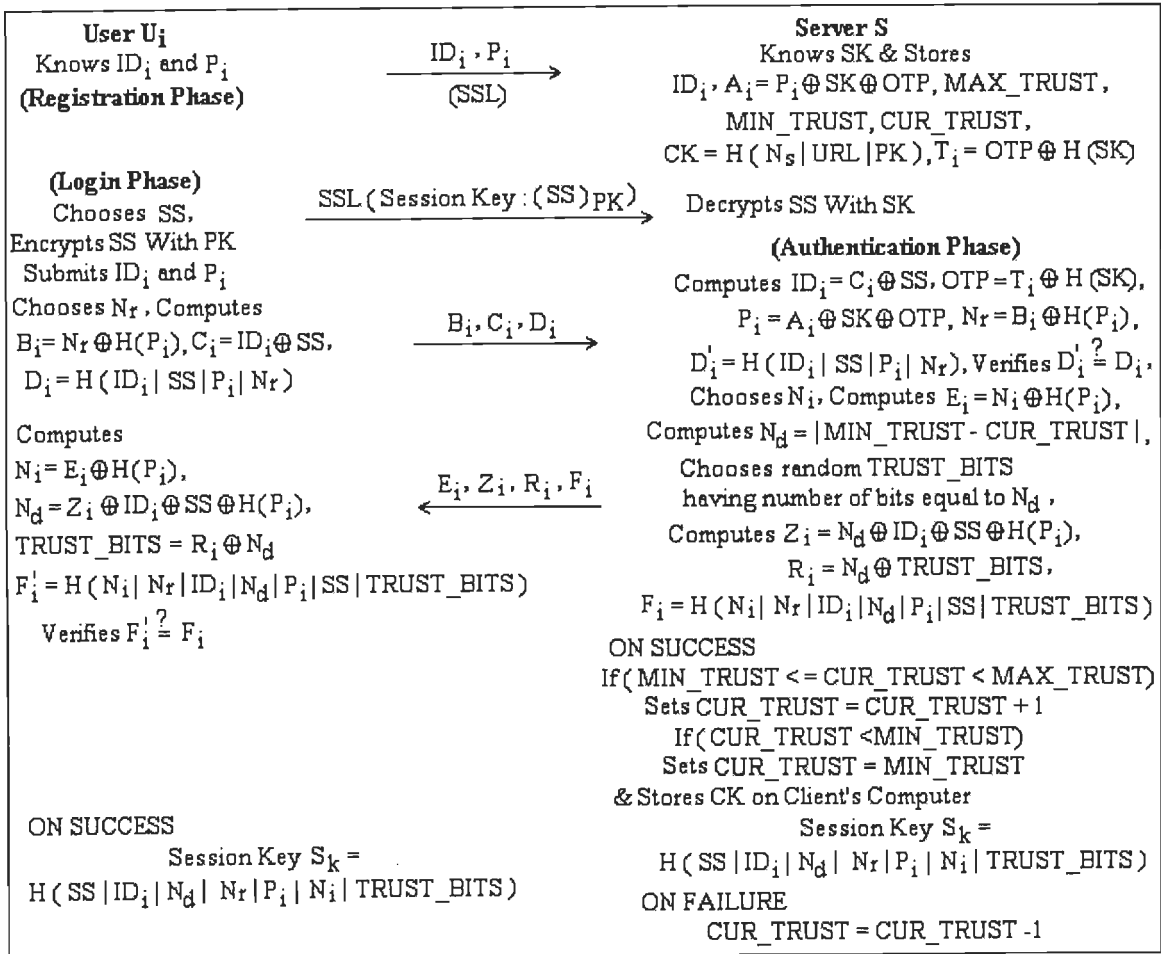


Figure 3.3: Protocol 2: Virtual password authentication protocol without cookie

#### 4. Password change phase

The password change phase is same as in Protocol 1. (See Section 3.2.1)

### 3.3 SECURITY ANALYSIS

The security of messages in online transaction inside communication channel is managed with SSL protocol. The proposed cookies based virtual password authentication protocol uses SSL protocol to establish SSL session key (SS) and then all the succeeding messages are communicated without using SSL protocol. This protocol provides good protection especially against online dictionary attacks. A good password authentication protocol should provide protection from different feasible attacks.

1. **Online dictionary attack:** In this type of attack, the attacker pretends to be legitimate client and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to generate  $K_i = H (ID_i | URL | PK | P_i | SS | CK)$  or  $\{B_i = N_r \oplus H (P_i), C_i = ID_i \oplus SS$  and  $D_i = H (ID_i | SS | P_i | N_r)\}$  corresponding to the user  $U_i$ , which are different for each new SSL session. With each failed login attempt, the difficulty of guessing TRUST\_BITS value increases because number of bits increases by one in TRUST\_BITS value after each login failure and sooner the guessing of TRUST\_BITS value will go out of the scope of the attacker as shown in Figure 3.2 and Figure 3.3. The legitimate user  $U_i$  can easily login on to the web server S, whatever may be the TRUST\_BITS and CUR\_TRUST values. Therefore, the proposed scheme is secure against online dictionary attack.
2. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess the user's identity and password from the recorded messages. The attacker obtains some identity and password verification information such as  $\{K_i = H (ID_i | URL | PK | P_i | SS | CK)\}$  or  $\{M_i = N_k \oplus H (ID_i | SS | P_i)$  and  $Q_i = H (ID_i | N_k | P_i | SS)\}$  or  $\{Z_i = N_d \oplus ID_i \oplus SS \oplus H (P_i), R_i = N_d \oplus TRUST\_BITS$  and  $V_i = H (ID_i | N_d | P_i | SS | TRUST\_BITS)\}$  or  $\{B_i = N_r \oplus H (P_i), C_i = ID_i \oplus SS$  and  $D_i = H (ID_i | SS | P_i | N_r)\}$  or  $\{E_i = N_i \oplus H (P_i), Z_i = N_d \oplus ID_i \oplus SS \oplus H (P_i), R_i = N_d \oplus TRUST\_BITS$  and  $F_i = H (N_i | N_r | ID_i | N_d | P_i | SS | TRUST\_BITS)\}$ . The attacker can not compute  $ID_i$  and  $P_i$  from these recorded messages. Therefore, the proposed protocol is secure against offline dictionary attack.
3. **Eavesdropping attack:** In this type of attack, the attacker first listens to all the communication between the client and the server and then tries to find out the client's identity  $ID_i$  and password  $P_i$ . The client's browser uses random nonce value  $N_r$  and

SSL session key  $SS$  for the generation of dynamic identity and password verifier information  $K_i = H(\text{ID}_i | \text{URL} | \text{PK} | \text{P}_i | \text{SS} | \text{CK})$  or  $\{B_i = N_r \oplus H(\text{P}_i), C_i = \text{ID}_i \oplus \text{SS}$  and  $D_i = H(\text{ID}_i | \text{SS} | \text{P}_i | N_r)\}$  corresponding to the user  $U_i$ , which are different for each new SSL session. The eavesdropper can not compute the user  $U_i$ 's identity  $\text{ID}_i$  and password  $\text{P}_i$  from any of the recorded message. Therefore, the proposed protocol is secure against eavesdropping attack.

4. **Denial of service attack:** In a specific type of denial of service attack, the server is cheated by the attacker to update the password verifier information with some false password verification information so that the legitimate user can not login successfully in subsequent login request to the server. The user  $U_i$  can change his password after the client and the server authenticate each other using the protocol shown in Figure 3.1 or Figure 3.2 or Figure 3.3. Therefore, the proposed protocol is secure against the user specific denial of service attack.
5. **Phishing attack:** In this type of attack, the attacker sends spoofed e-mails to different users from a website that is under the control of the attacker. Victim enters his valid login credentials into the fraudulent website that allows the attacker to transfer funds from the victim's account or cause other damages. The proposed protocol generates a new dynamic identity and password verifier information  $K_i = H(\text{ID}_i | \text{URL} | \text{PK} | \text{P}_i | \text{SS} | \text{CK})$  or  $\{B_i = N_r \oplus H(\text{P}_i), C_i = \text{ID}_i \oplus \text{SS}$  and  $D_i = H(\text{ID}_i | \text{SS} | \text{P}_i | N_r)\}$  corresponding to the user  $U_i$ , which are different for each new SSL session. The fraudulent server can ignore dynamic identity and password verifier information but can not produce valid credentials  $\{M_i = N_k \oplus H(\text{ID}_i | \text{SS} | \text{P}_i)$  and  $Q_i = H(\text{ID}_i | N_k | \text{P}_i | \text{SS})\}$  or  $\{Z_i = N_d \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{P}_i), R_i = N_d \oplus \text{TRUST\_BITS}$  and  $V_i = H(\text{ID}_i | N_d | \text{P}_i | \text{SS} | \text{TRUST\_BITS})$  or  $\{E_i = N_i \oplus H(\text{P}_i), Z_i = N_d \oplus \text{ID}_i \oplus \text{SS} \oplus H(\text{P}_i), R_i = N_d \oplus \text{TRUST\_BITS}$  and  $F_i = H(N_i | N_r | \text{ID}_i | N_d | \text{P}_i | \text{SS} | \text{TRUST\_BITS})\}$  meant for the user  $U_i$  because it does not have any such credentials. Therefore, the proposed protocol is secure against phishing attack.
6. **Pharming attack:** Pharming is a technique that fools the user by connecting his machine to a fake website even when the user submits correct domain name in to the web browser. This technique exploits vulnerabilities in the DNS servers to distribute the fake address information by DNS spoofing attack. Like phishing attacks, the attacker sets up a capture site to collect identity and password verifier information. The

attacker can cause the DNS caching server to return false information and direct the user to a malicious site. Malicious site can not impersonate as valid server because it can not produce valid credentials  $\{M_i = N_k \oplus H(ID_i | SS | P_i)$  and  $Q_i = H(ID_i | N_k | P_i | SS)\}$  or  $\{Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i), R_i = N_d \oplus TRUST\_BITS$  and  $V_i = H(ID_i | N_d | P_i | SS | TRUST\_BITS)\}$  or  $\{E_i = N_i \oplus H(P_i), Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i), R_i = N_d \oplus TRUST\_BITS$  and  $F_i = H(N_i | N_r | ID_i | N_d | P_i | SS | TRUST\_BITS)\}$  meant for the user  $U_i$ , which are unique for each new session. Therefore, the attacker can not launch pharming attack on the proposed protocol.

7. **Man-in-the-middle attack:** In this type of attack, the attacker intercepts the messages sent between the client and the server and replay these intercepted messages. The attacker can act as the client to the server or vice-versa with recorded messages. In the proposed protocol, the attacker can intercept the login request message  $\{K_i = H(ID_i | URL | PK | P_i | SS | CK)$  and  $CK\}$  or  $\{B_i = N_r \oplus H(P_i), C_i = ID_i \oplus SS$  and  $D_i = H(ID_i | SS | P_i | N_r)\}$  corresponding to the user  $U_i$ , which is sent by a user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message. The attacker can authenticate itself to server  $S$  as well as to legitimate user  $U_i$  by replaying old messages but can not compute the session key  $S_k = H(SS | P_i | N_k | CK | ID_i)$  or  $S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i)$  or  $S_k = H(SS | ID_i | N_d | N_r | P_i | N_i | TRUST\_BITS)$  because the attacker does not know the value of  $ID_i, P_i, SS, N_k, N_i, N_r, N_d$  and  $TRUST\_BITS$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
8. **Replay attack:** In this type of attack, the attacker first listens to the communication between the client and the server. Then the attacker tries to imitate the user to login on to the server by resending the captured messages. Replaying a message of one SSL session into another SSL session is useless because each SSL session generates a different dynamic identity and password verifier information corresponding to the same client because the session key  $SS$  is different for each new SSL session. Therefore, the messages can not be replayed successfully in any other SSL session. Moreover, the attacker can not compute the session key. Therefore, the proposed protocol is secure against message replay attack.
9. **Leak of verifier attack:** In this type of attack, the attacker may be able to steal verification table from the server. In case the password verifier information

$ID_i, A_i = P_i \oplus SK \oplus OTP, MAX\_TRUST, MIN\_TRUST, CUR\_TRUST,$   
 $CK = H(N_s | URL | PK)$  and  $T_i = OTP \oplus H(SK)$  is stolen by breaking into the server's database, the attacker does not have sufficient information to calculate the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  because the attacker has to guess  $SK$  and  $OTP$  correctly at the same time. It is not possible to guess  $SK$  and  $OTP$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against leak of verifier attack.

**10. Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the client's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can result in authentication failure between the client and the server but can not allow the attacker to gain any information about the client's password or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

**11. Brute force attack:** To launch brute force attack, an attacker first obtains some password verification information such as  $\{K_i = H(ID_i | URL | PK | P_i | SS | CK)\}$  from Figure 3.1 or Figure 3.2 protocol or  $\{B_i = N_r \oplus H(P_i), C_i = ID_i \oplus SS$  and  $D_i = H(ID_i | SS | P_i | N_r)\}$  from Figure 3.3 protocol. Even after recording these messages, the attacker has to guess minimum two parameters out of  $ID_i, P_i, N_r$  and  $SS$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against brute force attack.

### 3.4 PROPOSED INVERSE COOKIE BASED VIRTUAL PASSWORD AUTHENTICATION PROTOCOL

In general, the web server stores cookie on the user's computer if the legitimate user has successfully authenticated itself to the web server from that computer. On the other hand, the proposed protocol is termed as inverse cookie based virtual password authentication protocol because the web server stores cookie on the user's or the attacker's computer when the user or the attacker has not submitted correct identity and password for its authentication to the web server. An attacker can not launch online dictionary attack because the complexity of computation on the client side increases with each login failure

for an attacker so that it is very difficult for an attacker to impersonate as a legitimate user. The proposed protocol runs on top of the SSL protocol [34] and comprises four phases as follows. The notations used in this section are listed in Table 3.1.

We present two authentication protocols. Each protocol has four phases. These two protocols have same registration phase and password change phase and they differ in login phase and authentication phase. Protocol 1 does not use cookies whereas Protocol 2 makes use of cookies for the user's authentication. The user  $U_i$  has to follow the Protocol 1 if the user  $U_i$ 's computer does not contain cookie else the user  $U_i$  has to follow Protocol 2.

### 3.4.1 Protocol 1

This protocol is shown in Figure 3.4 and its various phases are described below.

#### 1. Registration phase

The registration phase is same as in Protocol 1 (See Section 3.2.1) except that the web server  $S$  stores  $CK$  as cookie information on the client's or the attacker's computer when the user  $U_i$  or the attacker fails to authenticate itself to the web server  $S$ . The web server  $S$  does not store cookie information on the client's computer when the user  $U_i$  authenticates itself to the web server successfully.

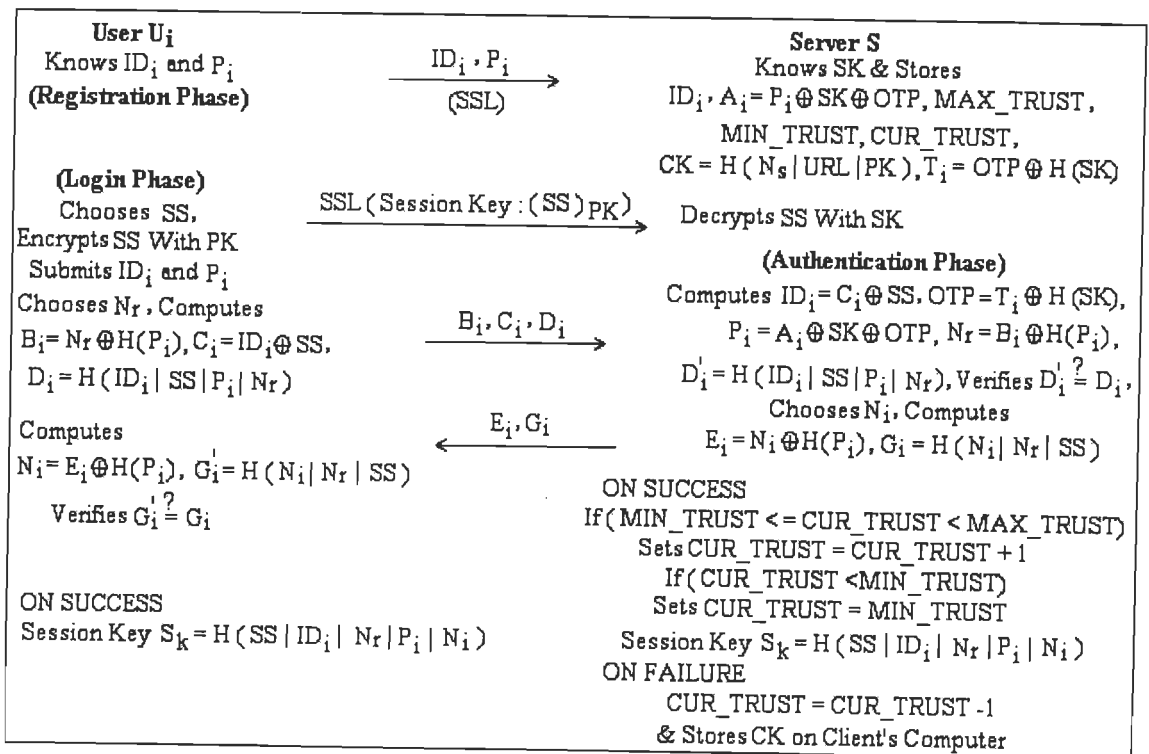


Figure 3.4: Protocol 1: Virtual password authentication protocol without cookie



## 2. Login phase

The user  $U_i$  agrees on SSL session key  $SS$  with the web server  $S$  using the SSL protocol as shown in login phase of Protocol 1 in Section 3.2.1. Then all the subsequent messages of this protocol are transmitted in the open without using SSL protocol.

The user  $U_i$  submits his identity  $ID_i$  and password  $P_i$  to the web browser. If the user  $U_i$ 's computer does not contain cookie  $CK$  then the user  $U_i$ 's web browser chooses random nonce value  $N_r$ , computes  $B_i = N_r \oplus H(P_i)$ ,  $C_i = ID_i \oplus SS$  and  $D_i = H(ID_i | SS | P_i | N_r)$ . The web browser of user  $U_i$  submits  $B_i$ ,  $C_i$  and  $D_i$  to the web server  $S$  as shown in Figure 3.4.

## 3. Authentication phase

The web server  $S$  computes  $ID_i$  from  $C_i$  as  $ID_i = C_i \oplus SS$  and recognizes the user  $U_i$  from its identity  $ID_i$ . After that, the web server  $S$  computes  $OTP$  as  $OTP = T_i \oplus H(SK)$  because the web server  $S$  knows its private key  $SK$ . Then the web server  $S$  computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and  $N_r$  from  $B_i$  as  $N_r = B_i \oplus H(P_i)$ . Afterwards, the web server  $S$  computes  $D_i' = H(ID_i | SS | P_i | N_r)$  and verifies it with the received value of  $D_i$ . If both values are equal, the web server  $S$  proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. The web server  $S$  chooses random nonce value  $N_i$  and computes  $E_i = N_i \oplus H(P_i)$ ,  $G_i = H(N_i | N_r | SS)$  and sends  $E_i$  and  $G_i$  to the web browser of user  $U_i$ . The web browser computes  $N_i$  from  $E_i$  as  $N_i = E_i \oplus H(P_i)$  because the web browser knows password  $P_i$  of the user  $U_i$ . Then the web browser computes  $G_i' = H(N_i | N_r | SS)$  and verifies the computed value of  $G_i'$  with the received value of  $G_i$  to validate that the messages are sent by the legitimate server  $S$  and not tampered during transmission. Hence the mutual authentication between the user  $U_i$  and the web server  $S$  is achieved as shown in Figure 3.4. Afterwards, the web server  $S$  checks  $CUR\_TRUST$  value in its database corresponding to the user identity  $ID_i$ . If  $CUR\_TRUST$  value stored in its database is more than or equal to  $MIN\_TRUST$  but less than  $MAX\_TRUST$  then the web server increases the  $CUR\_TRUST$  value by one ( $CUR\_TRUST = CUR\_TRUST + 1$ ) after successful authentication. If  $CUR\_TRUST$  value stored in its database is less than  $MIN\_TRUST$  then the web server resets the  $CUR\_TRUST$  value equal to  $MIN\_TRUST$  value after successful authentication. After successful authentication, the user  $U_i$  and the web server  $S$  agree on the common session key as  $S_k = H(SS | ID_i | N_r | P_i | N_i)$ .

Afterwards, all the subsequent messages between the user  $U_i$  and the web server  $S$  are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the web server  $S$  can retrieve the original message because both of them know the common session key. If the user  $U_i$  fails to authenticate itself to the web server  $S$  then the web server  $S$  decreases the CUR\_TRUST value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ) and stores the cookie CK on the client's computer.

#### 4. Password change phase

The password change phase is same as in Protocol 1. (See Section 3.2.1)

### 3.4.2 Protocol 2

This protocol is shown in Figure 3.5 and Figure 3.6 and its various phases are described below.

#### 1. Registration phase

The registration phase is same as in Protocol 1. (See Section 3.4.1)

#### 2. Login phase

The user  $U_i$  agrees on SSL session key  $SS$  with the web server  $S$  using the SSL protocol as shown in login phase of Protocol 1 in Section 3.2.1. Then all the subsequent messages of this protocol are transmitted in the open without using SSL protocol.

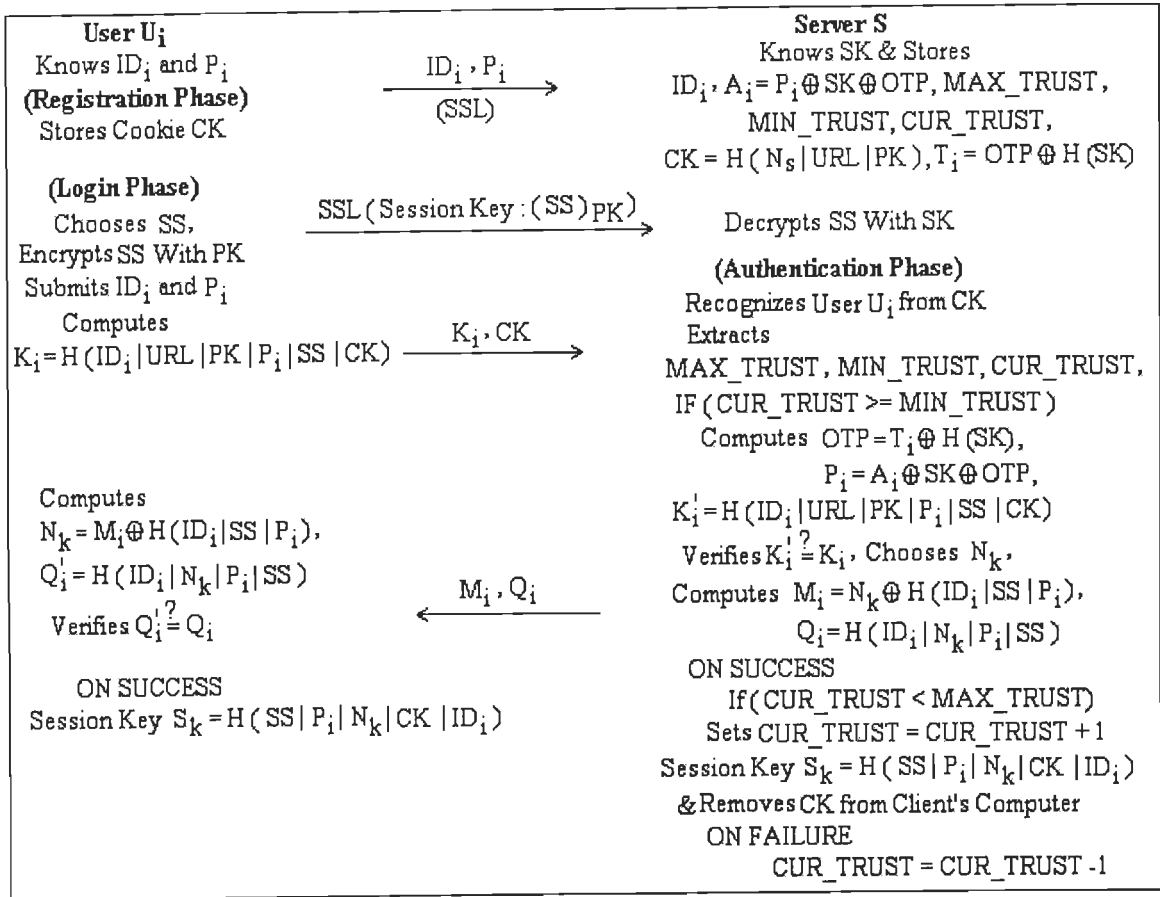
The User  $U_i$  submits his identity  $ID_i$  and password  $P_i$  to the web browser. If the user  $U_i$ 's computer contains cookie CK then the user  $U_i$ 's web browser computes dynamic identity and password verifier information  $K_i = H (ID_i | URL | PK | P_i | SS | CK)$  and submits  $K_i$  and CK to the web server  $S$  as shown in Figure 3.5 and Figure 3.6.

#### 3. Authentication phase

The web server  $S$  recognizes the user  $U_i$  from the received cookie CK and extracts MAX\_TRUST, MIN\_TRUST and CUR\_TRUST corresponding to cookie CK from its database.

#### Case 1:

If CUR\_TRUST value is more than or equal to MIN\_TRUST value then the web server  $S$  computes OTP as  $OTP = T_i \oplus H (SK)$  because the web server  $S$  knows its private key SK.



**Figure 3.5: Protocol 2: (Case 1) Virtual password authentication protocol with cookie**

Then the web server S computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and computes the dynamic identity and password verifier information  $K_i' = H(ID_i | URL | PK | P_i | SS | CK)$  and verifies the computed value of  $K_i'$  with the received value of  $K_i$ . If both values are equal, the web server S proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. Then the web server S chooses a random value of  $N_k$ , computes  $M_i = N_k \oplus H(ID_i | SS | P_i)$ ,  $Q_i = H(ID_i | N_k | P_i | SS)$  and sends  $M_i$  and  $Q_i$  to the web browser of user  $U_i$ . The web browser of user  $U_i$  computes  $N_k = M_i \oplus H(ID_i | SS | P_i)$ ,  $Q_i' = H(ID_i | N_k | P_i | SS)$  and verifies the computed value of  $Q_i'$  with the received value of  $Q_i$  to validate that the messages are sent by the legitimate web server S and not tampered during transmission. This equivalency authenticates the legitimacy of the user  $U_i$  and the web server S and the login request is accepted else the connection is interrupted. Hence the mutual authentication between the user  $U_i$  and the web server S is achieved as shown in Figure 3.5. Afterwards, the web server S checks  $CUR\_TRUST$  value in its database corresponding to the user identity  $ID_i$ . If the  $CUR\_TRUST$  value stored in the database of

web server S is less than MAX\_TRUST value then the CUR\_TRUST value is incremented by one ( $CUR\_TRUST = CUR\_TRUST + 1$ ) after successful login attempt by the user  $U_i$  on the web server S. Finally after successful authentication, the user  $U_i$  and the web server S agree on the common session key as  $S_k = H(SS | P_i | N_k | CK | ID_i)$  and the server S removes the cookie CK from the client's computer. Afterwards, all the subsequent messages between the user  $U_i$  and the web server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the web server S can retrieve the original message because both of them know this common session key. If the user  $U_i$  fails to authenticate itself to the web server S then the web server S decreases the CUR\_TRUST value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ).

**Case 2:**

If CUR\_TRUST value is less than MIN\_TRUST value then the web server S computes OTP as  $OTP = T_i \oplus H(SK)$  because the web server S knows its private key SK. Then the web server S computes  $P_i$  as  $P_i = A_i \oplus SK \oplus OTP$  and computes the dynamic identity and password verifier information  $K_i' = H(ID_i | URL | PK | P_i | SS | CK)$  and verifies the computed value of  $K_i'$  with the received value of  $K_i$ . If both values are equal, the web server S proceeds to the next step. Otherwise, the login request from the user  $U_i$  is rejected. Then the web server S computes  $N_d = MIN\_TRUST - CUR\_TRUST$  and chooses random TRUST\_BITS value having bits equal to the value of  $N_d$ . Suppose the value of  $N_d$  is 2 then the number of bits in TRUST\_BITS value will be 2. Then the web server S computes  $Z_i = N_d \oplus ID_i \oplus SS \oplus H(P_i)$ ,  $R_i = N_d \oplus TRUST\_BITS$ ,  $V_i = H(ID_i | N_d | P_i | SS | TRUST\_BITS)$  and sends  $Z_i$ ,  $R_i$  and  $V_i$  to the web browser of user  $U_i$ . The web browser of user  $U_i$  can compute the value of  $N_d$  as  $N_d = Z_i \oplus ID_i \oplus SS \oplus H(P_i)$ , TRUST\_BITS as  $TRUST\_BITS = R_i \oplus N_d$  and  $V_i' = H(ID_i | N_d | P_i | SS | TRUST\_BITS)$  and verifies the computed value of  $V_i'$  with the received value of  $V_i$ . Hence the mutual authentication between the user  $U_i$  and the web server S is achieved as shown in Figure 3.6. Finally after successful authentication, the user  $U_i$  and the web server S agree on the common session key as  $S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i)$  and the server S removes the cookie CK from the client's computer. Afterwards, all the subsequent messages between the user  $U_i$  and the web server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the web server S can retrieve the original message because both of them know the common session key. Then the web server S resets the CUR\_TRUST value corresponding

to user  $U_i$  equal to  $MIN\_TRUST$  value after successful authentication. If the user  $U_i$  fails to authenticate itself to the web server  $S$  then the web server  $S$  decreases the  $CUR\_TRUST$  value by one ( $CUR\_TRUST = CUR\_TRUST - 1$ ). The attacker has to guess the value of  $SS$ ,  $ID_i$ ,  $N_d$ ,  $TRUST\_BITS$  and  $P_i$  to compute the common session key as  $S_k = H(SS | ID_i | N_d | CK | TRUST\_BITS | P_i)$ . The computational efforts required by the attacker to find the  $TRUST\_BITS$  value increases exponentially with each login failure because the number of bits in  $TRUST\_BITS$  increases by one after each login failure.

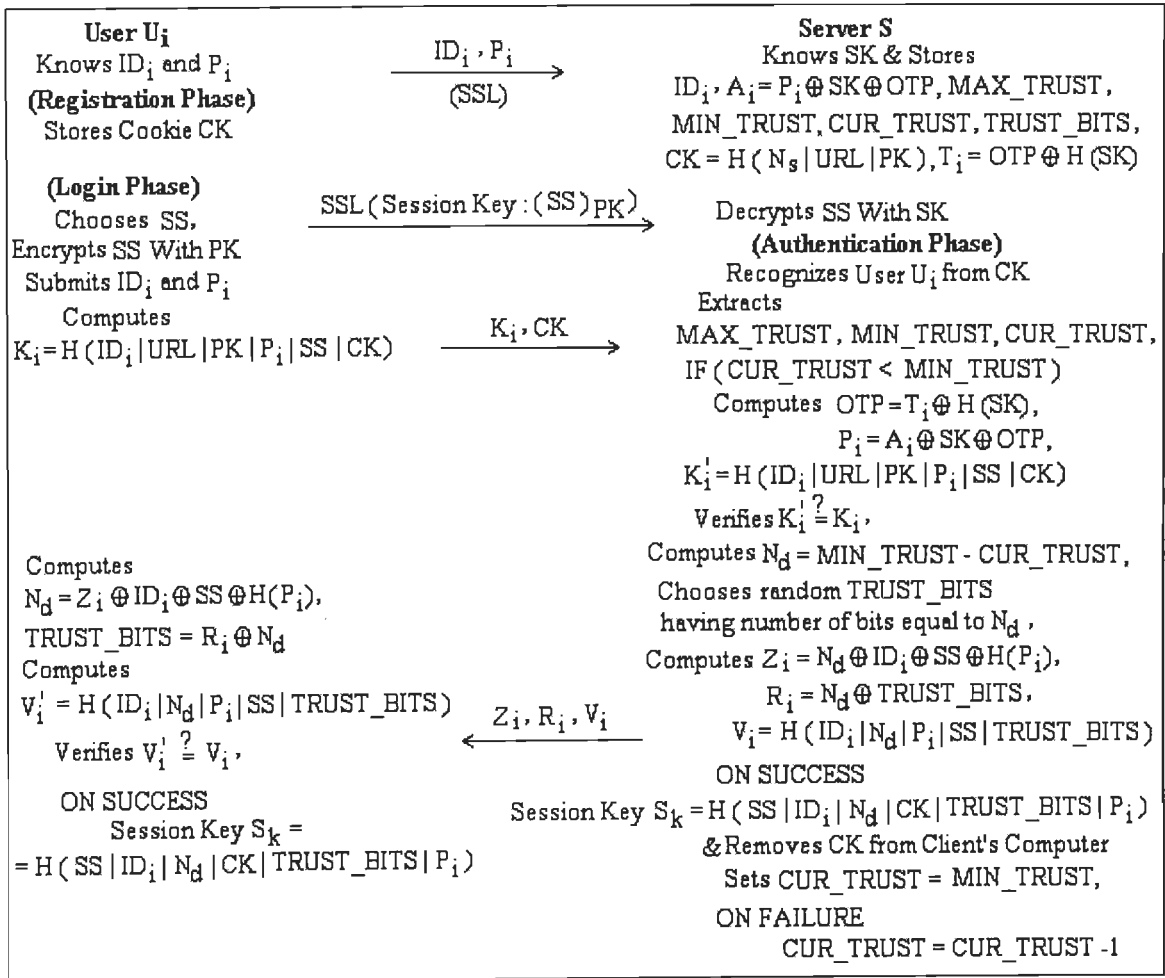


Figure 3.6: Protocol 2: (Case 2) Virtual password authentication protocol with cookie

#### 4. Password change phase

The password change phase is same as in Protocol 1. (See Section 3.2.1)

### 3.5 SECURITY ANALYSIS

The security analysis is same. (See Section 3.3)

### **3.6 CONCLUSION**

We have specified and analyzed a cookie and an inverse cookie based virtual password authentication protocols which are very effective to thwart online dictionary attacks because the computation cost to login on to the web server increases exponentially with each login failure for an attacker. The legitimate client can easily authenticate itself to the web server from any computer irrespective of whether that computer contains cookie or not. The proposed protocols are simple and fast if the user is using valid identity and correct password for its authentication. These protocols are practical and efficient because only one-way hash functions and XOR operations are used in its implementation. Security analysis proved that the proposed protocols are secure and practical.

**SSO PASSWORD BASED TWO-SERVER  
AUTHENTICATION PROTOCOL**

---

**4.1 INTRODUCTION**

Internet is a huge source of information and requires access control so that valuable information is accessible only to the authorized users. Therefore, it is essential to authenticate the identity of remote users. Password is the most extensively used authentication technique for the user's authentication. Single Sign-On (SSO) provides an environment in which the users sign in once and are able to access the services offered by different servers under the same administrative control. It allows the user to enter identity and password once within a specific time period to login on to multiple hosts and applications within an organization. The single login capability of SSO provides a unified view of each user's interactions in the organization and improves the quality of the services provided by that organization. This concept is used in accessing low risk information from multiple applications by authenticating the user to the server only once for a specific time period. On the other hand, high risk applications require strong authentication techniques like digital certificates, security tokens, smart cards and biometrics. Two common approaches to SSO are based on token and proxy that provide authentication as well as authorization. Token based approach uses centralized server to issue single encrypted token to the user after successful authentication. That single encrypted token can be used by the user for its authentication across multiple sites. In proxy based approach, the user login on to centralized server and presents the correct user credentials (e.g. certificate, token) to the respective application or the server.

The web server uses cookies for maintaining the information related to its clients to keep track of the client state. It provides a mechanism to pass the information between the user's computer and the service provider web server. The web server creates a cookie containing information related to the user and stores it on the user's computer that is accessed by the web server in subsequent login requests from the same user. The web server can choose different information fields related to the user and store it inside the cookie.

Window Live ID (Passport) is a SSO and one of the largest dedicated authentication service provided by Microsoft on the web. This authentication service manages accounts of all Hotmail and MSN Messenger users. A Passport enabled website does not need to handle the authentication itself and requires a user to use only one password for a group of sites. These websites delegate the task of user's authentication to Window Live ID which decides about the authenticity of the user that wants to sign in. Window Live ID issues a ticket to the user after verification that contains Passport Unique Identity (PUID) and timestamp of last sign-in. This ticket is encrypted with a key that Window Live ID shared with the website that requires authentication. Received ticket is decrypted by the website using its key shared with the Window Live ID to recognize the user from PUID and then check the authentication that occurred within a certain time period. These tickets are cached using the cookies so they can be used for a specific valid time period. Cookies are domain specific and hence the web servers can store tickets belonging to various domains as different cookies on the user's computer. Window Live ID also keeps cookies in its own domain to authenticate a user silently into another domain if that user has already logged into one of the domains successfully.

Most of the existing password authentication protocols are based on single-server model containing the user's password verifier information in its database. Password verification information stored on the single server is vulnerable to dictionary attack. The concept of plain multi-server model removes this common point of vulnerability but all the servers are equally exposed to the users and a user has to communicate in parallel with all the servers for its authentication. This communication model demands more communication bandwidth and requires user synchronization with multiple servers. This kind of setup is not suitable for resource constraint devices such as hand held devices and personal digital assistant. Another concept of gateway based multi-server model uses a gateway intermediately between the users and the servers. The concept of gateway does not require simultaneous parallel communication by a user with the multiple servers but increases another layer in the architecture. In 2006, Yang et al. [162] suggested two-server model for the user's authentication but without SSO authentication. In this chapter, the proposed SSO protocol also uses two-server model consisting of two servers that work together to authenticate the users. In the proposed protocol, different levels of trust are assigned to the servers as the authentication server is more exposed to the users than the control server. The back-end control server is not directly exposed to the users and thus it



is less likely to be attacked. The two-server model provides the flexibility to distribute the user passwords and the authentication functionality into two servers to eliminate the common point of vulnerability of the single-server model. Therefore, two-server model appears to be a reasonable choice for practical applications.

In this chapter, we present a secure and efficient SSO password based two-server authentication protocol. The aim of this chapter is to use two-server architecture so that password verification information is distributed between two servers and ticket is stored as cookie on the user's computer. The user can use this ticket to derive dynamic authentication information for its authentication to authentication servers. The user has to authenticate itself to the authentication and the control servers once to get the valid ticket for a specific time period. Then the user can use the same valid ticket to derive dynamic authentication information for its authentication to same or different authentication servers which are under the control of the same control server. The proposed authentication protocol is suitable for distributed network environment.

## **4.2 PROPOSED SSO AUTHENTICATION PROTOCOL**

Most of the password based user authentication protocols use single authentication server to store passwords or password verifier information. The attacker can compromise the authentication server and subject it to dictionary attack to find out the passwords of various users. The proposed SSO protocol uses two-server paradigm and hence is more secure against offline dictionary attack mounted on either of the two servers because the password verification information is distributed between two servers. This protocol can be used to strengthen existing single-server password system. The protocol is simple and fast if the user is using a valid ticket and correct passwords corresponding to the authentication server and the control server as shown in Figure 4.1 and Figure 4.2. The notations used in this section are listed in Table 4.1.

The proposed protocol provides password based SSO authentication using two servers namely Authentication Server ( $AS_i$ ) and Control Server (CS). The legitimate user can authenticate itself to  $AS_i$  and CS using previously shared passwords. After successful authentication, CS first generates and stores a ticket in its database and then issues this ticket to the  $AS_i$ , through which the user has requested the ticket. The validity time period (EXP\_TIME) for this ticket is decided by the CS.  $AS_i$  checks the authenticity of the ticket and then stores the ticket in its database and forwards it as cookie to the user's computer.

The user checks the validity of the received message and after verification it stores the received ticket as cookie on its computer. Then the user uses the same ticket in succeeding login attempts on same or different authentication servers which are under the control of same CS. The protocol has four phases.

**Table 4.1**  
**Notations**

$U_i$	User $U_i$
$AS_i$	$i^{th}$ Authentication server
CS	Control server
$ID_i$	Unique Identity of user $U_i$
$PS_i$	Password shared between user $U_i$ and $AS_i$
$P_i$	Password shared between user $U_i$ and CS
$H()$	One-way hash function
$Y_i$	Private key of $AS_i$
X	Private key of CS
OTP	One time secret stored on CS
$SK_i$	Secret key shared between $AS_i$ and CS
EXP_TIME	Expiration time of ticket
$\oplus$	XOR operation
	Concatenation
$N, N_i, N'$	Random nonce values
T	Current date and time of user $U_i$ 's computer

**1. Registration phase**

When a user  $U_i$  wants to become a legitimate client, the user  $U_i$  has to submit its identity  $ID_i$  and password  $PS_i$  to  $AS_i$ , and identity  $ID_i$  and password  $P_i$  to the CS independently via a secure communication channel. Then, these servers choose and compute some security parameters and store them on their databases. A user has to register to CS once and independently to different  $AS_i$ .

**2. Login and ticket request phase**

When a user  $U_i$  wants to get the valid ticket from the server CS, the user  $U_i$  sends its identity and valid password verifier information to  $AS_i$ .  $AS_i$  extracts its own password information from the received message and forwards the modified message to CS. CS verifies the password verification information from its database for the server  $AS_i$ 's and the user  $U_i$ 's authentication. Once the authentication among the user  $U_i$ ,  $AS_i$  and CS completes then the CS issues the ticket to the  $AS_i$ . Afterwards,  $AS_i$  verifies the authenticity

of the ticket. Then  $AS_i$  stores and forwards the ticket to the user  $U_i$ . Finally, the user  $U_i$  verifies the authenticity of the ticket and stores it as cookie on its computer. The user  $U_i$  can use this ticket to get the services of the  $AS_i$  and CS in any succeeding login attempt on to these servers.

### 3. Authentication phase

The user  $U_i$  can use this valid ticket to authenticate itself to  $AS_i$ . The user  $U_i$  can also use the same ticket for its authentication to different authentication servers which are under the control of same CS.

### 4. Password change phase

The user  $U_i$  has to authenticate itself to respective server  $AS_i$  or CS before changing the password.

#### 4.2.1 Registration phase

The user  $U_i$  sends its identity  $ID_i$  and password  $P_i$  to CS via a secure communication channel for its registration.

Step 1:  $U_i \rightarrow CS: ID_i, P_i$

CS has already stored  $OTP \oplus H(X)$  in its master database. CS does not need to remember the OTP and can retrieve it from  $OTP \oplus H(X)$  because CS knows its private key  $X$ . CS encrypts the password  $P_i$  of the user  $U_i$  using its private key  $X$  and one time secret OTP as  $P_i \oplus H(OTP | X)$  to defend the password  $P_i$  from stolen verifier attack and stores  $P_i \oplus H(OTP | X)$  corresponding to the user  $U_i$ 's identity  $ID_i$  in its client's database. All authentication servers register with CS and CS agrees on a unique secret key  $SK_i$  with each authentication server  $AS_i$ . The  $AS_i$  remembers the secret key  $SK_i$  and CS stores the secret key  $SK_i$  as  $SK_i \oplus H(X | OTP)$  corresponding to authentication server identity  $AS_i$  in its authentication server's database.

Then the user  $U_i$  has to register individually to different authentication servers with different passwords. The user  $U_i$  sends its identity  $ID_i$  and password  $PS_i$  to  $AS_i$  for registration via a secure communication channel.

Step 2:  $U_i \rightarrow AS_i: ID_i, PS_i$

The server  $AS_i$  encrypts the password  $PS_i$  of the user  $U_i$  using its private key  $Y_i$  and  $SK_i$  as  $PS_i \oplus Y_i \oplus SK_i$  to defend the password  $PS_i$  from stolen verifier attack and stores  $PS_i \oplus Y_i \oplus SK_i$  corresponding to the user's identity  $ID_i$  in its database.

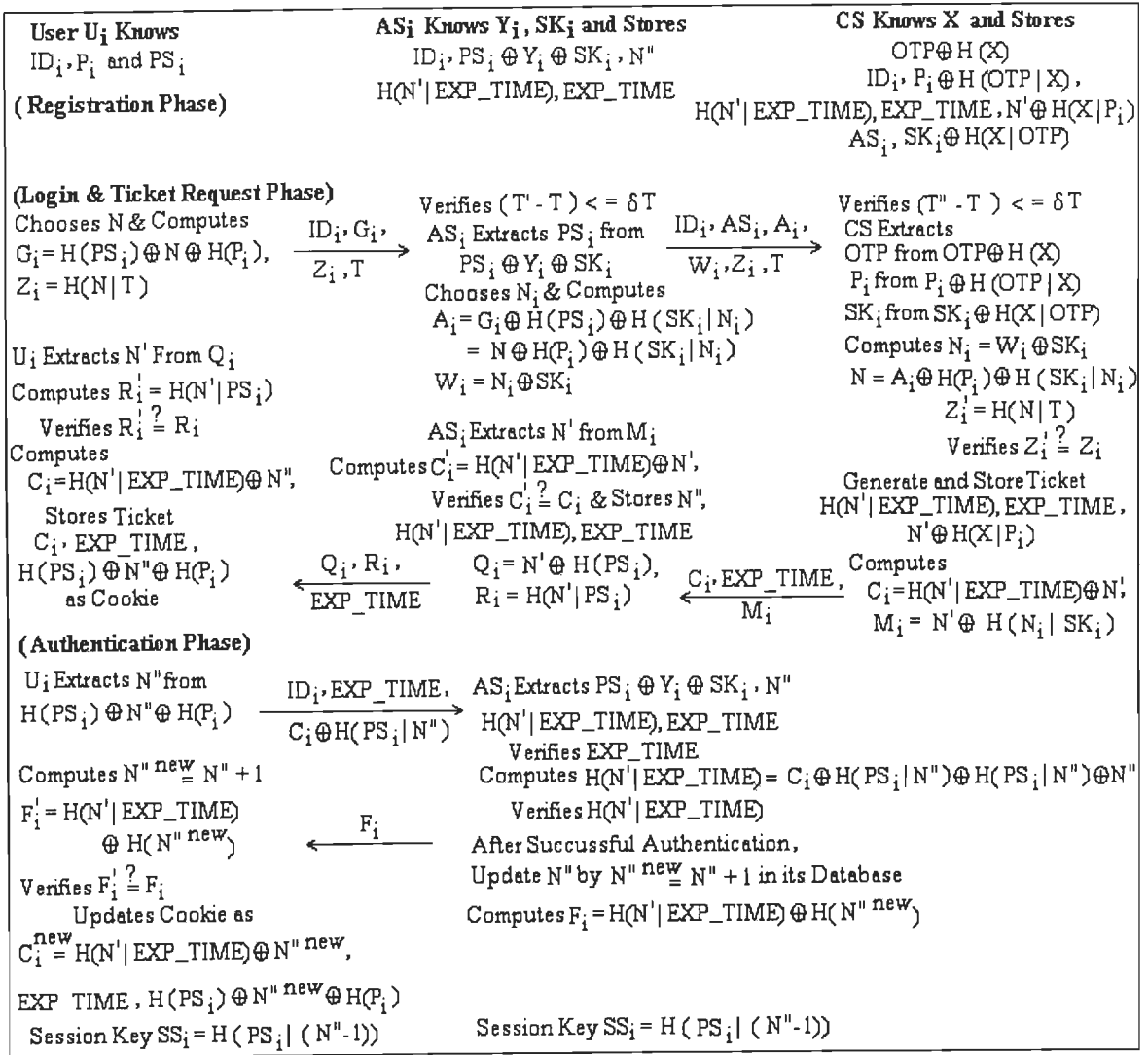


Figure 4.1: SSO password based two-server authentication protocol

#### 4.2.2 Login and ticket request phase

The user  $U_i$  chooses random nonce value  $N$ , computes  $G_i = H(PS_i) \oplus N \oplus H(P_i)$ ,  $Z_i = H(N|T)$  and sends  $ID_i, G_i, Z_i, T$  to  $AS_i$  to get the valid ticket, where  $T$  is current date and time of the user's computer.

Step 1:  $U_i \rightarrow AS_i: ID_i, G_i, Z_i, T$

After receiving the login request from the user  $U_i$ , the service provider server  $AS_i$  checks the validity of the timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $AS_i$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards,  $AS_i$  extracts  $PS_i$  from  $PS_i \oplus Y_i \oplus SK_i$  corresponding to the user  $U_i$ 's identity  $ID_i$  from its database. Then the  $AS_i$  generates random nonce value  $N_i$ , computes

$A_i = G_i \oplus H(PS_i) \oplus H(SK_i | N_i) = N \oplus H(P_i) \oplus H(SK_i | N_i)$  and  $W_i = N_i \oplus SK_i$ . Then  $AS_i$  sends  $ID_i, AS_i, A_i, W_i, Z_i$  and  $T$  to CS.

Step 2:  $AS_i \rightarrow CS: ID_i, AS_i, A_i, W_i, Z_i, T$

CS checks the validity of timestamp  $T$  by checking  $(T'' - T) \leq \delta T$ , where  $T''$  is current date and time of server CS and  $\delta T$  is permissible time interval for a transmission delay. CS extracts OTP from  $OTP \oplus H(X)$  and then extracts  $P_i$  from  $P_i \oplus H(OTP | X)$  corresponding to the user's identity  $ID_i$  from its client's database and also extracts  $SK_i$  from  $SK_i \oplus H(X | OTP)$  corresponding to authentication server identity  $AS_i$  from its authentication server's database. Then CS computes  $N_i$  from  $W_i$  as  $N_i = W_i \oplus SK_i$ ,  $N$  from  $A_i$  as  $N = A_i \oplus H(P_i) \oplus H(SK_i | N_i)$ ,  $Z_i' = H(N | T)$  and verifies the computed value of  $Z_i'$  with the received value of  $Z_i$  to check the authenticity of the received message. Once the received message is authenticated then CS generates a new ticket  $H(N' | EXP\_TIME)$  and stores  $H(N' | EXP\_TIME)$ ,  $EXP\_TIME$  and  $N' \oplus H(X | P_i)$  corresponding to the user's identity  $ID_i$  and  $P_i \oplus H(OTP | X)$  in its client's database, where  $N'$  is a random nonce value chosen by CS. Then CS computes  $C_i = H(N' | EXP\_TIME) \oplus N'$ ,  $M_i = N' \oplus H(N_i | SK_i)$  and sends  $C_i, EXP\_TIME$  and  $M_i$  to  $AS_i$ .

Step 3:  $CS \rightarrow AS_i: C_i, EXP\_TIME, M_i$

$AS_i$  extracts  $N'$  from the received value of  $M_i$  as  $N' = M_i \oplus H(N_i | SK_i)$ , computes  $C_i' = H(N' | EXP\_TIME) \oplus N'$  and then compares the computed value of  $C_i'$  with received value of  $C_i$ . This equivalency authenticates the legitimacy of CS and authenticity of the received message. After successful authentication,  $AS_i$  stores  $N''$ ,  $H(N' | EXP\_TIME)$  and  $EXP\_TIME$  corresponding to the user  $U_i$ 's identity  $ID_i$  and  $PS_i \oplus Y_i \oplus SK_i$  in its database. Here the value of  $N''$  is set equal to  $N'$ , when the ticket is initially issued to the user  $U_i$ . Then  $AS_i$  computes  $Q_i = N' \oplus H(PS_i)$ ,  $R_i = H(N' | PS_i)$  and sends  $Q_i, R_i$  and  $EXP\_TIME$  to the user  $U_i$ .

Step 4:  $AS_i \rightarrow U_i: Q_i, R_i, EXP\_TIME$

The user  $U_i$  knows  $PS_i$  and can extract  $N'$  from  $Q_i$  as  $N' = Q_i \oplus H(PS_i)$ , computes  $R_i' = H(N' | PS_i)$  and then verifies  $R_i'$  with the received value of  $R_i$  to check the legitimacy of  $AS_i$ . After verification, the user  $U_i$  computes the ticket information  $C_i = H(N' | EXP\_TIME) \oplus N''$  and stores the ticket information  $C_i, EXP\_TIME, H(PS_i) \oplus N'' \oplus H(P_i)$  as cookie on its computer corresponding to the server  $AS_i$ . Here the value of  $N''$  is set equal to  $N'$ , when the ticket is initially issued to the user  $U_i$ .

### 4.2.3 Authentication phase

The user  $U_i$  extracts the value of  $N''$  from  $H(PS_i) \oplus N'' \oplus H(P_i)$ , which is stored as the cookie corresponding to  $AS_i$ . The user  $U_i$  uses the cookie information corresponding to  $AS_i$  to generate the message  $ID_i, C_i \oplus H(PS_i | N'')$ ,  $EXP\_TIME$  and sends it to  $AS_i$ .

Step 1:  $U_i \rightarrow AS_i: ID_i, C_i \oplus H(PS_i | N''), EXP\_TIME$

$AS_i$  extracts  $N''$ ,  $H(N' | EXP\_TIME)$ ,  $EXP\_TIME$  and  $PS_i \oplus Y_i \oplus SK_i$  corresponding to the user  $U_i$ 's identity  $ID_i$  and verifies the received  $EXP\_TIME$  with the extracted value of  $EXP\_TIME$  to check the validity of the ticket. Then  $AS_i$  computes  $H(N' | EXP\_TIME)$  from received value  $C_i \oplus H(PS_i | N'')$  as  $H(N' | EXP\_TIME) = C_i \oplus H(PS_i | N'') \oplus H(PS_i | N'') \oplus N''$  and then compares it with the extracted ticket information  $H(N' | EXP\_TIME)$  to check the authenticity of the user  $U_i$ .

After each successful authentication of the user  $U_i$  to  $AS_i$ ,  $AS_i$  updates the ticket information as  $N''^{new} = N'' + 1$  corresponding to the user  $U_i$ 's identity  $ID_i$ ,  $PS_i \oplus Y_i \oplus SK_i$ ,  $H(N' | EXP\_TIME)$  and  $EXP\_TIME$  in its database. Afterwards,  $AS_i$  computes  $F_i = H(N' | EXP\_TIME) \oplus H(N''^{new})$  and sends  $F_i$  to the user  $U_i$ .

Step 2:  $AS_i \rightarrow U_i: F_i$

Then the user  $U_i$  computes  $N''^{new} = N'' + 1$ ,  $F_i' = H(N' | EXP\_TIME) \oplus H(N''^{new})$  and verifies the computed value of  $F_i'$  with the received value of  $F_i$  to check the legitimacy of  $AS_i$ . After successful verification, the user  $U_i$  updates the value of  $N''$  to  $N''^{new}$  in its cookie information corresponding to the authentication server  $AS_i$ . Therefore, the user  $U_i$  updates its cookie information corresponding to  $AS_i$  as  $C_i^{new} = H(N' | EXP\_TIME) \oplus N''^{new}$ ,  $EXP\_TIME$ ,  $H(PS_i) \oplus N''^{new} \oplus H(P_i)$  after each successful login attempt by the user  $U_i$ . Finally, the user  $U_i$  and the server  $AS_i$  agree on common session key  $SS_i = H(PS_i | (N'' - 1))$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server  $AS_i$  are  $XOR^{ed}$  with this session key. Therefore, either the user  $U_i$  or the server  $AS_i$  can retrieve the original messages between themselves because both of them know this common session key.

#### 4.2.3.1 Ticket based re-authentication phase

The user  $U_i$  uses the same ticket corresponding to  $AS_i$  for its authentication to another authentication server  $AS_k$  by computing  $N''$  from  $H(PS_i) \oplus N'' \oplus H(P_i)$ ,  $H(N' | EXP\_TIME)$  from  $C_i$  as  $H(N' | EXP\_TIME) = C_i \oplus N''$ , computes  $B_k =$

$H(N' | EXP\_TIME) \oplus H(PS_k) \oplus H(P_i)$  and then sends  $ID_i, EXP\_TIME, B_k, T$  to  $AS_k$ , where  $T$  is current date and time of the user's computer.

Step 1:  $U_i \rightarrow AS_k: ID_i, EXP\_TIME, B_k, T$

After receiving the ticket request from the user  $U_i$ , service provider server  $AS_k$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $AS_k$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards,  $AS_k$  extracts  $PS_k$  from  $PS_k \oplus Y_k \oplus SK_k$  corresponding to the user's identity  $ID_i$  from its database. Then  $AS_k$  generates random nonce value  $N_k$ , computes  $A_k = B_k \oplus H(PS_k) \oplus H(SK_k | N_k) = H(N' | EXP\_TIME) \oplus H(P_i) \oplus H(SK_k | N_k)$  and  $W_k = N_k \oplus SK_k$ . Then  $AS_k$  sends  $ID_i, AS_k, EXP\_TIME, A_k, W_k$  and  $T$  to CS.

Step 2:  $AS_k \rightarrow CS: ID_i, AS_k, EXP\_TIME, A_k, W_k, T$

CS checks the validity of timestamp  $T$  by checking  $(T'' - T) \leq \delta T$ , where  $T''$  is current date and time of the server CS and  $\delta T$  is permissible time interval for a transmission delay. CS extracts  $H(N' | EXP\_TIME), EXP\_TIME, N' \oplus H(X | P_i)$  and  $P_i \oplus H(OTP | X)$  corresponding to the user  $U_i$ 's identity  $ID_i$  from its client's database and extracts  $SK_k$  from  $SK_k \oplus H(X | OTP)$  corresponding to authentication server's identity  $AS_k$  from its authentication server's database. Then CS verifies the received  $EXP\_TIME$  with extracted value of  $EXP\_TIME$  to check the validity of the ticket. Afterwards, CS computes  $P_i$  from  $P_i \oplus H(OTP | X)$ ,  $N_k = W_k \oplus SK_k$ ,  $N'$  from  $N' \oplus H(X | P_i)$  and then computes  $H(N' | EXP\_TIME) = A_k \oplus H(P_i) \oplus H(SK_k | N_k)$  and verifies it with the extracted ticket information  $H(N' | EXP\_TIME)$  to check the authenticity of the user  $U_i$ . After checking the legitimacy of the user  $U_i$ , CS computes  $C_i = H(N' | EXP\_TIME) \oplus N'$ ,  $M_k = N' \oplus H(N_k | SK_k)$  and sends  $C_i, EXP\_TIME$  and  $M_k$  to  $AS_k$ .

Step 3:  $CS \rightarrow AS_k: C_i, EXP\_TIME, M_k$

Then  $AS_k$  extracts  $N'$  from the received value of  $M_k$  as  $N' = M_k \oplus H(N_k | SK_k)$  and computes  $C_i' = H(N' | EXP\_TIME) \oplus N'$  and then compares the computed value of  $C_i'$  with received value of  $C_i$ . This equivalency authenticates the legitimacy of CS and authenticity of the received message. After successful authentication,  $AS_k$  stores  $N'', H(N' | EXP\_TIME)$  and  $EXP\_TIME$  corresponding to the user  $U_i$ 's identity  $ID_i$  and  $PS_k \oplus Y_k \oplus SK_k$  in its database. Here the value of  $N''$  is set equal to  $N'$ , when the ticket is initially issued to the user  $U_i$  through the server  $AS_k$ . Then the  $AS_k$  computes  $Q_i = N' \oplus H(PS_k), R_i = H(N' | PS_k)$  and sends  $Q_i, R_i$  and  $EXP\_TIME$  to the user  $U_i$ .

Step 4:  $AS_k \rightarrow U_i: Q_i, R_i, EXP\_TIME$

The user  $U_i$  knows  $PS_k$  and can extract  $N'$  from  $Q_i$  as  $N' = Q_i \oplus H(PS_k)$ , computes  $R_i' = H(N' | PS_k)$  and then verifies  $R_i'$  with the received value of  $R_i$  to check the legitimacy of the  $AS_k$ . After verification, the user  $U_i$  computes the ticket information  $C_i = H(N' | EXP\_TIME) \oplus N''$  and stores the ticket information  $C_i, EXP\_TIME, H(PS_k) \oplus N'' \oplus H(P_i)$  as cookie on its computer corresponding to the server  $AS_k$ . In this way, the user  $U_i$  uses a valid ticket meant for one authentication server  $AS_i$  to authenticate itself to another authentication server  $AS_k$ .

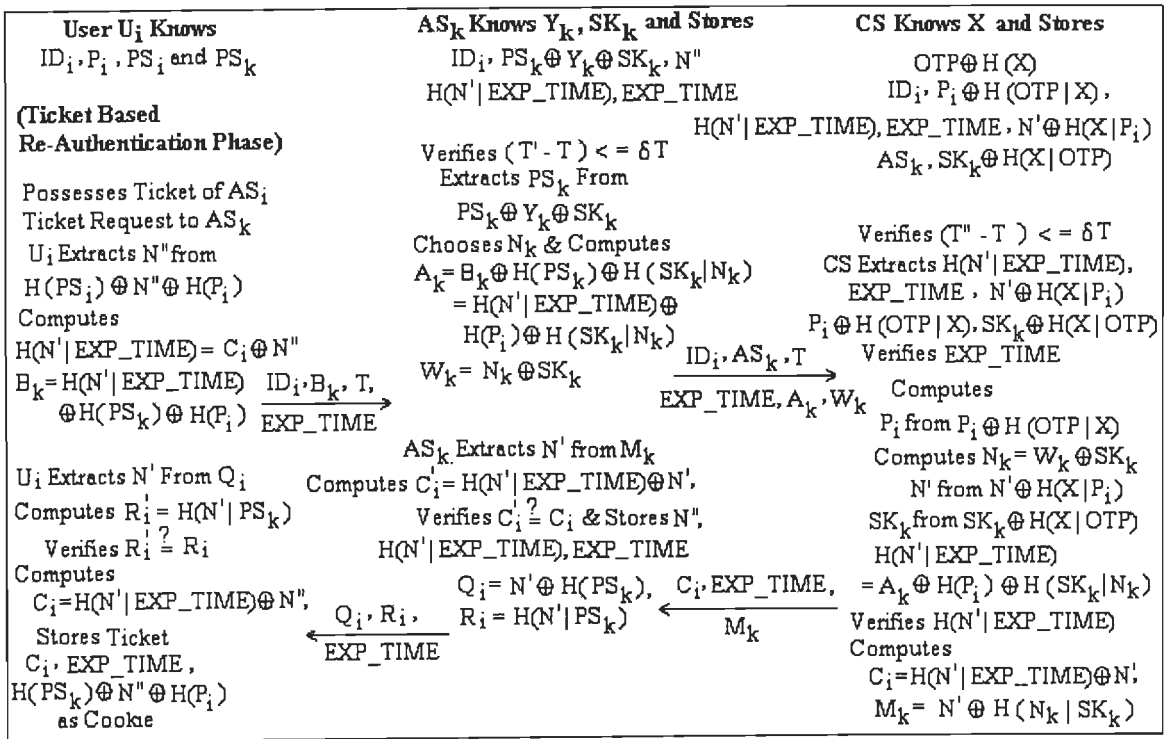


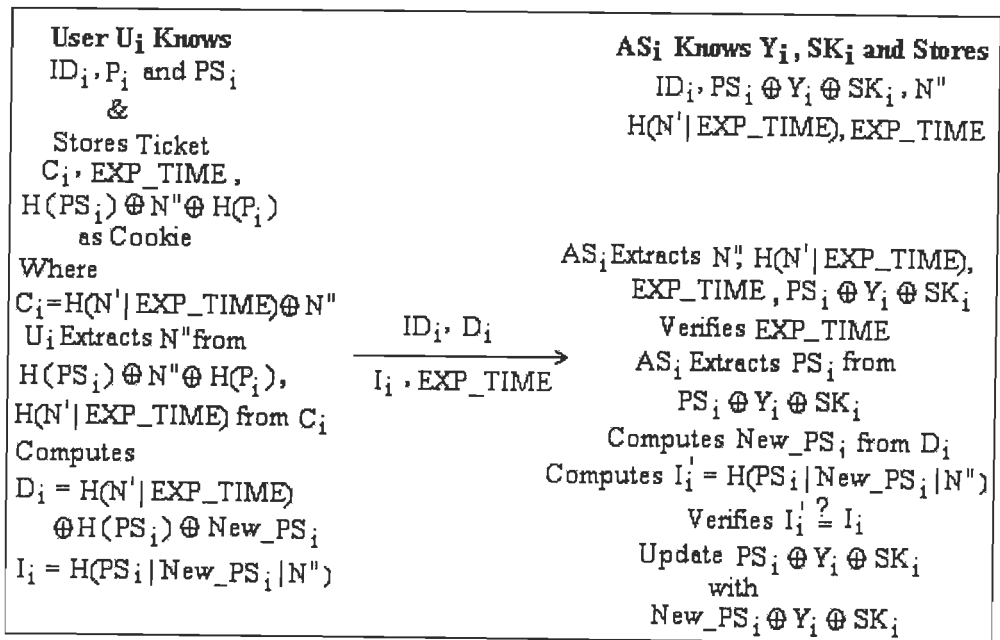
Figure 4.2: SSO password based two-server authentication protocol  
(Ticket based re-authentication phase)

#### 4.2.4 Password change phase

Protocol shown in Figure 4.3 is used by the user  $U_i$  to change its password with authentication server  $AS_i$ . The user  $U_i$  extracts  $N''$  from  $H(PS_i) \oplus N'' \oplus H(P_i)$  and then  $H(N' | EXP\_TIME)$  from valid ticket information stored on the user  $U_i$ 's computer corresponding to authentication server  $AS_i$  as  $H(N' | EXP\_TIME) = C_i \oplus N''$ . Afterwards, the user  $U_i$  computes  $D_i = H(N' | EXP\_TIME) \oplus H(PS_i) \oplus New\_PS_i$ ,  $I_i = H(PS_i | New\_PS_i | N'')$  and sends  $ID_i, D_i, I_i$  and  $EXP\_TIME$  to the corresponding



authentication server  $AS_i$ .  $AS_i$  extracts  $N''$ ,  $H(N' | EXP\_TIME)$ ,  $EXP\_TIME$  and  $PS_i \oplus Y_i \oplus SK_i$  corresponding to the user's identity  $ID_i$  and verifies the received  $EXP\_TIME$  with extracted value of  $EXP\_TIME$  to check the validity of the ticket. Then  $AS_i$  computes  $New\_PS_i$  from received value  $D_i$  as  $New\_PS_i = D_i \oplus H(N' | EXP\_TIME) \oplus H(PS_i)$ , computes  $I_i' = H(PS_i | New\_PS_i | N'')$  and compare the computed value of  $I_i'$  with the received value of  $I_i$  to check the legitimacy of the user  $U_i$ . Once the authenticity of user  $U_i$  is verified by authentication server  $AS_i$ , it updates the password  $PS_i$  of the user  $U_i$  with new modified password  $New\_PS_i$  by replacing  $PS_i \oplus Y_i \oplus SK_i$  corresponding to user  $U_i$ 's identity  $ID_i$  in its database with  $New\_PS_i \oplus Y_i \oplus SK_i$ .



**Figure 4.3: Authentication server's password change protocol**

Protocol shown in Figure 4.4 is used by the user  $U_i$  to change its password with CS. The user  $U_i$  extracts  $N''$  from  $H(PS_i) \oplus N'' \oplus H(P_i)$  and then  $H(N' | EXP\_TIME)$  from any valid ticket stored on the user  $U_i$ 's computer, which has been issued by CS corresponding to the user  $U_i$ 's identity  $ID_i$ . Then the user  $U_i$  computes  $E_i = H(N' | EXP\_TIME) \oplus H(P_i) \oplus New\_P_i$ ,  $L_i = H(New\_P_i | P_i)$  and sends  $ID_i, E_i, L_i$  and  $EXP\_TIME$  to CS. CS extracts  $H(N' | EXP\_TIME), EXP\_TIME, N' \oplus H(X | P_i)$  and  $P_i \oplus H(OTP | X)$  corresponding to the user  $U_i$ 's identity  $ID_i$ . CS verifies the received  $EXP\_TIME$  with extracted value of  $EXP\_TIME$  to check the validity of the ticket and computes  $P_i$  from  $P_i \oplus H(OTP | X)$ .

Then CS computes  $New\_P_i$  from received value  $E_i$  as  $New\_P_i = E_i \oplus H(N' | EXP\_TIME) \oplus H(P_i)$ , computes  $L_i' = H(New\_P_i | P_i)$  and compares the computed value of  $L_i'$  with the received value of  $L_i$  to check the legitimacy of the user  $U_i$ . Once the authenticity of the user  $U_i$  is verified by control server CS, it updates the password  $P_i$  of the user  $U_i$  with a new modified password  $New\_P_i$  by replacing  $P_i \oplus H(OTP | X)$  corresponding to the user  $U_i$ 's identity  $ID_i$  in its database with  $New\_P_i \oplus H(OTP | X)$ .

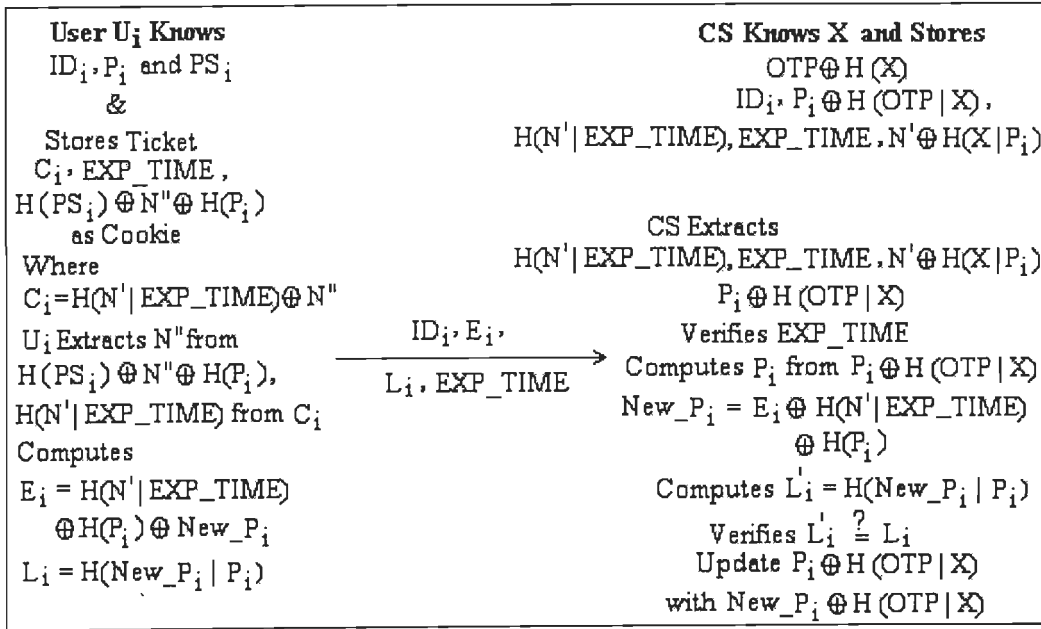


Figure 4.4: Control server's password change protocol

### 4.3 SECURITY ANALYSIS

The concept of two-tier authentication for the user makes it difficult for the attacker to guess out information pertaining to passwords and tickets. There may be a possibility that the ticket can be stolen by the attacker from the client's computer using plug-ins or other means. The dynamic nature of ticket makes it difficult for the attacker to find out any meaningful information from the eavesdropped messages. Moreover, the lifetime ( $EXP\_TIME$ ) of the ticket makes it valid for a limited time period as determined by the risk tolerance of the control server. A good password authentication scheme should have protection from different possible attacks relevant to that protocol.

- 1. Ticket theft:** The attacker can steal the ticket  $C_i = H(N' | EXP\_TIME) \oplus N''$ ,  $EXP\_TIME$  and  $H(PS_i) \oplus N'' \oplus H(P_i)$  from the client's computer using plug-ins or

other means. Then he tries to login on to corresponding authentication server  $AS_i$  or  $AS_k$  using this stolen ticket. Even after getting the valid ticket, the attacker has to guess at least two parameters out of  $PS_i$ ,  $PS_k$ ,  $N''$  and  $P_i$  correctly at the same time to compute the value of  $C_i \oplus H(PS_i | N'')$  or  $B_k = H(N' | EXP\_TIME) \oplus H(PS_k) \oplus H(P_i)$ . It is not possible to simultaneously guess two parameters correctly in real polynomial time. Moreover, the value of  $N''$  is changed to  $N'' + 1$  after each successful login attempt by the user  $U_i$  on to corresponding authentication server  $AS_i$  or  $AS_k$ . Therefore, the stolen ticket becomes useless if the user  $U_i$  login successfully on to corresponding authentication server  $AS_i$  or  $AS_k$  after ticket theft from the client's computer.

2. **Ticket poisoning attack:** The attacker may attempt to replace the valid ticket information  $C_i = H(N' | EXP\_TIME) \oplus N''$ ,  $EXP\_TIME$  and  $H(PS_i) \oplus N'' \oplus H(P_i)$  with some random values so that  $AS_i$  or user  $U_i$  stores wrong ticket information. In the proposed protocol, the server  $AS_i$  or the user  $U_i$  verifies the authenticity of received ticket information before updating the ticket information in their database. Since ticket information  $C_i = H(N' | EXP\_TIME) \oplus N''$  contains ticket verifier information as  $M_i = N' \oplus H(N_i | SK_i)$ ,  $EXP\_TIME$ ,  $C_i = H(N' | EXP\_TIME) \oplus N''$  from  $CS$  to  $AS_i$  and  $Q_i = N' \oplus H(PS_i)$ ,  $R_i = H(N' | PS_i)$  from  $AS_i$  to user  $U_i$ . Hence both  $AS_i$  and user  $U_i$  checks out the validity of ticket before storing or updating it in their databases. Therefore, the proposed protocol is free from ticket poisoning attack.
3. **Ticket independence:** The attacker can not compute a future ticket from any previous ticket related to the same user  $U_i$ . Any valid ticket is stored as  $C_i = H(N' | EXP\_TIME) \oplus N''$ ,  $EXP\_TIME$  and  $H(PS_i) \oplus N'' \oplus H(P_i)$  on the user's computer. Future ticket of the user  $U_i$  corresponding to same authentication server  $AS_i$  contains incremented value of  $N''$  as  $(N'' + 1)$  and hence the attacker can not compute future tickets from current tickets because the attacker requires to know at least two parameters out of  $PS_i$ ,  $N''$  and  $P_i$  correctly at the same time to generate the valid future tickets. It is not possible to simultaneously guess two parameters correctly in real polynomial time. Moreover, tickets of the same user for different authentication servers are different from each other.

4. **Online dictionary attack:** In this type of attack, the attacker pretends to be a legitimate user and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to get valid ticket corresponding to the authentication server  $AS_i$  or  $AS_k$ . Even after getting the valid ticket, the attacker has to guess at least two parameters out of  $PS_i$ ,  $PS_k$ ,  $N''$  and  $P_i$  correctly at the same time to compute the value of  $C_i \oplus H(PS_i | N'')$  or  $B_k = H(N' | EXP\_TIME) \oplus H(PS_k) \oplus H(P_i)$ . It is not possible to simultaneously guess two parameters correctly in real polynomial time. Moreover, the attacker can not compute password from messages generated from CS to  $AS_i$  and vice-versa. The attacker has to know  $PS_i$  and  $P_i$  correctly at the same time to request the fresh ticket from CS. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attacks.
5. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess user's passwords and ticket related information from recorded messages. The attacker first tries to obtain some password verification information such as  $H(PS_i) \oplus N'' \oplus H(P_i)$  or  $H(N' | EXP\_TIME) \oplus N'' \oplus H(PS_i | N'')$  from Figure 4.1 or  $B_k = H(N' | EXP\_TIME) \oplus H(PS_k) \oplus H(P_i)$  from Figure 4.2 or  $D_i = H(N' | EXP\_TIME) \oplus H(PS_i) \oplus New\_PS_i$  from Figure 4.3 or  $E_i = H(N' | EXP\_TIME) \oplus H(P_i) \oplus New\_P_i$  from Figure 4.4. Now the attacker has to guess at least two parameters correctly out of  $PS_i$ ,  $PS_k$ ,  $P_i$ ,  $N''$ ,  $N'$ ,  $New\_PS_i$  and  $New\_P_i$  at the same time. It is not possible to simultaneously guess two parameters correctly in real polynomial time. Also the value of  $N''$  is changed to  $(N'' + 1)$  after each successful authentication by the user  $U_i$  to  $AS_i$ . Moreover, the user  $U_i$ , the server  $AS_i$  and the server CS choose random nonce values as  $N$ ,  $N_i$  and  $N'$  during fresh ticket request. Therefore, the proposed protocol is secure against offline dictionary attack.
6. **Eavesdropping attack:** In this type of attack, the attacker listens to all the communication between the user and the server and then tries to find out  $PS_i$ ,  $P_i$ ,  $N''$  and  $N'$  values. The user  $U_i$  sends  $ID_i$ ,  $G_i = H(PS_i) \oplus N \oplus H(P_i)$ ,  $Z_i = H(N | T)$  or  $C_i \oplus H(PS_i | N'')$ ,  $EXP\_TIME$  to  $AS_i$  and  $AS_i$  sends  $ID_i$ ,  $AS_i$ ,  $A_i = N \oplus H(P_i) \oplus H(SK_i | N_i)$ ,  $Z_i$ ,  $T$  as password verification information to CS. An eavesdropper has

to guess at least two parameters correctly out of  $PS_i$ ,  $P_i$ ,  $N$ ,  $N''$  and  $SK_i$  at the same time. It is not possible to simultaneously guess two parameters correctly in real polynomial time in all four protocols shown in Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4. Therefore, the proposed protocol is secure against eavesdropping attack.

7. **Denial of service attack:** In denial of service attack, the server is cheated by the attacker to update false verification information for next login phase and hence legitimate user can not login successfully in subsequent login request to the server.  $AS_i$  updates the value of  $N''$  to  $(N'' + 1)$  in its database after authenticating the ticket information  $H(N' | EXP\_TIME)$  and password  $PS_i$  as shown in Figure 4.1. The legitimate user  $U_i$  can change the password with  $AS_i$  using the protocol shown in Figure 4.3 if the user  $U_i$  provides valid ticket information as  $H(N' | EXP\_TIME)$ , password  $PS_i$  and  $N''$  correctly at the same time. Also the legitimate user  $U_i$  can change the password with  $CS$  using the protocol shown in Figure 4.4 if the user  $U_i$  provides valid ticket information as  $H(N' | EXP\_TIME)$  and password  $P_i$  correctly at the same time. Therefore, the attacker can not launch denial of service attack on the proposed protocol.
8. **Man-in-the-middle attack:** In this type of attack, the attacker intercepts the messages sent between the client and the server and replay these intercepted messages with in the valid time frame window. An attacker can act as a client to the server or vice-versa with recorded messages. In the proposed protocol, the attacker can intercept the login request message  $ID_i$ ,  $G_i = H(PS_i) \oplus N \oplus H(P_i)$ ,  $EXP\_TIME$ ,  $Z_i = H(N | T)$ ,  $T$  or  $ID_i$ ,  $C_i \oplus H(PS_i | N_i)$ ,  $EXP\_TIME$  from the user  $U_i$  to its respective authentication server  $AS_i$ . Then he starts a new session with the server  $AS_i$  by sending a login request by replaying the login request message  $ID_i$ ,  $G_i = H(PS_i) \oplus N \oplus H(P_i)$ ,  $EXP\_TIME$ ,  $Z_i = H(N | T)$ ,  $T$  or  $ID_i$ ,  $C_i \oplus H(PS_i | N'')$ ,  $EXP\_TIME$ . The attacker can authenticate itself to server  $AS_i$  as well as to legitimate user  $U_i$  but can not compute the common session key  $SS_i = H(PS_i | (N'' - 1))$  because the attacker does not know the value of  $PS_i$  and  $N''$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
9. **Replay attack:** In this type of attack, the attacker first listens to communication between the client and the server and then tries to imitate the user to login on to the server by resending the captured messages transmitted between the client and

the server. Replaying the message  $ID_i$ ,  $G_i = H(PS_i) \oplus N \oplus H(P_i)$ ,  $EXP\_TIME$ ,  $Z_i = H(N | T)$ ,  $T$  or  $ID_i$ ,  $C_i \oplus H(PS_i | N_i)$ ,  $EXP\_TIME$  from the user  $U_i$  to its respective authentication server  $AS_i$  can be detected by  $AS_i$  because the value of  $N''$  is changed to  $(N'' + 1)$  after each successful login attempt by the user  $U_i$  to  $AS_i$ . Similarly, replayed message from  $AS_i$  to user  $U_i$  can also be detected by the user  $U_i$  because that message contains verifier information as  $F_i = H(N' | EXP\_TIME) \oplus H(N''^{new})$ . Moreover, the attacker can not compute the common session key  $SS_i = H(PS_i | (N'' - 1))$  because the attacker does not know the values of  $PS_i$  and  $N''$ . Therefore, the proposed protocol is secure against message replay attack.

**10. Leak of verifier attack:** In this type of attack, the attacker may be able to steal verification table from the server. In the proposed protocol, two servers are used and the attacker has to steal information from two servers simultaneously. Moreover, passwords are encrypted with the private keys ( $Y_i$  and  $SK_i$  or  $X$  and  $OTP$ ) of the servers. Also  $AS_i$  is updating the value of  $N''$  stored in its database to  $(N'' + 1)$  after each successful authentication by the user  $U_i$  to  $AS_i$ . In case verifier is stolen by breaking into the  $AS_i$ 's database, the attacker can not calculate the user's password  $PS_i$  because the user's password is stored as  $PS_i \oplus Y_i \oplus SK_i$  and the attacker does not know the private key  $Y_i$  and  $SK_i$  of the server  $AS_i$ . In case verifier is stolen by breaking into the server's CS database, the attacker can not calculate the user's password  $P_i$  because the user's password is stored as  $P_i \oplus H(OTP | X)$  and the attacker does not know the private key  $OTP$  and  $X$  of CS. Therefore, the proposed protocol is secure against leak of verifier attack.

**11. Brute force attack:** To launch brute force attack, the attacker first obtains some password verification information such as  $H(PS_i) \oplus N'' \oplus H(P_i)$  or  $H(N' | EXP\_TIME) \oplus N'' \oplus H(PS_i | N_i)$  from Figure 4.1 or  $B_k = H(N' | EXP\_TIME) \oplus H(PS_k) \oplus H(P_i)$  from Figure 4.2 or  $D_i = H(N' | EXP\_TIME) \oplus H(PS_i) \oplus New\_PS_i$  from Figure 4.3 or  $E_i = H(N' | EXP\_TIME) \oplus H(P_i) \oplus New\_P_i$  from Figure 4.4. Even after recording these messages, the attacker has to guess minimum two parameters correctly out of  $PS_i$ ,  $PS_k$ ,  $P_i$ ,  $N'$  and  $N''$  at the same time. It is not possible to simultaneously guess two parameters correctly in real polynomial time. Therefore, the proposed protocol is secure against brute force attack.

**12. Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on communication channel with the hope of discovering client's password or gaining unauthorized access. Modifying or inserting messages in proposed protocol can only cause authentication between the client and the server to fail but can not allow the attacker to gain any information about client's passwords  $PS_i$ ,  $P_i$  and ticket issued to that user. Therefore, the proposed protocol is secure against message modification or insertion attack.

#### 4.4 COST AND FUNCTIONALITY ANALYSIS

An efficient authentication scheme must take communication and computation cost into consideration during the user's authentication. To best of our knowledge, only Samar [123] suggested three general cookies based SSO approaches namely centralized cookie server, decentralized cookie server and centralized login server to provide SSO for web applications. Most of the two-server authentication protocols are based on smart cards [21][51][61][144]. The performance comparison of the proposed protocol with the smart card based two-server authentication schemes is summarized in Table 4.2. Assume that the  $ID_i$ ,  $PS_i$ ,  $P_i$ ,  $X$ ,  $Y_i$ ,  $OTP$  and  $SK_i$  are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$ ,  $T_S$  and  $T_X$  denote the time complexity for hash function, symmetric encryption and XOR operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_H \gg T_X$ . Time required for XOR operation is negligible as compared to hash operations and hence not considered in the table of comparison (Table 4.2). The computation cost of proposed protocol is computed after the valid ticket is stored as cookie on the user's computer. In the proposed protocol, the parameters stored in the cookie are  $C_i$ ,  $EXP\_TIME$ ,  $H(PS_i) \oplus N'' \oplus H(P_i)$  and the memory needed (E1) is 384 (= 3\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{ID_i, C_i \oplus H(PS_i | N_i), EXP\_TIME\}$  and  $\{F_i\}$  is 512 (= 4\*128) bits. The computation cost of the user (E3) and the service provider server (E4) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user is  $5T_H$  and that of the service provider server is  $3T_H$ . Table 4.2 shows that

the proposed protocol is computationally efficient as compared to the related two-server authentication protocols.

**Table 4.2**

**Cost and functionality comparison among related two-server authentication schemes**

	<b>Proposed Protocol</b>	<b>Chang &amp; Lee [21]</b>	<b>Hu et al. [51]</b>	<b>Juang [61]</b>	<b>Tsai [144]</b>
E1	384 bits	256 bits	384 bits	256 bits	128 bits
E2	4*128 bits	6*128 bits	8*128 bits	9*128 bits	13*128 bits
E3	5 $T_H$	4 $T_H$ + 3 $T_S$	6 $T_H$ + 1 $T_S$	3 $T_H$ + 3 $T_S$	6 $T_H$
E4	3 $T_H$	4 $T_H$ + 3 $T_S$	6 $T_H$ + 1 $T_S$	4 $T_H$ + 8 $T_S$	12 $T_H$

E1: Memory needed for the cookie or smart card.

E2: Communication cost of the authentication.

E3: Computation cost of the user.

E4: Computation cost of the service provider server.

## 4.5 CONCLUSION

Most of the password based authentication protocols are designed for single-server environment. Users have to login each time to access different services of the remote servers. To solve this problem, we proposed an efficient SSO password based two-server architecture in which the user has to login once to get a valid ticket. This valid ticket is used by the user for succeeding login attempts on same or different authentication server which is under the control of same control server. This time-bound ticket mechanism allows the legitimate user to login with less computational efforts in succeeding login attempts after getting the valid ticket from the authentication and the control server. The proposed protocol eliminates the main point of vulnerability as existing in most of the single-server password based authentication protocols. The control server is not much exposed to the users as well as attackers and hence less prone to attack. Therefore, this architecture increases the overall security of the system and resiliency to dictionary attack. The proposed protocol splits a password into two passwords and the user has to remember



one password for control server and different passwords for different authentication servers. This feature allows the users to have a strong password for CS and an easy to remember passwords for AS; and still have strong authentication because it is difficult for the attacker to guess both the passwords correctly at the same time. The proposed protocol has no compatibility problem with the single-server model as most of the existing password protocols use single server authentication architecture. Moreover, the proposed protocol does not use public key or symmetric key concept as is used in most of multi-server password based authentication protocols. The proposed protocol is an attempt to bridge the gap between single server and multi-server password based authentication protocols.

## STATIC IDENTITY BASED SMART CARD AUTHENTICATION PROTOCOLS

---

### 5.1 INTRODUCTION

Smart cards have been widely used in many e-commerce applications and network security protocols due to their low cost, portability, efficiency and the cryptographic properties. Smart card stores some sensitive data corresponding to the user that assist in user authentication. The user (card holder) inserts his smart card into a card reader machine and submits his identity and password. Then smart card and card reader machine perform some cryptographic operations using submitted arguments and the data stored inside the memory of smart card to verify the authenticity of the user.

In this chapter, a brief review of Yoon et al.'s scheme, Kim and Chung's scheme, Xu et al.'s scheme and Liu et al.'s scheme is given. Cryptanalysis of these schemes for different types of attacks is done and improved smart card based authentication protocols are proposed. The security analysis of the improved proposed protocols is presented. The cost and functionality comparison of the proposed protocols with the other related protocols is also presented.

### 5.2 REVIEW OF YOON ET AL.'S SCHEME

In this section, we examine a smart card based remote user authentication scheme proposed by Yoon et al. [170] in 2005. Yoon et al.'s scheme consists of four phases viz. registration phase, login phase, authentication phase and password change phase as summarized in Figure 5.1.

#### 1. Registration phase

The user  $U_i$  registers with the server  $S$  by submitting his identity  $ID_i$  and password  $P_i$  over a secure communication channel. The server  $S$  computes  $V_i = H (ID_i, T_{TSA}, x)$  and  $A_i = H (ID_i, T_{TSA}, x) \oplus P_i$ , where  $T_{TSA}$  is the trusted timestamp provided by a trusted time stamping authority,  $x$  is the secret key of the server  $S$ ,  $H ( )$  is a one-way hash function and

$\oplus$  represents the XOR operation. Then the server S issues the smart card containing security parameters ( $ID_i, V_i, A_i, H(\cdot)$ ) to the user  $U_i$  through a secure communication channel.

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server S and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card verifies the submitted  $ID_i^*$  with the stored value of  $ID_i$  in its memory. Then the smart card computes  $B_i = A_i \oplus P_i^*$  and verifies the computed value of  $B_i$  with the value of  $V_i$  stored in its memory. If both values match, the legitimacy of the user is assured and the smart card proceeds to the next step. Otherwise the login request from the user  $U_i$  is rejected. Afterwards, the smart card computes  $C_1 = H(B_i, T)$ , where T is current timestamp of user's smart card and sends the login request message ( $ID_i, C_1, T$ ) to the service provider server S.

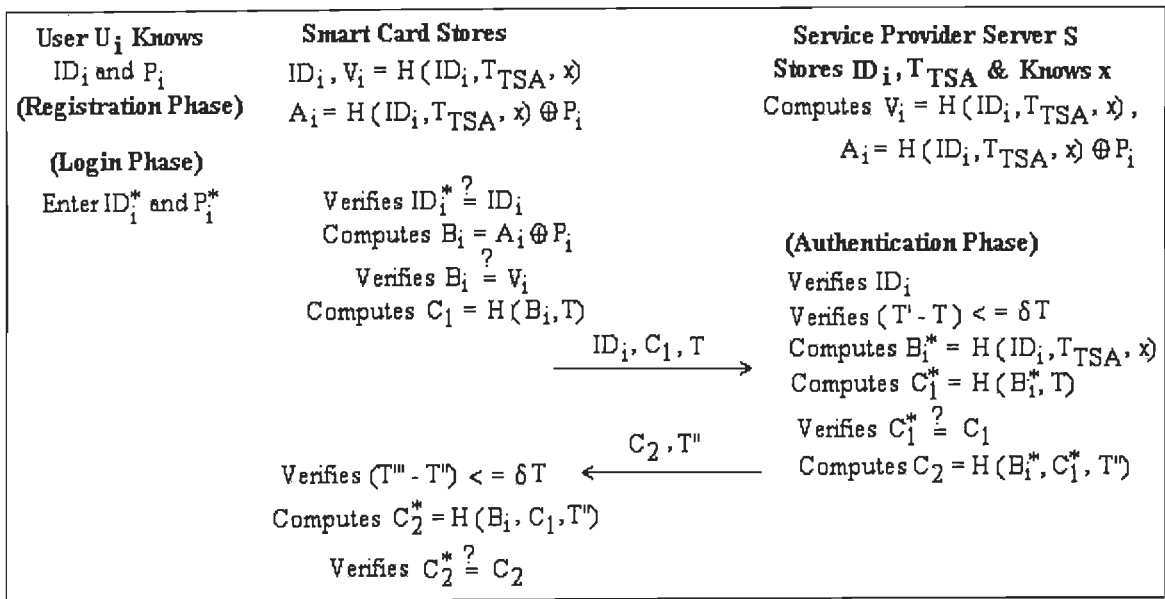


Figure 5.1: Yoon et al.'s scheme

## 3. Authentication phase

The service provider server S verifies the received  $ID_i$  with the stored value of  $ID_i$  in its database. Then the server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server S computes  $B_i^* = H(ID_i, T_{TSA}, x)$ ,  $C_1^* = H(B_i^*, T)$  and compares  $C_1^*$  with the received value of  $C_1$ . If they are not equal, the server

S rejects the login request and terminates this session. Otherwise the server S acquires the current timestamp  $T''$  and computes  $C_2 = H(B_i^*, C_1^*, T'')$  and sends the message  $(C_2, T'')$  back to the smart card of user  $U_i$ . On receiving the message  $(C_2, T'')$ , smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  denotes the current timestamp of user's smart card. Then the user  $U_i$ 's smart card computes  $C_2^* = H(B_i, C_1, T'')$  and compares it with received value of  $C_2$ . This equivalency authenticates the legitimacy of the service provider server S and the login request is accepted else the connection is interrupted.

#### 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader machine and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card verifies the submitted  $ID_i^*$  with the value of  $ID_i$  stored in its memory. Then smart card computes  $B_i = A_i \oplus P_i^* = H(ID_i, T_{TSA}, x)$  and compares the computed value of  $B_i$  with the stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of the card holder is verified, then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and  $A_i = H(ID_i, T_{TSA}, x) \oplus P_i$  stored in the smart card is replaced with  $A_i^{new} = H(ID_i, T_{TSA}, x) \oplus P_i^{new}$ .

##### 5.2.1 Cryptanalysis of Yoon et al.'s scheme

Yoon et al. [170] claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for stolen smart card attack, impersonation attack, parallel sessions attack and man-in-the-middle attack.

#### 1. Stolen smart card attack

A user may lose his smart card, which is found by the attacker or the attacker steals the user's smart card. The attacker can extract the stored values through some technique such as by monitoring their power consumption and reverse engineering techniques as pointed by Kocher et al. [67] and Messerges et al. [98]. As the smart card contains  $(ID_i, V_i, A_i, H(\ ))$ , suppose that the attacker is able to extract  $ID_i, V_i = H(ID_i, T_{TSA}, x)$  and  $A_i = H(ID_i, T_{TSA}, x) \oplus P_i$  from the memory of smart card. Then he can find the password  $P_i$  of the user  $U_i$  as  $P_i = V_i \oplus A_i$ . Now the attacker has the smart card, knows the identity  $ID_i$  and password  $P_i$  corresponding to the user  $U_i$  and hence can login on to the server S.

## 2. Impersonation attack

In the login phase of Yoon et al.'s scheme [170],  $B_i$  should be equal to the stored value of  $V_i$  in the smart card. That means the attacker need not know  $P_i$  to compute  $C_1$ , if the attacker has extracted  $V_i$  from the stolen smart card attack. Now the attacker can easily go through the steps in the login phase to forge a valid login request message as  $\{ID_i, C_1, T\}$ , where  $T$  is a current timestamp and  $C_1 = H(B_i, T) = H(V_i, T)$ . Therefore, the attacker can successfully make a valid login request and impersonates as a legitimate user  $U_i$ .

## 3. Parallel sessions attack

This attack can be launched if the server permits parallel sessions for a user. The attacker can masquerade as a legitimate user  $U_i$  by creating a valid login message from the eavesdropped communication between the user and the server without knowing the user's password. The attacker can intercept and record the login request message  $(ID_i, C_1, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i, C_1, T)$  within the valid time frame window. After receiving the login request, the server  $S$  checks the validity of  $ID_i$  and the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp. The server  $S$  computes  $B_i^* = H(ID_i, T_{TSA}, x)$ ,  $C_1^* = H(B_i^*, T)$  and compares  $C_1^*$  with received value of  $C_1$  to check the legitimacy of the user  $U_i$ . This equivalency authenticates the masquerading user.

## 4. Man-in-the-middle attack

In this type of attack, the attacker intercepts the messages sent between the user and the server and replays these intercepted messages within the valid time frame window. The attacker can act as a legitimate user to the server or vice-versa with recorded messages. He can intercept the login request  $(ID_i, C_1, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i, C_1, T)$  within the valid time frame window. After receiving the login request, the server  $S$  verifies the received  $ID_i$  with the stored value of  $ID_i$  in its database. Then the server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp. The server  $S$  computes  $B_i^* = H(ID_i, T_{TSA}, x)$ ,  $C_1^* = H(B_i^*, T)$  and compares  $C_1^*$  with the received value of  $C_1$  to check the legitimacy of the user  $U_i$ . This equivalency authenticates the masquerading user. The attacker can also

intercept the response message  $(C_2, T')$ , which is sent by the server  $S$  to the user  $U_i$ . Using this message, the attacker can masquerade as a legitimate server  $S$  to the legitimate user  $U_i$  by replaying this message within the valid time frame window. Now the attacker can act as a middle man and masquerades as legitimate user  $U_i$  to legitimate server  $S$  and vice-versa.

### 5.2.2 Proposed protocol

In this section, we describe a modified smart card based remote user authentication protocol which resolves the above security flaws of Yoon et al.'s [170] scheme. Figure 5.2 shows the entire protocol structure of the proposed authentication protocol. Legitimate user can easily login on to the service provider server using his smart card, identity and password.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  via a secure communication channel to register itself to the server  $S$ .

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  computes the security parameters  $V_i = H(ID_i | T_{TSA}) \oplus x \oplus H(P_i)$ ,  $A_i = H(ID_i | P_i) \oplus P_i$  and  $B_i = H(P_i) \oplus H(T_{TSA})$ , where  $|$  represents concatenation operation. Then the server  $S$  issues the smart card containing security parameters  $(V_i, A_i, B_i, H(\cdot))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i: \text{Smart card}$

#### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $A_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares the computed value of  $A_i^*$  with the stored value of  $A_i$  in its memory to verify the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $A_i^* \stackrel{?}{=} A_i$

After verification, smart card computes  $C_1 = V_i \oplus H(P_i)$  and  $C_2 = H(C_1 | T)$ , where  $T$  is current timestamp of user's smart card. Then smart card sends the login request message  $(ID_i^*, C_2, T)$  to the service provider server  $S$ .

Step 2: Smart card  $\rightarrow$  S:  $ID_i^*$ ,  $C_2$ , T

The user  $U_i$  extracts the value of  $H(T_{TSA})$  as  $H(T_{TSA}) = B_i \oplus H(P_i)$ , which is used by the user  $U_i$  for the computation of the agreed session key between the user  $U_i$  and the server S.

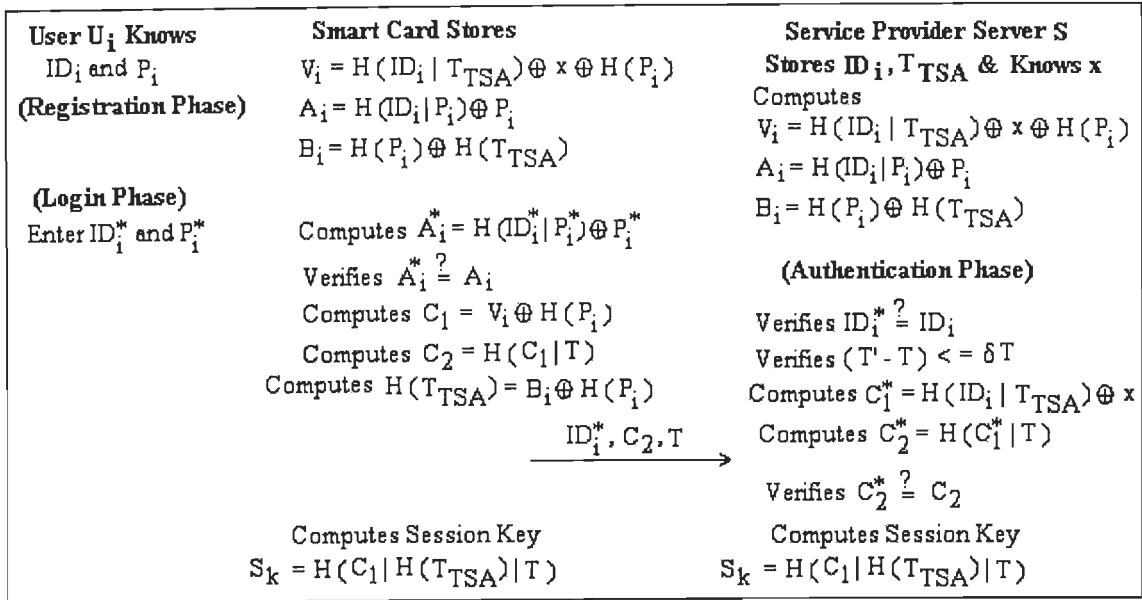


Figure 5.2: Proposed improvement in Yoon et al.'s scheme

### 3. Authentication phase

After receiving the login request from the user  $U_i$ , the service provider server S verifies the received  $ID_i^*$  with the stored value of  $ID_i$  in its database. The server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server S and  $\delta T$  is permissible time interval for a transmission delay. The server S extracts the value of  $T_{TSA}$  corresponding to the user  $U_i$ 's identity  $ID_i$ . Then server S computes  $C_1^* = H(ID_i | T_{TSA}) \oplus x$ ,  $C_2^* = H(C_1^* | T)$  and compares  $C_2^*$  with the received value of  $C_2$ .

Step 1: Server S checks  $C_2^* \stackrel{?}{=} C_2$

This equivalency authenticates the legitimacy of user  $U_i$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S agree on the common session key as  $S_k = H(C_1 | H(T_{TSA}) | T)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server S can retrieve the original message because both of them know the common session key.

#### 4. Password change phase

The user  $U_i$  can change his password without the help of the server  $S$ . The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $A_i^* = H (ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $A_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then  $V_i = H (ID_i | T_{TSA}) \oplus x \oplus H (P_i)$ ,  $A_i = H (ID_i | P_i) \oplus P_i$  and  $B_i = H (P_i) \oplus H (T_{TSA})$  stored in the smart card can be replaced with  $V_i^{new} = V_i \oplus H (P_i) \oplus H (P_i^{new})$ ,  $A_i^{new} = H (ID_i | P_i^{new}) \oplus P_i^{new}$  and  $B_i^{new} = B_i \oplus H (P_i) \oplus H (P_i^{new})$ .

#### 5.2.3 Security analysis

Smart card is a memory card that uses an embedded micro-processor from smart card reader machine to perform the required operations specified in the protocol. Kocher et al. [67] and Messerges et al. [98] have pointed out that present smart cards can not prevent the information stored in them from being extracted such as by monitoring their power consumption. Some other reverse engineering techniques are also available for extracting information from smart cards. This means that once a smart card is stolen by an attacker, he can extract the information stored in it. A good password authentication protocol should provide protection from different feasible attacks.

1. **Stolen smart card attack:** In case a user's smart card is stolen by the attacker, he can extract the information stored in its memory. The attacker can extract  $V_i = H (ID_i | T_{TSA}) \oplus x \oplus H (P_i)$ ,  $A_i = H (ID_i | P_i) \oplus P_i$  and  $B_i = H (P_i) \oplus H (T_{TSA})$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
2. **Impersonation attack:** In this type of attack, the attacker impersonates as a legitimate user and forges the authentication messages using the information obtained from the authentication protocol. The attacker can attempt to modify a login request message  $(ID_i^*, C_2, T)$  into  $(ID_i^*, C_2^*, T^*)$  so as to succeed in the authentication phase, where  $T^*$



is the attacker's current date and time. However, such a modification will fail in Step 1 of the authentication phase because the attacker has no way of obtaining the value of  $C_1 = H (ID_i^* | T_{TSA}) \oplus x$  to compute the valid parameter  $C_2^*$ . Moreover, the attacker can not compute the agreed session key  $S_k = H (C_1 | H (T_{TSA}) | T)$  between the user  $U_i$  and the server  $S$ . Therefore, the proposed protocol is secure against impersonation attack.

3. **Parallel session attack:** In this type of attack, the attacker first listens to communication between the user and the server. After that, he initiates a parallel session to imitate legitimate user to login on to the server by resending the captured messages transmitted between the user and the server with in the valid time frame window. He can masquerade as a legitimate user by replaying a login request message  $(ID_i^*, C_2, T)$  with in the valid time frame window. The attacker can not compute the agreed session key  $S_k = H (C_1 | H (T_{TSA}) | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $C_1$  and  $H (T_{TSA})$ . Therefore, the proposed protocol is secure against parallel session attack.
4. **Man-in-the-middle attack:** In this type of attack, the attacker intercepts the messages sent between the user and the server and replay these intercepted messages with in the valid time frame window. The attacker can act as the user to the server or vice-versa with recorded messages. In the proposed protocol, the attacker can intercept the login request message  $(ID_i^*, C_2, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i^*, C_2, T)$  with in the valid time frame window. The attacker can authenticate itself to the server  $S$  but can not compute the agreed session key  $S_k = H (C_1 | H (T_{TSA}) | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $C_1$  and  $H (T_{TSA})$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
5. **Replay attack:** In this type of attack, the attacker first listens to communication between the user and the server and then tries to imitate user to login on to the server by resending the captured messages transmitted between the user and the server. Replaying a login request message  $(ID_i^*, C_2, T)$  of one session into another session is useless because the user  $U_i$ 's smart card uses current timestamp value  $T$  in each new session, which makes all the messages dynamic and valid for small interval of time.

Old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.

6. **Leak of verifier attack:** In this type of attack, the attacker may be able to steal verification table from the server. If the attacker steals the verification table from the server, he can use the stolen verifiers to impersonate a participant of the authentication protocol. In the proposed protocol, the service provider server S knows secret  $x$  and only stores  $T_{TSA}$  corresponding to user  $U_i$ 's identity  $ID_i$  in its database. The attacker does not have any technique to find the value of  $x$ . In case verifier is stolen by breaking into smart card database, the attacker does not have sufficient information to calculate user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
7. **Server spoofing attack:** In server spoofing attack, the attacker can manipulate the sensitive data of legitimate users via setting up fake servers. In the proposed protocol, malicious server can not compute the session key  $S_k = H(C_1 | H(T_{TSA}) | T)$  because the malicious server does not know the values of  $C_1$  and  $H(T_{TSA})$ . Moreover, the session key is different for the same user in different login sessions. Therefore, the proposed protocol is secure against server spoofing attack.
8. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $V_i = H(ID_i | T_{TSA}) \oplus x \oplus H(P_i)$ ,  $A_i = H(ID_i | P_i) \oplus P_i$  and  $B_i = H(P_i) \oplus H(T_{TSA})$  from the memory of smart card. This malicious user can not compute the value of  $x$  or  $T_{TSA}$  from these parameters even if he knows the values of  $ID_i$  and  $P_i$ . Moreover, the value of  $T_{TSA}$  is unique corresponding to different users. Also, the malicious user can not generate smart card specific value of  $C_2 = H(C_1 | T)$  to masquerade as other legitimate user  $U_k$  to the service provider server S because the value of  $C_1$  is smart card specific and depends upon the values of  $ID_k$ ,  $x$  and  $T'_{TSA}$ . Therefore, the proposed protocol is secure against malicious user attack.
9. **Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the user's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can only cause authentication between the user and the server to fail but can not allow the attacker to gain any information about

the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

**10. Online dictionary attack:** In this type of attack, the attacker pretends to be the legitimate user and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to get the valid smart card and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to user  $U_i$ . Even after getting the valid smart card of user  $U_i$  by any means, the attacker gets a very few chances (normally a maximum of 3) to guess the identity  $ID_i$  and password  $P_i$  because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

**11. Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempt to guess the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  from recorded messages. The attacker first tries to obtain the user  $U_i$  verification information  $T$ ,  $C_2 = H((H(ID_i | T_{TSA}) \oplus x) | T)$  and then tries to guess the  $ID_i$ ,  $T_{TSA}$  and  $x$  by offline guessing. Even after gathering this information, the attacker has to guess all three parameters  $ID_i$ ,  $x$  and  $T_{TSA}$  correctly at the same time. It is not possible to guess all three parameters correctly at the same time in real polynomial time. In another option, the attacker requires valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.

#### 5.2.4 Cost and functionality analysis

An efficient authentication scheme must take communication and computation cost into consideration during user's authentication. The cost and functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 5.1 and Table 5.2. Assume that the identity  $ID_i$ , password  $P_i$  and timestamp value are all 128-bit long. Moreover, we assume that the output of the secure one-way hash function is 128-bit. Let  $T_H$  and  $T_X$  denote the time complexity for hash function and XOR operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_H \gg T_X$ .

Table 5.1

Cost comparison among related smart card based authentication schemes

	Proposed Protocol	Kim-Chung [65]	Hsiang-Shih [48]	Yoon-Ryu-Yoo [170]	Chein et al. [25]	Hwang-Lee-Tang [53]
E1	384 bits	384 bits	384 bits	384 bits	128 bits	256 bits
E2	3*128 bits	6 *128 bits	5 *128 bits	5 *128 bits	5 *128 bits	3*128 bits
E3	$4T_H + 4T_X$	$4T_H + 6T_X$	$4T_H + 4T_X$	$1T_H + 1T_X$	$1T_H + 2T_X$	$2T_H + 2T_X$
E4	$4T_H + 3T_X$	$4T_H + 6T_X$	$4T_H + 4T_X$	$2T_H + 1T_X$	$2T_H + 3T_X$	$2T_H + 2T_X$
E5	$4T_H + 1T_X$	$4T_H + 6T_X$	$4T_H + 3T_X$	$3T_H + 0T_X$	$3T_H + 3T_X$	$2T_H + 2T_X$

In the proposed protocol, the parameters stored in the smart card are  $V_i, A_i, B_i$  and the memory needed in the smart card (E1) is 384 (= 3\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{ID_i, C_2, T\}$  is 384 (= 3\*128) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration is  $4T_H + 4T_X$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user and that of the service provider server are  $4T_H + 3T_X$  and  $4T_H + 1T_X$  respectively. The proposed protocol requires less computation cost than that of latest schemes proposed by Kim and Chung [65] and Hsiang and Shih [48] and is secure against different possible attacks launched by the attacker.

Table 5.2

Functionality comparison among related smart card based authentication schemes

	Proposed Protocol	Kim-Chung [65]	Hsiang-Shih [48]	Yoon-Ryu-Yoo [170]	Chein et al. [25]	Hwang-Lee-Tang [53]
Stolen smart card attack	No	Yes	No	Yes	Yes	Yes
Impersonation attack	No	Yes	Yes	Yes	Yes	No
Parallel session attack	No	Yes	Yes	Yes	Yes	Yes
Man-in-the-middle attack	No	Yes	Yes	Yes	Yes	Yes
Session key agreement	Yes	No	No	No	No	No

### 5.3 REVIEW OF KIM AND CHUNG'S SCHEME

In this section, we examine a smart card based authentication scheme proposed by Kim and Chung [65] in 2009. Kim and Chung's scheme consists of four phases viz. registration phase, login phase, verification phase and password change phase as summarized in Figure 5.3.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  for registration over a secure communication channel. The server  $S$  computes  $K_1 = H(ID_i \oplus x) \oplus N$ ,  $K_2 = H(ID_i \oplus x \oplus N) \oplus H(P_i \oplus H(P_i))$  and  $R = K_1 \oplus H(P_i)$ , where  $N$  is a random unique number corresponding to the user  $U_i$  and  $x$  is the secret key of the server  $S$ . Then the server  $S$  issues the smart card containing secret parameters  $(K_1, K_2, R, H(\cdot))$  to the user  $U_i$  through a secure communication channel.

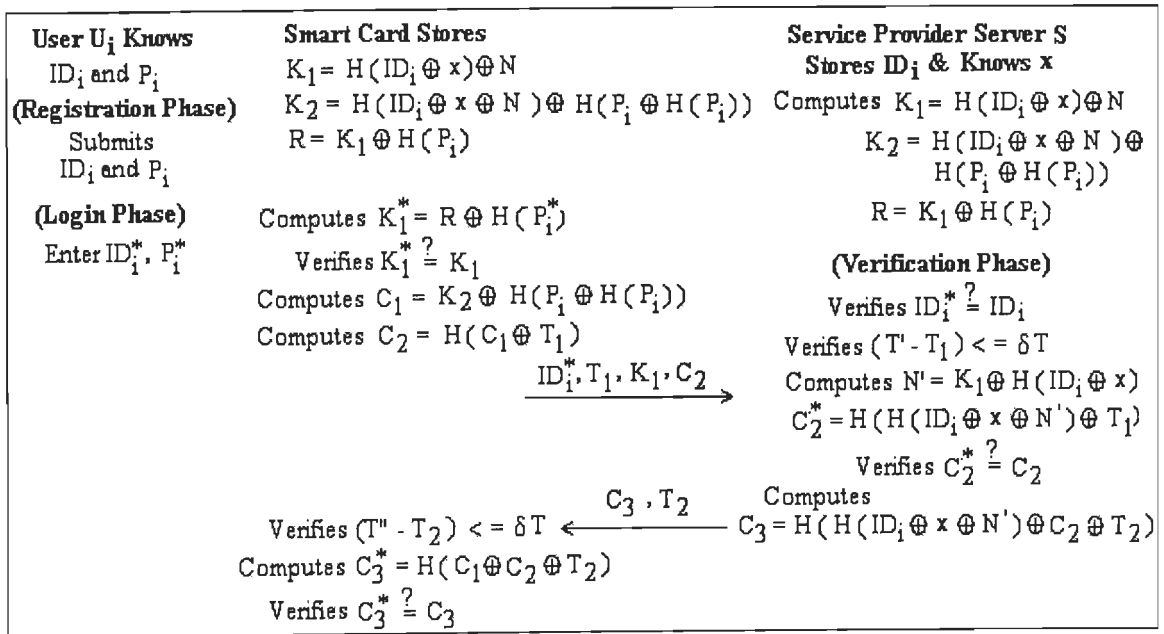


Figure 5.3: Kim and Chung's scheme

#### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $K_1^* = R \oplus H(P_i^*)$  and verifies the computed value of  $K_1^*$  with the stored value of  $K_1$  in its memory. If both values match, the legitimacy of the user is assured and smart card proceeds to the next step. Otherwise,

the login request from the user  $U_i$  is rejected. Afterwards, smart card computes  $C_1 = K_2 \oplus H (P_i \oplus H (P_i))$ ,  $C_2 = H (C_1 \oplus T_1)$ , where  $T_1$  is current timestamp of user's smart card and sends the login request message  $(ID_i^*, T_1, K_1, C_2)$  to the service provider server  $S$ .

### 3. Verification phase

After receiving the login request from the user  $U_i$ , the service provider server  $S$  verifies the received value of  $ID_i^*$  with stored value of  $ID_i$  in its database. Then the server  $S$  checks the validity of timestamp  $T_1$  by checking  $(T' - T_1) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes  $N' = K_1 \oplus H (ID_i \oplus x)$ ,  $C_2^* = H (H (ID_i \oplus x \oplus N') \oplus T_1)$  and checks if  $C_2^*$  is equal to the received value of  $C_2$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  acquires the current timestamp  $T_2$ , computes  $C_3 = H (H (ID_i \oplus x \oplus N') \oplus C_2 \oplus T_2)$  and sends the message  $(C_3, T_2)$  back to the smart card of user  $U_i$ . On receiving the message  $(C_3, T_2)$ , smart card checks the validity of timestamp  $T_2$  by checking  $(T'' - T_2) \leq \delta T$ , where  $T''$  denotes the user's smart card current timestamp. Then the user's smart card computes  $C_3^* = H (C_1 \oplus C_2 \oplus T_2)$  and compares the computed value of  $C_3^*$  with received value of  $C_3$ . This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted.

### 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $K_1^* = R \oplus H (P_i^*)$  and compares the computed value of  $K_1^*$  with the stored value of  $K_1$  in its memory to verifies the legitimacy of the user  $U_i$ . Once the authenticity of the card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and smart card computes  $R^{new} = K_1 \oplus H (P_i^{new})$  and  $K_2^{new} = K_2 \oplus H (P_i \oplus H (P_i)) \oplus H (P_i^{new} \oplus H (P_i^{new}))$ . Finally, the value of  $R$  and  $K_2$  stored in the smart card is replaced with  $R^{new}$  and  $K_2^{new}$  and password gets changed.

### 5.3.1 Cryptanalysis of Kim and Chung's scheme

Kim and Chung claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for masquerading user attack, masquerading server attack and offline dictionary attack using stolen smart card. Their protocol is also found to be susceptible to parallel session attack.

#### 1. Masquerading user attack

The attacker can intercept a valid login request message  $(ID_i^*, T_1, K_1, C_2)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_2 = H(C_1 \oplus T)$  to know the value of  $C_1$ , which is always same corresponding to the user  $U_i$ . After getting the value of  $C_1$ , the attacker can frame and send the fabricated valid login request message  $(ID_i^*, T_u, K_1, C_2^*)$  to the service provider server  $S$  without knowing the password  $P_i$  of the user  $U_i$ , where  $T_u$  is a current timestamp and  $C_2^* = H(C_1 \oplus T_u)$ . Therefore, the attacker can successfully make a valid login request by masquerading as a legitimate user  $U_i$  to the service provider server  $S$ .

#### 2. Masquerading server attack

The attacker can intercept a valid login request message  $(ID_i^*, T_1, K_1, C_2)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_2 = H(C_1 \oplus T)$  to know the value of  $C_1$ . After getting the value of  $C_1$ , the attacker can frame and send the fabricated valid response message  $(C_3, T_2)$  back to the smart card of user  $U_i$ , where  $T_2$  is a current timestamp and  $C_3 = H(C_1 \oplus C_2 \oplus T_2)$ . Therefore, the attacker can successfully masquerade as the service provider server  $S$  to the legitimate user  $U_i$ .

#### 3. Offline dictionary attack

A user  $U_i$  may lose his smart card, which is found by an attacker or an attacker steals the user  $U_i$ 's smart card. An attacker can extract the stored values through some technique like by monitoring their power consumption and reverse engineering techniques as pointed out by Kocher et al. [67] and Messerges et al. [98]. He can extract  $K_1 = H(ID_i \oplus x) \oplus N$ ,  $K_2 = H(ID_i \oplus x \oplus N) \oplus H(P_i \oplus H(P_i))$  and  $R = K_1 \oplus H(P_i)$  from the memory of smart card because smart card contains  $(K_1, K_2, R, H(\ ))$ . Then an attacker can find out the password information  $H(P_i)$  of user  $U_i$  as  $H(P_i) = R \oplus K_1$ . Now an attacker can guess

different values of  $P_i$  and checks its correctness by verifying it with the actual value of  $H(P_i)$ .

#### 4. Parallel session attack

The attacker can masquerade as a legitimate user by creating a valid login message from the eavesdropped communication between the user and the server without knowing the user's password. He can intercept the login request message  $(ID_i^*, T_1, K_1, C_2)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i^*, T_1, K_1, C_2)$  within the valid time frame window. After receiving the login request, the server  $S$  checks the validity of  $ID_i^*$  and timestamp  $T_1$  by checking  $(T' - T_1) \leq \delta T$ , where  $T'$  denotes the server's current timestamp. The server  $S$  computes  $N' = K_1 \oplus H(ID_i \oplus x)$ ,  $C_2^* = H(H(ID_i \oplus x \oplus N') \oplus T_1)$  and compares the computed value of  $C_2^*$  with the received value of  $C_2$  to check the legitimacy of the user  $U_i$ . This equivalency authenticates the masquerading user. Therefore, the attacker can successfully make a valid login request by masquerading as a legitimate user  $U_i$  to the service provider server  $S$ .

#### 5.3.2 Proposed protocol

In this section, we describe a modified smart card based remote user authentication protocol which resolves the above security flaws of Kim and Chung's [65] scheme. The proposed protocol precludes the weaknesses of Kim and Chung's scheme with improved security. Figure 5.4 shows the entire protocol structure of the new authentication protocol. Legitimate user  $U_i$  can easily login on to the service provider server using his smart card, identity and password. The proposed protocol consists of four phases viz. registration phase, login phase, verification and session key agreement phase and password change phase.

##### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  via a secure communication channel to register itself to the server  $S$ .

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  computes the security parameters  $K_1 = H(ID_i \oplus x) \oplus N$ ,  $K_2 = H(ID_i \oplus x \oplus N) \oplus H(P_i)$ ,  $K_3 = H(ID_i | x | N) \oplus H(ID_i | P_i)$  and  $R = K_1 \oplus H(ID_i | P_i) \oplus H(P_i)$ , where  $N$



is a random number unique to the user  $U_i$ . Then the server  $S$  issues the smart card containing security parameters  $(K_1, K_2, K_3, R, H ( ))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i$ : Smart card

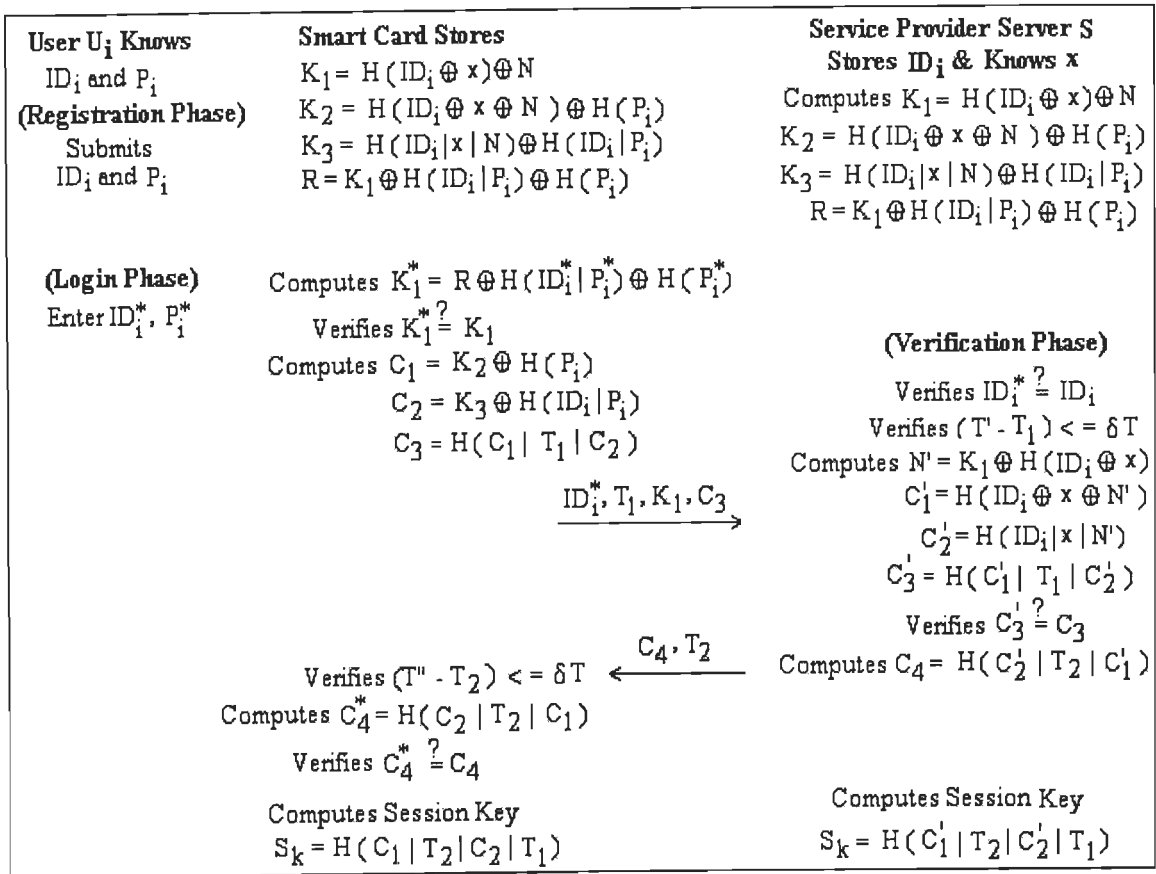


Figure 5.4: Proposed improvement in Kim and Chung's scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $K_1^* = R \oplus H (ID_i^* | P_i^*) \oplus H (P_i^*)$  and verifies the computed value of  $K_1^*$  with the stored value of  $K_1$  in its memory to verifies the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $K_1^* \stackrel{?}{=} K_1$

After verification, smart card computes  $C_1 = K_2 \oplus H (P_i)$ ,  $C_2 = K_3 \oplus H (ID_i | P_i)$  and  $C_3 = H (C_1 | T_1 | C_2)$ , where  $T_1$  is the user's smart card current timestamp. Then smart card sends the login request message  $(ID_i^*, T_1, K_1, C_3)$  to the service provider server  $S$ .

Step 2: Smart card  $\rightarrow S$ :  $ID_i^*, T_1, K_1, C_3$

### 3. Verification and session key agreement phase

After receiving the login request from the user  $U_i$ , the service provider server  $S$  verifies the received value of  $ID_i^*$  with stored value of  $ID_i$  in its database. The server  $S$  checks the validity of timestamp  $T_1$  by checking  $(T' - T_1) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes  $N' = K_1 \oplus H(ID_i \oplus x)$ ,  $C_1' = H(ID_i \oplus x \oplus N')$ ,  $C_2' = H(ID_i | x | N')$ ,  $C_3' = H(C_1' | T_1 | C_2')$  and checks if  $C_3'$  is equal to the received value of  $C_3$ .

Step 1: Server  $S$  checks  $C_3' \stackrel{?}{=} C_3$

If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  acquires the current timestamp  $T_2$  and computes  $C_4 = H(C_2' | T_2 | C_1')$  and sends the message  $(C_4, T_2)$  back to the smart card of user  $U_i$ .

Step 2:  $S \rightarrow$  Smart card:  $C_4, T_2$

On receiving the message  $(C_4, T_2)$ , the user  $U_i$ 's smart card checks the validity of timestamp  $T_2$  by checking  $(T'' - T_2) \leq \delta T$ , where  $T''$  is the user's smart card current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the user  $U_i$ 's smart card computes  $C_4^* = H(C_2 | T_2 | C_1)$  and compares the computed value of  $C_4^*$  with the received value of  $C_4$ .

Step 3: Smart card checks  $C_4^* \stackrel{?}{=} C_4$

This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server  $S$  agree on the common session key as  $S_k = H(C_1 | T_2 | C_2 | T_1)$ .

### 4. Password change phase

The user  $U_i$  can change his password without help of the server  $S$ . The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. Smart card computes  $K_1^* = R \oplus H(ID_i^* | P_i^*) \oplus H(P_i^*)$  and compares the computed value of  $K_1^*$  with the stored value of  $K_1$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of the card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $R^{new} = K_1 \oplus H(ID_i | P_i^{new}) \oplus H(P_i^{new})$ ,  $K_2^{new} = K_2 \oplus H(P_i) \oplus H(P_i^{new})$  and  $K_3^{new} = K_3 \oplus H(ID_i | P_i) \oplus H(ID_i | P_i^{new})$ . Afterwards, smart card replaces the values of  $R$ ,  $K_2$  and  $K_3$

stored in its memory with  $R^{new}$ ,  $K_2^{new}$  and  $K_3^{new}$ . The proposed protocol checks the validity of identity  $ID_i^*$  and password  $P_i^*$  before updating the password of the user  $U_i$ . In Kim and Chung's [65] scheme, smart card verifies only the password  $P_i$  of the user  $U_i$  and updates the password without verifying the identity  $ID_i$  of the user  $U_i$ .

### 5.3.3 Security analysis

A good password authentication protocol should provide protection from different possible attacks relevant to that protocol.

1. **Masquerading user attack:** In this type of attack, the attacker masquerades as a legitimate user and forges the authentication messages using the information obtained from the authentication scheme. The attacker can intercept a valid login request message  $(ID_i^*, T_1, K_1, C_3)$  of the user  $U_i$  from the public communication channel. The attacker has to guess the correct values of  $C_1$  and  $C_2$  at the same time to launch offline dictionary attack on  $C_3 = H(C_1 | T_1 | C_2)$  because the attacker requires correct values of  $C_1$  and  $C_2$  to frame fabricated valid login request message  $(ID_i^*, T_1', K_1, C_3^*)$ . It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against masquerading user attack.
2. **Masquerading server attack:** The attacker can intercept a valid login request message  $(ID_i^*, T_1, K_1, C_3)$  of the user  $U_i$  from the public communication channel. Now the attacker has to guess the correct values of  $C_1$  and  $C_2$  at the same time to launch offline dictionary attack on  $C_3 = H(C_1 | T_1 | C_2)$  because the attacker requires correct values of  $C_1$  and  $C_2$  to frame fabricated valid response message  $(C_4^*, T_2)$ , where  $C_4^* = H(C_2 | T_2 | C_1)$ . It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against masquerading server attack.
3. **Offline dictionary attack:** A user may lose his smart card, which is found by the attacker or the attacker steals the user's smart card. The attacker can extract the stored values through some technique such as by monitoring their power consumption and reverse engineering techniques as pointed by Kocher et al. [67] and Messerges et al. [98]. The attacker can extract  $K_1 = H(ID_i \oplus x) \oplus N$ ,  $K_2 = H(ID_i \oplus x \oplus N) \oplus H(P_i)$ ,  $K_3 = H(ID_i | x | N) \oplus H(ID_i | P_i)$  and  $R = K_1 \oplus H(ID_i | P_i) \oplus H(P_i)$  from the memory of smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and

$P_i$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.

4. **Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(ID_i^*, T_1, K_1, C_3)$  within the valid time frame window but cannot compute the agreed session key  $S_k = H(C_1 | T_2 | C_2 | T_1)$  because the attacker does not know the values of  $C_1$  and  $C_2$ . Therefore, the proposed protocol is secure against parallel session attack.
5. **Denial of service attack:** In this type of attack, the attacker updates password verification information on smart card to some arbitrary value so that the legitimate user cannot login successfully in subsequent login request to the server. In the proposed protocol, smart card checks the validity of the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. Suppose the attacker has obtained the smart card of the legitimate user  $U_i$ . The attacker can insert the smart card of user  $U_i$  into a card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $K_1^* = R \oplus H(ID_i | P_i) \oplus H(P_i)$  and compares the computed value of  $K_1^*$  with the stored value of  $K_1$  in its memory to verify the legitimacy of the user before the smart card accepts the password update request. It is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against denial of service attack.
6. **Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(ID_i^*, T_1, K_1, C_3)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i^*, T_1, K_1, C_3)$  within the valid time frame window. The attacker can authenticate himself to the server  $S$  as well as to the legitimate user  $U_i$  but cannot compute the session key  $S_k = H(C_1 | T_2 | C_2 | T_1)$  because the attacker does not know the values of  $C_1$  and  $C_2$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
7. **Replay attack:** Replaying a message of one session into another session is useless because the user's smart card and the server  $S$  use current timestamp values as  $T_1$  and  $T_2$  in each new session, which make all the messages dynamic and valid for a small

interval of time. Moreover, the attacker can not compute the agreed session key  $S_k = H(C_1 | T_2 | C_2 | T_1)$  because the attacker does not know the values of  $C_1$  and  $C_2$ . Old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.

8. **Leak of verifier attack:** In the proposed protocol, the service provider server  $S$  knows secret  $x$  and only stores user identity  $ID_i$  in its database. The attacker does not have any technique to find the value of  $x$ . In case verifier is stolen by breaking into smart card database, the attacker does not have sufficient information to calculate the user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
9. **Server spoofing attack:** Malicious server can not generate the valid value of  $C_4 = H(C_2' | T_2 | C_1')$  meant for smart card because malicious server does not know the values of  $C_2'$  and  $C_1'$  corresponding to that user's smart card. The proposed protocol provides mutual authentication between the user and the server. Therefore, the proposed protocol is secure against server spoofing attack.
10. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $K_1 = H(ID_i \oplus x) \oplus N$ ,  $K_2 = H(ID_i \oplus x \oplus N) \oplus H(P_i)$ ,  $K_3 = H(ID_i | x | N) \oplus H(ID_i | P_i)$  and  $R = K_1 \oplus H(ID_i | P_i) \oplus H(P_i)$  from his smart card. This malicious user can not compute the value of  $x$  because he does not know the value of  $N$ . The malicious user also can not generate smart card specific value of  $C_3 = H(C_1 | T_1 | C_2)$  to masquerade as other legitimate user  $U_k$  to the service provider server  $S$  because the values of  $C_1$  and  $C_2$  is smart card specific. Therefore, the proposed protocol is secure against malicious user attack.
11. **Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to that user's smart card. Even after getting the valid smart card by any mean, the attacker gets a very few chances (normally a maximum of 3) to guess the identity  $ID_i$  and password  $P_i$  because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

### 5.3.4 Cost and functionality analysis

An efficient authentication scheme must take communication and computation cost into consideration during user's authentication. The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 5.3. Assume that the identity  $ID_i$ , password  $P_i$ , secret keys  $x$ , timestamp values and output of secure one-way hash function are all 128-bit long. Let  $T_H$  and  $T_X$  denote the time complexity for hash function and XOR operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_H \gg T_X$ .

Table 5.3

Cost comparison among related smart card based authentication schemes

	Proposed Protocol	Kim-Chung [65]	Yoon-Yoo [171]	Lee et al. [74][77]	Chein et al. [25]
E1	512 bits	384 bits	256 bits	128 bits	128 bits
E2	6 * 128 bits	6 * 128 bits	5 * 128 bits	5 * 128 bits	5 * 128 bits
E3	$5T_H + 7T_X$	$4T_H + 6T_X$	$1T_H + 4T_X$	$1T_H + 2T_X$	$1T_H + 2T_X$
E4	$5T_H + 4T_X$	$4T_H + 6T_X$	$2T_H + 4T_X$	$3T_H + 3T_X$	$2T_H + 3T_X$
E5	$6T_H + 3T_X$	$4T_H + 6T_X$	$2T_H + 5T_X$	$4T_H + 3T_X$	$3T_H + 3T_X$

In the proposed protocol, the parameters stored in the smart card are  $K_1, K_2, K_3, R$  and the memory needed in the smart card (E1) is 512 (= 4\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{ID_i^*, T_1, K_1, C_3\}$  and  $\{C_4, T_2\}$  is 768 (= 6\*128) bits. The computation cost of registration (E3) is the total time of all the operations executed in the registration phase. The computation cost of registration is  $5T_H + 7T_X$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, both the computation cost of the user and that of the service provider server are  $5T_H + 4T_X$  and  $6T_H + 3T_X$  respectively. The proposed protocol requires more computation than that of Kim and Chung's [65] scheme but it is highly secure as compared to the related schemes [65][171][74][77][25]. The proposed protocol is free from masquerading user attack, masquerading server attack, offline dictionary attack, denial of service attack and parallel session attack, while the latest scheme proposed by Kim and Chung in 2009 suffers from these attacks. The functionality comparison of the

proposed protocol with the related smart card based authentication schemes is summarized in Table 5.4.

**Table 5.4**

**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Kim-Chung [65]</b>	<b>Yoon-Yoo [171]</b>	<b>Lee et al. [74][77]</b>	<b>Chein et al. [25]</b>
Identity Verification in Login Phase	Yes	No	No	No	No
Session Key Agreement	Yes	No	No	No	No
Masquerading User Attack	No	Yes	Yes	Yes	Yes
Masquerading Server Attack	No	Yes	Yes	Yes	Yes
Offline Dictionary Attack	No	Yes	Yes	Yes	Yes
Denial of Service Attack	No	Yes	Yes	Yes	Yes
Parallel Session Attack	No	Yes	Yes	No	Yes

## 5.4 REVIEW OF XU ET AL.’S SCHEME

In this section, we examine the smart card based remote user authentication scheme proposed by Xu et al. [157] in 2009. This scheme also consists of four phases as summarized in Figure 5.5.

### 1. Registration phase

The server selects two large prime numbers  $p$  and  $q$  such that  $p = 2q + 1$ . The server also chooses its secret key  $x \in Z_q^*$  and one-way hash function  $H(\cdot)$ . The user  $U_i$  chooses his identity  $ID_i$  and password  $P_i$  and submits  $ID_i$  and  $P_i$  to the server  $S$  for registration over a secure communication channel. The server  $S$  computes  $B_i = H(ID_i)^x + H(P_i) \pmod p$  and stores  $(ID_i, B_i, H(\cdot), p, q)$  into a smart card. The server  $S$  issues the smart card containing security parameters  $(ID_i, B_i, H(\cdot), p, q)$  to the user  $U_i$  through a secure communication channel.

### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submit his identity  $ID_i^*$  password  $P_i^*$ . The smart card chooses  $w \in_R Z_q^*$  and computes  $B_i' = (B_i - H(P_i^*))^w \pmod p$ ,  $W_i = H(ID_i^*)^w \pmod p$  and  $C_i = H(T | B_i' | W_i | ID_i^*)$ , where  $T$  is the user  $U_i$ 's smart card current timestamp and sends the login request message  $(ID_i^*, C_i, W_i, T)$  to the service provider server  $S$ .

### 3. Authentication phase

The service provider server S verifies the received value of  $ID_i^*$  with stored value of  $ID_i$  in its database. Then the server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$

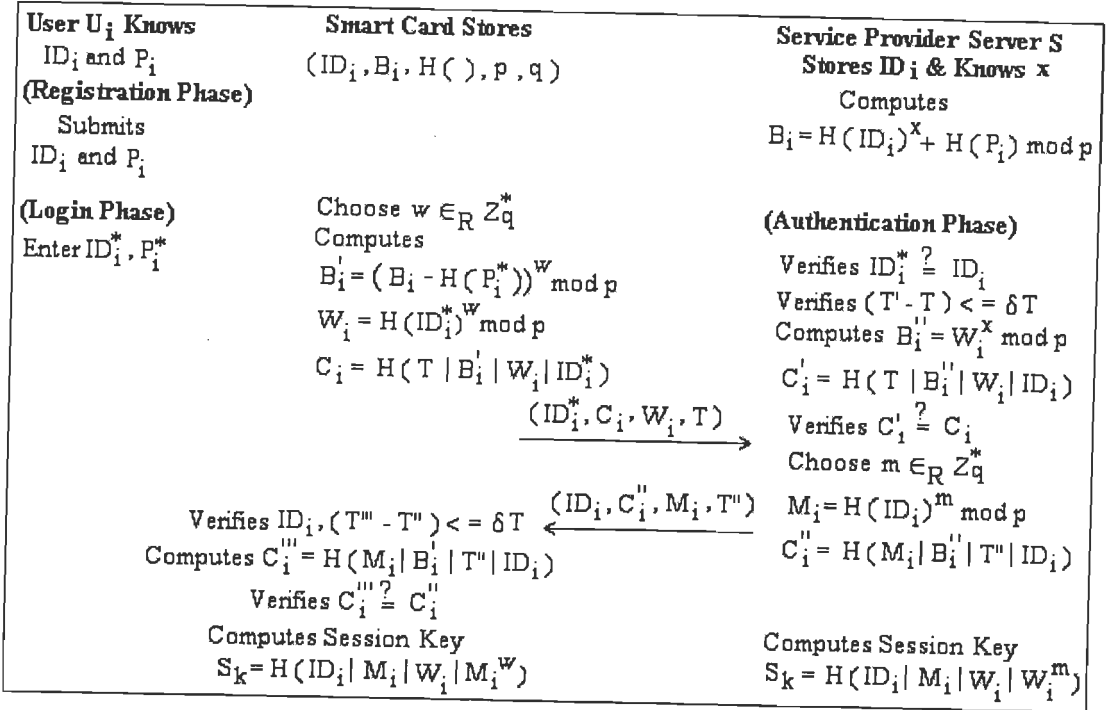


Figure 5.5: Xu et al.'s scheme

$\delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server S computes  $B_i'' = W_i^x \text{ mod } p$ ,  $C_i' = H(T | B_i'' | W_i | ID_i)$  and compares the computed value of  $C_i'$  with the received value of  $C_i$ . If they are not equal, the server S rejects the login request and terminates this session. Otherwise the server S chooses  $m \in_R Z_q^*$  and computes  $M_i = H(ID_i)^m \text{ mod } p$ ,  $C_i'' = H(M_i | B_i'' | T'' | ID_i)$ , where  $T''$  denotes the server's current timestamp and sends the message  $(ID_i, C_i'', M_i, T'')$  back to the smart card of user  $U_i$ . On receiving the message  $(ID_i, C_i'', M_i, T'')$ , smart card verifies the received value of  $ID_i$  and then checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  is the user  $U_i$ 's smart card current timestamp. Then the user  $U_i$ 's smart card computes  $C_i''' = H(M_i | B_i' | T'' | ID_i)$  and compares it with received value of  $C_i''$ . This equivalency authenticates the legitimacy of the service provider server S and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S compute  $S_k = H(ID_i | M_i | W_i | M_i^w) = H(ID_i | M_i | W_i | W_i^m)$  as session key.



#### 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i$  and password  $P_i$  corresponding to his smart card and requests to change his password to a new password  $P_i^{new}$ . The smart card interacts with the server  $S$  and after successful authentication, smart card replaces  $B_i = H(ID_i)^x + H(P_i) \bmod p$  with  $B_i^{new} = B_i - H(P_i) + H(P_i^{new}) \bmod p$  and password gets changed.

##### 5.4.1 Cryptanalysis of Xu et al.'s scheme

Xu et al. [157] claimed that their protocol can resist different attacks. However, this protocol is found to be flawed for forgery attack.

###### 1. Forgery attack

The attacker can intercept a valid login request message  $(ID_i, C_i, W_i, T)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_i = H(T | B_i' | W_i | ID_i)$  to know the value of  $B_i'$  because the attacker knows the values of  $T, W_i$  and  $ID_i$ . The correctly guessed value of  $B_i'$  is always same corresponding to the user  $U_i$ . After getting the correct value of  $B_i'$ , the attacker can frame and send fabricated valid login request message  $(ID_i, C_i, W_i', T_u)$  to the service provider server  $S$  without knowing the password  $P_i$  of the user  $U_i$ , where  $T_u$  is a current timestamp and  $C_i = H(T_u | B_i' | W_i' | ID_i)$  and  $W_i' = H(ID_i)^{w'}$ . Hence, the attacker can successfully make a valid login request to impersonate as a legitimate user  $U_i$  to the service provider server  $S$ . On receiving the message  $(ID_i, C_i'', M_i, T'')$  back from the server  $S$ , the attacker can compute the session key  $S_k = H(ID_i | M_i | W_i' | M_i^{w'})$  because the attacker knows the values of  $ID_i, M_i, W_i'$  and  $w'$ .

##### 5.4.2 Proposed protocol

In this section, we describe a modified smart card based remote user authentication protocol which resolves the above security flaw of Xu et al.'s [157] scheme. Figure 5.6 shows the entire protocol structure of the new authentication protocol. The proposed protocol consists of four phases viz. registration phase, login phase, verification and session key agreement phase and password change phase.

## 1. Registration phase

The server selects two large prime numbers  $p$  and  $q$  such that  $p = 2q + 1$ . The server also chooses its secret key  $x \in Z_q^*$  and one-way hash function  $H(\cdot)$ . The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  via a secure communication channel to register itself to the server  $S$ .

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  computes the security parameters  $B_i \equiv H(ID_i)^{x + y_i} \pmod p$  and  $C_i \equiv H(ID_i)^{y_i + P_i} \pmod p$  and stores  $(B_i, C_i, H(\cdot), p, q)$  into a smart card, where  $y_i$  is random value corresponding to user  $U_i$ . The server  $S$  issues the smart card containing security parameters  $(B_i, C_i, H(\cdot), p, q)$  to the user  $U_i$ .

Step 2:  $S \rightarrow U_i: \text{Smart card}$

User $U_i$ Knows $ID_i$ and $P_i$ (Registration Phase) Submits $ID_i$ and $P_i$	Smart Card Stores $(B_i, C_i, H(\cdot), p, q)$	Service Provider Server $S$ Knows $x$ Computes $B_i \equiv H(ID_i)^{x + y_i} \pmod p$ $C_i \equiv H(ID_i)^{y_i + P_i} \pmod p$ Stores $ID_i$
(Login Phase) Enter $ID_i^*, P_i^*$	Computes $C_i' \equiv C_i / H(ID_i)^{P_i} \pmod p$ $\equiv H(ID_i)^{y_i} \pmod p$ $B_i' \equiv B_i / C_i'$ $\equiv H(ID_i)^x \pmod p$ Choose $w \in_R Z_q^*$ $D_i \equiv B_i'^w \equiv H(ID_i)^{xw} \pmod p$ $E_i \equiv H(ID_i)^w \pmod p$ $M_i = H(B_i'   C_i'   D_i   T)$	(Verification Phase) Verifies $ID_i \stackrel{?}{=} ID_i$ Verifies $(T' - T) \leq \delta T$ $B_i' \equiv H(ID_i)^x \pmod p$ $C_i' \equiv B_i / B_i'$ $\equiv H(ID_i)^{y_i} \pmod p$ $D_i' \equiv E_i^x$ $\equiv H(ID_i)^{xw} \pmod p$ $M_i' = H(B_i'   C_i'   D_i'   T)$ Verifies $M_i' \stackrel{?}{=} M_i$
	Computes Session Key $S_k = H(D_i   C_i'   B_i'   T)$	$S_k = H(D_i   C_i'   B_i'   T)$

Figure 5.6: Proposed improvement in Xu et al.'s scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i$  and password  $P_i$ . The smart card computes  $C_i' \equiv C_i / H(ID_i)^{P_i} \pmod p \equiv H(ID_i)^{y_i} \pmod p$ ,  $B_i' \equiv B_i / C_i' \equiv H(ID_i)^x \pmod p$ ,  $D_i \equiv B_i'^w \equiv H(ID_i)^{xw} \pmod p$ ,  $E_i \equiv H(ID_i)^w \pmod p$  and  $M_i = H(B_i' | C_i' | D_i | T)$ , where smart card chooses  $w \in_R Z_q^*$  and  $T$  is the current timestamp of smart card. Then smart card sends the login request message  $(ID_i, B_i, E_i, M_i, T)$  to the server  $S$ .

Step 1: Smart card  $\rightarrow S: ID_i, B_i, E_i, M_i, T$

### 3. Verification and session key agreement phase

The service provider server S verifies the received value of  $ID_i$  with stored value of  $ID_i$  in its database. Then the server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. The server S computes  $B_i' \equiv H(ID_i)^x \pmod p$ ,  $C_i' \equiv B_i / B_i' \equiv H(ID_i)^{y_i} \pmod p$ ,  $D_i' \equiv E_i^x \equiv H(ID_i)^{xw} \pmod p$ ,  $M_i' = H(B_i' | C_i' | D_i' | T)$  and compares the computed value of  $M_i'$  with the received values of  $M_i$ .

Step 1: Server S checks  $M_i' \stackrel{?}{=} M_i$

This equivalency authenticates the legitimacy of the user and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S agree on the common session key as  $S_k = H(D_i | C_i' | B_i' | T)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server S can retrieve the original message because both of them know the common session key.

### 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i$  and password  $P_i$  corresponding to his smart card and requests to change his password to a new password  $P_i^{new}$ . The smart card interacts with the server S and after successful authentication, smart card replaces  $C_i \equiv H(ID_i)^{y_i} + P_i \pmod p$  with  $C_i^{new} \equiv [C_i / H(ID_i)^{P_i}] * H(ID_i)^{P_i^{new}} \pmod p$  and password gets changed.

#### 5.4.3 Security analysis

A good password authentication protocol should provide protection from different feasible attacks.

- 1. Forgery attack:** In this type of attack, the attacker impersonates as a legitimate user and forges the authentication messages using the information obtained from the authentication scheme. The attacker can attempt to modify a login request message  $(ID_i, B_i, E_i, M_i, T)$  into  $(ID_i, B_i, E_i, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of the verification and session key agreement phase because the attacker

has no way of obtaining the values of  $B_i'$ ,  $C_i'$  and  $D_i$  to compute the valid parameter  $M_i^*$ . Therefore, the proposed protocol is secure against forgery attack.

2. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $B_i \equiv H(ID_i)^{x+y_i} \pmod p$  and  $C_i \equiv H(ID_i)^{y_i + P_i} \pmod p$  from the memory of smart card. This malicious user can not generate smart card specific value of  $M_k = H(B_k' | C_k' | D_k | T)$  to masquerade as other legitimate user  $U_k$  to the service provider server  $S$  because the value of  $M_k$  is smart card specific and depends upon the values of  $B_k'$ ,  $C_k'$  and  $D_k$ . The malicious user does not have any method to calculate the values of  $B_k'$ ,  $C_k'$  and  $D_k$ . Therefore, the proposed protocol is secure against malicious user attack.
3. **Stolen smart card attack:** In case a user's smart card is stolen by the attacker, he can extract the information stored in its memory. The attacker can extract  $B_i \equiv H(ID_i)^{x+y_i} \pmod p$  and  $C_i \equiv H(ID_i)^{y_i + P_i} \pmod p$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
4. **Offline dictionary attack:** The attacker first tries to obtain the user verification information  $ID_i$ ,  $B_i \equiv H(ID_i)^{x+y_i} \pmod p$ ,  $E_i \equiv H(ID_i)^w \pmod p$ ,  $M_i = H(B_i' | C_i' | D_i | T)$ ,  $T$  and then try to guess the values of  $x$  and  $y_i$  by offline guessing. It is not possible to guess the values of  $x$  and  $y_i$  due to discrete logarithm problem even after gathering this information. In another option, the attacker requires valid smart card of the legitimate user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  correctly at the same time. It is not possible to verify the guessed values of  $ID_i$  and password  $P_i$  in the proposed protocol by offline dictionary attack because there is no verifier information stored in the smart card corresponding to correct values of  $ID_i$  and  $P_i$ . Therefore, the proposed protocol is secure against offline dictionary attack.
5. **Denial of service attack:** In the proposed protocol, smart card interacts with the server  $S$  and accepts the password update request after successful authentication by the server  $S$ . It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.

6. **Replay attack:** Replaying a login request message  $(ID_i, B_i, E_i, M_i, T)$  of one session into another session is useless because the user  $U_i$ 's smart card uses current timestamp value  $T$  in each new session, which makes the value of  $M_i = H(B_i' | C_i' | D_i | T)$  dynamic and valid for small interval of time. Old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.
7. **Leak of verifier attack:** In the proposed protocol, there is no secret verifier information stored on the server. In case verifier is stolen by breaking into smart card database, the attacker does not have sufficient information to calculate the user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
8. **Server spoofing attack:** In the proposed protocol, malicious server can not compute the session key  $S_k = H(D_i | C_i' | B_i' | T)$  because the malicious server does not know the values of  $D_i, C_i'$  and  $B_i'$ . Moreover, the session key is different for the same user in different login sessions. Therefore, the proposed protocol is secure against server spoofing attack.
9. **Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to the user  $U_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Moreover, an attacker gets very few chances (normally a maximum of 3) to guess the identity  $ID_i$  and password  $P_i$  because smart card gets locked after certain number of unsuccessful attempts. Therefore, the proposed protocol is secure against online dictionary attack.
10. **Parallel session attack:** In this type of attack, the attacker first listens to communication between the user and the server. After that, he initiates a parallel session to imitate legitimate user to login on to the server by resending the captured messages transmitted between the user and the server with in the valid time frame window. He can masquerade as a legitimate user by replaying a login request message  $(ID_i, B_i, E_i, M_i, T)$  with in the valid time frame window. The attacker can not compute

the agreed session key  $S_k = H(D_i | C_i' | B_i' | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $D_i$ ,  $C_i'$  and  $B_i'$ . Therefore, the proposed protocol is secure against parallel session attack.

**11. Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(ID_i, B_i, E_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(ID_i, B_i, E_i, M_i, T)$  within the valid time frame window. The attacker can authenticate itself to the server  $S$  but cannot compute the agreed session key  $S_k = H(D_i | C_i' | B_i' | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $D_i$ ,  $C_i'$  and  $B_i'$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

**12. Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the user's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can only cause authentication between the user and the server to fail but cannot allow the attacker to gain any information about the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

#### 5.4.4 Cost and functionality analysis

An efficient authentication scheme must take communication and computation cost into consideration during user's authentication. The cost and functionality comparison of the proposed protocol with the most related smart card based authentication schemes is summarized in Table 5.5 and table 5.6. Assume that the identity  $ID_i$ , password  $P_i$ ,  $w$ ,  $x$ ,  $y_i$  and timestamp values are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$ ,  $T_E$ ,  $T_X$ ,  $T_P$  and  $T_R$  denote the time complexity for hash function, exponential operation, XOR operation, public key operation and pseudo random function respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_P \gg T_E \gg T_H \approx T_R \gg T_X$ .

**Table 5.5**

**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Xu et al. [157]</b>	<b>Yang et al. [159]</b>	<b>Liao et al. [82]</b>	<b>Lee-Chiu [76]</b>	<b>Lee et al. [77]</b>
E1	512 bits	512 bits	896 bits	512 bits	640 bits	128 bits
E2	5 * 128 bits	8 * 128 bits	10 * 128 bits	6 * 128 bits	4 * 128 bits	5 * 128 bits
E3	2 T <sub>E</sub> + 1 T <sub>H</sub>	1 T <sub>E</sub> + 2 T <sub>H</sub>	2 T <sub>H</sub> + 1 T <sub>R</sub>	1 T <sub>E</sub> + 2 T <sub>H</sub>	1 T <sub>E</sub> + 2 T <sub>H</sub>	1 T <sub>H</sub> + 2 T <sub>X</sub>
E4	3 T <sub>E</sub> + 3 T <sub>H</sub>	3 T <sub>E</sub> + 5 T <sub>H</sub>	1 T <sub>E</sub> + 1 T <sub>P</sub>	1 T <sub>E</sub> + 3 T <sub>H</sub>	2 T <sub>E</sub> + 2 T <sub>H</sub> + 1 T <sub>X</sub>	3 T <sub>H</sub> + 3 T <sub>X</sub>
E5	2 T <sub>E</sub> + 3 T <sub>H</sub>	3 T <sub>E</sub> + 4 T <sub>H</sub>	1 T <sub>E</sub> + 1 T <sub>P</sub>	1 T <sub>E</sub> + 3 T <sub>H</sub>	1 T <sub>E</sub> + 2 T <sub>H</sub> + 1 T <sub>X</sub>	4 T <sub>H</sub> + 3 T <sub>X</sub>

In the proposed protocol, the parameters stored in the smart card are  $B_i$ ,  $C_i$ ,  $p$ ,  $q$  and the memory needed (E1) in the smart card is 512 (= 4\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{ID_i, B_i, E_i, M_i, T\}$  is 640 (= 5\*128) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration (E3) is  $2T_E + 1T_H$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user (E4) is  $3T_E + 3T_H$  and that of the service provider server (E5) is  $2T_E + 3T_H$ . The proposed protocol defends forgery attack and has less computation cost (E2, E4, E5) as compared to latest scheme proposed by Xu et al. [157] in 2009.

**Table 5.6**

**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Xu et al. [157]</b>	<b>Yang et al. [159]</b>	<b>Liao et al. [82]</b>	<b>Lee-Chiu [76]</b>	<b>Lee et al. [77]</b>
Forgery Attack	No	Yes	No	No	Yes	Yes
Man-in-the-Middle Attack	No	No	No	No	Yes	Yes
Offline Dictionary Attack	No	No	No	Yes	Yes	Yes
Session Key Agreement	Yes	Yes	Yes	No	No	No

## 5.5 REVIEW OF LIU ET AL.'S SCHEME

In this section, we examine the smart card based remote user authentication scheme proposed by Liu et al. [91] in 2008. Liu et al.'s scheme consists of three phases viz. initialization phase, registration phase, login and authentication phase.

### 1. Initialization phase

Key Information Center (KIC) generates secret parameters corresponding to the user, store them on the smart card and issue the smart card to the user. KIC is also responsible to change the passwords of registered users. It generates two large prime numbers  $p$  and  $q$  and computes  $n = p \cdot q$ . Then it chooses a public key  $e$  and finds a corresponding secret key  $d$  that satisfies  $e \cdot d \equiv 1 \pmod{(p-1) \cdot (q-1)}$ . The secret key  $d$  is sent to the server  $S$  over a secure communication channel. Afterwards, KIC finds an integer  $g$  that is a primitive element in  $GF(p)$  and  $GF(q)$ , where  $g$  is the public parameter of KIC. Finally, KIC sends the parameters  $n$ ,  $e$  and  $g$  to the server  $S$ .

### 2. Registration phase

The user  $U_i$  has to submit his password  $P_i$  to KIC for registration over a secure communication channel. KIC selects an identity  $ID_i$  corresponding to the user  $U_i$ . Then KIC computes  $CID_i = H(ID_i \oplus d)$ ,  $S_i \equiv ID_i^d \pmod{n}$  and  $h_i \equiv g^{P_i \cdot d} \pmod{n}$ , where  $H(\cdot)$  is a one-way hash function. Afterwards, KIC issues the smart card containing secret parameters  $(n, e, g, ID_i, CID_i, S_i, h_i)$  to the user  $U_i$  through a secure communication channel.

### 3. Login and authentication phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card compares the identity  $ID_i^*$  with the stored value of  $ID_i$  in its memory to verify the legitimacy of the user. Then the smart card computes  $SID_i = H(CID_i)$  and sends the login request message  $M_1 = \{ID_i, SID_i\}$  to the service provider server  $S$ . The service provider server  $S$  computes  $CID_i = H(ID_i \oplus d)$  and compares  $H(CID_i)$  with the received value of  $SID_i$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise the server  $S$  stores the parameters  $ID_i, SID_i$  and chooses nonce value  $N_S$  as a challenge to the user  $U_i$ . The server  $S$  computes  $S_N = N_S \oplus CID_i$  and sends the message  $M_2 = \{S_N\}$  back to the smart card of the user  $U_i$ . On receiving the message  $M_2$ , the smart card chooses a random nonce value  $N_C$  and computes  $N_S = S_N \oplus CID_i$ ,  $X_i \equiv g^{N_C \cdot P_i} \pmod{n}$  and  $Y_i \equiv S_i \cdot h_i^{N_C \cdot N_S} \pmod{n}$ . Then, the smart card sends the message  $M_3 = \{X_i, Y_i\}$  to the server  $S$ . On receiving the message  $M_3$ , the server  $S$  checks whether the equation  $Y_i^e \equiv ID_i \cdot X_i^{N_S} \pmod{n}$  holds. If it holds, the server  $S$  accepts the login request and computes  $Z_i \equiv (H(CID_i, X_i))^d \pmod{n}$  and sends the



message  $M_4 = \{Z_i\}$  back to the smart card. On receiving the message  $M_4$ , the smart card checks whether the equation  $Z_i^e \equiv H(CID_i, X_i) \pmod n$  holds or not. This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted.

### 5.5.1 Cryptanalysis of Liu et al.'s scheme

Liu et al. [91] claimed that their scheme can resist various types of known attacks. However, we found that their scheme is flawed for stolen smart card attack. Their scheme is also found to be flawed for man-in-the-middle attack by Sun et al. [140].

#### 1. Stolen smart card attack

A user  $U_i$  may lose his smart card, which is found by an attacker or an attacker steals the user's smart card. An attacker can extract the stored values through some technique like by monitoring their power consumption and reverse engineering techniques as pointed out by Kocher et al. [67] and Messerges et al. [98].

1. The attacker can extract the  $(n, e, g, ID_i, CID_i, S_i, h_i)$  parameters from the memory of a smart card.
2. Now the attacker computes  $SID_i = H(CID_i)$  and sends the login request message  $M_1 = \{ID_i, SID_i\}$  to the service provider server  $S$ .
3. The service provider server  $S$  computes and verifies the received value of  $SID_i$ .
4. Then the service provider server  $S$  chooses random nonce value  $N_S$  as a challenge to the smart card of the user  $U_i$ , computes  $S_N = N_S \oplus CID_i$  and sends the message  $M_2 = \{S_N\}$  back to the smart card of the user  $U_i$ .
5. Afterwards, the smart card chooses a random nonce value  $N_C$ , computes  $N_S = S_N \oplus CID_i$ ,  $X_i^* \equiv h_i^{N_C \cdot e} \pmod n$  and  $Y_i \equiv S_i \cdot h_i^{N_C \cdot N_S} \pmod n$ .

$$\begin{aligned}
 X_i^* &\equiv h_i^{N_C \cdot e} \pmod n \\
 &\equiv (g^{P_i \cdot d} \pmod n)^{N_C \cdot e} \pmod n \quad \text{because } h_i \equiv g^{P_i \cdot d} \pmod n \\
 &\equiv (g^{P_i \cdot d \cdot N_C \cdot e} \pmod n) \pmod n \\
 &\equiv (g^{P_i \cdot N_C} \pmod n) \pmod n \quad \text{because } g^{e \cdot d} \pmod n \equiv 1 \\
 &\equiv (g^{N_C \cdot P_i} \pmod n) \pmod n \\
 &\equiv X_i
 \end{aligned}$$

6. Then the smart card sends the message  $M_3 = \{X_i, Y_i\}$  to the server S.
7. The server S checks and verifies that the equation  $Y_i^e \equiv ID_i \cdot X_i^N \pmod n$  holds.
8. Then the server S computes  $Z_i \equiv (H(CID_i, X_i))^d \pmod n$  and sends the message  $M_4 = \{Z_i\}$  back to the smart card of the user  $U_i$ .
9. Now the attacker masquerading as the user  $U_i$  has authenticated itself to the service provider server S without knowing the password  $P_i$  of the user  $U_i$ .
10. That means once an attacker gets the smart card of the user  $U_i$ , he can masquerade as a legitimate user by authenticating itself to the server S without knowing the password  $P_i$  of the user  $U_i$  corresponding to user  $U_i$ 's smart card.

### 5.5.2 Proposed protocol

In this section, we describe a modified smart card based remote user authentication protocol which resolves the above security flaws of Liu et al.'s [91] scheme. Figure 5.7 shows the entire protocol structure of the new authentication protocol. The proposed protocol consists of four phases viz. registration phase, login phase, verification and session key agreement phase and password change phase.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server S via a secure communication channel to register itself to the server S.

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server S computes the security parameters  $Z_i \equiv g^{(ID_i | P_i) + H(P_i)} \pmod n$ ,  $B_i \equiv g^{(ID_i | x | y_i) + H(P_i)} \pmod n$  and  $C_i \equiv g^{x + y_i + P_i} \pmod n$ , where  $n$  is large prime number and  $g$  is a primitive element in  $GF(n)$ . The server S chooses its secret key  $x$  and  $H(\ )$  is a one-way hash function. Then the server S issues the smart card containing security parameters  $(Z_i, B_i, C_i, n, g, H(\ ))$  to the user  $U_i$ .

Step 2:  $S \rightarrow U_i: \text{Smart card}$

The server S also computes  $A_i \equiv g^{(ID_i | x | y_i) + y_i} \pmod n$  for each user and stores  $y_i \oplus x$  corresponding to  $A_i$  in its database. The server S chooses the value of  $y_i$  corresponding to each user in such a way so that the value of  $A_i$  must be unique for each user.

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $Z_i^* \equiv g^{(ID_i^* | P_i^*) + H(P_i^*)} \pmod n$  and compares it with the stored value of  $Z_i$  in its memory to verify the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $Z_i^* \stackrel{?}{=} Z_i$

After verification, the smart card computes  $B_i' \equiv B_i g^{-H(P_i)} \pmod n \equiv g^{(ID_i | x | y_i) + H(P_i)} \pmod n$ ,  $C_i' \equiv C_i g^{-P_i} \pmod n \equiv g^{x + y_i} \pmod n$ ,  $D_i \equiv B_i'$ ,  $C_i' \pmod n \equiv g^{(ID_i | x | y_i) + x + y_i} \pmod n$ ,  $E_i \equiv g^{w + H(B_i' | T)} \pmod n$  and  $M_i = H(B_i' | C_i' | T)$ , where smart card chooses  $w \in_R Z_n^*$  and  $T$  is current timestamp of the smart card. Then the smart card sends the login request message  $(D_i, E_i, M_i, T)$  to the server  $S$ .

Step 2: Smart card  $\rightarrow$   $S: D_i, E_i, M_i, T$

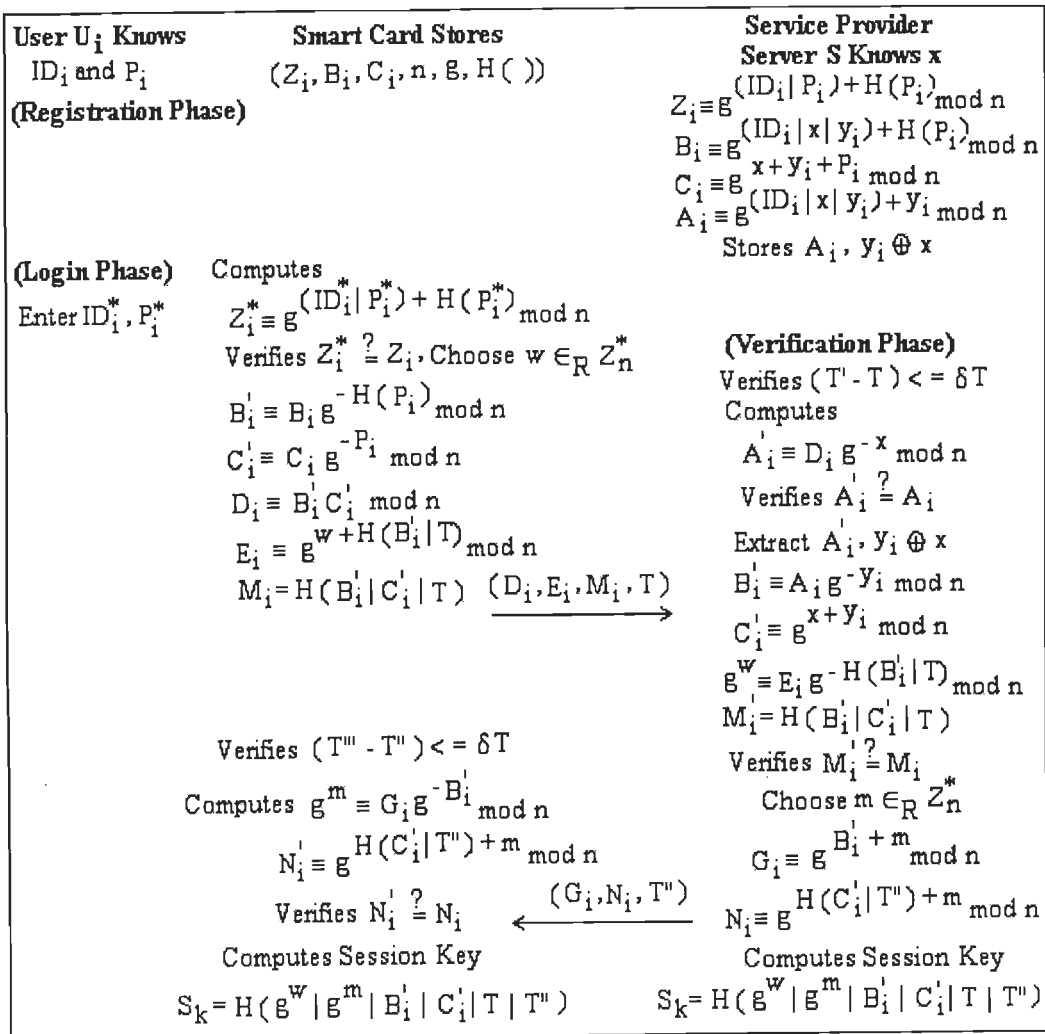


Figure 5.7: Proposed improvement in Liu et al.'s scheme

### 3. Verification and session key agreement phase

After receiving the login request from the user  $U_i$ , the service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current timestamp of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. The server  $S$  computes  $A_i' \equiv D_i g^{-x} \pmod n$  and compares  $A_i'$  with the stored values of  $A_i$  in its database.

Step 1: Server  $S$  checks  $A_i' \stackrel{?}{=} A_i$

If no match found, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  extracts  $y_i$  from  $y_i \oplus x$  corresponding to  $A_i$  from its database. Now the server  $S$  computes  $B_i' \equiv A_i g^{-y_i} \pmod n \equiv g^{(ID_i | x | y_i)} \pmod n$ ,  $C_i' \equiv g^{x + y_i} \pmod n$ ,  $g^w \equiv E_i g^{-H(B_i' | T)} \pmod n$ ,  $M_i' = H(B_i' | C_i' | T)$  and compares  $M_i'$  with the received values of  $M_i$ .

Step 2: Server  $S$  checks  $M_i' \stackrel{?}{=} M_i$

Now the server  $S$  chooses  $m \in_{\mathcal{R}} Z_n^*$  and acquires the current timestamp  $T''$  and computes  $G_i \equiv g^{B_i' + m} \pmod n$ ,  $N_i \equiv g^{H(C_i' | T'') + m} \pmod n$  and sends the message  $(G_i, N_i, T'')$  back to the smart card of the user  $U_i$ .

Step 3:  $S \rightarrow$  Smart card:  $G_i, N_i, T''$

On receiving the message  $(G_i, N_i, T'')$ , the user  $U_i$ 's smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  is current timestamp of the smart card. Then the smart card extracts  $g^m \equiv G_i g^{-B_i'} \pmod n$  and computes  $N_i' \equiv g^{H(C_i' | T'')} \cdot g^m \pmod n \equiv g^{H(C_i' | T'') + m} \pmod n$  and compares the computed value of  $N_i'$  with the received value of  $N_i$  to verify the legitimacy of the service provider server  $S$ .

Step 4: Smart card checks  $N_i' \stackrel{?}{=} N_i$

This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server  $S$  agree on the common session key as  $S_k = H(g^w | g^m | B_i' | C_i' | T | T'')$ .

### 4. Password change phase

The user  $U_i$  can change his password without the help of the server  $S$ . The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $Z_i^* \equiv g^{(ID_i^* | P_i^*) + H(P_i^*)} \pmod n$  and compares the computed value of  $Z_i^*$  with the stored value of  $Z_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of the card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart

card asks the card holder to resubmit a new password  $P_i'$  and then the smart card computes  $Z_i^{new} \equiv g^{(ID_i | P_i') + H(P_i')} \pmod n$ ,  $B_i^{new} \equiv B_i g^{-H(P_i')} g^{+H(P_i')} \equiv g^{(ID_i | x | y_i) + H(P_i')} \pmod n$  and  $C_i^{new} \equiv C_i g^{-P_i} g^{+P_i'} \equiv g^{x+y_i+P_i'} \pmod n$ . Afterwards, the smart card updates the values of  $Z_i$ ,  $B_i$  and  $C_i$  stored in its memory with  $Z_i^{new}$ ,  $B_i^{new}$  and  $C_i^{new}$  and password gets changed.

### 5.5.3 Security analysis

A good password authentication protocol should provide protection from different possible attacks relevant to that protocol.

1. **Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by an attacker, he can extract the information stored in its memory. An attacker can extract  $Z_i \equiv g^{(ID_i | P_i) + H(P_i)} \pmod n$ ,  $B_i \equiv g^{(ID_i | x | y_i) + H(P_i)} \pmod n$  and  $C_i \equiv g^{x+y_i+P_i} \pmod n$  from the memory of smart card. Even after gathering this information, the attacker has to guess out  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out the two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
2. **Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(D_i, E_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(D_i, E_i, M_i, T)$  with in the valid time frame window. The attacker can authenticate itself to the server  $S$  as well as to the legitimate user  $U_i$  but can not compute the session key  $S_k = H(g^w | g^m | B_i' | C_i' | T | T')$  because the attacker does not know the value of  $g^w$ ,  $g^m$ ,  $B_i'$  and  $C_i'$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
3. **Impersonation attack:** In this type of attack, the attacker impersonates as the legitimate user and forges the authentication message using the information obtained from the authentication scheme. The attacker can attempt to modify a login request message  $(D_i, E_i, M_i, T)$  into  $(D_i, E_i^*, M_i^*, T^*)$  so as to succeed in the authentication, where  $T^*$  is the attacker's current date and time. However, such a modification will fail in Step 2 of the verification and session key agreement phase because the attacker has no way of obtaining the values of  $ID_i$ ,  $P_i$ ,  $x$  and  $y_i$  to compute the valid parameters  $E_i^*$  and  $M_i^*$ . Therefore, the proposed protocol is secure against impersonation attack.

4. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $Z_i \equiv g^{(ID_i | P_i) + H(P_i)} \bmod n$ ,  $B_i \equiv g^{(ID_i | x | y_i) + H(P_i)} \bmod n$  and  $C_i \equiv g^{x + y_i + P_i} \bmod n$  from the memory of smart card. This malicious user can not generate smart card specific values of  $B_k' \equiv g^{(ID_k | x | y_k)} \bmod n$  and  $C_k' \equiv g^{x + y_k} \bmod n$  to masquerade as other legitimate user  $U_k$  to the service provider server  $S$  because the values of  $B_k'$  and  $C_k'$  is smart card specific and depend upon the values of  $ID_k$ ,  $x$  and  $y_k$ . The malicious user does not have any method to calculate the values of  $ID_k$ ,  $x$  and  $y_k$ . Therefore, the proposed protocol is secure against malicious user attack.
5. **Offline dictionary attack:** The attacker first tries to obtain some user or server verification information such as  $D_i \equiv B_i' \cdot C_i' \bmod n \equiv g^{(ID_i | x | y_i) + x + y_i} \bmod n$ ,  $E_i \equiv g^{w + H(B_i' | T)} \bmod n$ ,  $M_i = H(B_i' | C_i' | T)$ ,  $T$ ,  $G_i \equiv g^{B_i' + m} \bmod n$ ,  $N_i \equiv g^{H(C_i' | T') + m} \bmod n$ ,  $T''$  and then tries to guess the  $ID_i$ ,  $P_i$ ,  $x$  and  $y_i$  by offline guessing. Even after gathering this information, the attacker has to guess  $ID_i$ ,  $P_i$ ,  $x$  and  $y_i$  correctly at the same time. In another option, the attacker requires valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at same time. Therefore, the proposed protocol is secure against offline dictionary attack.
6. **Denial of service attack:** In the proposed protocol, the smart card checks the validity of user identity  $ID_i$  and password  $P_i$  before password update procedure. The attacker inserts the smart card into a smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $Z_i^* \equiv g^{(ID_i^* | P_i^*) + H(P_i^*)} \bmod n$  and compares it with the stored value of  $Z_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
7. **Replay attack:** Replaying a message of one session into another session is useless because the user  $U_i$ 's smart card and the server  $S$  uses current timestamp values  $T$  and  $T''$  in each new session, which make the values of  $E_i$ ,  $M_i$  and  $N_i$  dynamic and valid for small interval of time. Hence old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.

- 8. Leak of verifier attack:** In the proposed protocol, the service provider server  $S$  knows secret  $x$  and stores  $y_i \oplus x$  corresponding to the user  $U_i$ 's  $A_i$  value in its database. The attacker does not have any way to find out the value of  $x$  and hence can not calculate  $y_i$  from  $y_i \oplus x$ . Moreover, the attacker can not calculate  $ID_i$ ,  $x$  and  $y_i$  from  $A_i \equiv g^{(ID_i | x | y_i) + y_i} \bmod n$ . In case verifier is stolen by breaking into smart card database, the attacker does not have sufficient information to calculate the user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
- 9. Server spoofing attack:** In server spoofing attack, the attacker can manipulate the sensitive data of legitimate users via setting up fake servers. The proposed protocol provides mutual authentication to withstand the server spoofing attack. Malicious server can not generate the valid value of  $G_i \equiv g^{B_i' + m} \bmod n$  and  $N_i \equiv g^{H(C_i' | T'') + m} \bmod n$  meant for the smart card of user  $U_i$  because malicious server has to know the value of  $B_i'$  and  $C_i'$  to generate the valid values of  $G_i$  and  $N_i$  corresponding to the user  $U_i$ 's smart card. Therefore, the proposed protocol is secure against server spoofing attack.
- 10. Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$ . Even after getting the valid smart card of user  $U_i$  by any means, the attacker gets very few chances (normally a maximum of 3) to guess the identity  $ID_i$  and password  $P_i$  because the smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time. Therefore, the proposed protocol is secure against online dictionary attack.
- 11. Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(D_i, E_i, M_i, T)$  within the valid time frame window. However, the attacker can not compute the agreed session key  $S_k = H(g^w | g^m | B_i' | C_i' | T | T'')$  because the attacker does not know the values of  $g^w$ ,  $g^m$ ,  $B_i'$  and  $C_i'$ . Therefore, the proposed protocol is secure against parallel session attack.

#### 5.5.4 Cost and functionality analysis

An efficient authentication scheme must take communication and computation cost into consideration during user's authentication. The cost comparison of the proposed protocol

with the most related smart card based authentication schemes is summarized in Table 5.7. Assuming that the identity  $ID_i$ , password  $P_i$ , random number ( $w$  or  $m$ ),  $x$ ,  $y_i$ , timestamp, nonce values are all 128-bit long and prime modular operation is 1024-bit length as in most of practical implementations. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$ ,  $T_E$  and  $T_X$  denote the time complexity for hash function, exponential operation and XOR operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_E \gg T_H \gg T_X$ . In our proposed protocol, the parameters stored in the smart card are  $Z_i$ ,  $B_i$ ,  $C_i$ ,  $n$ ,  $g$  and the memory needed (E1) in the smart card is 640 (=  $5 \cdot 128$ ) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{D_i, E_i, M_i, T\}$  and  $\{G_i, N_i, T''\}$  is 896 (=  $7 \cdot 128$ ) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration (E3) is  $4T_E + 1T_H + 1T_X$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user (E4) is  $6T_E + 5T_H$  and that of the service provider server (E5) is  $6T_E + 4T_H + 1T_X$ .

**Table 5.7**

**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Xu et al. [157]</b>	<b>Liu et al. [91]</b>	<b>Shen et al. [128]</b>	<b>Yang-Shieh [160]</b>
E1	640 bits	512 bits	896 bits	896 bits	896 bits
E2	$7 \cdot 128$ bits	$8 \cdot 128$ bits	$6 \cdot 128$ bits	$10 \cdot 128$ bits	$8 \cdot 128$ bits
E3	$4T_E + 1T_H + 1T_X$	$1T_E + 2T_H$	$2T_E + 1T_H + 1T_X$	$2T_E + 1T_H + 1T_X$	$2T_E$
E4	$6T_E + 5T_H$	$3T_E + 5T_H$	$3T_E + 2T_H + 1T_X$	$3T_E + 2T_H$	$2T_E + 1T_H$
E5	$6T_E + 4T_H + 1T_X$	$3T_E + 4T_H$	$2T_E + 3T_H + 2T_X$	$3T_E + 3T_H + 1T_X$	$2T_E + 1T_H$

The functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 5.8. The proposed protocol requires some additional computation costs (E3, E4, E5) as compared to other related schemes but it is highly secure as compared to the other schemes.



Table 5.8

Functionality comparison among related smart card based authentication schemes

	Proposed Protocol	Xu et al. [157]	Liu et al. [91]	Shen et al. [128]	Yang-Shieh [160]
Stolen Smart Card Attack	No	No	Yes	Yes	Yes
Man-in-the-Middle Attack	No	No	Yes	Yes	Yes
Forgery Attack	No	Yes	No	Yes	Yes
Identity Protection	Yes	No	No	No	No
Offline Dictionary Attack	No	No	No	No	No
Mutual Authentication	Yes	Yes	Yes	Yes	No
Session Key Agreement	Yes	Yes	No	No	No

## 5.6 CONCLUSION

In this chapter, we presented cryptanalysis of Yoon et al.'s scheme, Kim and Chung's scheme, Xu et al.'s scheme and Liu et al.'s scheme by showing that their schemes are vulnerable to different attacks. The improvements to these schemes are proposed. Security analysis proved that the proposed protocols are more secure and practical.

## DYNAMIC IDENTITY BASED SMART CARD AUTHENTICATION PROTOCOLS

---

### 6.1 INTRODUCTION

A number of static identity based smart card authentication schemes have been proposed to improve security, efficiency and cost. The static identity leaks out partial information about the user's authentication messages to the attacker. On the other hand, the dynamic identity based smart card authentication schemes provide multi-factor authentication based on acquiring the smart card, knowing the identity and password and hence more suitable to e-commerce applications.

In this chapter, a brief review of Liao et al.'s scheme, Liou et al.'s scheme, Wang et al.'s scheme, Lee et al.'s scheme and Hsiang & Shih's scheme is given. Cryptanalysis of these schemes for different types of attacks is done and improved dynamic identity based smart card authentication protocols are proposed. The security analysis of the improved proposed protocols is presented. The cost and functionality comparison of the proposed protocols with the other related protocols is also presented.

### 6.2 REVIEW OF LIAO ET AL.'S SCHEME

In this section, we examine a dynamic identity based smart card authentication scheme proposed by Liao et al. [81] in 2005. Liao et al.'s scheme consists of four phases viz. registration phase, login phase, verification phase and password change phase as summarized in Figure 6.1.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password verifier information  $H(P_i)$  to the server  $S$  for registration over a secure communication channel. The server  $S$  computes  $N_i = H(P_i) \oplus H(x | ID_i)$ , where  $x$  is the secret key of the remote server  $S$ ,  $\oplus$  represents XOR logic,  $|$  represents concatenation operation and  $H(\ )$  is a one-way hash function. Then the server  $S$  issues the smart card containing secret parameters  $(H(\ ), N_i, y)$  to the

user  $U_i$  through a secure communication channel, where  $y$  is the remote server's secret number stored in each registered user's smart card.

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and then submit his password  $P_i^*$ . The smart card computes  $CID_i = H(P_i^*) \oplus H(N_i \oplus y \oplus T)$ ,  $E_i = H(CID_i \oplus H(P_i^*))$  and  $C_i = H(T \oplus N_i \oplus E_i \oplus y)$ , where  $T$  is current date and time of user's smart card and sends the login request message  $(CID_i, N_i, C_i, T)$  to the server  $S$ .

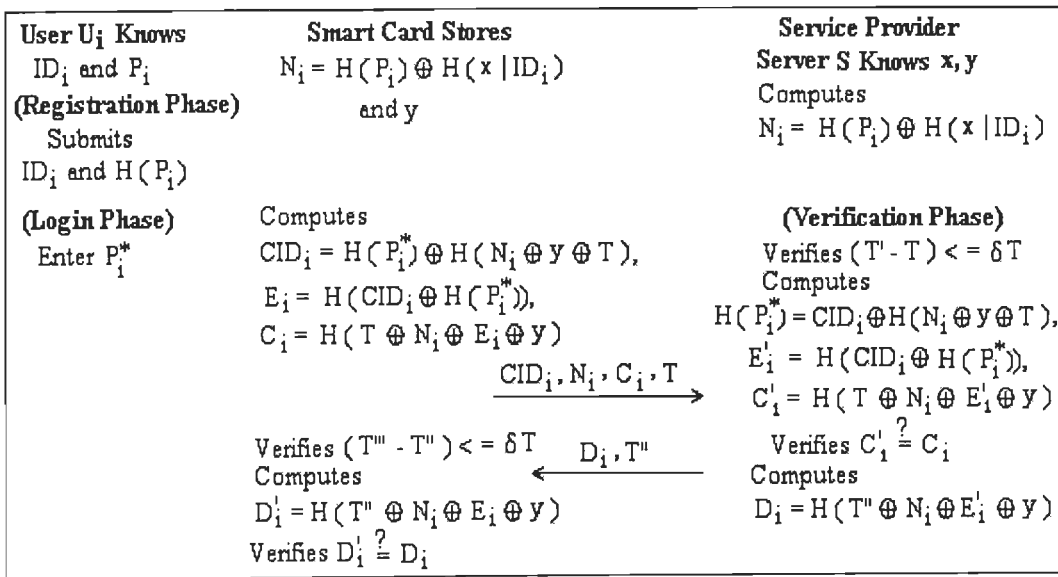


Figure 6.1: Liao et al.'s scheme

## 3. Verification phase

The service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes  $H(P_i^*) = CID_i \oplus H(N_i \oplus y \oplus T)$ ,  $E_i' = H(CID_i \oplus H(P_i^*))$  and  $C_i' = H(T \oplus N_i \oplus E_i' \oplus y)$  and compares the computed value of  $C_i'$  with the received value of  $C_i$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  computes  $D_i = H(T'' \oplus N_i \oplus E_i' \oplus y)$ , where  $T''$  denotes the server's current timestamp and sends the message  $(D_i, T'')$  back to the smart card of user  $U_i$ . On receiving the message  $(D_i, T'')$ , smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  denotes the user's smart card current timestamp. Then the user  $U_i$ 's smart card computes

$D_i' = (T'' \oplus N_i \oplus E_i \oplus y)$  and compares it with the received value of  $D_i$ . This equivalency authenticates the legitimacy of the server  $S$  and the login request is accepted else the connection is interrupted.

#### 4. Password change phase

The user  $U_i$  can change his password without the server's help. A user  $U_i$  inserts his smart card into a card reader and enters his password  $P_i$  corresponding to his smart card and requests to change his password to a new password  $P_i^{new}$ . The smart card computes  $N_i^{new} = N_i \oplus H(P_i) \oplus H(P_i^{new})$  and replaces the values of  $N_i$  stored in its memory with  $N_i^{new}$ .

#### 6.2.1 Cryptanalysis of Liao et al.'s scheme

Liao et al. [81] claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for malicious user attack, Ku and Chang's impersonation attack [70], Awasthi's stolen smart card attack [8] and offline password-guessing attack. Moreover, Liao et al.'s scheme does not maintain the user's anonymity and its password change phase is insecure. In 2006, Yoon and Yoo [172] demonstrated a reflection attack on Liao et al.'s scheme that breaks the mutual authentication.

##### 1. Malicious user attack

An attacker can extract the stored values through some technique like by monitoring their power consumption and reverse engineering techniques as pointed out by Kocher et al. [67] and Messerges et al. [98]. Therefore, a malicious privileged user  $U_k$  can extract the value of  $y$  from his own smart card.

1. Now this malicious privileged user  $U_k$  can intercept the login request message  $(CID_i, N_i, C_i, T)$  of the user  $U_i$  from the public communication channel.
2. This malicious user  $U_k$  can compute the password verifier information of the user  $U_i$  as  $H(P_i) = CID_i \oplus H(N_i \oplus y \oplus T)$  because the malicious user knows the values of  $N_i$ ,  $y$  and  $T$ .
3. Then the malicious user  $U_k$  can compute the values of  $CID_i' = H(P_i) \oplus H(N_i \oplus y \oplus T')$ ,  $E_i' = H(CID_i' \oplus H(P_i))$ ,  $C_i' = H(T' \oplus N_i \oplus E_i' \oplus y)$  and hence can frame fabricated login request message  $(CID_i', N_i, C_i', T')$ . Afterwards, the malicious user  $U_k$  can send this fabricated login request message to the server  $S$ .

4. The service provider server S checks the validity of timestamp T' by checking  $(T'' - T') \leq \delta T$ , where T'' denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay.

5. Afterwards, the server S computes:

$$H(P_i) = CID_i' \oplus H(N_i \oplus y \oplus T')$$

$$E_i' = H(CID_i' \oplus H(P_i))$$

$$C_i'' = H(T' \oplus N_i \oplus E_i' \oplus y)$$

Server S checks  $C_i'' \stackrel{?}{=} C_i'$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request from malicious user  $U_k$  is accepted by the service provider server S.

## 2. Impersonation attack

Ku and Chang [70] demonstrated impersonation attack on Das et al.'s scheme [28]. This attack is also applicable on Liao et al.'s scheme. An attacker can perform impersonation attack as follows.

1. An attacker can intercept a login request message  $(CID_i, N_i, C_i, T)$  of the user  $U_i$  from the public communication channel.
2. Now the attacker gets the current timestamp  $T'$  and computes  $\delta T = T \oplus T'$ ,  $N_i' = N_i \oplus \delta T$  and  $CID_i' = CID_i \oplus \delta T$ .
3. Then an attacker can frame the message  $(CID_i', N_i', C_i, T')$  and sends this login request message to the server S.
4. The server S checks the validity of the timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the server's current timestamp. Then the server S computes:

$$H(P_i') = CID_i' \oplus H(N_i' \oplus y \oplus T')$$

$$= CID_i \oplus \delta T \oplus H(N_i \oplus \delta T \oplus y \oplus T \oplus \delta T)$$

$$= CID_i \oplus \delta T \oplus H(N_i \oplus y \oplus T)$$

$$= H(P_i) \oplus \delta T$$

$$E_i' = H(CID_i' \oplus H(P_i'))$$

$$= H(CID_i \oplus \delta T \oplus H(P_i) \oplus \delta T)$$

$$= H(CID_i \oplus H(P_i))$$

$$= E_i$$

$$\begin{aligned}
C_i' &= H(T' \oplus N_i' \oplus E_i' \oplus y) \\
&= H(T \oplus \delta T \oplus N_i \oplus \delta T \oplus E_i \oplus y) \\
&= H(T \oplus N_i \oplus E_i \oplus y) \\
&= C_i
\end{aligned}$$

The server S compares this computed value of  $C_i'$  with the received value of  $C_i$ . On successful verification, the server S accepts the forged login authentication request. Therefore, the attacker can impersonate as the legitimate user  $U_i$ .

### 3. Stolen smart card attack

Awasthi [8] demonstrated stolen smart card attack on Das et al.'s scheme [28]. This attack is also applicable on Liao et al.'s scheme. Suppose a user  $U_i$  may lose his smart card, which is found by an attacker or an attacker steals the user  $U_i$ 's smart card. The attacker can insert stolen smart card into a card reader to login on to the server S and then submits arbitrary string  $P_i'$  as password. This arbitrary string  $P_i'$  will pass the server verifier test as follows.

1. The smart card computes:

$$\begin{aligned}
CID_i' &= H(P_i') \oplus H(N_i \oplus y \oplus T) \\
E_i' &= H(CID_i' \oplus H(P_i')) \\
C_i' &= H(T \oplus N_i \oplus E_i' \oplus y)
\end{aligned}$$

Here T is current date and time of the smart card.

2. Then the smart card can send the login request message  $(CID_i', N_i, C_i', T)$  to the service provider server S.
3. The service provider server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay.
4. Afterwards, the server S computes:

$$\begin{aligned}
H(P_i') &= CID_i' \oplus H(N_i \oplus y \oplus T) \\
E_i' &= H(CID_i' \oplus H(P_i')) \\
C_i'' &= H(T \oplus N_i \oplus E_i' \oplus y)
\end{aligned}$$

Server S checks  $C_i'' \stackrel{?}{=} C_i'$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted by the service provider server S.

#### 4. Offline password guessing attack

The malicious privileged user  $U_k$  can extract the value of  $y$  from his own smart card.

1. Now this malicious privileged user  $U_k$  can intercept the login request message  $(CID_i, N_i, C_i, T)$  of the user  $U_i$  from the public communication channel.
2. This malicious user  $U_k$  can extract the password verifier information of the user  $U_i$  as  $H(P_i) = CID_i \oplus H(N_i \oplus y \oplus T)$  because the malicious user  $U_k$  knows the values of  $N_i$ ,  $y$  and  $T$ .
3. Then the malicious user  $U_k$  can launch offline dictionary attack on  $H(P_i)$  to know the password  $P_i$  corresponding to the user  $U_i$ .
4. In case the user  $U_i$ 's smart card is stolen by this malicious user, he can masquerade as a legitimate user to the service provider server  $S$  because the malicious user  $U_k$  possesses the smart card of user  $U_i$  and knows the password  $P_i$  corresponding to the smart card of user  $U_i$ .

#### 5. Insecure password change phase

The password change phase of Liao et al.'s [81] scheme is insecure like that of Das et al.'s [28] scheme. If an attacker manages to obtain the smart card of user  $U_i$  for a very short time, he can change the password of user  $U_i$ . An attacker can insert the smart card of user  $U_i$  into a card reader and submits a random string  $P_i'$  as password and requests to change the password with a new password  $P_i^{new}$  without knowing the correct password  $P_i$ .

1. The smart card computes  $N_i^{new} = N_i \oplus H(P_i') \oplus H(P_i^{new})$ .
2. The smart card replaces the value of  $N_i$  stored in its memory with  $N_i^{new}$  and the password gets changed.

The password change phase of Liao et al.'s scheme does not verify the authenticity of password before replacing the value of  $N_i$  in the memory of smart card. Now the registered legitimate user  $U_i$  can not make a valid login request even after getting his smart card back because his old password  $P_i$  has been replaced with some random password by the attacker.

#### 6. User's anonymity

The user  $U_i$  can insert his smart card into a card reader to login on to the server  $S$  and submit his password  $P_i$ .

1. The smart card computes:

$$CID_i = H(P_i) \oplus H(N_i \oplus y \oplus T)$$

$$E_i = H(CID_i \oplus H(P_i))$$

$$C_i = H(T \oplus N_i \oplus E_i \oplus y)$$

Here T is current date and time of the smart card.

2. Then smart card sends the login request message  $(CID_i, N_i, C_i, T)$  to the service provider server S.

The value of  $N_i$  remains same for different login request messages belonging to the same user. Therefore, login request message belonging to same user can be traced out and can be interlinked to derive some information related to the user  $U_i$ . Therefore, Liao et al.'s scheme is not able to preserve the user's anonymity.

## 7. Reflection attack

Yoon and Yoo [172] demonstrated reflection attack on Liao et al.'s scheme. The attacker reuses the user  $U_i$ 's login request message as the response message of a fake server as follows.

1. By observing the login request message  $(CID_i, N_i, C_i, T)$  and the response message  $(D_i, T''')$  of the user  $U_i$ , it is clear that the difference between  $C_i$  and  $D_i$  is only the timestamps, where  $C_i = H(T \oplus N_i \oplus E_i \oplus y)$  and  $D_i = H(T''' \oplus N_i \oplus E_i \oplus y)$ .
2. An attacker can intercept a login request message  $(CID_i, N_i, C_i, T)$  of the user  $U_i$  from the public communication channel.
3. Now the attacker impersonates as the server and reuses the user  $U_i$ 's  $C_i$  and  $T$  values as the response message  $(C_i, T)$  and sends it back to the user  $U_i$  immediately. The response message will pass the server's verification with high probability.

### 6.2.2 Proposed protocol

In this section, we describe a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Liao et al.'s [81] scheme. Figure 6.2 shows the entire protocol structure of the new authentication scheme.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server S via a secure communication channel to register itself to the server S.



Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server S chooses random value  $y_i$  and computes the security parameters  $N_i = H(P_i) \oplus H(y_i | ID_i) \oplus H(x)$ ,  $B_i = y_i \oplus H(P_i)$  and  $V_i = H(ID_i | P_i) \oplus P_i$  and  $D_i = H(y_i | ID_i)$ . The server S chooses the value of  $y_i$  corresponding to each user in such a way that the value of  $D_i$  must be unique for each user. The server S stores  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $D_i$  in its database. Then the server S issues the smart card containing security parameters  $(N_i, B_i, V_i, H(\cdot))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i$ : Smart card

User $U_i$ Knows	Smart Card Stores	Service Provider Server S Knows $x$ Chooses $y_i$
$ID_i$ and $P_i$	$N_i = H(P_i) \oplus H(y_i   ID_i) \oplus H(x)$	Computes $N_i = H(P_i) \oplus H(y_i   ID_i) \oplus H(x)$
<b>(Registration Phase)</b>	$B_i = y_i \oplus H(P_i)$	$B_i = y_i \oplus H(P_i)$
Submits	$V_i = H(ID_i   P_i) \oplus P_i$	$V_i = H(ID_i   P_i) \oplus P_i, D_i = H(y_i   ID_i)$
$ID_i$ and $P_i$		Stores $D_i, y_i \oplus x, ID_i \oplus H(x)$
<b>(Login Phase)</b>	Computes $V_i^* = H(ID_i^*   P_i^*) \oplus P_i^*$	<b>(Verification Phase)</b>
Enter $ID_i^*$ and $P_i^*$	Verifies $V_i^* \stackrel{?}{=} V_i$	Verifies $(T' - T) \leq \delta T$
	Computes $y_i = B_i \oplus H(P_i)$ ,	Computes $D_i^* = CID_i \oplus H(H(x)   T)$
	$H(x) = N_i \oplus H(P_i) \oplus H(y_i   ID_i)$	Extracts
	$CID_i = H(y_i   ID_i) \oplus H(H(x)   T)$	$D_i^*, y_i \oplus x, ID_i \oplus H(x)$
	$M_i = H(H(x)   H(y_i)   T)$	Computes
	$\xrightarrow{CID_i, M_i, T}$	$M_i^* = H(H(x)   H(y_i)   T)$
	Computes Session Key	Verifies $M_i^* \stackrel{?}{=} M_i$
	$S_k = H(ID_i   y_i   H(x)   T)$	Computes Session Key
		$S_k = H(ID_i   y_i   H(x)   T)$

Figure 6.2: Proposed improvement in Liao et al.'s scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server S and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $V_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verifies the legitimacy of user  $U_i$ .

Step 1: Smart card checks  $V_i^* \stackrel{?}{=} V_i$

After verification, the smart card computes  $y_i = B_i \oplus H(P_i)$ ,  $H(x) = N_i \oplus H(P_i) \oplus H(y_i | ID_i)$ ,  $CID_i = H(y_i | ID_i) \oplus H(H(x) | T)$  and  $M_i = H(H(x) | H(y_i) | T)$ , where T is current date and time of the user's smart card. Then the smart card sends login request message  $(CID_i, M_i, T)$  to the service provider server S.

Step 2: Smart card  $\rightarrow S: CID_i, M_i, T$

### 3. Verification and session key agreement phase

After receiving the login request message from the user  $U_i$ , the service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. The server  $S$  computes  $D_i^* = CID_i \oplus H(H(x) | T)$  and finds  $D_i$  corresponding to  $D_i^*$  in its database and then extracts  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $D_i^*$  from its database. Now the server  $S$  computes  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$  because the server  $S$  knows the value of  $x$ . Then, the server  $S$  computes  $M_i^* = H(H(x) | H(y_i) | T)$  and compares the computed value of  $M_i^*$  with the received value of  $M_i$ .

Step 1: Server  $S$  checks  $M_i^* \stackrel{?}{=} M_i$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server  $S$  agree on the common session key as  $S_k = H(ID_i | y_i | H(x) | T)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server  $S$  are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server  $S$  can retrieve the original message because both of them know the common session key.

### 4. Password change phase

The user  $U_i$  can change his password without the help of the server  $S$ . The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $V_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $N_i^{new} = N_i \oplus H(P_i) \oplus H(P_i^{new})$ ,  $B_i^{new} = B_i \oplus H(P_i) \oplus H(P_i^{new})$  and  $V_i^{new} = H(ID_i | P_i^{new}) \oplus P_i^{new}$ . Thereafter, smart card replaces the values of  $N_i$ ,  $B_i$  and  $V_i$  stored in its memory with  $N_i^{new}$ ,  $B_i^{new}$  and  $V_i^{new}$  and password gets changed.

#### 6.2.3 Security analysis

A good password authentication protocol should provide protection from different feasible attacks.

1. **Malicious user attack:** A malicious privileged user  $U_k$  having his own smart card can gather information like  $N_k = H(P_k) \oplus H(y_k | ID_k) \oplus H(x)$ ,  $B_k = y_k \oplus H(P_k)$  and  $V_k = H(ID_k | P_k) \oplus P_k$  from the memory of smart card. This malicious user can not generate smart card specific values of  $CID_i = H(y_i | ID_i) \oplus H(H(x) | T)$  and  $M_i = H(H(x) | H(y_i) | T)$  to masquerades as other legitimate user  $U_i$  to the service provider server  $S$  because the values of  $CID_i$  and  $M_i$  is smart card specific and depend upon the values of  $ID_i$ ,  $y_i$  and  $H(x)$ . Although malicious privileged user  $U_k$  can extract  $H(x)$  from his own smart card but he does not have any method to calculate the values of  $ID_i$  and  $y_i$ . Therefore, the proposed protocol is secure against malicious user attack.
2. **Impersonation attack:** The attacker can attempt to modify a login request message  $(CID_i, M_i, T)$  of user  $U_i$  into  $(CID_i^*, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of the verification and session key agreement phase because an attacker has no way of obtaining the values of  $ID_i$ ,  $y_i$  and  $H(x)$  to compute the valid parameters  $CID_i^*$  and  $M_i^*$ . Therefore, the proposed protocol is secure against impersonation attack.
3. **Stolen smart card attack:** In case a user's smart card is stolen by the attacker, he can extract the information stored in its memory. The attacker can extract  $N_i = H(P_i) \oplus H(y_i | ID_i) \oplus H(x)$ ,  $B_i = y_i \oplus H(P_i)$  and  $V_i = H(ID_i | P_i) \oplus P_i$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
4. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempt to guess the user  $U_i$ 's identity  $ID_i$ , password  $P_i$  and other secret parameters from recorded messages. The attacker first tries to obtain the user  $U_i$ 's verification information  $CID_i = H(y_i | ID_i) \oplus H(H(x) | T)$ ,  $M_i = H(H(x) | H(y_i) | T)$ ,  $T$  and then try to guess the values of  $ID_i$ ,  $y_i$  and  $H(x)$  by offline guessing. Even after gathering this information, the attacker has to guess at least two parameters correctly at the same time out of  $ID_i$ ,  $y_i$  and  $H(x)$ . It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.

5. **Denial of service attack:** In the proposed protocol, smart card checks the validity of user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. An attacker can insert the smart card into the smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $V_i^* = H (ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
6. **Replay attack:** Replaying a login request message  $(CID_i, M_i, T)$  of one session into another session is useless because the user  $U_i$ 's smart card uses current timestamp value  $T$  in each new session, which makes the messages  $CID_i$  and  $M_i$  dynamic and valid for small interval of time. Old messages can not be replayed successfully in other sessions and hence the proposed protocol is secure against message replay attack.
7. **Leak of verifier attack:** In the proposed protocol, the service provider server  $S$  knows secret  $x$  and stores  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $D_i$  in its database. The attacker does not have any technique to find out the value of  $x$  and hence can not calculate  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$ . In case verifier is stolen by breaking into smart card database, the attacker does not have sufficient information to calculate the user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
8. **Server spoofing attack:** In the proposed protocol, malicious server can not compute the session key  $S_k = H (ID_i | y_i | H(x) | T)$  because the malicious server does not know the values of  $ID_i$ ,  $y_i$  and  $H(x)$ . Moreover, the session key is different for same user in different login sessions. Therefore, the proposed protocol is secure against server spoofing attack.
9. **Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to user  $U_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

- 10. Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(CID_i, M_i, T)$  within the valid time frame window but cannot compute the agreed session key  $S_k = H(ID_i | y_i | H(x) | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $ID_i$ ,  $y_i$  and  $H(x)$ . Therefore, the proposed protocol is secure against parallel session attack.
- 11. Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(CID_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(CID_i, M_i, T)$  within the valid time frame window. The attacker can authenticate himself to the server  $S$  but cannot compute the agreed session key  $S_k = H(ID_i | y_i | H(x) | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $ID_i$ ,  $y_i$  and  $H(x)$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.
- 12. Reflection attack:** Yoon and Yoo [172] demonstrated reflection attack on Liao et al.'s scheme [81] by reusing the user  $U_i$ 's login request message  $(CID_i, N_i, C_i, T)$  as the response message  $(C_i, T)$  of a fake server. In the proposed protocol, only the user  $U_i$  authenticates itself to the server  $S$ . After successful authentication, the user  $U_i$  and the server  $S$  agree on the common session key as  $S_k = H(ID_i | y_i | H(x) | T)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server  $S$  are XOR<sup>ed</sup> with the session key. Therefore, the proposed protocol is secure against reflection attack.

#### 6.2.4 Cost and functionality analysis

An efficient authentication scheme must take communication and computation cost into consideration during user's authentication. The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.1. Assume that the identity  $ID_i$ , password  $P_i$ ,  $x$  and  $y_i$  values are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$ ,  $T_E$  and  $T_S$  denote the time complexity for hash function, exponential operation and symmetric key encryption respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_E \gg T_H$ . In the proposed protocol, the parameters stored in the smart card are  $N_i$ ,  $B_i$ ,  $V_i$  and the memory needed in the smart card (E1) is 384 (= 3\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting

message  $(CID_i, M_i, T)$  is  $384 (= 3 \times 128)$  bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration is  $4T_H$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user  $7T_H$  and that of the service provider server are  $5T_H$ .

**Table 6.1**  
**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Liao et al. [81]</b>	<b>Liou et al. [88]</b>	<b>Yoon-Yoo [172]</b>	<b>Chein-Chen [26]</b>	<b>Das et al. [28]</b>
E1	384 bits	256 bits	384 bits	384 bits	384 bits	256 bits
E2	$3 \times 128$ bits	$6 \times 128$ bits	$5 \times 128$ bits	$6 \times 128$ bits	$4 \times 128$ bits	$4 \times 128$ bits
E3	$4 T_H$	$2 T_H$	$3 T_H$	$3 T_H$	$2 T_H$	$2 T_H$
E4	$7 T_H$	$5 T_H$	$4 T_H$	$6 T_H$	$2 T_E + 2 T_S$	$4 T_H$
E5	$5 T_H$	$4 T_H$	$5 T_H$	$4 T_H$	$2 T_H + 2 T_E + 2 T_S$	$3 T_H$

The functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.2. The proposed protocol requires nearly the same computation as other related schemes [81][88][172][28] and requires very less computation as compared to Chien and Chen’s scheme [26] but it is highly secure as compared to the related schemes.

**Table 6.2**  
**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Liao et al. [81]</b>	<b>Liou et al. [88]</b>	<b>Yoon-Yoo [172]</b>	<b>Chein-Chen [26]</b>	<b>Das et al. [28]</b>
Malicious User Attack	No	Yes	Yes	No	No	Yes
Impersonation Attack	No	Yes	Yes	No	No	Yes
Stolen Smart Card Attack	No	Yes	Yes	No	Yes	Yes
Offline Dictionary Attack	No	Yes	Yes	Yes	No	Yes
User’s Anonymity	Yes	Yes	Yes	Yes	Yes	Yes
Reflection Attack	No	Yes	No	No	No	No
Session Key Agreement	Yes	No	No	No	Yes	No
Man-in-the-middle Attack	No	Yes	Yes	No	No	No

**6.3 REVIEW OF LIOU ET AL’S SCHEME**

In this section, we examine a dynamic identity based smart card authentication scheme proposed by Liou et al. [88] in 2006. Liou et al.’s scheme consists of four phases viz.

registration phase, login phase, verification phase and password change phase as summarized in Figure 6.3.

### 1. Registration phase

The user  $U_i$  has to submit his password  $P_i$  to the server  $S$  for registration over a secure communication channel. The server  $S$  computes  $M_i = H(P_i) \oplus H(y)$ ,  $N_i = H(P_i) \oplus H(x)$ , where  $x$  is secret key of the remote server  $S$ . Then the server  $S$  issues the smart card containing secret parameters  $(H(\cdot), M_i, N_i, y)$  to the user  $U_i$  through a secure communication channel, where  $y$  is the server's secret number stored in each registered user's smart card.

### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and then submit his password  $P_i^*$ . The smart card computes  $CID_i = H(P_i^*) \oplus H(M_i \oplus N_i \oplus T)$  and  $E_i = M_i \oplus H(T \oplus y)$ , where  $T$  is current date and time of the user's smart card and sends the login request message  $(CID_i, E_i, T)$  to the service provider server  $S$ .

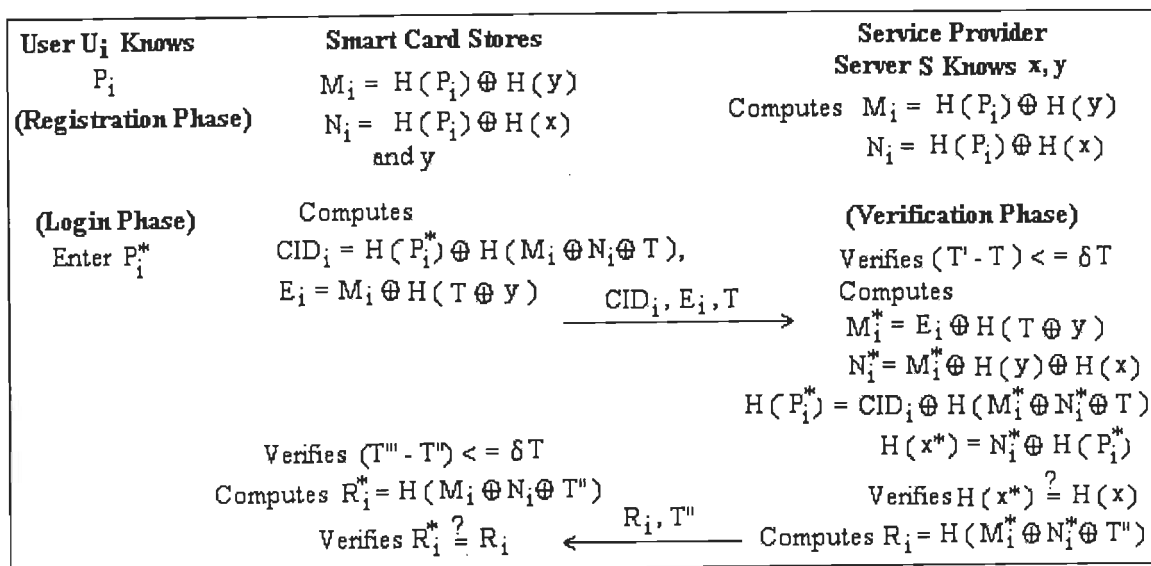


Figure 6.3: Liou et al.'s scheme

### 3. Verification phase

The service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval

for a transmission delay. Afterwards, the server  $S$  computes  $M_i^* = E_i \oplus H(T \oplus y)$ ,  $N_i^* = M_i^* \oplus H(y) \oplus H(x)$ ,  $H(P_i^*) = CID_i \oplus H(M_i^* \oplus N_i^* \oplus T)$ ,  $H(x^*) = N_i^* \oplus H(P_i^*)$  and compares the computed value of  $H(x^*)$  with the known value of  $H(x)$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  computes  $R_i = H(M_i^* \oplus N_i^* \oplus T'')$ , where  $T''$  denotes the server's current timestamp and sends the message  $(R_i, T'')$  back to the smart card of user  $U_i$ . On receiving the message  $(R_i, T'')$ , smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  denotes the user  $U_i$ 's smart card current timestamp. Then the user  $U_i$ 's smart card computes  $R_i^* = H(M_i \oplus N_i \oplus T'')$  and compares it with the received value of  $R_i$ . This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted.

#### 4. Password change phase

The user  $U_i$  can change his password without the server's help. The user  $U_i$  inserts his smart card into a card reader and submits his password  $P_i^*$  corresponding to his smart card. Smart card computes  $H(P_i^*)$  and extracts  $M_i, y$  from its memory to compute  $H(P_i) = M_i \oplus H(y)$ . Then smart card compares the computed value of  $H(P_i)$  with  $H(P_i^*)$  to verify the legitimacy of the user  $U_i$ . If both values match, the authenticity of card holder is verified and then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to submit a new password  $P_i^{new}$ . Then the smart card computes the values  $H(x) = H(P_i) \oplus N_i$ ,  $M_i^{new} = H(P_i^{new}) \oplus H(y)$  and  $N_i^{new} = H(P_i^{new}) \oplus H(x)$ . Finally, the smart card replaces the values of  $M_i$  and  $N_i$  stored in its memory with  $M_i^{new}$  and  $N_i^{new}$  and password gets changed.

#### 6.3.1 Cryptanalysis of Liou et al.'s scheme

Liou et al. [88] claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for impersonation attack, malicious user attack, offline password guessing attack and man-in-the-middle attack.

##### 1. Impersonation attack

Ku and Chang's [70] demonstrated impersonation attack on Das et al.'s scheme [28]. This attack is also applicable on Liou et al.'s [88] scheme. An attacker can perform impersonation attack as follows.



1. The attacker can intercept a login request message  $(CID_i, E_i, T)$  of the user  $U_i$  from the public communication channel.
2. Now the attacker gets the current timestamp  $T'$  and computes  $\delta T = T \oplus T'$ ,  $E_i' = E_i \oplus \delta T$  and  $CID_i' = CID_i \oplus \delta T$ .
3. Then an attacker frames the message  $(CID_i', E_i', T')$  and sends this login request message to the server  $S$ .
4. The server  $S$  checks the validity of the timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the server's current timestamp. Then the server  $S$  computes:

$$\begin{aligned} M_i' &= E_i' \oplus H(T' \oplus y) \\ &= E_i \oplus \delta T \oplus H(T' \oplus y) \\ &= M_i \oplus \delta T \end{aligned}$$

$$\begin{aligned} N_i' &= M_i' \oplus H(y) \oplus H(x) \\ &= M_i \oplus \delta T \oplus H(y) \oplus H(x) \\ &= N_i \oplus \delta T \end{aligned}$$

$$\begin{aligned} H(P_i') &= CID_i' \oplus H(M_i' \oplus N_i' \oplus T') \\ &= CID_i \oplus \delta T \oplus H(M_i \oplus \delta T \oplus N_i \oplus \delta T \oplus T') \\ &= CID_i \oplus \delta T \oplus H(M_i \oplus N_i \oplus T') \\ &= H(P_i) \oplus \delta T \end{aligned}$$

$$\begin{aligned} H(x) &= N_i' \oplus H(P_i') \\ &= N_i \oplus \delta T \oplus H(P_i) \oplus \delta T \\ &= N_i \oplus H(P_i) \end{aligned}$$

The server  $S$  compares this computed value of  $H(x)$  with the known value of  $H(x)$ . On this successful verification, the server  $S$  accepts the forged login authentication request. Therefore, the attacker can impersonate as the legitimate user  $U_i$ .

## 2. Malicious user attack

An attacker can extract the stored values through some technique like by monitoring their power consumption and reverse engineering techniques as pointed out by Kocher et al. [67] and Messerges et al. [98]. Therefore, a malicious privileged user  $U_k$  can extract  $M_k = H(P_k) \oplus H(y)$ ,  $N_k = H(P_k) \oplus H(x)$  and  $y$  from his own smart card. He can find out  $H(x) = N_k \oplus H(P_k)$  because the malicious user  $U_k$  knows his own password  $P_k$  corresponding to his smart card.

1. Now this malicious privileged user  $U_k$  can intercept the login request message  $(CID_i, E_i, T)$  of the user  $U_i$  from the public communication channel.
2. This malicious user  $U_k$  can compute the password verifier information of the user  $U_i$  as  $H(P_i) = CID_i \oplus H(H(y) \oplus H(x) \oplus T)$  because the malicious user knows  $y$  and  $H(x)$ .

$$\begin{aligned} \text{Since } H(M_i \oplus N_i \oplus T) &= H(H(P_i) \oplus H(y) \oplus H(P_i) \oplus H(x) \oplus T) \\ &= H(H(y) \oplus H(x) \oplus T) \end{aligned}$$

Then the malicious user can compute the values of  $M_i = H(P_i) \oplus H(y)$ ,  $N_i = H(P_i) \oplus H(x)$  and hence can frame fabricated login request message  $(CID_i', E_i', T')$  corresponding to the user  $U_i$ , where  $CID_i' = H(P_i) \oplus H(M_i \oplus N_i \oplus T')$  and  $E_i' = M_i \oplus H(T' \oplus y)$ . Afterwards, the malicious user  $U_k$  can send this fabricated login request message to server  $S$ .

3. The service provider server  $S$  checks the validity of timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes:

$$\begin{aligned} M_i &= E_i' \oplus H(T' \oplus y) \\ N_i &= M_i \oplus H(y) \oplus H(x) \\ H(P_i) &= CID_i' \oplus H(M_i \oplus N_i \oplus T') \\ H(x) &= N_i \oplus H(P_i) \end{aligned}$$

Then the server  $S$  compares this computed value of  $H(x)$  with the known value of  $H(x)$ . This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted by the service provider server  $S$ .

### 3. Offline dictionary attack

A user  $U_i$  may lose his smart card, which is found by an attacker or an attacker steals the user's smart card. He can extract  $M_i = H(P_i) \oplus H(y)$ ,  $N_i = H(P_i) \oplus H(x)$  and  $y$  from the memory of user  $U_i$ 's smart card because smart card contains  $(M_i, N_i, y, H(\ ))$ . Then the attacker can find the password information  $H(P_i)$  of the user  $U_i$  as  $H(P_i) = M_i \oplus H(y)$ . Now the attacker can guess different values of  $P_i$  and check its correctness by verifying it with the actual value of  $H(P_i)$ .

### 4. Man-in-the-middle attack

In this type of attack, an attacker can intercept the messages sent between the user and the server and replay these intercepted messages within the valid time frame window. An attacker can act as a user to the server or vice-versa with recorded messages.

1. The malicious privileged user  $U_k$  can intercept the login request message  $(CID_i, E_i, T)$  of the user  $U_i$  to the server  $S$  from the public communication channel.
2. Then this malicious privileged user  $U_k$  can start a new session with the server  $S$  by sending a login request message  $(CID_k, E_k, T')$ .
3. After receiving the login request, the server  $S$  check the validity of timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the server's current timestamp. Then the server  $S$  computes:

$$M_k = E_k \oplus H(T \oplus y)$$

$$N_k = M_k \oplus H(y) \oplus H(x)$$

$$H(P_k) = CID_k \oplus H(M_k \oplus N_k \oplus T)$$

$$H(x) = N_k \oplus H(P_k)$$

Then the server  $S$  compares this computed value of  $H(x)$  with the known value of  $H(x)$ . This equivalency authenticates the legitimacy of the user  $U_k$  and the login request is accepted by the service provider server  $S$ .

4. The server  $S$  computes  $R_k = H(M_k \oplus N_k \oplus T'')$  and sends the message  $(R_k, T'')$  back to the user  $U_k$ .
5. Now the user  $U_k$  immediately sends the message  $(R_k, T'')$  to the user  $U_i$ .
6. The user  $U_i$  checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  denotes the user  $U_i$ 's smart card current timestamp. The user  $U_i$  computes  $R_i = H(M_i \oplus N_i \oplus T'')$  and compares it with the received value of  $R_k$ .

$$R_i = H(M_i \oplus N_i \oplus T'')$$

$$= H(H(P_i) \oplus H(y) \oplus H(P_i) \oplus H(x) \oplus T'')$$

$$= H(H(y) \oplus H(x) \oplus T'')$$

$$R_k = H(M_k \oplus N_k \oplus T'')$$

$$= H(H(P_k) \oplus H(y) \oplus H(P_k) \oplus H(x) \oplus T'')$$

$$= H(H(y) \oplus H(x) \oplus T'')$$

This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted by the user  $U_i$ . Thus the user  $U_k$  acts as middle-man between the user  $U_i$  and the server  $S$  and masquerades as the legitimate server  $S$  to the user  $U_i$ .

### 6.3.2 Proposed protocol

In this section, we describe a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Liou et al.'s [88] scheme. The proposed protocol consists of four phases viz. registration phase, login phase, authentication phase and password change phase as summarized in Figure 6.4.

#### 1. Registration phase

The user  $U_i$  has to submit his unique identity  $ID_i$  and password  $P_i$  to the server  $S$  for registration over a secure communication channel.

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  computes the security parameters  $A_i = H(x | y_i)$ ,  $B_i = H(ID_i | P_i) \oplus P_i \oplus H(x | y_i)$ ,  $C_i = H(x | y_i) \oplus H(P_i)$  and  $D_i = H(ID_i | P_i) \oplus H(x)$ . The server  $S$  chooses the value of  $y_i$  corresponding to each user in such a way that the value of  $A_i$  must be unique for each user. The server  $S$  stores  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $A_i$  in its database. Then the server  $S$  issues the smart card containing security parameters  $(B_i, C_i, D_i, H(\cdot))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i: \text{Smart card}$

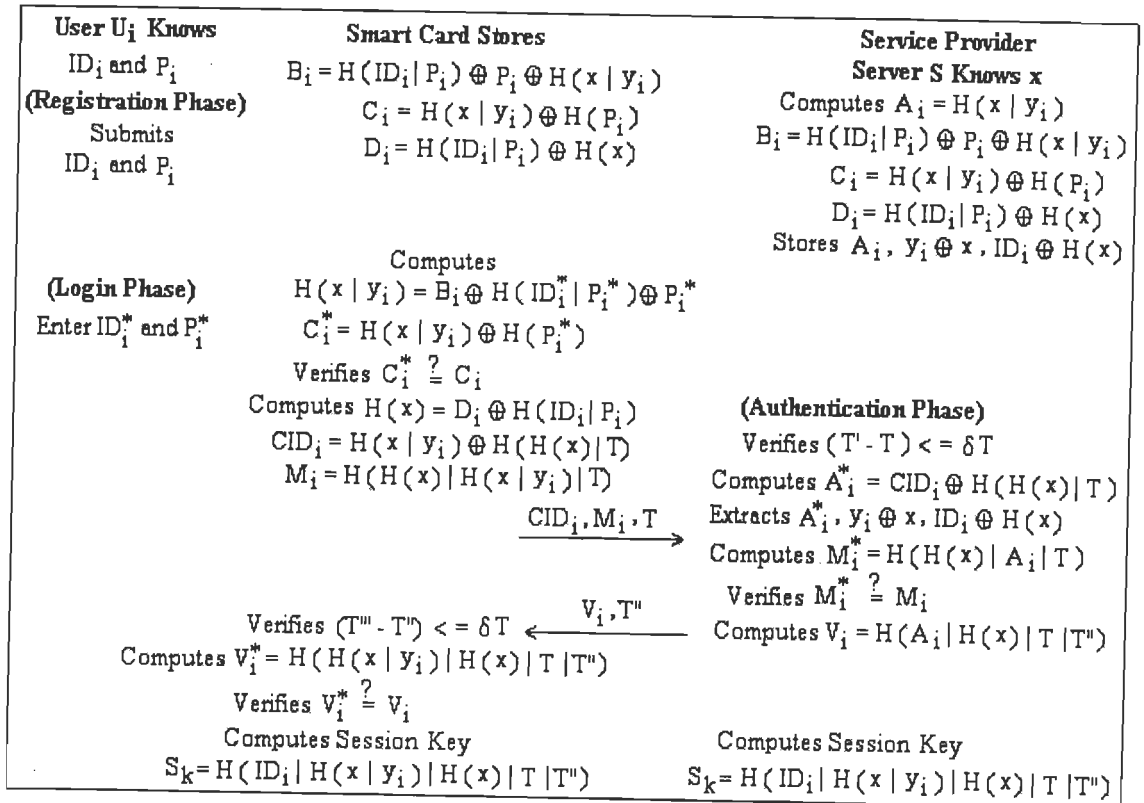


Figure 6.4: Proposed improvement in Liou et al.'s scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $H(x | y_i) = B_i \oplus H(ID_i^* | P_i^*) \oplus P_i^*$ ,  $C_i^* = H(x | y_i) \oplus H(P_i^*)$  and compares the computed value of  $C_i^*$  with the stored value of  $C_i$  in its memory to verifies the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $C_i^* \stackrel{?}{=} C_i$

After verification, the smart card computes  $H(x) = D_i \oplus H(ID_i | P_i)$ ,  $CID_i = H(x | y_i) \oplus H(H(x) | T)$  and  $M_i = H(H(x) | H(x | y_i) | T)$ , where  $T$  is current date and time of the user's smart card. Then the smart card sends login request message  $(CID_i, M_i, T)$  to the service provider server  $S$ .

Step 2: Smart card  $\rightarrow$   $S: CID_i, M_i, T$

## 3. Authentication phase

After receiving the login request message from the user  $U_i$ , the service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. The server  $S$  computes  $A_i^* = CID_i \oplus H(H(x) | T)$  and finds  $A_i$  corresponding to  $A_i^*$  in its database and then extracts  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $A_i$  from its database. Now the server  $S$  computes  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$  because the server  $S$  knows the value of  $x$ . Then the server  $S$  computes  $M_i^* = H(H(x) | A_i | T)$  and compares the computed value of  $M_i^*$  with the received value of  $M_i$ .

Step 1: Server  $S$  checks  $M_i^* \stackrel{?}{=} M_i$

If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  acquires the current timestamp  $T''$  and computes  $V_i = H(A_i | H(x) | T | T'')$  and sends the message  $(V_i, T'')$  back to the smart card of the user  $U_i$ .

Step 2:  $S \rightarrow$  Smart card:  $V_i, T''$

On receiving the message  $(V_i, T'')$ , the user  $U_i$ 's smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  is current date and time of the smart card. Then the smart card computes  $V_i^* = H(H(x | y_i) | H(x) | T | T'')$  and compares it with the received value of  $V_i$ .

Step 3: Smart card checks  $V_i^* \stackrel{?}{=} V_i$

This equivalency authenticates the legitimacy of the service provider server S and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S agree on the common session key as  $S_k = H (ID_i | H (x | y_i) | H (x) | T | T')$ .

#### 4. Password change phase

The user  $U_i$  can change his password without the server's help. The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $H (x | y_i) = B_i \oplus H (ID_i^* | P_i^*) \oplus P_i^*$ ,  $C_i^* = H (x | y_i) \oplus H (P_i^*)$  and compares the computed value of  $C_i^*$  with the stored value of  $C_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $B_i^{new} = H (ID_i | P_i^{new}) \oplus P_i^{new} \oplus H (x | y_i)$ ,  $C_i^{new} = H (x | y_i) \oplus H (P_i^{new})$  and  $D_i^{new} = D_i \oplus H (ID_i | P_i) \oplus H (ID_i | P_i^{new})$ . Thereafter, smart card replaces the values of  $B_i$ ,  $C_i$  and  $D_i$  stored in its memory with  $B_i^{new}$ ,  $C_i^{new}$  and  $D_i^{new}$  and password gets changed.

#### 6.3.3 Security analysis

A good password authentication protocol should provide protection from different possible attacks relevant to that protocol.

1. **Impersonation attack:** An attacker can attempt to modify a login request message  $(CID_i, M_i, T)$  into  $(CID_i^*, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of the authentication phase because the attacker requires to know the values  $A_i$  and  $H (x)$  to compute the valid parameters  $CID_i^*$  and  $M_i^*$ . Moreover, the attacker requires to know  $ID_i$  to compute the session key  $S_k = H (ID_i | H (x | y_i) | H (x) | T | T')$  between the user  $U_i$  and the server S. Therefore, the proposed protocol is secure against impersonation attack.
2. **Malicious user attack:** A malicious privileged user  $U_k$  having his own smart card can gather information like  $B_k = H (ID_k | P_k) \oplus P_k \oplus H (x | y_k)$ ,  $C_k = H (x | y_k) \oplus H (P_k)$  and  $D_k = H (ID_k | P_k) \oplus H (x)$  from the memory of smart card. This malicious user can not

generate smart card specific values of  $CID_i = H(x | y_i) \oplus H(H(x) | T)$  and  $M_i = H(H(x) | H(x | y_i) | T)$  to masquerades as other legitimate user  $U_i$  to the service provider server  $S$  because the values of  $CID_i$  and  $M_i$  is smart card specific and depend upon the values of  $x$  and  $y_i$ . Although, the malicious user  $U_k$  can extract  $H(x)$  from his own smart card but he does not have any method to calculate the values of  $x$  and  $y_i$ . Moreover, the malicious user  $U_k$  should know  $ID_i$  to compute the session key  $S_k = H(ID_i | H(x | y_i) | H(x) | T | T')$  between the user  $U_i$  and the server  $S$ . Therefore, the proposed protocol is secure against malicious user attack.

3. **Stolen smart card attack:** In case a user's smart card is stolen by an attacker, he can extract the information stored in the smart card. An attacker can extract  $B_i = H(ID_i | P_i) \oplus P_i \oplus H(x | y_i)$ ,  $C_i = H(x | y_i) \oplus H(P_i)$  and  $D_i = H(ID_i | P_i) \oplus H(x)$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, an attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
4. **Offline dictionary attack:** In offline dictionary attack, an attacker can record messages and attempts to guess the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  from the recorded messages. An attacker first tries to obtains some user or server verification information such as  $CID_i = H(x | y_i) \oplus H(H(x) | T)$ ,  $M_i = H(H(x) | H(x | y_i) | T)$ ,  $V_i = H(A_i | H(x) | T | T')$  and then tries to guess  $x$  and  $y_i$  by offline dictionary attack. Even after gathering this information, the attacker has to guess both these parameters correctly at the same time. It is not possible to guess both parameters correctly at the same time. Therefore, the proposed protocol is secure against offline dictionary attack.
5. **Man-in-the-middle attack:** In the proposed protocol, an attacker can intercept the login request message  $(CID_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by replaying the login request message  $(CID_i, M_i, T)$  with in the valid time frame window. An attacker can authenticate itself to the server  $S$  as well as to the legitimate user  $U_i$  but can not compute the session key  $S_k = H(ID_i | H(x | y_i) | H(x) | T | T')$  because the attacker does not know the values of  $ID_i$ ,  $x$  and  $y_i$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

6. **Denial of service attack:** In the proposed protocol, smart card checks the validity of user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. An attacker can insert the smart card into the smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $H(x | y_i) = B_i \oplus H(ID_i^* | P_i^*) \oplus P_i^*$ ,  $C_i^* = H(x | y_i) \oplus H(P_i^*)$  and compares the computed value of  $C_i^*$  with the stored value of  $C_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
7. **Replay attack:** Replaying a message of one session into another session is useless because the user  $U_i$ 's smart card and the server  $S$  uses current timestamp values as  $T$  and  $T''$  in each new session, which make all the messages  $CID_i$ ,  $M_i$  and  $V_i$  dynamic and valid for small interval of time. Old replayed messages are not valid in current session and hence proposed protocol is secure against message replay attack.
8. **Leak of verifier attack:** In the proposed protocol, the service provider server  $S$  knows the secret  $x$  and stores  $y_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to user  $U_i$ 's  $A_i = H(x | y_i)$  value in its database. An attacker does not have any technique to find out the value of  $x$  and hence can not calculate  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$ . Therefore, the proposed protocol is secure against leak of verifier attack.
9. **Server spoofing attack:** Malicious server can not generate the valid value of  $V_i = H(A_i | H(x) | T | T'')$  meant for the smart card of user  $U_i$  because the malicious server has to know the values of  $x$  and  $y_i$  to generate the valid value of  $V_i$  corresponding to user  $U_i$ 's smart card. Moreover, the malicious server requires to know  $ID_i$  to compute the session key. The proposed protocol provides mutual authentication to withstand the server spoofing attack. Therefore, the proposed protocol is secure against server spoofing attack.
10. **Online dictionary attack:** In the proposed protocol, an attacker has to get the valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, an attacker gets a very few chances to guess the identity  $ID_i$  and password  $P_i$  because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess identity



$ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

**11. Parallel session attack:** The attacker can masquerade as legitimate user  $U_i$  by replaying a login request message  $(CID_i, M_i, T)$  with in the valid time frame window. However, an attacker can not compute the agreed session key  $S_k = H (ID_i | H (x | y_i) | H (x) | T | T'')$  because the attacker does not know the values of  $ID_i, x$  and  $y_i$ . Therefore, the proposed protocol is secure against parallel session attack.

### 6.3.4 Cost and functionality analysis

The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.3. Assume that the identity  $ID_i$ , password  $P_i$ ,  $x$  and  $y_i$  values are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H, T_E$  and  $T_S$  denote the time complexity for hash function, exponential operation and symmetric key encryption respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_E \gg T_H$ . In the proposed protocol, the parameters stored in the smart card are  $B_i, C_i, D_i$  and the memory needed in the smart card (E1) is 384 ( $= 3 * 128$ ) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $(CID_i, M_i, T)$  and  $(V_i, T'')$  is 640 ( $= 5 * 128$ ) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration is  $4T_H$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, both the computation cost of the user and that of the service provider server are  $6T_H$  and  $5T_H$  respectively.

**Table 6.3**

**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Liou et al. [88]</b>	<b>Yoon-Yoo [172]</b>	<b>Liao et al. [81]</b>	<b>Chein-Chen [26]</b>	<b>Das et al. [28]</b>
E1	384 bits	384 bits	384 bits	256 bits	384 bits	256 bits
E2	5 * 128 bits	5 * 128 bits	6 * 128 bits	6 * 128 bits	4 * 128 bits	4 * 128 bits
E3	4 $T_H$	3 $T_H$	3 $T_H$	2 $T_H$	2 $T_H$	2 $T_H$
E4	6 $T_H$	4 $T_H$	6 $T_H$	5 $T_H$	2 $T_E + 2 T_S$	4 $T_H$
E5	5 $T_H$	5 $T_H$	4 $T_H$	4 $T_H$	2 $T_H + 2 T_E + 2 T_S$	3 $T_H$

The functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.4. The proposed protocol requires nearly the same computation as other related schemes [88][172][81][28] and requires very less computation as compared to Chien and Chen scheme [26] but it is highly secure as compared to the related schemes.

**Table 6.4**

**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Liou et al. [88]</b>	<b>Yoon-Yoo [172]</b>	<b>Liao et al. [81]</b>	<b>Chien-Chen [26]</b>	<b>Das et al. [28]</b>
User's Anonymity	Yes	Yes	Yes	Yes	Yes	Yes
Session Key Agreement	Yes	No	No	No	Yes	No
Impersonation Attack	No	Yes	No	Yes	No	Yes
Malicious User Attack	No	Yes	No	Yes	No	Yes
Offline Dictionary Attack	No	Yes	Yes	Yes	No	Yes
Man-in-the-middle Attack	No	Yes	No	Yes	No	No
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	No

## 6.4 REVIEW OF WANG ET AL.'S SCHEME

In this section, we examine the smart card based remote user authentication scheme proposed by Wang et al. [151] in 2009. Wang et al.'s scheme consists of four phases viz. registration phase, login phase, verification phase and password change phase as summarized in Figure 6.5.

### 1. Registration phase

The user  $U_i$  registers with the server  $S$  by submitting his unique identity  $ID_i$  over a secure communication channel. The server  $S$  computes  $N_i = H(P_i) \oplus H(x) \oplus ID_i$ , where  $x$  is the secret key of remote server  $S$  and  $P_i$  is the password of user  $U_i$  chosen by the remote server  $S$ . Then the server  $S$  issues the smart card containing secret parameters  $(H(\cdot), N_i, y)$  and also sends the password  $P_i$  to the user  $U_i$  through a secure communication channel, where  $y$  is the remote server's secret number stored in each registered user's smart card.

### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the service provider server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $CID_i = H(P_i^*) \oplus H(N_i \oplus y \oplus T) \oplus ID_i^*$ , where  $T$  is current date and time of the user's smart card and sends the login request message  $(ID_i^*, CID_i, N_i, T)$  to the server  $S$ .

### 3. Verification phase

The service provider server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where T' denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server S computes  $H(P_i^{**}) = CID_i \oplus H(N_i \oplus y \oplus T) \oplus ID_i^*$ ,  $ID_i^{**} = N_i \oplus H(P_i^{**}) \oplus H(x)$  and compares the computed value of  $ID_i^{**}$  with the received value of  $ID_i^*$ . If they are not equal, the server S rejects the login request and terminates this session. Otherwise, the server S computes  $A_i = H(H(P_i^{**}) \oplus y \oplus T')$  and sends the message  $(A_i, T')$  back to the smart card of user  $U_i$ . On receiving the message  $(A_i, T')$ , smart card checks the validity of timestamp T' by checking  $(T'' - T') \leq \delta T$ , where T'' denotes the user  $U_i$ 's smart card current timestamp. Then the user  $U_i$ 's smart card computes  $A_i' = H(H(P_i^*) \oplus y \oplus T')$  and compares it with the received value of  $A_i$ . This equivalency authenticates the legitimacy of the service provider server S and the login request is accepted else the connection is interrupted.

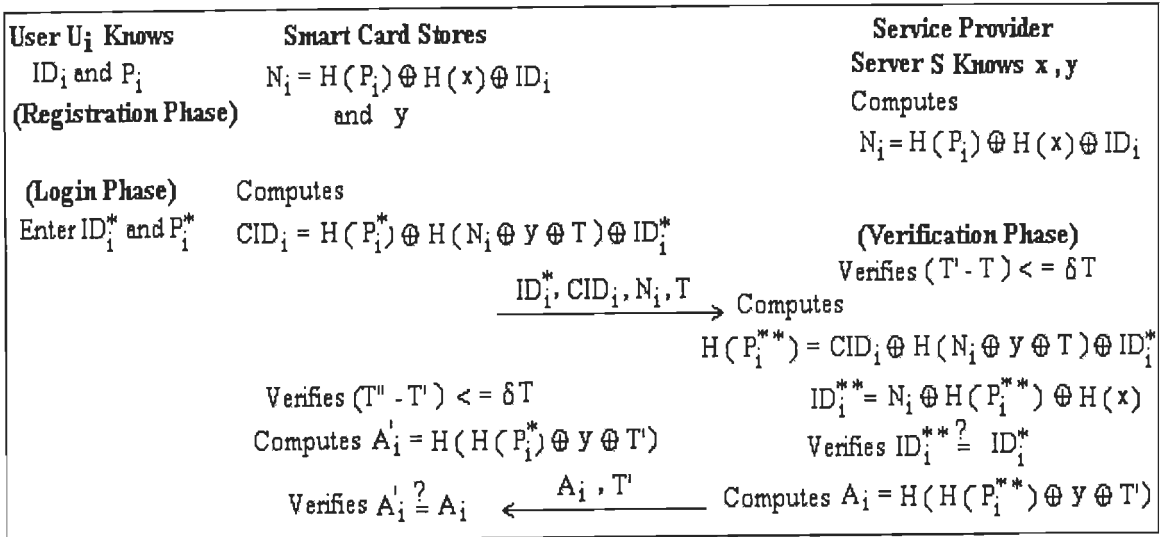


Figure 6.5: Wang et al.'s scheme

### 4. Password change phase

The user  $U_i$  can change his password without the server's help. A user  $U_i$  inserts his smart card into a card reader and submits his password  $P_i$  corresponding to his smart card and request to change his password with a new password  $P_i^{new}$ . The smart card computes  $N_i^{new} = N_i \oplus H(P_i) \oplus H(P_i^{new})$  and replaces the values of  $N_i$  stored in its memory with  $N_i^{new}$  and the password gets changed.

### 6.4.1 Cryptanalysis of Wang et al.'s scheme

Wang et al. [151] claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for impersonation attack, stolen smart card attack, offline password guessing attack and denial of service attack. Their scheme also fails to preserve the user's anonymity in contrast to Das et al.'s scheme.

#### 1. Impersonation attack

Ku and Chang's [70] demonstrated impersonation attack on Das et al.'s scheme [28]. This attack is also applicable on Wang et al.'s [151] scheme. An attacker can perform impersonation attack as follows.

1. An attacker can intercept a login request message  $(ID_i^*, CID_i, N_i, T)$  of the user  $U_i$  from the public communication channel.
2. Now the attacker gets the current timestamp  $T'$  and computes  $\delta T = T \oplus T'$ ,  $N_i' = N_i \oplus \delta T$  and  $CID_i' = CID_i \oplus \delta T$ .
3. Then an attacker can frame the message  $(ID_i^*, CID_i', N_i', T')$  and sends this login request message to the server  $S$ .
4. The server  $S$  checks the validity of the timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the server's current timestamp. Then the server  $S$  computes:

$$\begin{aligned} H(P_i^{**}) &= CID_i' \oplus H(N_i' \oplus y \oplus T') \oplus ID_i^* \\ &= CID_i \oplus \delta T \oplus H(N_i \oplus \delta T \oplus y \oplus T \oplus \delta T) \oplus ID_i^* \\ &= CID_i \oplus \delta T \oplus H(N_i \oplus y \oplus T) \oplus ID_i^* \\ &= H(P_i^*) \oplus \delta T \end{aligned}$$

$$\begin{aligned} ID_i^{**} &= N_i' \oplus H(x) \oplus H(P_i^{**}) \\ &= N_i \oplus \delta T \oplus H(x) \oplus H(P_i^*) \oplus \delta T \\ &= N_i \oplus H(x) \oplus H(P_i^*) \\ &= ID_i^* \end{aligned}$$

The server  $S$  compares this computed value of  $ID_i^{**}$  with the received value of  $ID_i^*$ . On this successful verification, the server  $S$  accepts the forged login authentication request. Therefore, the attacker can impersonate as the legitimate user  $U_i$ .

## 2. Stolen smart card attack

A malicious privileged user  $U_k$  having his own smart card can gather information  $N_k = H(P_k) \oplus H(x) \oplus ID_k$  from his own smart card. He can find out the value of  $H(x)$  as  $H(x) = N_k \oplus H(P_k) \oplus ID_k$  because the malicious user  $U_k$  knows his own identity  $ID_k$  and password  $P_k$  corresponding to his smart card. In case another user  $U_i$ 's smart card is stolen by this malicious user, he can extract the information  $N_i = H(P_i) \oplus H(x) \oplus ID_i$  from the memory of smart card.

1. Now the malicious user  $U_k$  chooses any arbitrary combination of identity  $ID_i^*$  and password  $P_i^*$  so that  $N_i^* = H(P_i^*) \oplus H(x) \oplus ID_i^*$  is equal to extracted value of  $N_i$ .
2. Then the malicious user  $U_k$  can insert the stolen smart card into a card reader to login on to the server  $S$  and can submit fabricated identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $CID_i = H(P_i^*) \oplus H(N_i \oplus y \oplus T) \oplus ID_i^*$ , where  $T$  is current date and time of the user's smart card and sends the login request message  $(ID_i^*, CID_i, N_i, T)$  to the server  $S$ .
3. The service provider server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes:

$$\begin{aligned} H(P_i^{**}) &= CID_i \oplus H(N_i \oplus y \oplus T) \oplus ID_i^* \\ &= H(P_i^*) \\ ID_i^{**} &= N_i \oplus H(x) \oplus H(P_i^{**}) \\ &= N_i^* \oplus H(x) \oplus H(P_i^*) \quad \text{because } N_i^* = N_i \end{aligned}$$

Then server  $S$  compares  $ID_i^{**}$  with the received value of  $ID_i^*$ . This equivalency authenticates the malicious user  $U_k$  to the service provider server  $S$ .

4. Then the server  $S$  computes  $A_i = H(H(P_i^{**}) \oplus y \oplus T')$  and sends the message  $(A_i, T')$  back to the smart card of user  $U_i$ . On receiving the message  $(A_i, T')$ , smart card checks the validity of timestamp  $T'$  by checking  $(T'' - T') \leq \delta T$ , where  $T''$  denotes the user  $U_i$ 's smart card current timestamp. Then the user  $U_i$ 's smart card computes  $A_i' = H(H(P_i^*) \oplus y \oplus T')$  and compares it with the received value of  $A_i$ . This equivalency authenticates the legitimacy of the server  $S$  and the login request is accepted.

### 3. Offline password guessing attack

The malicious privileged user  $U_k$  having his own smart card can gather information  $N_k = H(P_k) \oplus H(x) \oplus ID_k$  from smart card. He can find out the value of  $H(x)$  as  $H(x) = N_k \oplus H(P_k) \oplus ID_k$  because the malicious user  $U_k$  knows his own identity  $ID_k$  and password  $P_k$ .

1. Now this malicious privileged user  $U_k$  can intercept the login request message  $(ID_i, CID_i, N_i, T)$  of the user  $U_i$  from the public communication channel.
2. This malicious user  $U_k$  can extract the password verifier information of the user  $U_i$  as  $H(P_i) = N_i \oplus H(x) \oplus ID_i$  because the malicious user  $U_k$  knows the values of  $H(x)$ ,  $ID_i$  and  $N_i$ .
3. Then malicious user  $U_k$  can launch offline dictionary attack on  $H(P_i)$  to know the password  $P_i$  corresponding to the smart card of user  $U_i$ .
4. In case user  $U_i$ 's smart card is stolen by this malicious user, he can masquerade as a legitimate user to the server  $S$  because the malicious user possesses the smart card of user  $U_i$ , knows the identity  $ID_i$  and password  $P_i$  corresponding to the user  $U_i$ .

### 4. Denial of service attack

The password change phase of Wang et al.'s [151] scheme is insecure like that of Das et al.'s [28] scheme. If an attacker manages to obtain the smart card of user  $U_i$  for a very short time, he can change the password of user  $U_i$ . An attacker can insert the smart card of the user  $U_i$  into a card reader and submits a random string  $K$  as password and request to change the password with a new password  $P_i^{new}$  without knowing the correct password  $P_i$ . The smart card computes  $N_i^{new} = N_i \oplus H(K) \oplus H(P_i^{new})$  and updates the values of  $N_i$  stored in its memory with  $N_i^{new}$  and the password gets changed. The password change phase of Wang et al.'s [151] scheme does not verify the authenticity of password before replacing the value of  $N_i$  in the memory of smart card. Once the value of  $N_i$  is updated in the memory of smart card, legitimate user  $U_i$  can not login successfully even after getting his smart card back because the value of  $ID_i$  can not be verified in verification phase. Hence, denial of service attack can be launched on the user  $U_i$ 's smart card.

### 5. User's anonymity

The user's identity  $ID_i^*$  is transferred in the clear text in login request message  $(ID_i^*, CID_i, N_i, T)$  and hence the different login request messages belonging to the same user can be

traced out and can be interlinked to derive some information related to the user  $U_i$ . Hence Wang et al.'s [151] scheme is not able to preserve the user's anonymity.

### 6.4.2 Proposed protocol

In this section, we describe a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Wang et al.'s [151] scheme. Figure 6.6 shows the entire protocol structure of the new authentication scheme.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  via a secure communication channel to register itself to the server  $S$ .

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  chooses random value  $y_i$  and computes the security parameters  $N_i = H(ID_i | P_i) \oplus H(x)$ ,  $A_i = H(ID_i | P_i) \oplus P_i \oplus H(y_i)$ ,  $B_i = y_i \oplus ID_i \oplus P_i$  and  $D_i = H(H(ID_i | y_i) \oplus x)$ . The server  $S$  chooses the value of  $y_i$  corresponding to each user in such a way so that the value of  $D_i$  must be unique for each user. The server  $S$  stores  $y_i \oplus x$  and  $ID_i \oplus H(x | y_i)$  corresponding to  $D_i$  in its database. Then the server  $S$  issues the smart card containing security parameters  $(N_i, A_i, B_i, H(\cdot))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i: \text{Smart card}$

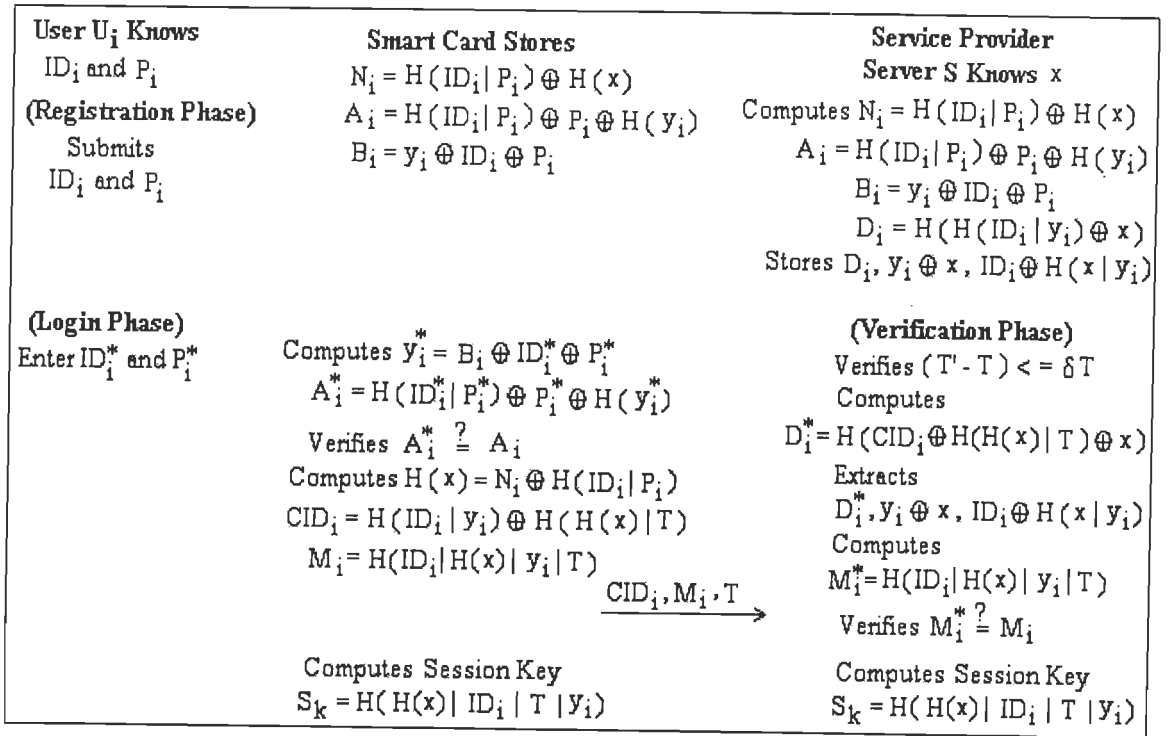
#### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $y_i^* = B_i \oplus ID_i^* \oplus P_i^*$ ,  $A_i^* = H(ID_i^* | P_i^*) \oplus P_i^* \oplus H(y_i^*)$  and compares the computed value of  $A_i^*$  with the stored value of  $A_i$  in its memory to verify the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $A_i^* \stackrel{?}{=} A_i$

After verification, smart card computes  $H(x) = N_i \oplus H(ID_i | P_i)$ ,  $CID_i = H(ID_i | y_i) \oplus H(H(x) | T)$  and  $M_i = H(ID_i | H(x) | y_i | T)$ , where  $T$  is current date and time of the user's smart card. Then smart card sends login request message  $(CID_i, M_i, T)$  to the server  $S$ .

Step 2: Smart card  $\rightarrow S: CID_i, M_i, T$



**Figure 6.6: Proposed improvement in Wang et al.'s scheme**

### 3. Verification and session key agreement phase

After receiving the login request message from the user  $U_i$ , the service provider server S checks the validity of timestamp T by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server S and  $\delta T$  is permissible time interval for a transmission delay. The server S computes  $D_i^* = H(CID_i \oplus H(H(x) | T) \oplus x)$  and finds  $D_i$  corresponding to  $D_i^*$  in its database and then extracts  $y_i \oplus x$  and  $ID_i \oplus H(x | y_i)$  corresponding to  $D_i^*$  from its database. Now the server S computes  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x | y_i)$  because the server S knows the value of  $x$ . Then, the server S computes  $M_i^* = H(ID_i | H(x) | y_i | T)$  and compares the computed value of  $M_i^*$  with the received value of  $M_i$ .

Step 1: Server S checks  $M_i^* \stackrel{?}{=} M_i$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S agree on the common session key as  $S_k = H(H(x) | ID_i | T | y_i)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server S can retrieve the original messages between themselves because both of them know the common session key.



#### 4. Password change phase

The user  $U_i$  can change his password without the help of the server  $S$ . The identity and the password of the user are verified before the password update procedure. The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $y_i^* = B_i \oplus ID_i^* \oplus P_i^*$ ,  $A_i^* = H(ID_i^* | P_i^*) \oplus P_i^* \oplus H(y_i^*)$  and compares the computed value of  $A_i^*$  with the stored value of  $A_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $N_i^{new} = N_i \oplus H(ID_i | P_i) \oplus H(ID_i | P_i^{new})$ ,  $A_i^{new} = H(ID_i | P_i^{new}) \oplus P_i^{new} \oplus H(y_i^*)$ ,  $B_i^{new} = y_i^* \oplus ID_i \oplus P_i^{new}$ . Thereafter, smart card updates the values of  $N_i$ ,  $A_i$  and  $B_i$  stored in its memory with  $N_i^{new}$ ,  $A_i^{new}$  and  $B_i^{new}$  and the password gets changed.

##### 6.4.3 Security analysis

A good password authentication protocol should provide protection from different feasible attacks.

- 1. Impersonation attack:** The attacker can attempt to modify a login request message  $(CID_i, M_i, T)$  of the user  $U_i$  into  $(CID_i^*, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of the verification and session key agreement phase because the attacker has no way of obtaining the values of  $ID_i$ ,  $H(x)$  and  $y_i$  to compute the valid parameters  $CID_i^*$  and  $M_i^*$  corresponding to the user  $U_i$ . Therefore, the proposed protocol is secure against impersonation attack.
- 2. Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by the attacker, he can extract the information stored in its memory. The attacker can extract  $N_i = H(ID_i | P_i) \oplus H(x)$ ,  $A_i = H(ID_i | P_i) \oplus P_i \oplus H(y_i)$  and  $B_i = y_i \oplus ID_i \oplus P_i$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.

3. **Offline dictionary attack:** The attacker first tries to obtain the user  $U_i$ 's verification information  $CID_i = H(ID_i | y_i) \oplus H(H(x) | T)$ ,  $M_i = H(ID_i | H(x) | y_i | T)$ ,  $T$  and then try to guess the values of  $ID_i$ ,  $y_i$  and  $H(x)$  by offline guessing. Even after gathering this information, the attacker has to guess all three parameters  $ID_i$ ,  $y_i$  and  $H(x)$  correctly at the same time. It is not possible to guess all three parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.
4. **Denial of service attack:** In the proposed protocol, smart card of user  $U_i$  checks the validity of user identity  $ID_i$  and password  $P_i$  before password update procedure. An attacker has to insert the smart card of user  $U_i$  into the smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly. Since the smart card computes  $y_i^* = B_i \oplus ID_i^* \oplus P_i^*$ ,  $A_i^* = H(ID_i^* | P_i^*) \oplus P_i^* \oplus H(y_i^*)$  and compares the computed value of  $A_i^*$  with the stored value of  $A_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
5. **Malicious user attack:** A malicious privileged user  $U_k$  having his own smart card can gather information like  $N_k = H(ID_k | P_k) \oplus H(x)$ ,  $A_k = H(ID_k | P_k) \oplus P_k \oplus H(y_k)$  and  $B_k = y_k \oplus ID_k \oplus P_k$  from the memory of smart card. This malicious user  $U_k$  can not generate smart card specific values of  $CID_i = H(ID_i | y_i) \oplus H(H(x) | T)$  and  $M_i = H(ID_i | H(x) | y_i | T)$  to masquerade as other legitimate user  $U_i$  to the server  $S$  because the values of  $CID_i$  and  $M_i$  is smart card specific and depend upon the values of  $ID_i$ ,  $y_i$  and  $H(x)$ . Although malicious user  $U_k$  can extract  $H(x)$  from his own smart card but he does not have any method to calculate the values of  $ID_i$  and  $y_i$ . Therefore, the proposed protocol is secure against malicious user attack.
6. **Replay attack:** Replaying a login request message  $(CID_i, M_i, T)$  of one session into another session is useless because the user  $U_i$ 's smart card uses current timestamp value  $T$  in each new session, which makes all the messages  $CID_i$  and  $M_i$  dynamic and valid for small interval of time. Old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.

7. **Leak of verifier attack:** In the proposed protocol, the service provider server S knows secret  $x$  and stores  $y_i \oplus x$  and  $ID_i \oplus H(x | y_i)$  corresponding to  $D_i$  in its database. The attacker does not have any technique to find out the value of  $x$  and hence can not calculate  $y_i$  from  $y_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x | y_i)$ . Therefore, the proposed protocol is secure against leak of verifier attack.
8. **Server spoofing attack:** In the proposed protocol, malicious server can not compute the session key  $S_k = H(H(x) | ID_i | T | y_i)$  because the malicious server does not know the value of  $H(x)$ ,  $ID_i$  and  $y_i$ . Moreover, the session key is different for the same user in different login sessions. Therefore, the proposed protocol is secure against server spoofing attack.
9. **Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card of the user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.
10. **Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(CID_i, M_i, T)$  with in the valid time frame window but can not compute the agreed session key  $S_k = H(H(x) | ID_i | T | y_i)$  between the user  $U_i$  and the server S because the attacker does not know the values of  $H(x)$ ,  $ID_i$  and  $y_i$ . Therefore, the proposed protocol is secure against parallel session attack.
11. **Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(CID_i, M_i, T)$  from the user  $U_i$  to the server S. Then he starts a new session with the server S by sending a login request by replaying the login request message  $(CID_i, M_i, T)$  with in the valid time frame window. The attacker can authenticate itself to the server S but can not compute the agreed session key  $S_k = H(H(x) | ID_i | T | y_i)$  between the user  $U_i$  and the server S because the attacker does not know the values of  $H(x)$ ,  $ID_i$  and  $y_i$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

#### 6.4.4 Cost and functionality analysis

The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.5. Assume that the identity  $ID_i$ , password  $P_i$ ,  $x$ ,  $y_i$  and timestamp values are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$ ,  $T_E$  and  $T_S$  denote the time complexity for hash function, exponential operation and symmetric key encryption respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_E \gg T_H$ . In the proposed protocol, the parameters stored in the smart card are  $N_i$ ,  $A_i$ ,  $B_i$  and the memory needed (E1) in the smart card is 384 ( $= 3*128$ ) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{CID_i, M_i, T\}$  is 384 ( $= 3*128$ ) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration is  $6T_H$ . The computation cost of the user and the service provider server is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user (E4) is  $6T_H$  and that of the service provider server (E5) is  $6T_H$ . The functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.6. The proposed protocol requires nearly the same computation as other related schemes [151][88][172][81][28] and requires very less computation as compared to Chien and Chen scheme [26] but it is highly secure as compared to the related schemes.

**Table 6.5**  
**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Wang et al. [151]</b>	<b>Liou et al. [88]</b>	<b>Yoon-Yoo [172]</b>	<b>Liao et al. [81]</b>	<b>Chien-Chen [26]</b>	<b>Das et al. [28]</b>
E1	384 bits	256 bits	384 bits	384 bits	256 bits	384 bits	256 bits
E2	3 * 128 bits	6 * 128 bits	5 * 128 bits	6 * 128 bits	6 * 128 bits	4 * 128 bits	4 * 128 bits
E3	$6 T_H$	$2 T_H$	$3 T_H$	$3 T_H$	$2 T_H$	$2 T_H$	$2 T_H$
E4	$6 T_H$	$3 T_H$	$4 T_H$	$6 T_H$	$5 T_H$	$2 T_E + 2 T_S$	$4 T_H$
E5	$6 T_H$	$3 T_H$	$5 T_H$	$4 T_H$	$4 T_H$	$2 T_H + 2 T_E + 2 T_S$	$3 T_H$

Table 6.6

Functionality comparison among related smart card based authentication schemes

	Proposed Protocol	Wang et al. [151]	Liou et al. [88]	Yoon-Yoo [172]	Liao et al. [81]	Chien-Chen [26]	Das et al. [28]
User's Anonymity	Yes	No	Yes	Yes	Yes	Yes	Yes
Session Key Agreement	Yes	No	No	No	No	Yes	No
Impersonation Attack	No	Yes	Yes	No	Yes	No	Yes
Malicious User Attack	No	Yes	Yes	No	Yes	No	Yes
Offline Dictionary Attack	No	Yes	Yes	Yes	Yes	No	Yes
Stolen Smart Card Attack	No	Yes	Yes	No	Yes	Yes	Yes
Denial of Service Attack	No	Yes	No	No	Yes	No	Yes

### 6.5 REVIEW OF LEE ET AL.'S SCHEME

In this section, we examine the smart card based remote user authentication scheme proposed by Lee et al. [77] in 2005. Lee et al.'s scheme consists of three phases viz. registration phase, login phase and verification phase as summarized in Figure 6.7.

#### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  for registration over a secure communication channel. The server  $S$  computes  $R_i = H(ID_i \oplus x) \oplus P_i$ , where  $x$  is the secret key of remote server  $S$ . Then the server  $S$  issues the smart card containing secret parameters  $(H(\cdot), R_i)$  to the user  $U_i$  through a secure communication channel. The server  $S$  also stores identity  $ID_i$  of user  $U_i$  in its database.

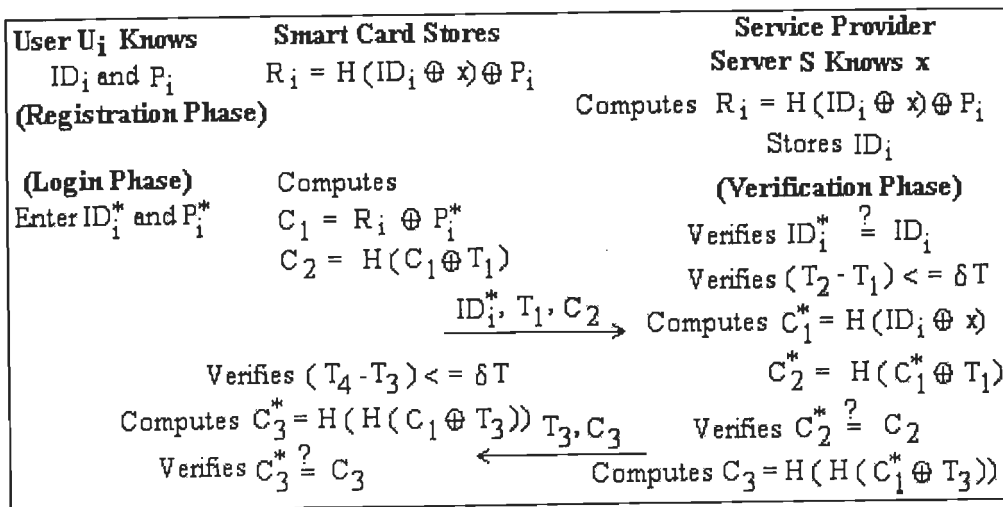


Figure 6.7: Lee et al.'s scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and then submit his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $C_1 = R_i \oplus P_i^*$  and  $C_2 = H(C_1 \oplus T_1)$ , where  $T_1$  is current date and time of the user's smart card and sends the login request message  $(ID_i^*, T_1, C_2)$  to the service provider server  $S$ .

## 3. Verification phase

The service provider server  $S$  verifies the received value of  $ID_i^*$  with the stored value of  $ID_i$  in its database. Then the server  $S$  verifies the validity of timestamp  $T_1$  by checking  $(T_2 - T_1) \leq \delta T$ , where  $T_2$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes  $C_1^* = H(ID_i \oplus x)$  and  $C_2^* = H(C_1^* \oplus T_1)$  and compares the computed value of  $C_2^*$  with the received value of  $C_2$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise the server  $S$  acquires the current timestamp  $T_3$  and computes  $C_3 = H(H(C_1^* \oplus T_3))$  and sends the message  $(T_3, C_3)$  back to the smart card of user  $U_i$ . On receiving the message  $(T_3, C_3)$ , smart card checks the validity of timestamp  $T_3$  by checking  $(T_4 - T_3) \leq \delta T$ , where  $T_4$  is current date and time of the user  $U_i$ 's smart card. Then the user  $U_i$ 's smart card computes  $C_3^* = H(H(C_1 \oplus T_3))$  and compares it with received value of  $C_3$ . This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted.

### 6.5.1 Cryptanalysis of Lee et al.'s scheme

Lee et al. [77] claimed that their protocol can resist various known attacks. However, we found that their protocol is flawed for impersonation attack, malicious user attack and reflection attack. Lee et al.'s scheme also fails to protect the user's anonymity in insecure communication channel.

#### 1. Impersonation attack

The attacker can intercept a valid login request message  $(ID_i^*, T_1, C_2)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_2 = H(C_1 \oplus T)$  to know the value of  $C_1$ , which is always same corresponding to user  $U_i$ . After guessing the correct value of  $C_1$ , the attacker can frame and send fabricated valid login request message  $(ID_i^*, T_u, C_2)$  to the server  $S$  without knowing the password  $P_i$  of the

user  $U_i$ , where  $T_u$  is a current timestamp and  $C_2 = H(C_1 \oplus T_u)$ . Therefore, the attacker can successfully make valid login request to impersonate as legitimate user  $U_i$  to the server  $S$ .

## 2. Malicious user attack

The malicious privileged user  $U_k$  having his own smart card can extract the stored value of  $R_k$  from its memory. Then malicious user  $U_k$  can launch offline dictionary attack on  $R_k = H(ID_k \oplus x) \oplus P_k$  to know the secret key  $x$  of the server  $S$  because the user  $U_k$  knows its identity  $ID_k$  and password  $P_k$ . Now this malicious user  $U_k$  can intercept the valid login request message  $(ID_i^*, T_1, C_2)$  of the user  $U_i$  from the public communication channel. The malicious user  $U_k$  can compute  $C_1 = H(ID_i^* \oplus x)$  and  $C_2 = H(C_1 \oplus T_u)$ , where  $T_u$  is current date and time. Then malicious user  $U_k$  can frame fabricated login request message  $(ID_i^*, T_u, C_2)$  corresponding to user  $U_i$  and sends it to the server  $S$ . The server  $S$  checks the validity of timestamp  $T_u$  by checking  $(T'' - T_u) \leq \delta T$ , where  $T''$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  computes the value of  $C_1^* = H(ID_i^* \oplus x)$ ,  $C_2^* = H(C_1^* \oplus T_u)$  and compares the computed value of  $C_2^*$  with the received value of  $C_2$ . This equivalency authenticates the legitimacy of the user  $U_i$  and login request is accepted by the server  $S$ .

## 3. Reflection attack

Reflection attack on Lee et al.'s scheme can be demonstrated as follows. The attacker reuses user  $U_i$ 's login request message as the response message of the fake server as follows. By observing the login request message  $(ID_i^*, T_1, C_2)$  and the response message  $(T_3, C_3)$ , it is clear that the  $C_2$  and  $C_3$  is symmetric to each other and difference between  $C_2$  and  $C_3$  is the timestamps and one additional hash operation, where  $C_2 = H(C_1 \oplus T_1)$  and  $C_3 = H(H(C_1 \oplus T_3))$ . An attacker can intercept a valid login request message  $(ID_i^*, T_1, C_2)$  of the user  $U_i$  from the public communication channel. Now the attacker can compute  $C_3 = H(C_2) = H(H(C_1 \oplus T_1))$  and impersonates as the server  $S$  and sends the response message  $(T_1, C_3)$  back to the user  $U_i$  immediately. The response message will pass the server's verification with high probability.

## 4. User's anonymity

The user's identity  $ID_i^*$  is transmitted clearly in login request message  $(ID_i^*, T_1, C_2)$  to the server  $S$  and hence the different login request messages belonging to the same user can be

traced out and can be interlinked to derive some information related to the user  $U_i$ . Therefore, Lee et al.'s scheme is not able to preserve the user's anonymity.

### 6.5.2 Proposed protocol

In this section, we describe a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Lee et al.'s [77] scheme. Figure 6.8 shows the entire protocol structure of new authentication scheme and consists of four phases viz. registration phase, login phase, authentication and session key agreement phase and password change phase.

#### 1. Registration phase

The user  $U_i$  selects a random number  $b$  to compute  $A_i = H(ID_i | b)$  and submits  $A_i$  to the server  $S$  for registration over a secure communication channel, where  $ID_i$  is the identity of the user  $U_i$ .

Step 1:  $U_i \rightarrow S: A_i$

The server  $S$  computes the security parameters  $F_i = A_i \oplus y_i$ ,  $B_i = A_i \oplus H(y_i) \oplus H(x)$  and  $C_i = H(A_i | H(x) | H(y_i))$ , where  $x$  is the secret key of remote server  $S$ . The server  $S$  chooses the value of  $y_i$  corresponding to each user in such a way so that the value of  $C_i$  must be unique for each user. The server  $S$  stores  $y_i \oplus x$  corresponding to  $C_i$  in its database. Then server  $S$  issues the smart card containing security parameters  $(F_i, B_i, H())$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i: \text{Smart card}$

Then user  $U_i$  computes security parameters  $D_i = b \oplus H(ID_i | P_i)$ ,  $E_i = H(ID_i | H(P_i)) \oplus P_i$  and enters the values of  $D_i$  and  $E_i$  in his smart card. Finally, the smart card contains security parameters as  $(D_i, E_i, F_i, B_i, H())$  stored in its memory.

#### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $E_i^* = H(ID_i^* | H(P_i^*)) \oplus P_i^*$  and compares it with stored value of  $E_i$  in its memory to verify the legitimacy of user  $U_i$ .

Step 1: Smart card checks  $E_i^* \stackrel{?}{=} E_i$

After verification, smart card computes  $b = D_i \oplus H(ID_i | P_i)$ ,  $A_i = H(ID_i | b)$ ,  $y_i = F_i \oplus A_i$ ,  $H(x) = B_i \oplus A_i \oplus H(y_i)$ ,  $C_i = H(A_i | H(x) | H(y_i))$ ,  $CID_i = H(H(x) | T) \oplus C_i$  and



$M_i = H(H(x) | H(y_i) | T)$ , where  $T$  is current date and time of the user's smart card. Then smart card sends the login request message  $(CID_i, M_i, T)$  to the server  $S$ .

Step 2: Smart card  $\rightarrow$   $S: CID_i, M_i, T$

### 3. Authentication and session key agreement phase

After receiving the login request from user  $U_i$ , the server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. The server  $S$  computes  $C_i^* = CID_i \oplus H(H(x) | T)$  and extracts  $y_i$  from  $y_i \oplus x$  corresponding to  $C_i^*$  from its database. If the value of  $C_i^*$  does not match with any value of  $C_i$  in the database of server  $S$ , the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  computes  $M_i^* = H(H(x) | H(y_i) | T)$  and compares  $M_i^*$  with the received value of  $M_i$  to check the authenticity of received message.

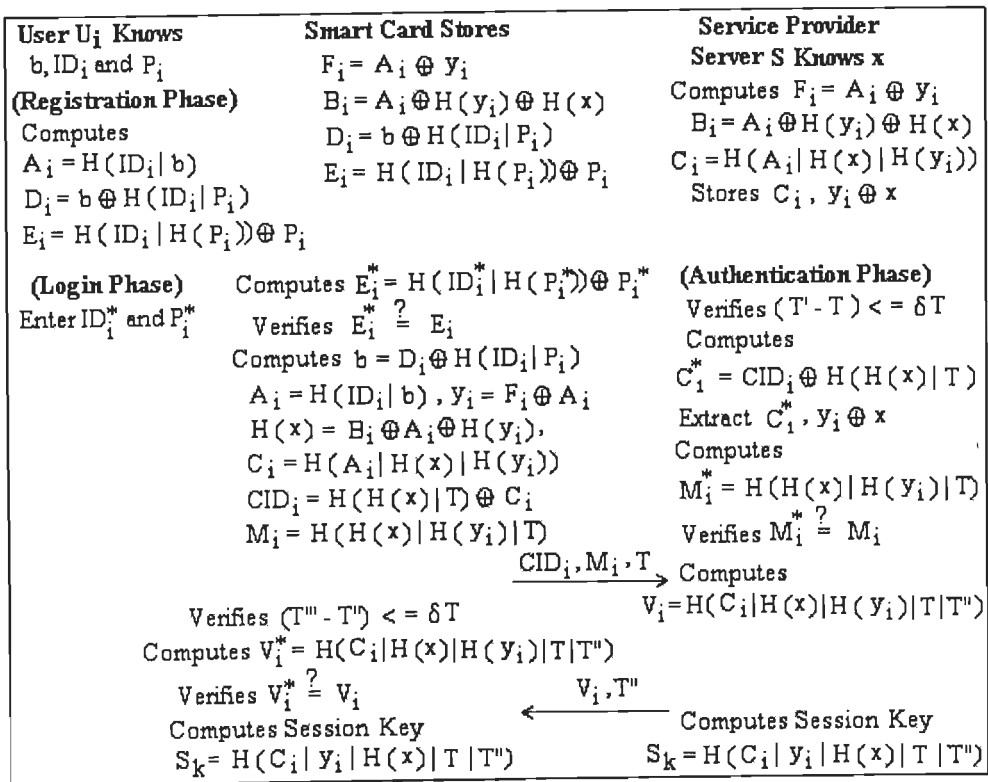


Figure 6.8: Proposed improvement in Lee et al.'s scheme

Step 1: Server  $S$  checks  $M_i^* \stackrel{?}{=} M_i$

If they are not equal, the server  $S$  rejects the login request and terminates this session.

Otherwise, the server  $S$  acquires the current timestamp  $T''$  and computes  $V_i =$

$H(C_i | H(x) | H(y_i) | T | T'')$  and sends the message  $(V_i, T'')$  back to the smart card of user  $U_i$ .

Step 2:  $S \rightarrow$  Smart card:  $V_i, T''$

On receiving the message  $(V_i, T'')$ , the user  $U_i$ 's smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  is current date and time of the user's smart card. Then smart card computes  $V_i^* = H(C_i | H(x) | H(y_i) | T | T'')$  and compares it with the received value of  $V_i$ .

Step 3: Smart card checks  $V_i^* \stackrel{?}{=} V_i$

This equivalency authenticates the legitimacy of the server  $S$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server  $S$  agree on the common session key as  $S_k = H(C_i | y_i | H(x) | T | T'')$ .

#### 4. Password change phase

The user  $U_i$  can change his password without the help of server  $S$ . The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$ . Smart card computes  $E_i^* = H(ID_i^* | H(P_i^*)) \oplus P_i^*$  and compares it with the stored value of  $E_i$  in its memory to verify the legitimacy of the  $U_i$ . Once the authenticity of card holder is verified then the  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then  $D_i = b \oplus H(ID_i | P_i)$  and  $E_i = H(ID_i | H(P_i)) \oplus P_i$  stored in the smart card can be replaced with  $D_i^{new} = D_i \oplus H(ID_i | P_i) \oplus H(ID_i | P_i^{new})$  and  $E_i^{new} = H(ID_i | H(P_i^{new})) \oplus P_i^{new}$  and the password gets changed.

#### 6.5.3 Security analysis

A good password authentication protocol should provide protection from different possible attacks relevant to that protocol.

1. **Impersonation attack:** The attacker can attempt to modify a login request message  $(CID_i, M_i, T)$  of the user  $U_i$  into  $(CID_i^*, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of authentication and session key agreement phase because the

attacker has no way of obtaining the values of  $H(x)$ ,  $C_i$  and  $H(y_i)$  to compute the valid parameter  $CID_i^*$  and  $M_i^*$  corresponding to the user  $U_i$ . Therefore, the proposed protocol is secure against impersonation attack.

2. **Malicious user attack:** A malicious privileged user  $U_k$  having his own smart card can gather information like  $D_k = b_k \oplus H(ID_k | P_k)$ ,  $E_k = H(ID_k | H(P_k)) \oplus P_k$ ,  $F_k = A_k \oplus y_k$  and  $B_k = A_k \oplus H(y_k) \oplus H(x)$  from the memory of smart card. This malicious user can not generate smart card specific values of  $CID_i = H(H(x) | T) \oplus C_i$  and  $M_i = H(H(x) | H(y_i) | T)$  to masquerades as other legitimate user  $U_i$  to the server  $S$  because the values of  $CID_i$  and  $M_i$  is smart card specific and depend upon the values of  $ID_i$ ,  $P_i$ ,  $b_i$ ,  $H(x)$  and  $H(y_i)$ . Although malicious user can extract  $H(x)$  from his own smart card but he does not have any method to calculate the values of  $ID_i$ ,  $P_i$ ,  $b_i$  and  $H(y_i)$ . Therefore, the proposed protocol is secure against malicious user attack.
3. **Reflection attack:** In this type of attack, the attacker reuses login request message ( $CID_i$ ,  $M_i$ ,  $T$ ) of the user  $U_i$  as the response message ( $V_i$ ,  $T''$ ) of a fake server. In the proposed protocol, there is no symmetry in the values of  $CID_i = H(H(x) | T) \oplus C_i$ ,  $M_i = H(H(x) | H(y_i) | T)$  and  $V_i = H(C_i | H(x) | H(y_i) | T | T'')$ . Therefore, the proposed protocol is secure against reflection attack.
4. **Identity protection:** The proposed approach provides identity protection in the sense that instead of sending the real identity  $ID_i$  of user  $U_i$  for authentication, the dynamic pseudo identification  $CID_i = H(H(x) | T) \oplus C_i$  is generated by the smart card corresponding to the legitimate user  $U_i$  for its authentication with the server  $S$ . The real identity information about the user  $U_i$  is not transmitted in the login request message. This approach provides the privacy and unlinkability among different login requests belonging to same user. The attacker can not link different sessions belonging to the same user.
5. **Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by an attacker, he can extract the information stored in its memory. An attacker can extract  $D_i = b \oplus H(ID_i | P_i)$ ,  $E_i = H(ID_i | H(P_i)) \oplus P_i$ ,  $F_i = A_i \oplus y_i$  and  $B_i = A_i \oplus H(y_i) \oplus H(x)$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess

out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.

6. **Offline dictionary attack:** An attacker first tries to obtain the user  $U_i$ 's or server verification information such as  $T$ ,  $T''$ ,  $CID_i = H(H(x) | T) \oplus C_i$ ,  $M_i = H(H(x) | H(y_i) | T)$ ,  $V_i = H(C_i | H(x) | H(y_i) | T | T'')$  and then try to guess the values of  $ID_i$ ,  $P_i$ ,  $b$ ,  $x$  and  $y_i$  by offline dictionary attack. Even after gathering this information, the attacker has to guess at least two parameters correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.
7. **Denial of service attack:** In the proposed protocol, smart card checks the validity of user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. Since the smart card computes  $E_i^* = H(ID_i^* | H(P_i^*)) \oplus P_i^*$  and compares it with the stored value of  $E_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
8. **Replay attack:** Replaying a message of one session into another session is useless because the user  $U_i$ 's smart card and the server  $S$  uses current timestamp values ( $T$  and  $T''$ ) in each new session, which make all the messages  $CID_i$ ,  $M_i$  and  $V_i$  dynamic and valid for small interval of time. Old messages can not be replayed successfully in any other session. Therefore, the proposed protocol is secure against message replay attack.
9. **Leak of verifier attack:** In the proposed protocol, the server  $S$  knows secret  $x$  and stores  $y_i \oplus x$  corresponding to user  $U_i$ 's  $C_i = H(A_i | H(x) | H(y_i))$  value in its database. The attacker does not have any technique to find out the value of  $x$  and hence can not calculate  $y_i$  from  $y_i \oplus x$ . Moreover, the attacker can not compute  $ID_i$ ,  $b$  or  $y_i$  from  $C_i$ . In case verifier is stolen by breaking into smart card memory, the attacker does not have sufficient information to calculate user  $U_i$ 's identity  $ID_i$  and password  $P_i$ . Therefore, the proposed protocol is secure against leak of verifier attack.
10. **Server spoofing attack:** The proposed protocol provides mutual authentication between user  $U_i$ 's smart card and server  $S$  to withstand the server spoofing attack. Malicious server can not generate the valid value of  $V_i = H(C_i | H(x) | H(y_i) | T | T'')$

meant for smart card of user  $U_i$  because malicious server has to know the values of  $ID_i$ ,  $b$ ,  $H(x)$  and  $H(y_i)$  to generate the valid value of  $V_i$  corresponding to user  $U_i$ 's smart card. Therefore, the proposed protocol is secure against server spoofing attack.

**11. Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, the attacker gets a very few chances (normally a maximum of 3) to guess the identity  $ID_i$  and password  $P_i$  because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

**12. Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(CID_i, M_i, T)$  with in the valid time frame window. The attacker can not compute the agreed session key  $S_k = H(C_i | y_i | H(x) | T | T')$  because the attacker does not know the values of  $ID_i$ ,  $b$ ,  $y_i$  and  $H(x)$ . Therefore, the proposed protocol is secure against parallel session attack.

**13. Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(CID_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request message  $(CID_i, M_i, T)$  with in the valid time frame window. The attacker can authenticate itself to server  $S$  as well as to legitimate user  $U_i$  but can not compute the session key  $S_k = H(C_i | y_i | H(x) | T | T')$  because the attacker does not know the values of  $ID_i$ ,  $b$ ,  $y_i$  and  $H(x)$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

#### 6.5.4 Cost and functionality analysis

The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.7. Assume that the identity  $ID_i$ , password  $P_i$ ,  $x$ ,  $y_i$ , timestamp values and output of secure one-way hash function are all 128-bit long. Let  $T_H$ ,  $T_E$  and  $T_X$  denote the time complexity for hash function, exponential operation and XOR operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_E \gg T_H \gg T_X$ .

**Table 6.7**

**Cost comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Xu et al. [157]</b>	<b>Kim-Chung [65]</b>	<b>Yoon-Yoo [171]</b>	<b>Lee et al. [77]</b>	<b>Chein et al. [25]</b>
E1	512 bits	512 bits	384 bits	256 bits	128 bits	128 bits
E2	5 * 128 bits	8 * 128 bits	6 * 128 bits	5 * 128 bits	5 * 128 bits	5 * 128 bits
E3	$7T_H + 6T_X$	$1T_E + 2T_H$	$4T_H + 6T_X$	$1T_H + 4T_X$	$1T_H + 2T_X$	$1T_H + 2T_X$
E4	$10T_H + 6T_X$	$3T_E + 5T_H$	$4T_H + 6T_X$	$2T_H + 4T_X$	$3T_H + 3T_X$	$2T_H + 3T_X$
E5	$6T_H + 2T_X$	$3T_E + 4T_H$	$4T_H + 6T_X$	$2T_H + 5T_X$	$4T_H + 3T_X$	$3T_H + 3T_X$

In the proposed protocol, the parameters stored in the smart card are  $D_i, E_i, F_i, B_i$  and the memory needed (E1) in the smart card is 512 (= 4\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{CID_i, M_i, T\}$  and  $\{V_i, T''\}$  is 640 (= 5\*128) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration (E3) is  $7T_H + 6T_X$ . The computation cost of the user (E4) and the service provider server (E5) is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user (E4) is  $10T_H + 6T_X$  and that of the service provider server (E5) is  $6T_H + 2T_X$ .

**Table 6.8**

**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Xu et al. [157]</b>	<b>Kim-Chung [65]</b>	<b>Yoon-Yoo [171]</b>	<b>Lee et al. [77]</b>	<b>Chein et al. [25]</b>
Identity Protection	Yes	No	No	No	No	No
Impersonation Attack	No	Yes	Yes	Yes	Yes	Yes
Malicious User Attack	No	Yes	Yes	Yes	Yes	Yes
Reflection Attack	No	No	No	No	Yes	Yes
Offline Dictionary Attack	No	No	Yes	Yes	Yes	Yes
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	Yes
Session Key Agreement	Yes	Yes	No	No	No	No

The proposed protocol requires less computation than that of Xu et al.'s [157] scheme and requires more computation than that of Kim and Chung's [65] scheme but it is highly secure as compared to the related schemes. The proposed protocol provides identity protection and free from impersonation attack, malicious user attack and offline dictionary attack, while the latest schemes [65][157] proposed in 2009 suffers from these attacks. The functionality comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.8.

## 6.6 REVIEW OF HSIANG AND SHIH'S SCHEME

In this section, we examine a smart card based remote user authentication scheme proposed by Hsiang and Shih [48] in 2009. Hsiang and Shih's scheme consists of four phases viz. registration phase, login phase, authentication phase and password change phase as summarized in Figure 6.9.

### 1. Registration phase

The user  $U_i$  selects a random number  $b$  and computes  $H(b \oplus P_i)$  and submit his identity  $ID_i$ ,  $H(P_i)$  and  $H(b \oplus P_i)$  to the server  $S$  for registration over a secure communication channel. If it is user  $U_i$ 's initial registration, the server  $S$  creates an entry for user  $U_i$  in its database and stores  $n = 0$  for this user. Otherwise, the server  $S$  sets  $n = n + 1$  in the existing entry for the user  $U_i$  (Re-registration for lost or stolen smart card). The server  $S$  computes  $EID_i = (ID_i | n)$ ,  $B_i = H(EID_i \oplus x)$ ,  $R_i = B_i \oplus H(b \oplus P_i)$  and  $V_i = H(B_i \oplus H(P_i))$ , where  $x$  is the master secret of the server  $S$ . The server  $S$  stores  $n$  corresponding to  $ID_i$  in its database. Then, the server  $S$  issues the smart card containing secret parameters  $(R_i, V_i, H(\ ))$  to the user  $U_i$  through a secure communication channel. Afterwards, the user  $U_i$  enters the value of  $b$  in his smart card. Finally, the smart card contains security parameters as  $(R_i, V_i, b, H(\ ))$  stored in its memory.

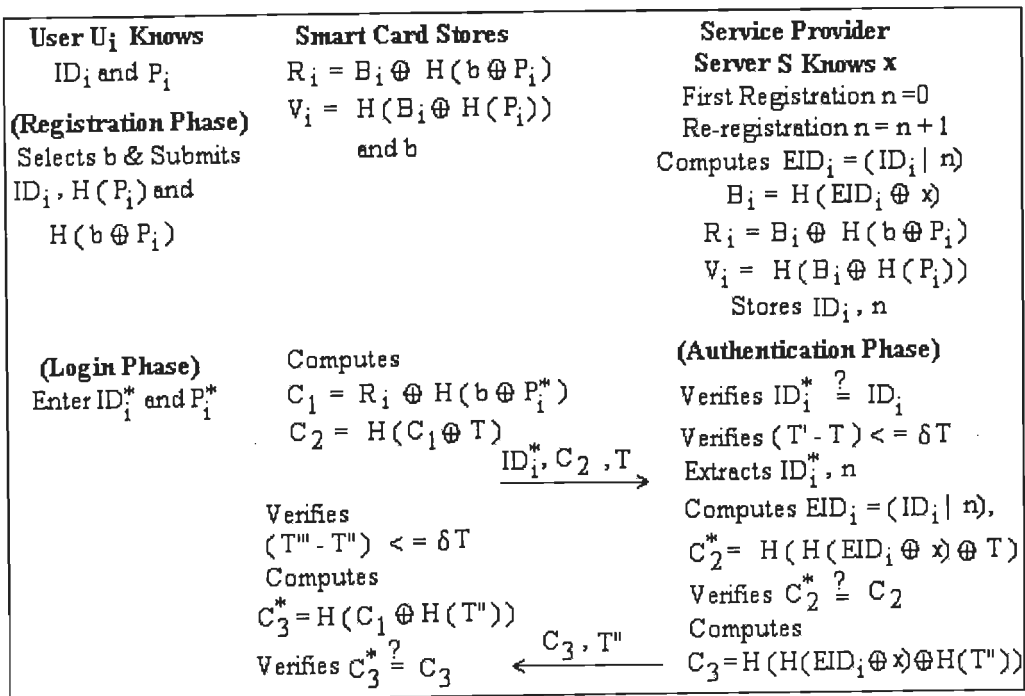


Figure 6.9: Hsiang and Shih's scheme

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . Then, the smart card computes  $C_1 = R_i \oplus H(b \oplus P_i^*)$  and  $C_2 = H(C_1 \oplus T)$ , where  $T$  is current date and time of the user's smart card and sends the login request message  $(ID_i^*, C_2, T)$  to the service provider server  $S$ .

## 3. Authentication phase

The service provider server  $S$  verifies the received value of  $ID_i^*$  with the stored value of  $ID_i$  in its database. Then the server  $S$  verifies the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  denotes the server's current timestamp and  $\delta T$  is permissible time interval for a transmission delay. Afterwards, the server  $S$  extracts the value of  $n$  corresponding to  $ID_i^*$  from its database to compute  $EID_i = (ID_i | n)$ ,  $C_2^* = H(H(EID_i \oplus x) \oplus T)$  and compares the computed value of  $C_2^*$  with the received value of  $C_2$ . If they are not equal, the server  $S$  rejects the login request and terminates this session. Otherwise, the server  $S$  acquires the current timestamp  $T''$  to compute  $C_3 = H(H(EID_i \oplus x) \oplus H(T''))$  and sends the message  $(C_3, T'')$  back to the smart card of the user  $U_i$ . On receiving the message  $(C_3, T'')$ , smart card checks the validity of timestamp  $T''$  by checking  $(T''' - T'') \leq \delta T$ , where  $T'''$  denotes the user  $U_i$ 's smart card current timestamp. Then, the user  $U_i$ 's smart card computes  $C_3^* = H(C_1 \oplus H(T''))$  and compares it with received value of  $C_3$ . This equivalency authenticates the legitimacy of the service provider server  $S$  and the login request is accepted else the connection is interrupted.

## 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $C_1 = R_i \oplus H(b \oplus P_i^*)$ ,  $V_i^* = H(C_1 \oplus H(P_i^*))$  and compares the computed value of  $V_i^*$  with stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $R_i^{new} = C_1 \oplus H(b \oplus P_i^{new})$  and  $V_i^{new} = H(C_1 \oplus H(P_i^{new}))$ . Thereafter, smart card replaces the values of  $R_i$  and  $V_i$  stored in its memory with  $R_i^{new}$  and  $V_i^{new}$  and password gets changed.



### 6.6.1 Cryptanalysis of Hsiang and Shih's scheme

Hsiang and Shih [48] claimed that their scheme can resist various known attacks. However, we found that their scheme is flawed for impersonation attack and offline guessing attack. This scheme also delays the checking of legitimacy of the user to authentication phase and also fails to preserve the user's anonymity.

#### 1. Impersonation attack

The attacker can intercept a valid login request message  $(ID_i^*, C_2, T)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_2 = H(C_1 \oplus T)$  to know the value of  $C_1$ , which is always equal to  $B_i$ . After guessing the value of  $C_1$ , the attacker can frame and send fabricated valid login request message  $(ID_i^*, C_2, T_u)$  to the server  $S$  without knowing the password  $P_i$  of the user  $U_i$ , where  $T_u$  is a current timestamp and  $C_2 = H(C_1 \oplus T_u) = H(B_i \oplus T_u)$ . Hence, the attacker can successfully make a valid login request to impersonate as a legitimate user  $U_i$  to the service provider server  $S$ .

#### 2. Offline guessing attack

A malicious privileged user  $U_k$  having his own smart card can gather information  $R_k = B_k \oplus H(b \oplus P_k)$  and  $b$  from his own smart card. He can find out the value of  $B_k$  as  $B_k = R_k \oplus H(b \oplus P_k)$  because the malicious user  $U_k$  knows the values of  $b$  and his own password  $P_k$  corresponding to his smart card. Then, this malicious user launches offline dictionary attack on  $B_k = H(EID_k \oplus x) = H((ID_k | n) \oplus x)$  to find the value of  $x$  because malicious user  $U_k$  knows his identity  $ID_k$  and value of  $n$ . Now this malicious user  $U_k$  has intercepted a valid login request message  $(ID_i, C_2, T)$  of the user  $U_i$  from the public communication channel. Now he can launch offline dictionary attack on  $C_2 = H(C_1 \oplus T) = H(((ID_i | n) \oplus x) \oplus T)$  by guessing the value of  $n$  (value of  $n$  is predictable starting from 0 to some positive integer like 1, 2 or 3) to know the value of  $C_1$ , which is always equal to  $B_i$ . After guessing the correct value of  $C_1$ , an attacker can frame and send fabricated valid login request message  $(ID_i^*, C_2, T_u)$  to the service provider server  $S$ , where  $T_u$  is a current timestamp and  $C_2 = H(B_i \oplus T_u)$ . Hence, the malicious user  $U_k$  can successfully make a valid login request to masquerades as a legitimate user  $U_i$ .

#### 3. Delayed user verification

The legitimacy of the user  $U_i$  has not checked in the login phase of Hsiang and Shih's scheme. The authenticity of the user  $U_i$  being verified by checking the received value of

$C_2$  in the authentication phase. Suppose the attacker gets the smart card of any user, he can flood different login request message to the server  $S$  from different card reader machines by submitting any invalid identity and random password. That causes denial of service by the server to other legitimate users. Instead, the legitimacy of the user  $U_i$  should be checked by the smart card in login phase by computing  $C_1 = R_i \oplus H(b \oplus P_i^*)$ ,  $V_i^* = H(C_1 \oplus H(P_i^*))$  and verifying the computed value of  $V_i^*$  with stored value of  $V_i$  in its memory. If both values match, the legitimacy of the user  $U_i$  is assured and smart card proceeds to the next step. Otherwise the login request from the user  $U_i$  is rejected.

#### 4. User's anonymity

The user  $U_i$ 's identity  $ID_i^*$  is transferred in clear text during login request message  $(ID_i^*, C_2, T)$  and hence the different login request messages belonging to the same user can be traced out and can be interlinked to derive some information related to the user  $U_i$ . Therefore, Hsiang and Shih's scheme is not able to preserve the user's anonymity.

#### 6.6.2 Proposed protocol

In this section, we describe a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Hsiang and Shih's [48] scheme. Figure 6.10 shows the entire protocol structure of the new authentication protocol.

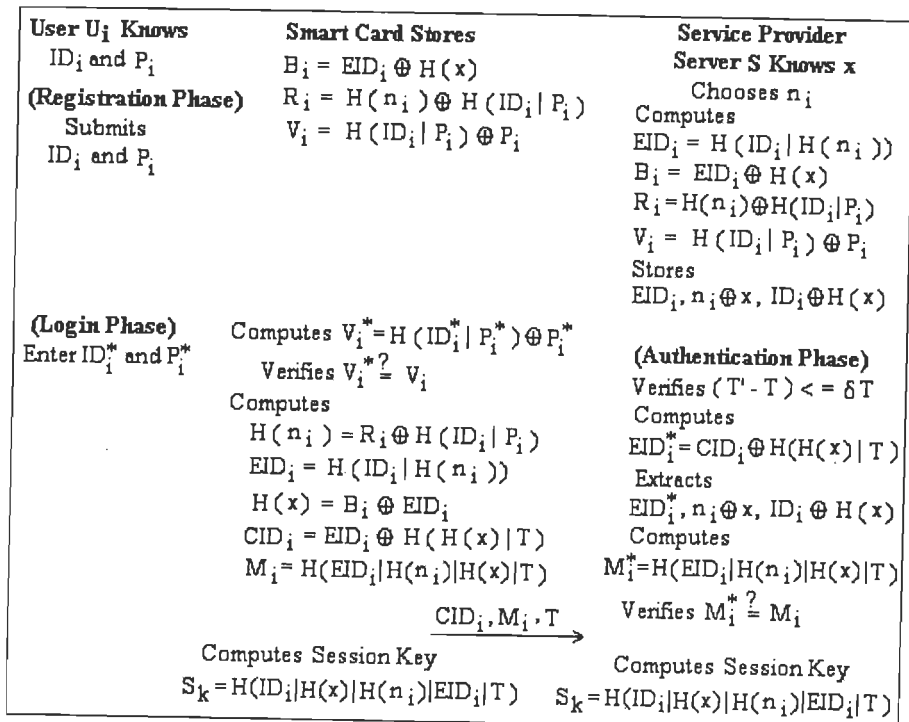


Figure 6.10: Proposed improvement in Hsiang and Shih's scheme

### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the server  $S$  via a secure communication channel to register itself to the server  $S$ .

Step 1:  $U_i \rightarrow S: ID_i, P_i$

The server  $S$  chooses random value  $n_i$  and computes the security parameters  $EID_i = H(ID_i | H(n_i))$ ,  $B_i = EID_i \oplus H(x)$ ,  $R_i = H(n_i) \oplus H(ID_i | P_i)$  and  $V_i = H(ID_i | P_i) \oplus P_i$ . The server  $S$  chooses the value of  $n_i$  corresponding to each user in such a way that the value of  $EID_i$  must be unique for each user. The server  $S$  stores  $n_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $EID_i$  in its database. Then the server  $S$  issues the smart card containing security parameters  $(B_i, R_i, V_i, H(\ ))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $S \rightarrow U_i: \text{Smart card}$

### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S$  and submits his identity  $ID_i^*$  and password  $P_i^*$ . The smart card computes  $V_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verify legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $V_i^* \stackrel{?}{=} V_i$

After verification, the smart card computes  $H(n_i) = R_i \oplus H(ID_i | P_i)$ ,  $EID_i = H(ID_i | H(n_i))$ ,  $H(x) = B_i \oplus EID_i$ ,  $CID_i = EID_i \oplus H(H(x) | T)$  and  $M_i = H(EID_i | H(n_i) | H(x) | T)$ , where  $T$  is current date and time of the user's smart card. Then the smart card sends login request message  $(CID_i, M_i, T)$  to the service provider server  $S$ .

Step 2: Smart card  $\rightarrow S: CID_i, M_i, T$

### 3. Authentication phase

After receiving the login request message from the user  $U_i$ , the server  $S$  checks the validity of timestamp  $T$  by checking  $(T' - T) \leq \delta T$ , where  $T'$  is current date and time of the server  $S$  and  $\delta T$  is permissible time interval for a transmission delay. The server  $S$  computes  $EID_i^* = CID_i \oplus H(H(x) | T)$  and finds  $EID_i$  corresponding to  $EID_i^*$  in its database and then extracts  $n_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $EID_i^*$  from its database. Now the server  $S$  computes  $n_i$  from  $n_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$  because the server  $S$  knows the value of  $x$ . Then, the server  $S$  computes  $M_i^* = H(EID_i | H(n_i) | H(x) | T)$  and compares the computed value of  $M_i^*$  with the received value of  $M_i$ .

Step 1: Server S checks  $M_i^* \stackrel{?}{=} M_i$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$  and the server S agree on the common session key as  $S_k = H (ID_i | H (x) | H (n_i) | EID_i | T)$ . Afterwards, all the subsequent messages between the user  $U_i$  and the server S are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server S can retrieve the original message because both of them know the common session key.

#### 4. Password change phase

The user  $U_i$  can change his password without the help of the server S. The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. The smart card computes  $V_i^* = H (ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct the smart card to change his password. Afterwards, the smart card asks the card holder to resubmit a new password  $P_i^{new}$  and then smart card computes  $R_i^{new} = R_i \oplus H (ID_i | P_i) \oplus H (ID_i | P_i^{new})$  and  $V_i^{new} = H (ID_i | P_i^{new}) \oplus P_i^{new}$ . Thereafter, smart card updates the values of  $R_i$  and  $V_i$  stored in its memory with  $R_i^{new}$  and  $V_i^{new}$ . The identity and the password of the user is verified before the password update procedure, while Hsiang and Shih's [48] scheme only verifies the password of the user before the password update procedure.

##### 6.6.3 Security analysis

A good password authentication protocol should provide protection from different possible attacks relevant to that protocol.

- 1. Impersonation attack:** The attacker can attempt to modify a login request message  $(CID_i, M_i, T)$  of user  $U_i$  into  $(CID_i^*, M_i^*, T^*)$ , where  $T^*$  is the attacker's current date and time, so as to succeed in the authentication phase. However, such a modification will fail in Step 1 of the authentication and session key agreement phase because the attacker has no way of obtaining the values of  $ID_i$ ,  $H (n_i)$  and  $H (x)$  to compute the valid parameters  $CID_i^*$  and  $M_i^*$  corresponding to the user  $U_i$ . Therefore, the proposed protocol is secure against impersonation attack.

2. **Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by the attacker, he can extract the information stored in its memory. The attacker can extract  $B_i = EID_i \oplus H(x)$ ,  $R_i = H(n_i) \oplus H(ID_i | P_i)$  and  $V_i = H(ID_i | P_i) \oplus P_i$  from the memory of user  $U_i$ 's smart card. Even after gathering this information, the attacker has to guess  $ID_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
3. **Offline dictionary attack:** The attacker first tries to obtain the user  $U_i$ 's verification information  $CID_i = EID_i \oplus H(H(x) | T)$ ,  $M_i = H(EID_i | H(n_i) | H(x) | T)$ ,  $T$  and then try to guess the values of  $ID_i$ ,  $H(n_i)$  and  $H(x)$  by offline guessing. Even after gathering this information, the attacker has to guess all three parameters  $ID_i$ ,  $H(n_i)$  and  $H(x)$  correctly at the same time. It is not possible to guess all three parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.
4. **Denial of service attack:** In the proposed protocol, smart card checks the validity of user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. An attacker can insert the smart card into a smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $V_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $V_i$  in its memory to verify the legitimacy of the user  $U_i$  before the smart card accepts the password update request. It is not possible to guess out identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.
5. **Malicious user attack:** A malicious privileged user  $U_k$  having his own smart card can gather information like  $B_k = EID_k \oplus H(x)$ ,  $R_k = H(n_k) \oplus H(ID_k | P_k)$  and  $V_k = H(ID_k | P_k) \oplus P_k$  from the memory of smart card. This malicious user  $U_k$  can not generate smart card specific values of  $CID_i = EID_i \oplus H(H(x) | T)$  and  $M_i = H(EID_i | H(n_i) | H(x) | T)$  to masquerades as other legitimate user  $U_i$  to the service provider server  $S$  because the values of  $CID_i$  and  $M_i$  is smart card specific and depend upon the values of  $ID_i$ ,  $H(n_i)$  and  $H(x)$ . Although malicious user can extract  $H(x)$  from his own smart card but he does not have any method to calculate the values

of  $ID_i$  and  $H(n_i)$  corresponding to user  $U_i$ . Therefore, the proposed protocol is secure against malicious user attack.

6. **Replay attack:** Replaying a login request message  $(CID_i, M_i, T)$  of one session into another session is useless because the user  $U_i$ 's smart card uses current timestamp value  $T$  in each new session, which makes all the messages  $CID_i$  and  $M_i$  dynamic and valid for small interval of time. Old messages can not be replayed successfully in any other session and hence the proposed protocol is secure against message replay attack.
7. **Leak of verifier attack:** In the proposed protocol, the service provider server  $S$  knows secret  $x$  and stores  $n_i \oplus x$  and  $ID_i \oplus H(x)$  corresponding to  $EID_i$  in its database. The attacker does not have any technique to find out the value of  $x$  and hence can not calculate  $n_i$  from  $n_i \oplus x$  and  $ID_i$  from  $ID_i \oplus H(x)$ . Therefore, the proposed protocol is secure against leak of verifier attack.
8. **Server spoofing attack:** In the proposed protocol, malicious server can not compute the session key  $S_k = H(ID_i | H(x) | H(n_i) | EID_i | T)$  because the malicious server does not know the value of  $ID_i$ ,  $H(x)$  and  $H(n_i)$ . Moreover, the session key is different for the same user in different login sessions. Therefore, the proposed protocol is secure against server spoofing attack.
9. **Online dictionary attack:** In the proposed protocol, the attacker has to get the valid smart card of user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.
10. **Parallel session attack:** The attacker can masquerade as a legitimate user  $U_i$  by replaying a login request message  $(CID_i, M_i, T)$  within the valid time frame window but can not compute the agreed session key  $S_k = H(ID_i | H(x) | H(n_i) | EID_i | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $ID_i$ ,  $H(x)$  and  $H(n_i)$ . Therefore, the proposed protocol is secure against parallel session attack.
11. **Man-in-the-middle attack:** In the proposed protocol, the attacker can intercept the login request message  $(CID_i, M_i, T)$  from the user  $U_i$  to the server  $S$ . Then he starts a new session with the server  $S$  by sending a login request by replaying the login request

message  $(CID_i, M_i, T)$  with in the valid time frame window. The attacker can authenticate itself to the server  $S$  but can not compute the agreed session key  $S_k = H (ID_i | H (x) | H (n_i) | EID_i | T)$  between the user  $U_i$  and the server  $S$  because the attacker does not know the values of  $ID_i, H (x)$  and  $H (n_i)$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

#### 6.6.4 Cost and functionality analysis

The cost comparison of the proposed protocol with the related smart card based authentication schemes is summarized in Table 6.9. Assume that the identity  $ID_i$ , password  $P_i, b, x, n_i$  and timestamp values are all 128-bit long. Moreover, we assume that the output of secure one-way hash function is 128-bit. Let  $T_H$  and  $T_X$  denote the time complexity for hash function and XOR operation respectively. Typically, time complexity

Table 6.9

Cost comparison among related smart card based authentication schemes

	Proposed Protocol	Hsiang-Shih [48]	Yoon et al. [168]	Ku-Chen [69]	Chein et al. [25]
E1	384 bits	384 bits	384 bits	256 bits	128 bits
E2	3 * 128 bits	5 * 128 bits	5 * 128 bits	5 * 128 bits	5 * 128 bits
E3	4 $T_H$ + 5 $T_X$	4 $T_H$ + 4 $T_X$	2 $T_H$ + 3 $T_X$	2 $T_H$ + 3 $T_X$	1 $T_H$ + 2 $T_X$
E4	5 $T_H$ + 4 $T_X$	4 $T_H$ + 4 $T_X$	3 $T_H$ + 4 $T_X$	3 $T_H$ + 4 $T_X$	2 $T_H$ + 3 $T_X$
E5	5 $T_H$ + 3 $T_X$	4 $T_H$ + 3 $T_X$	3 $T_H$ + 3 $T_X$	3 $T_H$ + 3 $T_X$	3 $T_H$ + 3 $T_X$

associated with these operations can be roughly expressed as  $T_H \gg T_X$ . In the proposed protocol, the parameters stored in the smart card are  $B_i, R_i, V_i$  and the memory needed (E1) in the smart card is 384 (= 3\*128) bits. The communication cost of authentication (E2) includes the capacity of transmitting message involved in the authentication scheme. The capacity of transmitting message  $\{CID_i, M_i, T\}$  is 384 (= 3\*128) bits. The computation cost of registration (E3) is the total time of all operations executed in the registration phase. The computation cost of registration is  $4T_H + 5T_X$ . The computation cost of the user and the service provider server is the time spent by the user and the service provider server during the process of authentication. Therefore, the computation cost of the user (E4) is  $5T_H + 4T_X$  and that of the service provider server (E5) is  $5T_H + 3T_X$ . The functionality comparison of the proposed protocol with the relevant smart card based

authentication schemes is summarized in Table 6.10. The proposed protocol requires nearly the same computation as required by Hsiang and Shih scheme but it is highly secure as compared to the related schemes.

**Table 6.10**  
**Functionality comparison among related smart card based authentication schemes**

	<b>Proposed Protocol</b>	<b>Hsiang-Shih [48]</b>	<b>Yoon et al. [168]</b>	<b>Ku-Chen [69]</b>	<b>Chein et al. [25]</b>
Identity Protection	Yes	No	No	No	No
Session Key Agreement	Yes	No	No	No	No
Impersonation Attack	No	Yes	Yes	Yes	Yes
Offline Dictionary Attack	No	Yes	Yes	Yes	Yes
Delayed User Verification	No	Yes	Yes	Yes	Yes
User's Anonymity	Yes	No	No	No	No

## 6.7 CONCLUSION

In this chapter, we presented cryptanalysis of Liao et al.'s scheme, Liou et al.'s scheme, Wang et al.'s scheme, Lee et al.'s scheme and Hsiang & Shih's scheme by showing that their schemes are vulnerable to different attacks. The improvements to these schemes are proposed. Security analysis proved that the proposed protocols are more secure and practical.



## DYNAMIC IDENTITY BASED AUTHENTICATION PROTOCOLS FOR MULTI-SERVER ARCHITECTURE

---

### 7.1 INTRODUCTION

Most of the existing password authentication protocols are based on single-server model in which the server stores the user's password verifier information in its database. Password verifier information stored on the single server is mainly susceptible to stolen verifier attack. The concept of multi-server model removes this common point of susceptibility. In this chapter, two protocols are proposed for multi-server model consisting of two servers (service provider server and control server), these servers work together to authenticate the users. Yang et al. [162] also suggested similar kind of two-server model for the user's authentication. In the proposed protocols, different levels of trust are assigned to the servers and the service provider server is more exposed to the clients than the control server. The back-end control server is not directly accessible to the clients and thus it is less likely to be attacked. The two-server model provides the flexibility to distribute user passwords and the authentication functionality into two servers to eliminate the main point of vulnerability of the single-server model. Therefore, two-server model appears to be a reasonable choice for practical applications.

Password is the most commonly used authentication technique in authentication protocols. A number of static identity based remote user authentication protocols have been proposed in the literature to improve security, efficiency and cost. The user may change his password but can not change his identity in password authentication protocols. During communication, the static identity leaks out partial information about the user's authentication messages to the attacker. Most of the password authentication protocols for multi-server environment are based on static identity and the attacker can use this information to trace and identify different requests belonging to the same user. On the other hand, the dynamic identity based authentication protocols provide two-factor authentication based on the identity and password and hence more suitable to e-commerce applications. The aim of this chapter is to provide dynamic identity based secure and

computation efficient authentication protocols with user's anonymity for multi-server environment using smart cards. They protect the user's identity in insecure communication channel and hence can be applied directly to e-economic applications.

In this chapter, a brief review of two dynamic identity based authentication protocols (Liao & Wang's protocol and Hsiang & Shih's protocol) using smart cards for multi-server architecture is given. It describes the cryptanalysis of these protocols for different types of attacks and improved protocols are proposed. The security analysis of the proposed improved protocols is presented. The cost and functionality comparison of the proposed protocols with the other related protocols is also presented.

## 7.2 REVIEW OF LIAO AND WANG PROTOCOL

In this section, we describe the dynamic identity based remote user authentication protocol using smart cards for multi-server environment proposed by Liao and Wang [83] in 2009. Their protocol includes three phases viz. registration phase, login phase and mutual verification and session key agreement phase. The notations used in this section are listed in Table 7.1 and the protocol is shown in Figure 7.1.

**Table 7.1**  
**Notations**

$U_i$	User $U_i$
$S_J$	$J^{\text{th}}$ server
RC	Registration center
$ID_i$	Unique identification of user $U_i$
$P_i$	Password of user $U_i$
$SID_J$	Unique identification of server $S_J$
$CID_i$	Dynamic identity of user $U_i$
$H()$	One-way hash function
$x$	Master secret of registration center
$y$	Shared secret key of registration center & all servers
$\oplus$	XOR operation
	Concatenation

### 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the registration center RC so that he can access the resources of the service provider server  $S_J$ . The RC computes

$T_i = H(ID_i | x)$ ,  $V_i = T_i \oplus H(ID_i | P_i)$ ,  $B_i = H(P_i) \oplus H(x)$  and  $D_i = H(T_i)$ . Then the RC issues the smart card containing secret parameters ( $V_i$ ,  $B_i$ ,  $D_i$ ,  $H(\cdot)$ ,  $y$ ) to the user  $U_i$  through a secure communication channel.

## 2. Login phase

The user  $U_i$  submits his identity  $ID_i^*$ , password  $P_i^*$  and the server identity  $SID_J$  to smart card in order to login on to the service provider server  $S_J$ . The smart card computes  $T_i^* = V_i \oplus H(ID_i^* | P_i^*)$ ,  $D_i^* = H(T_i^*)$  and then verifies the equality of computed value of  $D_i^*$  with the stored value of  $D_i$  in its memory. If both values of  $D_i$  match, the legitimacy of the user is assured and smart card proceeds to the next step. Otherwise the login request from the user  $U_i$  is rejected. Then smart card generates a nonce value  $N_i$  and computes  $CID_i = H(P_i) \oplus H(T_i | y | N_i)$ ,  $P_{ij} = T_i \oplus H(y | N_i | SID_J)$  and  $Q_i = H(B_i | y | N_i)$ . Afterwards, smart card sends the login request message ( $CID_i$ ,  $P_{ij}$ ,  $Q_i$ ,  $N_i$ ) to the service provider server  $S_J$ .

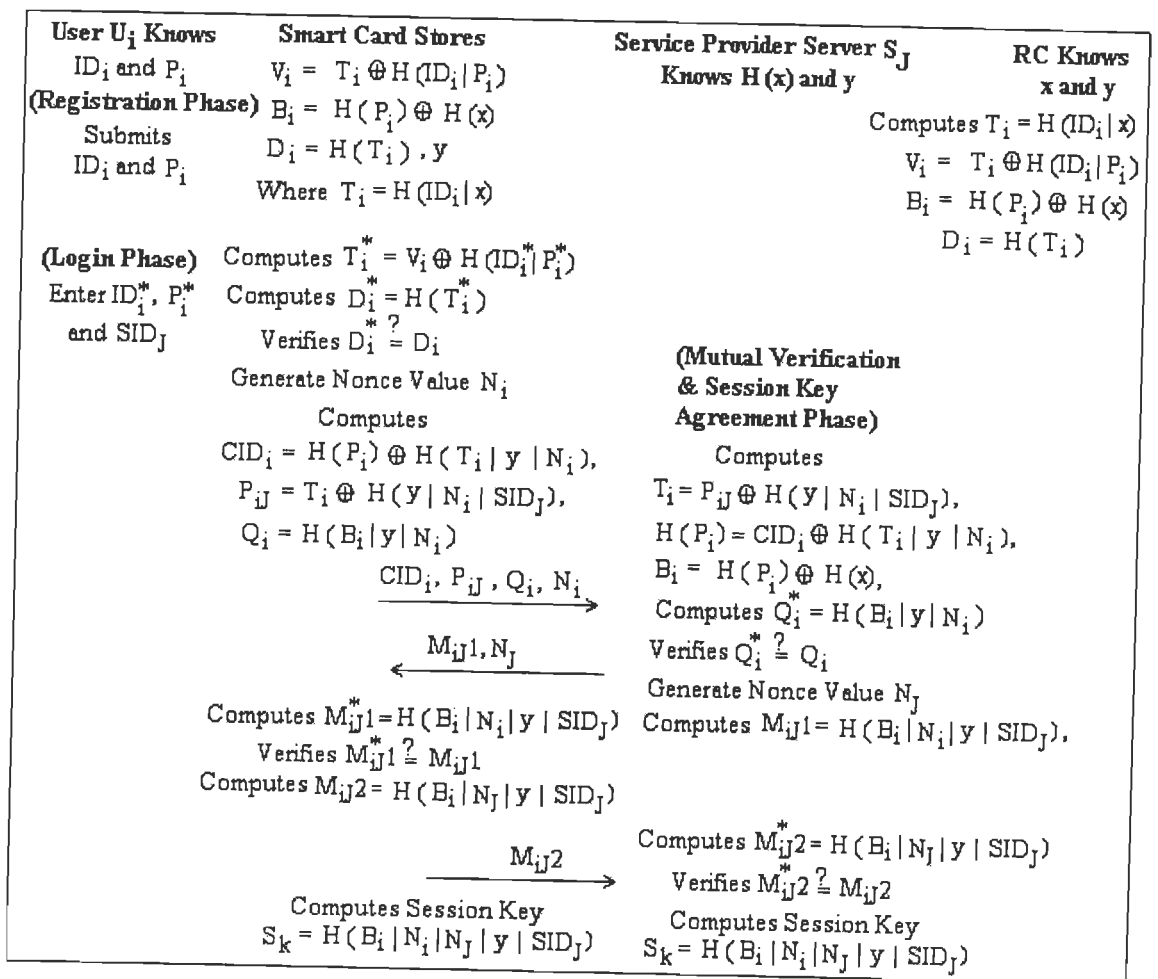


Figure 7.1: Liao and Wang's dynamic identity based multi-server authentication protocol

### 3. Mutual verification and session key agreement phase

The service provider server  $S_J$  computes  $T_i = P_{ij} \oplus H(y | N_i | SID_J)$ ,  $H(P_i) = CID_i \oplus H(T_i | y | N_i)$ ,  $B_i = H(P_i) \oplus H(x)$  and  $Q_i^* = H(B_i | y | N_i)$  and then compares the computed value of  $Q_i^*$  with the received value of  $Q_i$ . If they are not equal, the server  $S_J$  rejects the login request and terminates this session. Otherwise, the server  $S_J$  generates nonce value  $N_J$  and computes  $M_{ij1} = H(B_i | N_i | y | SID_J)$  and sends the message  $(M_{ij1}, N_J)$  back to smart card of the user  $U_i$ . On receiving the message  $(M_{ij1}, N_J)$ , the user  $U_i$ 's smart card computes  $M_{ij1}^* = H(B_i | N_i | y | SID_J)$  and compares the computed value of  $M_{ij1}^*$  with the received value of  $M_{ij1}$ . This equivalency authenticates the legitimacy of the service provider server  $S_J$  else the connection is interrupted. Then the user  $U_i$ 's smart card computes  $M_{ij2} = H(B_i | N_J | y | SID_J)$  and sends  $M_{ij2}$  back to the service provider server  $S_J$ . On receiving the message  $M_{ij2}$ , the service provider server  $S_J$  computes  $M_{ij2}^* = H(B_i | N_J | y | SID_J)$  and compares the computed value of  $M_{ij2}^*$  with the received value of  $M_{ij2}$ . This equivalency assures the legitimacy of the user  $U_i$ . After finishing mutual authentication, the user  $U_i$  and the service provider server  $S_J$  computes  $S_k = H(B_i | N_i | N_J | y | SID_J)$  as the session key.

#### 7.2.1 Cryptanalysis of Liao and Wang's protocol

Liao and Wang [83] claimed that their protocol provides identity privacy and can resist various known attacks. This protocol protects the identity of the user efficiently. However, we found that this protocol is flawed for malicious server attack and malicious user attack.

##### 1. Malicious server attack

Suppose that a service provider server  $S_J$  is malicious. This malicious server  $S_J$  can compute the values of  $T_i$ ,  $H(P_i)$  and  $B_i$  corresponding to the user  $U_i$  during mutual verification and session key agreement phase. The server  $S_J$  also knows  $H(\ )$  function,  $y$  and  $H(x)$  because Liao and Wang mentioned that  $y$  is the shared key among the users, the servers and the registration center and  $H(x)$  is used by the legitimate server  $S_J$  to compute  $B_i = H(P_i) \oplus H(x)$ . The server  $S_J$  can record  $CID_i = H(P_i) \oplus H(T_i | y | N_i)$ ,  $Q_i = H(B_i | y | N_i)$ ,  $N_i$  during login request message from the user  $U_i$  and computes  $P_{im} = T_i \oplus H(y | N_i | SID_m)$  corresponding to the user  $U_i$ . Afterwards, the malicious server  $S_J$  sends the login request message  $(CID_i, P_{im}, Q_i, N_i)$  to the service provider server  $S_m$  by

masquerading as the user  $U_i$ . The service provider server  $S_m$  authenticates the received messages by calculating  $Q_i^*$  from the received messages and checks its equivalency with the received value of  $Q_i$ . After that, the server  $S_m$  generates a nonce value  $N_m$  and computes  $M_{im1} = H(B_i | N_i | y | SID_m)$  and sends the message  $(M_{im1}, N_m)$  back to the malicious server  $S_j$  who is masquerading as the user  $U_i$ . On receiving the message  $(M_{im1}, N_m)$ , the malicious server  $S_j$  computes  $M_{im2} = H(B_i | N_m | y | SID_m)$  and sends  $M_{im2}$  back to the service provider server  $S_m$ . On receiving the message  $M_{im2}$ , the service provider server  $S_m$  computes  $M_{im2}^* = H(B_i | N_m | y | SID_m)$  and compares it with the received value of  $M_{im2}$ . This equivalency assures the legitimacy of the user  $U_i$ . After the completion of mutual authentication phase, the malicious server masquerading as the user  $U_i$  and the service provider server  $S_m$  computes  $S_k = H(B_i | N_i | N_m | y | SID_m)$  as the session key.

## 2. Malicious user attack

The malicious user  $U_m$  can extract information like  $y$  and  $B_m = H(P_m) \oplus H(x)$  from his own smart card. He can also intercept the login request message  $(CID_i, P_{ij}, Q_i, N_i)$  of the user  $U_i$  to the service provider  $S_j$ . This malicious user  $U_m$  can compute  $H(x) = B_m \oplus H(P_m)$ ,  $T_i = P_{ij} \oplus H(y | N_i | SID_j)$ ,  $H(P_i) = CID_i \oplus H(T_i | y | N_i)$  and  $B_i = H(P_i) \oplus H(x)$ . Now this malicious user  $U_m$  can choose random nonce value  $N_m$  and computes  $CID_i = H(P_i) \oplus H(T_i | y | N_m)$ ,  $P_{ij} = T_i \oplus H(y | N_m | SID_j)$ ,  $Q_i = H(B_i | y | N_m)$  and masquerades as the legitimate user  $U_i$  by sending the login request message  $(CID_i, P_{ij}, Q_i, N_m)$  to the service provider server  $S_j$ . The service provider server  $S_j$  computes  $T_i = P_{ij} \oplus H(y | N_m | SID_j)$ ,  $H(P_i) = CID_i \oplus H(T_i | y | N_m)$ ,  $B_i = H(P_i) \oplus H(x)$ ,  $Q_i^* = H(B_i | y | N_m)$  and compares the equality of computed value of  $Q_i^*$  with the received value of  $Q_i$  to verify the legitimacy of the user  $U_i$ . Afterwards, the server  $S_j$  generates nonce value  $N_j$ , computes  $M_{ij1} = H(B_i | N_m | y | SID_j)$  and sends the message  $(M_{ij1}, N_j)$  back to the malicious user  $U_m$  who is masquerading as the user  $U_i$ . On receiving the message  $(M_{ij1}, N_j)$ , the malicious user  $U_m$  computes  $M_{ij2} = H(B_i | N_j | y | SID_j)$  and sends  $M_{ij2}$  back to the service provider server  $S_j$ . On receiving the message  $M_{ij2}$ , the service provider server  $S_j$  computes  $M_{ij2}^* = H(B_i | N_j | y | SID_j)$  and compares the computed value of  $M_{ij2}^*$  with the received value of  $M_{ij2}$  to verify the legitimacy of the user  $U_i$ . After finishing mutual authentication phase, the malicious user  $U_m$  masquerading as the user  $U_i$

and the service provider server  $S_j$  computes  $S_k = H(B_i | N_m | N_j | y | SID_j)$  as the session key.

### 7.2.2 Proposed protocol

In this section, we propose a dynamic identity based authentication protocol for multi-server architecture using smart cards that is free from all the attacks considered behind. The legitimate user  $U_i$  can easily login on to the service provider server using his smart card, identity and password. The notations used in this section are listed in Table 7.2. This protocol consists of four phases viz. registration phase, login phase, authentication & session key agreement phase and password change phase as summarized in Figure 7.2.

**Table 7.2**

**Notations**

$U_i$	User $U_i$
$S_m$	$m^{\text{th}}$ Service provider server
CS	Control server
$ID_i$	Unique identification of user $U_i$
$P_i$	Password of user $U_i$
$SID_m$	Unique identification of server $S_m$
$y_i$	Random value chosen by CS for user $U_i$
$H()$	One-way hash function
$x$	Master secret parameter of server CS
$N_1$	Random nonce value generated by user's smart card
$N_2$	Random nonce value generated by server $S_m$
$N_3$	Random nonce value generated by server CS
$\oplus$	XOR operation
	Concatenation

#### 1. Registration phase

When the user  $U_i$  wants to become a legitimate client, the user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the control server CS via a secure communication channel. The control server CS chooses and computes some security parameters and stores them on smart card of the user  $U_i$ . Then the control server CS issues smart card to the user  $U_i$ . The control server CS also sends some security parameters corresponding to newly registered user  $U_i$  to all service provider servers.

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader and submits his identity  $ID_i$ , password  $P_i$  and identity  $SID_m$  of the service provider server  $S_m$  to login on to the service provider server  $S_m$ . Smart card verifies authenticity of the user  $U_i$  and sends the user's and the server's verifier information to the destination server  $S_m$ .

## 3. Authentication and session key agreement phase

The service provider server  $S_m$  forwards the user's and the server's verifier information to the control server CS. Once CS authenticates the user  $U_i$  and the service provider server  $S_m$  then the control server CS sends some security parameters back to the server  $S_m$ . The server  $S_m$  verifies the authenticity of the control server CS using these security parameters. Then the server  $S_m$  sends some security parameters back to smart card of the user  $U_i$ . Using these security parameters, smart card of the user  $U_i$  verifies the legitimacy of the server  $S_m$  and the control server CS. Finally the CS, the service provider server  $S_m$  and the user  $U_i$  agree on the common session key.

## 4. Password change phase

The user  $U_i$  has to authenticate itself to smart card before requesting the password change.

## 1. Registration phase

The user  $U_i$  has to submit his identity  $ID_i$  and password  $P_i$  to the control server CS for its registration over a secure communication channel.

Step 1:  $U_i \rightarrow CS: ID_i, P_i$

The control server CS computes the security parameters  $Z_i = H(ID_i | P_i) \oplus H^2(x)$ ,  $V_i = y_i \oplus ID_i \oplus H(x)$ ,  $B_i = H(ID_i | P_i) \oplus P_i \oplus y_i$  and  $C_i = H(y_i) \oplus ID_i \oplus x$ , where  $x$  is the secret key of the CS and  $y_i$  is the random value chosen by the CS for the user  $U_i$ . The server CS chooses the value of  $y_i$  corresponding to the user  $U_i$  in such a way so that the value of  $C_i$  must be unique for each user. The server CS stores  $y_i \oplus x$  corresponding to  $C_i$  in its client's database. Then the server CS issues smart card containing security parameters  $(Z_i, V_i, B_i, H())$  to the user  $U_i$  through a secure communication channel.

Step 2:  $CS \rightarrow U_i: \text{Smart card}$

All service provider servers register themselves with CS and CS agrees on a unique secret key  $SK_m$  with each service provider server  $S_m$ . The server  $S_m$  remembers the secret

key  $SK_m$  and CS stores the secret key  $SK_m$  as  $SK_m \oplus H(x | SID_m)$  corresponding to service provider server identity  $SID_m$  in its service provider server's database.

The CS sends  $ID_i$  and  $H(y_i)$  corresponding to newly registered user  $U_i$  to all service provider servers. Each service provider server stores  $ID_i$  and  $H(y_i)$  in its database.

Step 3: CS  $\rightarrow S_m: ID_i, H(y_i)$

User $U_i$ Knows	Smart Card Stores	Service Provider Server $S_m$ Knows $SK_m$	CS Knows $x$
$ID_i$ and $P_i$	$Z_i = H(ID_i   P_i) \oplus H^2(x)$	Stores $ID_i, H(y_i)$	Computes $Z_i = H(ID_i   P_i) \oplus H^2(x)$
(Registration Phase)	$V_i = y_i \oplus ID_i \oplus H(x)$		$V_i = y_i \oplus ID_i \oplus H(x)$
Submits	$B_i = H(ID_i   P_i) \oplus P_i \oplus Y_i$		$B_i = H(ID_i   P_i) \oplus P_i \oplus Y_i$
$ID_i$ and $P_i$			$C_i = H(y_i) \oplus ID_i \oplus x$
(Login Phase)	Computes		Stores $C_i, y_i \oplus x$
Enter $ID_i^*, P_i^*$	$Y_i = B_i \oplus H(ID_i^*   P_i^*) \oplus P_i^*$	(Authentication & Session Key Agreement Phase)	Stores $SID_m, SK_m \oplus H(x   SID_m)$
and $SID_m$	$H(x) = V_i \oplus Y_i \oplus ID_i^*$	Generate Nonce Value $N_2$	Extract $SID_m, SK_m \oplus H(x   SID_m)$
	$Z_i^* = H(ID_i^*   P_i^*) \oplus H^2(x)$	Computes $G_i = N_2 \oplus SK_m$	$N_1 = M_i \oplus H^2(x), N_2 = G_i \oplus SK_m$
	Verifies $Z_i^* \stackrel{?}{=} Z_i$		$C_i^* = CID_i \oplus N_1 \oplus H(x) \oplus x$
	Generate Nonce Value $N_1$		Extract $C_i^*, y_i \oplus x$
	Computes $CID_i = V_i \oplus y_i \oplus H(y_i) \oplus N_1$	$SID_m, CID_i, M_i, E_i, G_i \rightarrow$	$ID_i = C_i \oplus H(y_i) \oplus x$
	$M_i = H^2(x) \oplus N_1$	Computes	$E_i^* = H(y_i   H(x)   N_1   ID_i   SID_m)$
	$E_i = H(y_i   H(x)   N_1   ID_i   SID_m)$	$N_1 \oplus N_3 = A_i \oplus H(SK_m   N_2)$	Verifies $E_i^* \stackrel{?}{=} E_i$
	$SID_m, CID_i, M_i, E_i \rightarrow$	$ID_i = D_i \oplus H(N_1 \oplus N_2 \oplus N_3)$	Generate Nonce Value $N_3$
		Extract $ID_i, H(y_i)$	Computes $A_i = N_1 \oplus N_3 \oplus H(SK_m   N_2)$
	Computes	Computes	$D_i = ID_i \oplus H(N_1 \oplus N_2 \oplus N_3)$
	$F_i^* = H[H(N_1 \oplus N_2 \oplus N_3)   ID_i   H(y_i)]$	Verifies $F_i^* \stackrel{?}{=} F_i$	$F_i = H[H(N_1 \oplus N_2 \oplus N_3)   ID_i   H(y_i)]$
	$N_2 \oplus N_3 = T_i \oplus H(y_i   ID_i   H(x)   N_1)$		$T_i = N_2 \oplus N_3 \oplus H(y_i   ID_i   H(x)   N_1)$
	$F_i^* = H[H(N_1 \oplus N_2 \oplus N_3)   ID_i   H(y_i)]$		$A_i, D_i, F_i, T_i \leftarrow$
	Verifies $F_i^* \stackrel{?}{=} F_i$	Computes Session Key	Computes Session Key
	Computes Session Key	$S_k = H(ID_i   (N_1 \oplus N_2 \oplus N_3)   H(y_i))$	$S_k = H(ID_i   (N_1 \oplus N_2 \oplus N_3)   H(y_i))$
$S_k = H(ID_i   (N_1 \oplus N_2 \oplus N_3)   H(y_i))$			

Figure 7.2: Proposed improvement in Liao and Wang's authentication protocol

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader and submits his identity  $ID_i^*$ , password  $P_i^*$  and the server identity  $SID_m$  to smart card in order to login on to the service provider server  $S_m$ . Then smart card computes  $y_i = B_i \oplus H(ID_i^* | P_i^*) \oplus P_i^*$ ,  $H(x) = V_i \oplus y_i \oplus ID_i^*$ ,  $Z_i^* = H(ID_i^* | P_i^*) \oplus H^2(x)$  and compares the computed value of  $Z_i^*$  with the stored value of  $Z_i$  in its memory to verify the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $Z_i^* \stackrel{?}{=} Z_i$

After verification, smart card generates random nonce value  $N_1$  and computes  $CID_i = V_i \oplus y_i \oplus H(y_i) \oplus N_1$ ,  $M_i = H^2(x) \oplus N_1$  and  $E_i = H(y_i | H(x) | N_1 | ID_i | SID_m)$ . Then



smart card sends the login request message  $(SID_m, CID_i, M_i, E_i)$  to the service provider server  $S_m$ .

Step 2: Smart card  $\rightarrow S_m: SID_m, CID_i, M_i, E_i$

### 3. Authentication and session key agreement phase

After receiving the login request from the user  $U_i$ , the server  $S_m$  generates random nonce value  $N_2$ , computes  $G_i = N_2 \oplus SK_m$  and sends the login request message  $(SID_m, CID_i, M_i, E_i, G_i)$  to the control server CS.

Step 1:  $S_k \rightarrow CS: SID_m, CID_i, M_i, E_i, G_i$

The control server CS extracts  $SK_m$  from  $SK_m \oplus H(x | SID_m)$  corresponding to  $SID_m$  in its service provider server's database. Then CS computes  $N_1 = M_i \oplus H^2(x)$ ,  $N_2 = G_i \oplus SK_m$ ,  $C_i^* = CID_i \oplus N_1 \oplus H(x) \oplus x$  and finds the matching value of  $C_i$  corresponding to  $C_i^*$  from its client database.

Step 2: Server CS checks  $C_i^* \stackrel{?}{=} C_i$

If the value of  $C_i^*$  does not match with any value of  $C_i$  in its client database, the CS rejects the login request and terminates this session. Otherwise, the CS extracts  $y_i$  from  $y_i \oplus x$  corresponding to  $C_i^*$  from its client database. Then the CS computes  $ID_i = C_i \oplus H(y_i) \oplus x$ ,  $E_i^* = H(y_i | H(x) | N_1 | ID_i | SID_m)$  and compares the computed value of  $E_i^*$  with the received value of  $E_i$  to verify the legitimacy of the user  $U_i$  and the service provider server  $S_m$ .

Step 3: Server CS checks  $E_i^* \stackrel{?}{=} E_i$

If they are not equal, the CS rejects the login request and terminates this session. Otherwise the CS generates random nonce value  $N_3$ , computes  $A_i = N_1 \oplus N_3 \oplus H(SK_m | N_2)$ ,  $D_i = ID_i \oplus H(N_1 \oplus N_2 \oplus N_3)$ ,  $F_i = H[H(N_1 \oplus N_2 \oplus N_3) | ID_i | H(y_i)]$ ,  $T_i = N_2 \oplus N_3 \oplus H(y_i | ID_i | H(x) | N_1)$  and sends the message  $(A_i, D_i, F_i, T_i)$  back to the service provider server  $S_m$ . The server  $S_m$  computes  $N_1 \oplus N_3 = A_i \oplus H(SK_m | N_2)$  from  $A_i$  and  $ID_i = D_i \oplus H(N_1 \oplus N_2 \oplus N_3)$  from  $D_i$ . Then the server  $S_m$  extracts  $H(y_i)$  corresponding to  $ID_i$  from its database. Afterwards, the server  $S_m$  computes  $F_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | ID_i | H(y_i)]$  and compares the computed value of  $F_i^*$  with the received value of  $F_i$  to verify the legitimacy of the control server CS.

Step 4: Server  $S_m$  checks  $F_i^* \neq F_i$

Then the server  $S_m$  sends  $(F_i, T_i)$  to smart card of the user  $U_i$ . Then smart card computes  $N_2 \oplus N_3 = T_i \oplus H(y_i | ID_i | H(x) | N_1)$ ,  $F_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | ID_i | H(y_i)]$  and compares the computed value of  $F_i^*$  with the received value of  $F_i$ .

Step 5: Smart card checks  $F_i^* \neq F_i$

This equivalency authenticates the legitimacy of the control server CS, the server  $S_m$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$ 's smart card, the server  $S_m$  and the control server CS agree on the common session key as  $S_k = H(ID_i | (N_1 \oplus N_2 \oplus N_3) | H(y_i))$ . Afterwards, all the subsequent messages between the user  $U_i$ , the server  $S_m$  and the CS are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server  $S_m$  or the server CS can retrieve the original message because all of them know the common session key.

#### 4. Password change phase

The user  $U_i$  can change his password without the help of control server CS. The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. Smart card computes  $y_i = B_i \oplus H(ID_i^* | P_i^*) \oplus P_i^*$ ,  $H(x) = V_i \oplus y_i \oplus ID_i^*$ ,  $Z_i^* = H(ID_i^* | P_i^*) \oplus H^2(x)$  and compares the computed value of  $Z_i^*$  with the stored value of  $Z_i$  in its memory to verifies the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified, the smart card asks the card holder to resubmit a new password  $P_i^{new}$ . Finally, the value of  $Z_i = H(ID_i | P_i) \oplus H^2(x)$  and  $B_i = H(ID_i | P_i) \oplus P_i \oplus y_i$  stored in the smart card is updated with  $Z_i^{new} = Z_i \oplus H(ID_i | P_i) \oplus H(ID_i | P_i^{new})$  and  $B_i^{new} = B_i \oplus H(ID_i | P_i) \oplus P_i \oplus H(ID_i | P_i^{new}) \oplus P_i^{new}$  and the password gets changed.

#### 7.2.3 Security analysis

A good password authentication scheme should provide protection from different possible attacks relevant to that protocol.

1. **Malicious server attack:** A malicious privileged server  $S_m$  can monitor the authentication process of the user  $U_i$  and can gather information related to the user  $U_i$ . The malicious server  $S_m$  can gather information  $CID_i = V_i \oplus y_i \oplus H(y_i) \oplus N_1$ ,  $M_i = H^2(x) \oplus N_1$  and  $E_i = H(y_i | H(x) | N_1 | ID_i | SID_m)$  during login phase corresponding to the legitimate user  $U_i$ . This malicious server  $S_m$  can not compute  $ID_i$ ,

$y_i$  and  $x$  from this information. This malicious server  $S_m$  can compute the identity  $ID_i$  from  $D_i$  and can extract  $H(y_i)$  corresponding to  $ID_i$  from its database corresponding to the user  $U_i$  during authentication and session key agreement phase. To masquerade as the legitimate user  $U_i$ , this malicious server  $S_m$  who knows the identity  $ID_i$  has to guess  $y_i$  and  $H(x)$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. In another option, this malicious server  $S_m$  has to get smart card of the user  $U_i$  and has to guess the correct identity  $ID_i$  and password  $P_i$  in order to login on to the server  $S_m$ . It is not possible to guess both these parameters correctly at the same time in real polynomial time even after getting the smart card of legitimate user  $U_i$ . Therefore, the proposed protocol is secure against malicious server attack.

2. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $Z_i = H(ID_i | P_i) \oplus H^2(x)$ ,  $V_i = y_i \oplus ID_i \oplus H(x)$  and  $B_i = H(ID_i | P_i) \oplus P_i \oplus y_i$  from the memory of smart card. The malicious user  $U_i$  can compute the value of  $H(x)$  from this information. The values of  $CID_m$ ,  $M_m$  and  $E_m$  is smart card specific and the malicious user  $U_i$  requires to know the values of  $H(x)$ ,  $y_m$  and  $ID_m$  to masquerades as the legitimate user  $U_m$ . Therefore, this malicious user  $U_i$  has to guess  $y_m$  and  $ID_m$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against malicious user attack.
3. **Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by an attacker, he can extract the information stored in the smart card. An attacker can extract  $Z_i = H(ID_i | P_i) \oplus H^2(x)$ ,  $V_i = y_i \oplus ID_i \oplus H(x)$  and  $B_i = H(ID_i | P_i) \oplus P_i \oplus y_i$  from the memory of smart card. Even after gathering this information, the attacker has to guess minimum two parameters out of  $ID_i$ ,  $H(x)$ ,  $y_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
4. **Identity protection:** Our approach provides identity protection in the sense that instead of sending the real identity  $ID_i$  of the user  $U_i$  in authentication, the pseudo identification  $CID_i = V_i \oplus y_i \oplus H(y_i) \oplus N_1$  is generated by smart card corresponding to the legitimate user  $U_i$  for its authentication to the service provider server  $S_m$  and the control server CS. There is no real identity information about the user during the login

and authentication & session key agreement phase. This approach provides the privacy and unlinkability among different login requests belonging to the same user. The attacker can not link different sessions belonging to the same user.

5. **Online dictionary attack:** In this type of attack, the attacker pretends to be legitimate user and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to get the valid smart card of the user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to the user  $U_i$ . Even after getting the valid smart card by any mean, an attacker gets a very few chances (normally a maximum of 3) to guess the identity and password because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.
6. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess user  $U_i$ 's identity  $ID_i$  and password  $P_i$  from recorded messages. An attacker first tries to obtains identity and password verification information such as  $Z_i = H (ID_i | P_i) \oplus H^2 (x)$ ,  $B_i = H (ID_i | P_i) \oplus P_i \oplus y_i$  and then try to guess the identity  $ID_i$  and password  $P_i$  by offline guessing. Here an attacker has to guess the identity  $ID_i$  and password  $P_i$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.
7. **Impersonation attack:** In this type of attack, the attacker impersonates as the legitimate user and forges the authentication messages using the information obtained from the authentication protocol. An attacker has to guess  $ID_i$ ,  $H (x)$  and  $y_i$  to masquerades as a legitimate user  $U_i$  to login on to the service provider server  $S_m$  to access the resources of the server  $S_m$ . It is not possible to guess all these parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against impersonation attack.
8. **Replay attack:** In this type of attack, the attacker first listens to communication between the user and the server and then tries to imitate the user to login on to the server by resending the captured messages transmitted between the user and the server.

Replaying a message of one session into another session is useless because the user's smart card, the server  $S_m$  and the control server CS choose different nonce values ( $N_1, N_2, N_3$ ) in each new session, which make all messages dynamic and valid for that session only. Therefore, replaying old dynamic identity and user's verifier information is useless. Moreover, the attacker can not compute the session key  $S_k = H (ID_i | (N_1 \oplus N_2 \oplus N_3) | H (y_i))$  because the user  $U_i$ 's smart card, the server  $S_m$  and the control server CS contribute different nonce values ( $N_1, N_2, N_3$ ) in each new session and the attacker does not know the values of  $ID_i, N_1, N_2, N_3$  and  $H (y_i)$ . Therefore, the proposed protocol is secure against replay attack.

9. **Leak of verifier attack:** In this type of attack, the attacker may able to steal the verification table from the server. If the attacker steals the verification table from the server, he can use the stolen verifiers to impersonate a participant of the scheme. In the proposed protocol, the service provider server  $S_m$  knows  $SK_m$ , stores  $ID_i$  and  $H (y_i)$  in its database. Similarly the control server CS knows the value of  $x$ , stores  $y_i \oplus x$  corresponding to  $C_i$  in its client's database,  $SK_m \oplus H (x | SID_m)$  corresponding to server identity  $SID_m$  in its service provider server's database. The attacker can not compute the values of  $x$  and  $y_i$  from the verifier information stored on the service provider server and the control server. In case verifier is stolen by breaking into smart card database, an attacker does not have sufficient information to calculate the user's identity and password. Therefore, the proposed protocol is secure against leak of verifier attack.
10. **Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the user's password or gaining unauthorized access. Modifying or inserting messages in proposed protocol can only cause authentication between the client and the server to fail but can not allow the attacker to gain any information about the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.
11. **Mutual authentication:** The goal of mutual authentication is to establish an agreed session key among the user  $U_i$ , the service provider server  $S_m$  and the control server CS. All three parties contribute their random nonce values as  $N_1, N_2$

and  $N_3$  for the derivation of session key  $S_k = H (ID_i | (N_1 \oplus N_2 \oplus N_3) | H (y_i))$ . The control server CS authenticates the user  $U_i$  using verifier information as  $E_i^* = H (y_i | H (x) | N_1 | ID_i | SID_m)$ , the service provider server  $S_m$  authenticates the server CS using  $F_i^* = H [H (N_1 \oplus N_2 \oplus N_3) | ID_i | H (y_i)]$  and the user  $U_i$  authenticates the server  $S_m$  and the server CS using  $F_i^* = H [H (N_1 \oplus N_2 \oplus N_3) | ID_i | H (y_i)]$ . The proposed protocol satisfies strong mutual authentication.

#### 7.2.4 Cost and functionality analysis

An efficient authentication protocol must take communication and computation cost into consideration during user's authentication. The cost comparison of the proposed protocol with the related smart card based authentication protocols is summarized in Table 7.3.

Table 7.3

Cost comparison among related smart card based multi-server authentication schemes

	Proposed Protocol	Liao & Wang [83]	Hsiang & Shih [49]	Chang & Lee [21]	Juang [61]	Lin et al. [87]
E1	384 bits (0.375  n )	512 bits (0.5  n )	640 bits (0.625  n )	256 bits (0.25  n )	256 bits (0.25  n )	(4t + 1)  n  bits
E2	9 *128 bits (1.125  n )	7*128 bits (0.875  n )	14 *128 bits (1.75  n )	5*128 bits (0.625  n )	9 *128 bits (1.125  n )	7*1024 bits (7  n )
E3	4T <sub>H</sub>	5T <sub>H</sub>	6 T <sub>H</sub>	2T <sub>H</sub>	T <sub>H</sub>	5tT <sub>E</sub>
E4	8 T <sub>H</sub>	9T <sub>H</sub>	10 T <sub>H</sub>	4T <sub>H</sub> + 3T <sub>S</sub>	3T <sub>H</sub> +3T <sub>S</sub>	2T <sub>E</sub>
E5	14T <sub>H</sub>	6T <sub>H</sub>	13 T <sub>H</sub>	4T <sub>H</sub> + 3T <sub>S</sub>	4T <sub>H</sub> + 8T <sub>S</sub>	7T <sub>E</sub>

Assume that the identity  $ID_i$ , password  $P_i$ ,  $x$ ,  $y_i$ , nonce values ( $N_1, N_2, N_3$ ) are all 128-bit long, prime modular operation is 1024-bits long as in most of practical implementations and  $t$  is the number of servers. Moreover, we assume that the output of secure one-way hash function and the block size of secure symmetric cryptosystem are 128-bit. Let  $T_H$ ,  $T_S$  and  $T_E$  are defined as the time complexity for hash function, symmetric encryption/decryption and exponential operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_E \gg T_H$ . In the proposed protocol, the parameters stored in the smart card are  $Z_i, V_i, B_i$  and the memory needed (E1) in the smart card is 384 (= 3\*128) bits. The communication cost of authentication (E2) includes the number of communication parameters involved in the authentication protocol. The number of communication parameters are  $\{SID_m, CID_i, M_i, E_i, G_i, A_i, D_i, F_i, T_i\}$  and hence the communication cost of authentication (E2) is 1152 (= 9\*128) bits. The computation cost of registration (E3) is the total time of all

operations executed by the user  $U_i$  in the registration phase. The computation cost of registration (E3) is  $4T_H$ . The computation cost of the user (E4) is the time spent by the user during the process of authentication. Therefore, the computation cost of the user (E4) is  $8T_H$ . The computation cost of the service provider server and the control server (E5) is the time spent by the service provider server and the control server during the process of authentication. Therefore, the computation cost of the service provider server and the control server (E5) is  $14T_H$ . The proposed protocol uses the control server CS and the service provider server  $S_m$  for the user's authentication that is why the computation cost of the servers (E5) is high as compared to Liao and Wang protocol [83]. On the other hand, the protocol proposed by Liao and Wang in 2009 totally relies on the service provider server  $S_m$  for the user's authentication and hence susceptible to malicious server attack and malicious user attack. The proposed protocol maintains the user's anonymity by generating dynamic identity and free from different attacks. The proposed protocol requires very less computation as compared to other related protocols [49][21][61][87] and also highly secure as compared to these related protocols. The functionality comparison of the proposed protocol with the related smart card based authentication protocols is summarized in Table 7.4.

Table 7.4

Functionality comparison among related smart card based multi-server authentication schemes

	Proposed Protocol	Liao & Wang [83]	Hsiang & Shih [49]	Chang & Lee [21]	Juang [61]	Lin et al. [87]
User's Anonymity	Yes	Yes	Yes	No	No	No
Computation Cost	Low	Low	Low	Low	Low	High
Single Registration	Yes	Yes	Yes	Yes	Yes	No
Session Key Agreement	Yes	Yes	Yes	Yes	Yes	No
Correct Password Update	Yes	Yes	No	No	No	No
No Time Synchronization	Yes	Yes	Yes	Yes	Yes	No
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	No
Multi Factor Security	Yes	Yes	Yes	No	No	No
Malicious Server Attack	No	Yes	No	Yes	Yes	No
Malicious User Attack	No	Yes	Yes	Yes	Yes	No

### 7.3 REVIEW OF HSIANG AND SHIH PROTOCOL

In this section, we describe the dynamic identity based remote user authentication protocol using smart cards for multi-server environment proposed by Hsiang and Shih [49] which is an improvement of Liao and Wang's protocol [83]. Their protocol includes four phases viz. registration phase, login phase, mutual verification & session key agreement phase

and password change phase. The notations used in this section are listed in Table 7.5 and the protocol is shown in Figure 7.3.

### 1. Registration phase

The user  $U_i$  selects a random number  $b$ , computes  $E_i = H(b \oplus P_i)$  and submits  $ID_i$  and  $E_i$  to the registration center RC for registration over a secure communication channel.

Step 1:  $U_i \rightarrow RC: ID_i, E_i$

**Table 7.5**  
**Notations**

$U_i$	User $U_i$
$S_J$	$J^{\text{th}}$ server
RC	Registration center
$ID_i$	Unique identification of user $U_i$
$P_i$	Password of user $U_i$
$SID_J$	Unique identification of server $S_J$
$CID_i$	Dynamic identity of user $U_i$
$H()$	One-way hash function
$x$	Master secret of registration center
$y \& r$	Secret number known to registration center
$\oplus$	XOR operation
	Concatenation

The RC computes the security parameters  $T_i = H(ID_i | x)$ ,  $V_i = T_i \oplus H(ID_i \oplus H(b \oplus P_i))$ ,  $A_i = H(H(b \oplus P_i) | r) \oplus H(x \oplus r)$ ,  $B_i = A_i \oplus H(b \oplus P_i)$ ,  $R_i = H(H(b \oplus P_i) | r)$  and  $H_i = H(T_i)$ . Then the RC issues the smart card containing security parameters  $(V_i, B_i, R_i, H_i, H())$  to the user  $U_i$  through a secure communication channel.

Step 2:  $RC \rightarrow U_i$ : Smart card

After that, user  $U_i$  enters the value of  $b$  in his smart card. Finally, the smart card contains security parameters as  $(V_i, B_i, R_i, H_i, H(), b)$  stored in its memory.

Step 3:  $U_i \rightarrow$  Smart card:  $b$

All service provider servers register themselves with RC. The RC computes  $H(SID_J | y)$  for service provider server  $S_J$  and sends this information to the server  $S_J$  over a secure communication channel. Similarly RC computes these server specific keys for all service provider servers and sends to them over a secure communication channel.



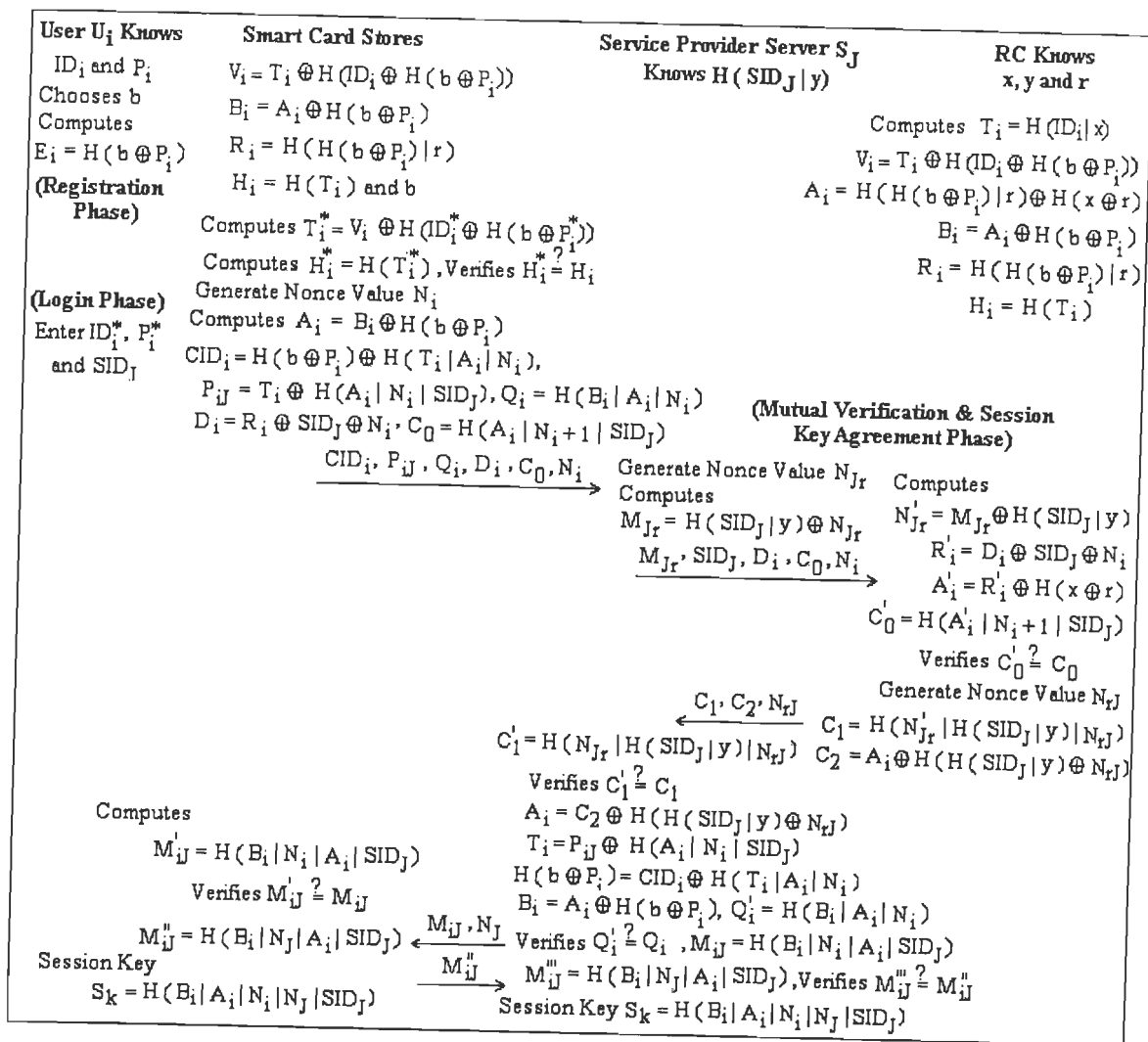


Figure 7.3: Hsiang and Shih's dynamic identity based multi-server authentication protocol

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S_J$  and submits his identity  $ID_i^*$ , password  $P_i^*$  and server identity  $SID_J$ . The smart card computes  $T_i^* = V_i \oplus H(ID_i^* \oplus H(b \oplus P_i^*))$ ,  $H_i^* = H(T_i^*)$  and compares  $H_i^*$  with the stored value of  $H_i$  in its memory to verifies the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $H_i^* \stackrel{?}{=} H_i$

After verification, smart card generates random nonce value  $N_i$  and computes  $A_i = B_i \oplus H(b \oplus P_i)$ ,  $CID_i = H(b \oplus P_i) \oplus H(T_i | A_i | N_i)$ ,  $P_{iJ} = T_i \oplus H(A_i | N_i | SID_J)$ ,  $Q_i = H(B_i | A_i | N_i)$ ,  $D_i = R_i \oplus SID_J \oplus N_i$  and  $C_0 = H(A_i | N_i + 1 | SID_J)$ . Afterwards, Smart card sends the login request message  $(CID_i, P_{iJ}, Q_i, D_i, C_0, N_i)$  to the service provider server  $S_J$ .

Step 2: Smart card  $\rightarrow S_J: CID_i, P_{iJ}, Q_i, D_i, C_0, N_i$

### 3. Mutual verification and session key agreement phase

The service provider server  $S_J$  generates random nonce value  $N_{Jr}$ , computes  $M_{Jr} = H(SID_J | y) \oplus N_{Jr}$  and then sends the message  $(M_{Jr}, SID_J, D_i, C_0, N_i)$  to the registration center RC.

Step 1:  $S_J \rightarrow RC: M_{Jr}, SID_J, D_i, C_0, N_i$

On receiving the message  $(M_{Jr}, SID_J, D_i, C_0, N_i)$ , the RC computes  $N_{Jr}' = M_{Jr} \oplus H(SID_J | y)$ ,  $R_i' = D_i \oplus SID_J \oplus N_i$ ,  $A_i' = R_i' \oplus H(x \oplus r)$ ,  $C_0' = H(A_i' | N_i + 1 | SID_J)$  and compares the computed value of  $C_0'$  with the received value of  $C_0$ . If they are not equal, the registration center RC rejects the login request and terminates this session.

Step 2: Registration center checks  $C_0' = C_0$

Otherwise the RC generates nonce value  $N_{rJ}$  and computes  $C_1 = H(N_{Jr}' | H(SID_J | y) | N_{rJ})$ ,  $C_2 = A_i \oplus H(H(SID_J | y) \oplus N_{rJ})$  and sends the message  $(C_1, C_2, N_{rJ})$  back to the server  $S_J$ . On receiving the message  $(C_1, C_2, N_{rJ})$ , the service provider server  $S_J$  computes  $C_1' = H(N_{Jr} | H(SID_J | y) | N_{rJ})$  and compares the computed value of  $C_1'$  with the received value of  $C_1$ . If they are not equal, the service provider server  $S_J$  rejects the login request and terminates this session.

Step 3: Service provider server  $S_J$  checks  $C_1' = C_1$

Then the server  $S_J$  computes  $A_i = C_2 \oplus H(H(SID_J | y) \oplus N_{rJ})$ ,  $T_i = P_{iJ} \oplus H(A_i | N_i | SID_J)$ ,  $H(b \oplus P_i) = CID_i \oplus H(T_i | A_i | N_i)$ ,  $B_i = A_i \oplus H(b \oplus P_i)$ ,  $Q_i' = H(B_i | A_i | N_i)$  and compares the computed value of  $Q_i'$  with the value of  $Q_i$  received in login request message. If they are not equal, the server  $S_J$  rejects the login request and terminates this session.

Step 4: Service provider server  $S_J$  checks  $Q_i' = Q_i$

Otherwise the server  $S_J$  generates random nonce value  $N_J$ , computes  $M_{iJ} = H(B_i | N_i | A_i | SID_J)$  and sends the message  $(M_{iJ}, N_J)$  back to smart card of the user  $U_i$ . On receiving the message  $(M_{iJ}, N_J)$ , the user  $U_i$ 's smart card computes  $M_{iJ}' = H(B_i | N_i | A_i | SID_J)$  and compares it with the received value of  $M_{iJ}$ . If they are not equal, the user  $U_i$ 's smart card rejects the login request and terminates this session.

Step 5: Smart card checks  $M_{iJ}' = M_{iJ}$

Otherwise the user  $U_i$ 's smart card computes  $M_{iJ}'' = H(B_i | N_J | A_i | SID_J)$  and sends the message  $M_{iJ}'''$  back to the service provider server  $S_J$ . Then the server  $S_J$  computes

$M_{ij}''' = H (B_i | N_j | A_i | SID_j)$  and compares it with the received value of  $M_{ij}''$ . If they are not equal, the server  $S_j$  rejects the login request and terminates this session.

Step 5: Service provider server  $S_j$  checks  $M_{ij}''' \stackrel{?}{=} M_{ij}''$

This equivalency authenticates the legitimacy of the user  $U_i$  and the login request is accepted else the connection is interrupted. Finally after mutual authentication, the user  $U_i$ 's smart card and the server  $S_j$  agree on the common session key as  $S_k = H (B_i | A_i | N_i | N_j | SID_j)$ .

#### 4. Password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. Then smart card computes  $T_i^* = V_i \oplus H (ID_i^* \oplus H (b \oplus P_i^*))$ ,  $H_i^* = H (T_i^*)$  and compares the computed value of  $H_i^*$  with the stored value of  $H_i$  in its memory to verifies the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct smart card to change his password. Afterwards, smart card asks the card holder to resubmit a new password  $P_i^{new}$ , then  $V_i = T_i \oplus H (ID_i \oplus H (b \oplus P_i))$  and  $B_i = H ( H (b \oplus P_i) | r) \oplus H (x \oplus r) \oplus H (b \oplus P_i)$  stored in smart card can be updated with  $V_i^{new} = T_i \oplus H (ID_i \oplus H (b \oplus P_i^{new}))$  and  $B_i^{new} = B_i \oplus H (b \oplus P_i) \oplus H (b \oplus P_i^{new})$  and password gets changed.

#### 7.3.1 Cryptanalysis of Hsiang and Shih's protocol

Hsiang and Shih [49] claimed that their protocol provides identity privacy and can resist various known attacks. This protocol protects the identity of the user efficiently. However, we found that this protocol is flawed for replay attack, impersonation attack and stolen smart card attack. Moreover, the password change phase of Hsiang and Shih's protocol is incorrect.

##### 1. Replay attack

A malicious privileged user  $U_k$  having his own smart card can gather information  $(V_k, B_k, R_k, H_k, H ( ), b_k)$  from his own smart card. He can compute the value of  $A_k$  as  $A_k = B_k \oplus H (b_k \oplus P_k)$  because this malicious user  $U_k$  knows the value of  $b_k$  and his own password  $P_k$  corresponding to his smart card. Then this malicious user  $U_k$  can compute the value of  $H (x \oplus r)$  as  $H (x \oplus r) = A_k \oplus R_k$ . Now this malicious user  $U_k$  can intercept a valid login request message  $(CID_i, P_{ij}, Q_i, D_i, C_0, N_i)$  of the user  $U_i$  from the public

communication channel. Then the malicious user  $U_k$  can compute  $R_i = D_i \oplus SID_J \oplus N_i$ ,  $A_i = R_i \oplus H(x \oplus r)$ ,  $T_i = P_{ij} \oplus H(A_i | N_i | SID_J)$ ,  $H(b \oplus P_i) = CID_i \oplus H(T_i | A_i | N_i)$  and  $B_i = A_i \oplus H(b \oplus P_i)$  corresponding to the user  $U_i$ . The malicious user  $U_k$  can replay this valid login request message  $(CID_i, P_{ij}, Q_i, D_i, C_0, N_i)$  to the server  $S_J$  by masquerading as the user  $U_i$  at some time latter. This valid login request message is verified by the registration center RC and the server  $S_J$ . After verification of login request message, the server  $S_J$  computes  $M_{ij} = H(B_i | N_i | A_i | SID_J)$  and sends the message  $(M_{ij}, N_J)$  to the user  $U_k$  who is masquerading as the user  $U_i$ . The masquerading user  $U_k$  can verify the received value of  $M_{ij}$  because he knows the values of  $B_i, N_i, A_i$  and  $SID_J$ . Then the masquerading user  $U_k$  can compute  $M_{ij}'' = H(B_i | N_J | A_i | SID_J)$  and sends the message  $M_{ij}''$  back to the server  $S_J$ . Then the server  $S_J$  computes  $M_{ij}''' = H(B_i | N_J | A_i | SID_J)$  and verifies it with the received value of  $M_{ij}''$ . This equivalency authenticates the legitimacy of the user  $U_i$ , the service provider server  $S_J$  and the login request is accepted. Finally after mutual authentication, the malicious user  $U_k$  masquerading as the user  $U_i$  and the server  $S_J$  agree on the common session key as  $S_k = H(B_i | A_i | N_i | N_J | SID_J)$ .

## 2. Impersonation attack

A malicious privileged user  $U_k$  having his own smart card can gather information  $(V_k, B_k, R_k, H_k, H(\cdot), b_k)$  from his own smart card. He can compute the value of  $H(x \oplus r)$  as shown in the replay attack. Now this malicious user  $U_k$  can intercept a valid login request message  $(CID_i, P_{ij}, Q_i, D_i, C_0, N_i)$  of the user  $U_i$  from the public communication channel. Then the malicious user  $U_k$  can compute  $R_i = D_i \oplus SID_J \oplus N_i$ ,  $A_i = R_i \oplus H(x \oplus r)$ ,  $T_i = P_{ij} \oplus H(A_i | N_i | SID_J)$ ,  $H(b \oplus P_i) = CID_i \oplus H(T_i | A_i | N_i)$  and  $B_i = A_i \oplus H(b \oplus P_i)$  corresponding to the user  $U_i$ . This malicious user  $U_k$  can choose random nonce value  $N_i'$  and computes  $CID_i = H(b \oplus P_i) \oplus H(T_i | A_i | N_i')$ ,  $P_{im} = T_i \oplus H(A_i | N_i' | SID_m)$ ,  $Q_i = H(B_i | A_i | N_i')$ ,  $D_i = R_i \oplus SID_m \oplus N_i'$  and  $C_0 = H(A_i | N_i' + 1 | SID_m)$ . Now this malicious user  $U_k$  can send valid login request message  $(CID_i, P_{im}, Q_i, D_i, C_0, N_i')$  by masquerading as the user  $U_i$  to the server  $S_m$ . This valid login request message is verified by the registration center RC and the server  $S_m$ . After verification of login request message, the server  $S_m$  computes  $M_{im} = H(B_i | N_i' | A_i | SID_m)$  and sends the message  $(M_{im}, N_m)$  to the user  $U_k$  who is masquerading as the user  $U_i$ . The masquerading user  $U_k$  can verify the received value of  $M_{im}$  because he knows the values of  $B_i, N_i', A_i$  and

$SID_m$ . Then the masquerading user  $U_k$  can compute  $M_{im}'' = H (B_i | N_m | A_i | SID_m)$  and sends the message  $M_{im}''$  back to the server  $S_m$ . Then the server  $S_m$  computes  $M_{im}''' = H (B_i | N_m | A_i | SID_m)$  and verifies it with the received value of  $M_{im}''$ . This equivalency authenticates the legitimacy of the user  $U_i$ , the service provider server  $S_m$  and the login request is accepted. Finally after mutual authentication, the malicious user  $U_k$  masquerading as the user  $U_i$  and the server  $S_m$  agree on the common session key as  $S_k = H (B_i | A_i | N_i' | N_m | SID_m)$ .

### 3. Stolen smart card attack

A malicious privileged user  $U_k$  having his own smart card can gather information  $(V_k, B_k, R_k, H_k, H ( ), b_k)$  from his own smart card. He can find out the value of  $H (x \oplus r)$  as shown in the replay attack. Now this malicious user  $U_k$  can intercept a valid login request message  $(CID_i, P_{ij}, Q_i, D_i, C_0, N_i)$  of the user  $U_i$  from the public communication channel. Then the malicious user  $U_k$  can compute  $R_i = D_i \oplus SID_j \oplus N_i$ ,  $A_i = R_i \oplus H (x \oplus r)$ ,  $T_i = P_{ij} \oplus H (A_i | N_i | SID_j)$ ,  $H (b \oplus P_i) = CID_i \oplus H (T_i | A_i | N_i)$  and  $B_i = A_i \oplus H (b \oplus P_i)$  corresponding to the user  $U_i$ .

1. In case the user  $U_i$ 's smart card is stolen by this malicious user  $U_k$ , he can extract the information  $(V_i, B_i, H_i, R_i, H ( ), b)$  from the memory of smart card.
2. Then the malicious user  $U_k$  can launch offline dictionary attack on  $V_i = T_i \oplus H (ID_i \oplus H (b \oplus P_i))$  to know the identity  $ID_i$  of the user  $U_i$  because the malicious user  $U_k$  knows the values of  $T_i$  and  $H (b \oplus P_i)$  corresponding to the user  $U_i$ .
3. Then this malicious user  $U_k$  can launch offline dictionary attack on  $H (b \oplus P_i)$  to know the password  $P_i$  of the user  $U_i$  because the malicious user  $U_k$  knows the value of  $b$  from smart card of the user  $U_i$ .

Now this malicious user  $U_k$  possesses the valid smart card of user  $U_i$ , knows the identity  $ID_i$ , password  $P_i$  corresponding to the user  $U_i$  and hence can login on to any service provider server.

### 4. Incorrect password change phase

The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. Then smart card computes  $T_i^* = V_i \oplus H (ID_i^* \oplus H (b \oplus P_i^*))$ ,  $H_i^* = H (T_i^*)$  and compares  $H_i^*$  with the stored value of  $H_i$

in its memory to verifies the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified then the user  $U_i$  can instruct smart card to change his password. Afterwards, smart card asks the card holder to resubmit a new password  $P_i^{new}$ , then  $V_i = T_i \oplus H (ID_i \oplus H (b \oplus P_i))$  and  $B_i = H ( H (b \oplus P_i) | r) \oplus H (x \oplus r) \oplus H (b \oplus P_i)$  stored in the smart card can be replaced with  $V_i^{new} = T_i \oplus H (ID_i \oplus H (b \oplus P_i^{new}))$  and  $B_i^{new} = B_i \oplus H (b \oplus P_i) \oplus H (b \oplus P_i^{new}) = H ( H (b \oplus P_i) | r) \oplus H (x \oplus r) \oplus H (b \oplus P_i^{new})$ . The  $B_i^{new}$  value contains older password  $P_i$  in  $H ( H (b \oplus P_i) | r)$ . Therefore, the modified  $B_i^{new}$  is not correct. Moreover, smart card of the user  $U_i$  does not know the value of  $r$  and hence can not compute the correct value of  $B_i^{new}$ . Moreover, the value of  $R_i = H ( H (b \oplus P_i) | r)$  also contains password  $P_i$ , which has not been updated by smart card of the user  $U_i$  in password change phase. Smart card does not know the value of  $r$  and hence can not compute the correct new  $R_i^{new}$  value. Therefore, the password change phase of Hsiang and Shih's protocol is incorrect.

### 7.3.2 Proposed protocol

In this section, we propose a dynamic identity based authentication protocol for multi-server architecture using smart cards that is free from all the attacks considered behind. The legitimate user  $U_i$  can easily login on to the service provider server using his smart card, identity and password. The notations used in this section are listed in Table 7.2. This protocol consists of four phases viz. registration phase, login phase, authentication & session key agreement phase and password change phase as summarized in Figure 7.4.

#### 1. Registration phase

When the user  $U_i$  wants to become a legitimate client, the user  $U_i$  has to submit his identity and password verifier information to the control server CS via a secure communication channel. Then the CS chooses and computes some security parameters and stores them on the smart card of the user  $U_i$ . Then the CS issues smart card to the user  $U_i$ . Also the user  $U_i$  computes and stores some security parameters on his smart card.

#### 2. Login phase

The user  $U_i$  inserts his smart card into a card reader and submits his identity  $ID_i$ , password  $P_i$  and identity  $SID_m$  of service provider server  $S_m$  to login on to the service provider server  $S_m$ . Smart card verifies authenticity of the user  $U_i$  and sends the user's and the server's verifier information to the destination server  $S_m$ .

### 3. Authentication and session key agreement phase

The service provider server  $S_m$  forwards the user's and the server's verifier information to the CS. Once CS authenticates the user  $U_i$  and the service provider server  $S_m$  then the CS sends some security parameters back to the server  $S_m$ . The server  $S_m$  verifies the authenticity of the CS using these security parameters. Then the server  $S_m$  sends some security parameters back to smart card of the user  $U_i$ . Using these security parameters, smart card of the user  $U_i$  verifies the legitimacy of the server  $S_m$  and the CS. Finally the CS, the service provider server  $S_m$  and the user  $U_i$  agree on the common session key.

### 4. Password change phase

The user  $U_i$  has to authenticate itself to smart card before requesting the password change.

### 1. Registration phase

The user  $U_i$  selects a random number  $b$ , computes  $A_i = H(\text{ID}_i | b)$ ,  $B_i = H(b \oplus P_i)$  and submits  $A_i$  and  $B_i$  to the control server CS for registration over a secure communication channel.

Step 1:  $U_i \rightarrow \text{CS}: A_i, B_i$

The CS computes the security parameters  $F_i = A_i \oplus y_i$ ,  $G_i = B_i \oplus H(y_i) \oplus H(x)$  and  $C_i = A_i \oplus H(y_i) \oplus x$ , where  $x$  is the secret key of the CS and  $y_i$  is the random value chosen by the CS for the user  $U_i$ . The server CS chooses the value of  $y_i$  corresponding to the user  $U_i$  in such a way so that the value of  $C_i$  must be unique for each user. Then the CS stores  $y_i \oplus x$  corresponding to  $C_i$  in its client's database. Then the CS issues smart card containing security parameters  $(F_i, G_i, H(\ ))$  to the user  $U_i$  through a secure communication channel.

Step 2:  $\text{CS} \rightarrow U_i: \text{Smart card}$

After that, the user  $U_i$  computes security parameters  $D_i = b \oplus H(\text{ID}_i | P_i)$ ,  $E_i = H(\text{ID}_i | P_i) \oplus P_i$  and enters the value of  $D_i$  and  $E_i$  in his smart card. Finally, the smart card contains security parameters as  $(D_i, E_i, F_i, G_i, H(\ ))$  stored in its memory.

Step 3:  $U_i \rightarrow \text{Smart card}: D_i, E_i$

All service provider servers register themselves with CS and CS agrees on a unique secret key  $SK_m$  with each service provider server  $S_m$ . The server  $S_m$  remembers the secret key  $SK_m$  and CS stores the secret key  $SK_m$  as  $SK_m \oplus H(x | \text{SID}_m)$  corresponding to service provider server identity  $\text{SID}_m$  in its service provider server's database.

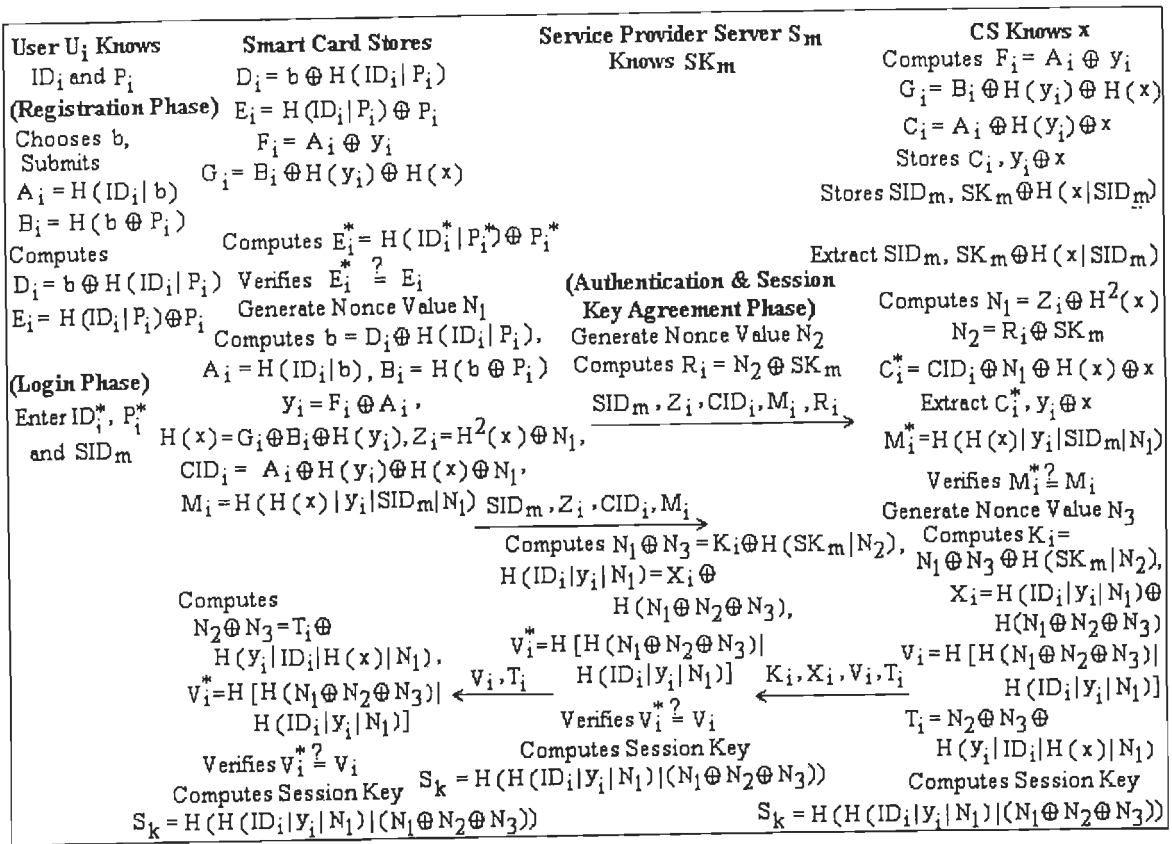


Figure 7.4: Proposed improvement in Hsiang and Shih's authentication protocol

## 2. Login phase

The user  $U_i$  inserts his smart card into a card reader to login on to the server  $S_m$  and submits his identity  $ID_i^*$ , password  $P_i^*$  and server identity  $SID_m$ . The smart card computes  $E_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $E_i$  in its memory to verify the legitimacy of the user  $U_i$ .

Step 1: Smart card checks  $E_i^* \stackrel{?}{=} E_i$

After verification, smart card generates random nonce value  $N_1$  and computes  $b = D_i \oplus H(ID_i | P_i)$ ,  $A_i = H(ID_i | b)$ ,  $B_i = H(b \oplus P_i)$ ,  $y_i = F_i \oplus A_i$ ,  $H(x) = G_i \oplus B_i \oplus H(y_i)$ ,  $Z_i = H^2(x) \oplus N_1$ ,  $CID_i = A_i \oplus H(y_i) \oplus H(x) \oplus N_1$  and  $M_i = H(H(x) | y_i | SID_m | N_1)$ . Then smart card sends the login request message  $(SID_m, Z_i, CID_i, M_i)$  to the service provider server  $S_m$ .

Step 2: Smart card  $\rightarrow S_m: SID_m, Z_i, CID_i, M_i$

## 3. Authentication and session key agreement phase

After receiving the login request from the user  $U_i$ , the server  $S_m$  generates random nonce value  $N_2$ , computes  $R_i = N_2 \oplus SK_m$  and sends the login request message  $(SID_m, Z_i, CID_i, M_i, R_i)$  to the CS.



Step 1:  $S_m \rightarrow CS: SID_m, Z_i, CID_i, M_i, R_i$

The CS extracts  $SK_m$  from  $SK_m \oplus H(x | SID_m)$  corresponding to  $SID_m$  in its service provider server's database. Then CS computes  $N_1 = Z_i \oplus H^2(x)$ ,  $N_2 = R_i \oplus SK_m$ ,  $C_i^* = CID_i \oplus N_1 \oplus H(x) \oplus x$  and finds the matching value of  $C_i$  corresponding to  $C_i^*$  from its client database.

Step 2: Server CS checks  $C_i^* \stackrel{?}{=} C_i$

If the value of  $C_i^*$  does not match with any value of  $C_i$  in its client database, the CS rejects the login request and terminates this session. Otherwise, the CS extracts  $y_i$  from  $y_i \oplus x$  corresponding to  $C_i^*$  from its client database. Then the CS computes  $M_i^* = H(H(x) | y_i | SID_m | N_1)$  and compares  $M_i^*$  with the received value of  $M_i$  to verify the legitimacy of the user  $U_i$  and the service provider server  $S_k$ .

Step 3: Control server CS checks  $M_i^* \stackrel{?}{=} M_i$

If they are not equal, the control server CS rejects the login request and terminates this session. Otherwise the CS generates random nonce value  $N_3$ , computes  $K_i = N_1 \oplus N_3 \oplus H(SK_m | N_2)$ ,  $X_i = H(ID_i | y_i | N_1) \oplus H(N_1 \oplus N_2 \oplus N_3)$ ,  $V_i = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$ ,  $T_i = N_2 \oplus N_3 \oplus H(y_i | ID_i | H(x) | N_1)$  and sends the message  $(K_i, X_i, V_i, T_i)$  back to the service provider server  $S_m$ . The server  $S_m$  computes  $N_1 \oplus N_3 = K_i \oplus H(SK_m | N_2)$  from  $K_i$  and  $H(ID_i | y_i | N_1) = X_i \oplus H(N_1 \oplus N_2 \oplus N_3)$  from  $X_i$ . Then the server  $S_m$  computes  $V_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$  and compares the computed value of  $V_i^*$  with the received value of  $V_i$  to verify the legitimacy of the control server CS.

Step 4: Server  $S_m$  checks  $V_i^* \stackrel{?}{=} V_i$

Then the server  $S_m$  sends  $(V_i, T_i)$  to smart card of the user  $U_i$ . Then smart card computes  $N_2 \oplus N_3 = T_i \oplus H(y_i | ID_i | H(x) | N_1)$ ,  $V_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$  and compares the computed value of  $V_i^*$  with the received value of  $V_i$ .

Step 5: Smart card checks  $V_i^* \stackrel{?}{=} V_i$

This equivalency authenticates the legitimacy of the control server CS, the server  $S_m$  and the login request is accepted else the connection is interrupted. Finally, the user  $U_i$ 's smart card, the server  $S_m$  and the control server CS agree on the common session key as  $S_k = H(H(ID_i | y_i | N_1) | (N_1 \oplus N_2 \oplus N_3))$ . Afterwards, all the subsequent messages

between the user  $U_i$ , the server  $S_m$  and the CS are XOR<sup>ed</sup> with the session key. Therefore, either the user  $U_i$  or the server  $S_m$  or the server CS can retrieve the original message because all of them know the common session key.

#### 4. Password change phase

The user  $U_i$  can change his password without the help of control server CS. The user  $U_i$  inserts his smart card into a card reader and enters his identity  $ID_i^*$  and password  $P_i^*$  corresponding to his smart card. Smart card computes  $E_i^* = H (ID_i^* | P_i^*) \oplus P_i^*$  and compares the computed value of  $E_i^*$  with the stored value of  $E_i$  in its memory to verifies the legitimacy of the user  $U_i$ . Once the authenticity of card holder is verified, smart card computes the values of  $b$ ,  $H (y_i)$ ,  $H (x)$  and then asks the card holder to resubmit a new password  $P_i^{new}$ . Finally, the values of  $D_i = b \oplus H (ID_i | P_i)$ ,  $E_i = H (ID_i | P_i) \oplus P_i$  and  $G_i = H (b \oplus P_i) \oplus H (y_i) \oplus H (x)$  stored in the smart card is updated with  $D_i^{new} = b \oplus H (ID_i | P_i^{new})$ ,  $E_i^{new} = H (ID_i | P_i^{new}) \oplus P_i^{new}$  and  $G_i^{new} = H (b \oplus P_i^{new}) \oplus H (y_i) \oplus H (x)$  and password gets changed.

#### 7.3.3 Security analysis

A good password authentication scheme should provide protection from different possible attacks relevant to that protocol.

1. **Replay attack:** In this type of attack, the attacker first listens to communication between the user and the server and then tries to imitate the user to login on to the server by resending the captured messages transmitted between the user and the server. Replaying a message of one session into another session is useless because the user's smart card, the server  $S_m$  and the control server CS choose different nonce values ( $N_1$ ,  $N_2$ ,  $N_3$ ) in each new session, which make all messages dynamic and valid for that session only. Therefore, replaying old dynamic identity and user's verifier information is useless. Moreover, the attacker can not compute the session key  $S_k = H (H (ID_i | y_i | N_1) | (N_1 \oplus N_2 \oplus N_3))$  because the user  $U_i$ 's smart card, the server  $S_m$  and the control server CS contributes different nonce values ( $N_1$ ,  $N_2$ ,  $N_3$ ) in each new session and the attacker does not know the values of  $ID_i$ ,  $y_i$ ,  $N_1$ ,  $N_2$  and  $N_3$ . Therefore, the proposed protocol is secure against replay attack.

2. **Impersonation attack:** In this type of attack, the attacker impersonates as the legitimate user and forges the authentication messages using the information obtained from the authentication protocol. An attacker has to guess  $A_i$ ,  $H(x)$  and  $y_i$  to masquerades as a legitimate user  $U_i$  to login on to the service provider server  $S_m$  to access the resources of the server  $S_m$ . It is not possible to guess all these parameters correctly at the same time in real polynomial time. Moreover, the attacker can not compute  $A_i$ ,  $H(x)$  and  $y_i$  from intercepted communication parameters  $Z_i$ ,  $CID_i$ ,  $M_i$ ,  $R_i$ ,  $K_i$ ,  $X_i$ ,  $V_i$ ,  $T_i$  over insecure communication channel. Therefore, the proposed protocol is secure against impersonation attack.
3. **Stolen smart card attack:** In case a user  $U_i$ 's smart card is stolen by an attacker, he can extract the information stored in the smart card. An attacker can extract  $D_i = b \oplus H(ID_i | P_i)$ ,  $E_i = H(ID_i | P_i) \oplus P_i$ ,  $F_i = A_i \oplus y_i$  and  $G_i = B_i \oplus H(y_i) \oplus H(x)$  from the memory of smart card. Even after gathering this information, an attacker has to guess minimum two parameters out of  $ID_i$ ,  $H(x)$ ,  $y_i$  and  $P_i$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against stolen smart card attack.
4. **Malicious server attack:** A malicious privileged server  $S_m$  can monitor the authentication process of the user  $U_i$  and can gather information related to the user  $U_i$ . The malicious server  $S_m$  can gather information  $Z_i = H^2(x) \oplus N_1$ ,  $CID_i = A_i \oplus H(y_i) \oplus H(x) \oplus N_1$  and  $M_i = H(H(x) | y_i | SID_m | N_1)$  during login phase corresponding to the legitimate user  $U_i$ . This malicious server  $S_m$  can not compute  $ID_i$ ,  $y_i$  and  $x$  from this information. The malicious server  $S_m$  can not compute  $ID_i$ ,  $y_i$  and  $x$  from  $K_i = N_1 \oplus N_3 \oplus H(SK_m | N_2)$ ,  $X_i = H(ID_i | y_i | N_1) \oplus H(N_1 \oplus N_2 \oplus N_3)$ ,  $V_i = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$  and  $T_i = N_2 \oplus N_3 \oplus H(y_i | ID_i | H(x) | N_1)$ . Therefore, the proposed protocol is secure against malicious server attack.
5. **Malicious user attack:** A malicious privileged user  $U_i$  having his own smart card can gather information like  $D_i = b \oplus H(ID_i | P_i)$ ,  $E_i = H(ID_i | P_i) \oplus P_i$ ,  $F_i = A_i \oplus y_i$  and  $G_i = B_i \oplus H(y_i) \oplus H(x)$  from the memory of smart card. The malicious user  $U_i$  can compute the value of  $H(x)$  from this information. The value of  $CID_m$  and  $M_m$  is smart card specific and the malicious user  $U_i$  requires to know the values of  $H(x)$ ,  $y_m$  and  $A_m$

to masquerades as the legitimate user  $U_m$ . Therefore, this malicious user  $U_i$  has to guess  $y_m$  and  $A_m$  correctly at the same time. It is not possible to guess out two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against malicious user attack.

6. **Leak of verifier attack:** In this type of attack, the attacker may able to steal the verification table from the server. If the attacker steals the verification table from the server, he can use the stolen verifiers to impersonate a participant of the scheme. In the proposed protocol, the service provider server  $S_m$  knows  $SK_m$  and does not store any information in its database. Similarly the control server CS knows the value of  $x$ , stores  $y_i \oplus x$  corresponding to  $C_i$  in its client's database,  $SK_m \oplus H(x | SID_m)$  corresponding to server identity  $SID_m$  in its service provider server's database. The attacker can not compute the values of  $x$  and  $y_i$  from the verifier information stored on the control server. In case verifier is stolen by breaking into smart card database, an attacker does not have sufficient information to calculate the user's identity and password. Therefore, the proposed protocol is secure against leak of verifier attack.
7. **Offline dictionary attack:** In offline dictionary attack, the attacker can record messages and attempts to guess user  $U_i$ 's identity  $ID_i$  and password  $P_i$  from recorded messages. An attacker first tries to obtains identity and password verification information such as  $D_i = b \oplus H(ID_i | P_i)$ ,  $E_i = H(ID_i | P_i) \oplus P_i$ ,  $F_i = A_i \oplus y_i$  and  $G_i = B_i \oplus H(y_i) \oplus H(x)$  and then try to guess the identity  $ID_i$  and password  $P_i$  by offline guessing. Here an attacker has to guess the identity  $ID_i$  and password  $P_i$  correctly at the same time. It is not possible to guess two parameters correctly at the same time in real polynomial time. Therefore, the proposed protocol is secure against offline dictionary attack.
8. **Online dictionary attack:** In this type of attack, the attacker pretends to be legitimate user and attempts to login on to the server by guessing different words as password from a dictionary. In the proposed protocol, the attacker has to get the valid smart card of the user  $U_i$  and then has to guess the identity  $ID_i$  and password  $P_i$  corresponding to the user  $U_i$ . Even after getting the valid smart card of user  $U_i$  by any mean, an attacker gets a very few chances (normally a maximum of 3) to guess the identity and password because smart card gets locked after certain number of unsuccessful attempts. Moreover, it is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same

time in real polynomial time. Therefore, the proposed protocol is secure against online dictionary attack.

9. **Identity protection:** Our approach provides identity protection in the sense that instead of sending the real identity  $ID_i$  of the user  $U_i$  in authentication, the pseudo identification  $CID_i = A_i \oplus H(y_i) \oplus H(x) \oplus N_1$  is generated by smart card corresponding to the legitimate user  $U_i$  for its authentication to the service provider server  $S_m$  and the control server CS. There is no real identity information about the user during the login and authentication & session key agreement phase. This approach provides the privacy and unlinkability among different login requests belonging to the same user. The attacker can not link different sessions belonging to the same user.
10. **Mutual authentication:** The goal of mutual authentication is to establish an agreed session key among the user  $U_i$ , the service provider server  $S_m$  and the control server CS. All three parties contribute their random nonce values as  $N_1$ ,  $N_2$  and  $N_3$  for the derivation of session key  $S_k = H(H(ID_i | y_i | N_1) | (N_1 \oplus N_2 \oplus N_3))$ . The control server CS authenticates the user  $U_i$  using verifier information as  $M_i^* = H(H(x) | y_i | SID_m | N_1)$ , the service provider server  $S_m$  authenticates the server CS using  $V_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$  and the user  $U_i$  authenticates the server  $S_m$  and the server CS using  $V_i^* = H[H(N_1 \oplus N_2 \oplus N_3) | H(ID_i | y_i | N_1)]$ . The proposed protocol satisfies strong mutual authentication.
11. **Denial of service attack:** In this type of attack, an attacker updates identity and password verification information on smart card to some arbitrary value and hence legitimate user can not login successfully in subsequent login request to the server. In the proposed protocol, smart card checks the validity of user  $U_i$ 's identity  $ID_i$  and password  $P_i$  before password update procedure. An attacker can insert the stolen smart card of the user  $U_i$  into smart card reader and has to guess the identity  $ID_i$  and password  $P_i$  correctly corresponding to the user  $U_i$ . Since the smart card computes  $E_i^* = H(ID_i^* | P_i^*) \oplus P_i^*$  and compares it with the stored value of  $E_i$  in its memory to verifies the legitimacy of the user  $U_i$  before smart card accepts password update request. It is not possible to guess identity  $ID_i$  and password  $P_i$  correctly at the same time in real polynomial time even after getting the smart card of the user  $U_i$ . Therefore, the proposed protocol is secure against denial of service attack.

**12. Parallel session attack:** In this type of attack, an attacker first listens to communication between the client and the server. After that, he initiates a parallel session to imitate legitimate user to login on to the server by resending the captured messages transmitted between the client and the server within the valid time frame window. He can masquerade as legitimate user  $U_i$  by replaying a login request message  $(SID_m, Z_i, CID_i, M_i)$  but cannot compute the agreed session key  $S_k = H(H(ID_i | y_i | N_1) | (N_1 \oplus N_2 \oplus N_3))$  because an attacker does not know the values of  $ID_i, y_i, N_1, N_2$  and  $N_3$ . Therefore, the proposed protocol is secure against parallel session attack.

**13. Man-in-the-middle attack:** In this type of attack, the attacker intercepts the messages sent between the client and the server and replays these intercepted messages. An attacker can act as a client to the server or vice-versa with recorded messages. In the proposed protocol, an attacker can intercept the login request message  $(SID_m, Z_i, CID_i, M_i)$  from the user  $U_i$  to the server  $S_m$ . Then he starts a new session with the server  $S_m$  by sending a login request by replaying the login request message  $(SID_m, Z_i, CID_i, M_i)$ . An attacker can authenticate itself to the control server CS but cannot compute the session key  $S_k = H(H(ID_i | y_i | N_1) | (N_1 \oplus N_2 \oplus N_3))$  because an attacker does not know the values of  $ID_i, y_i, N_1, N_2$  and  $N_3$ . Therefore, the proposed protocol is secure against man-in-the-middle attack.

**14. Message modification or insertion attack:** In this type of attack, the attacker modifies or inserts some messages on the communication channel with the hope of discovering the user's password or gaining unauthorized access. Modifying or inserting messages in the proposed protocol can only cause authentication between the client and the server to fail but cannot allow the attacker to gain any information about the user  $U_i$ 's identity  $ID_i$  and password  $P_i$  or gain unauthorized access. Therefore, the proposed protocol is secure against message modification or insertion attack.

#### 7.3.4 Cost and functionality analysis

An efficient authentication protocol must take communication and computation cost into consideration during user's authentication. The cost comparison of the proposed protocol with the related smart card based authentication protocols is summarized in Table 7.6. Assume that the identity  $ID_i$ , password  $P_i$ ,  $x, y_i$ , nonce values  $(N_1, N_2, N_3)$  are all 128-bit long, prime modular operation is 1024-bits long as in most of practical implementations

and  $t$  is the number of servers. Moreover, we assume that the output of secure one-way hash function and the block size of secure symmetric cryptosystem are 128-bit. Let  $T_H$ ,  $T_S$  and  $T_E$  are defined as the time complexity for hash function, symmetric encryption/decryption and exponential operation respectively. Typically, time complexity associated with these operations can be roughly expressed as  $T_S \gg T_E > T_H$ . In the proposed protocol, the parameters stored in the smart card are  $D_i, E_i, F_i, G_i$  and the memory needed (E1) in the smart card is 512 (=  $4 \cdot 128$ ) bits. The communication cost of authentication (E2) includes the number of communication parameters involved in the authentication protocol. The number of communication parameters are  $\{SID_m, Z_i, CID_i, M_i, R_i, K_i, X_i, V_i, T_i\}$  and hence the communication cost of authentication (E2) is 1152 (=  $9 \cdot 128$ ) bits. The computation cost of registration (E3) is the total time of all operations executed by the user  $U_i$  in the registration phase. The computation cost of registration (E3) is  $5T_H$ . The computation cost of the user (E4) is the time spent by the user during the process of authentication. Therefore, the computation cost of the user (E4) is  $11T_H$ . The computation cost of the service provider server and the control server (E5) is the time spent by the service provider server and the control server during the process of authentication. Therefore, the computation cost of the service provider server and the control server (E5) is  $14T_H$ .

**Table 7.6**

**Cost comparison among related smart card based multi-server authentication schemes**

	<b>Proposed Protocol</b>	<b>Hsiang &amp; Shih [49]</b>	<b>Liao &amp; Wang [83]</b>	<b>Chang &amp; Lee [21]</b>	<b>Juang [61]</b>	<b>Lin et al. [87]</b>
E1	512 bits ( $0.5  n $ )	640 bits ( $0.625  n $ )	512 bits ( $0.5  n $ )	256 bits ( $0.25  n $ )	256 bits ( $0.25  n $ )	$(4t + 1)  n $ bits
E2	$9 \cdot 128$ bits ( $1.125  n $ )	$14 \cdot 128$ bits ( $1.75  n $ )	$7 \cdot 128$ bits ( $0.875  n $ )	$5 \cdot 128$ bits ( $0.625  n $ )	$9 \cdot 128$ bits ( $1.125  n $ )	$7 \cdot 1024$ bits ( $7  n $ )
E3	$5T_H$	$6T_H$	$5T_H$	$2T_H$	$T_H$	$5tT_E$
E4	$11T_H$	$10T_H$	$9T_H$	$4T_H + 3T_S$	$3T_H + 3T_S$	$2T_E$
E5	$14T_H$	$13T_H$	$6T_H$	$4T_H + 3T_S$	$4T_H + 8T_S$	$7T_E$

The proposed protocol uses the control server CS and the service provider server  $S_m$  for the user's authentication and still having less computation costs (E1, E2, E3) and nearly the same computation costs for (E4, E5) as compared to Hsiang and Shih's protocol as shown in Table 7.6. Moreover, the proposed protocol maintains the user's anonymity by

generating dynamic identity and free from different attacks. The proposed protocol requires very less computation as compared to other related protocols [21][61][87] and also highly secure as compared to these related protocols. The functionality comparison of the proposed protocol with the related smart card based authentication protocols is summarized in Table 7.7.

**Table 7.7**

**Functionality comparison among related smart card based multi-server authentication schemes**

	<b>Proposed Protocol</b>	<b>Hsiang &amp; Shih [49]</b>	<b>Liao &amp; Wang [83]</b>	<b>Chang &amp; Lee [21]</b>	<b>Juang [61]</b>	<b>Lin et al. [87]</b>
User's Anonymity	Yes	Yes	Yes	No	No	No
Computation Cost	Low	Low	Low	Low	Low	High
Single Registration	Yes	Yes	Yes	Yes	Yes	No
Session Key Agreement	Yes	Yes	Yes	Yes	Yes	No
Correct Password Update	Yes	No	Yes	No	No	No
No Time Synchronization	Yes	Yes	Yes	Yes	Yes	No
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	No
Multi Factor Security	Yes	Yes	Yes	No	No	No
Replay Attack	No	Yes	Yes	Yes	Yes	Yes
Impersonation Attack	No	Yes	Yes	Yes	Yes	Yes
Stolen Smart Card Attack	No	Yes	Yes	Yes	Yes	Yes

## 7.4 CONCLUSION

In this chapter, we presented cryptanalysis of Liao & Wang’s protocol and Hsiang & Shih’s protocol by showing that their protocols are vulnerable to different attacks. The improvements to these protocols are proposed. Security analysis proved that the proposed protocols are more secure and practical.



## CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

---

### 8.1 CONTRIBUTIONS OF THE THESIS

Instances of password theft are growing significantly mainly due to phishing and dictionary attacks. This is sufficient to shake the customer's confidence in e-commerce. Therefore, corporate network and e-commerce applications require secure and practical remote user authentication solutions. In this study, we analyzed various currently available password based authentication schemes over insecure communication channels. Most of these schemes do not fulfill security requirements and can not resist in different attack scenarios. In this thesis, we have proposed an anti-phishing protocol, two cookies based virtual password authentication protocols and a SSO based two-server authentication protocol. Improvements to several static and dynamic identity based authentication protocols have also been suggested. Finally, two dynamic identity based authentication protocols for multi-server architecture are proposed.

It is important to detect the phishing sites early because most of them are short-lived and cause the damage in the short time span between appearing online and vanishing. Instances of phishing attacks are rapidly growing in number. Phishing is doing direct damage to the financial industry and is also affecting the expansion of e-commerce. Confidence of clients in e-commerce and other online transactions can be enhanced by negating phishing attacks. We presented a cryptanalysis of Gouda et al.'s protocol and showed that their protocol is susceptible to offline dictionary attack, denial of service attack and man-in-the-middle attack in the presence of an active attacker. We suggested an improvement to Gouda et al.'s protocol that can resist offline dictionary attack and denial of service attack. It is however found that Gouda et al.'s protocol is not repairable for man-in-the-middle attack. Therefore, we proposed a new single password based anti-phishing protocol that resolves aforementioned problems and is secure against different types of attacks. In this protocol, the client can use a single password for different online accounts and that password can not be detected by any of the malicious servers or the

attacker. The client machine's browser generates a dynamic identity and a dynamic password for each new login request to the server. The dynamic identity and dynamic password generated for a client are different in different sessions of the Secure Socket Layer (SSL) protocol. Naive users either find it difficult to understand or ignore web browser security indicators but this protocol is equally secure for security ignorant users, who are not very conversant with the browser's security indicators. The presented protocol is a step towards bridging the gap between skilled and unskilled users in online transactions. This protocol will be helpful to the naive users in detecting the phishing website quickly. This protocol can be easily integrated into different types of services such as banking and enterprise applications.

Password based authentication protocols are susceptible to dictionary attacks by means of automated programs because most of the user chosen passwords are limited to the user's personal domain. The online dictionary attacks are one of the major concerns in password based authentication protocols. The efficient and effective use of cookies helps in designing secure authentication protocols. We proposed a cookie based and an inverse cookie based virtual password authentication protocols. In cookie based virtual password authentication protocol, the web server stores a cookie on the user's computer if the user has successfully authenticated himself to the web server from that computer. On the other hand, in an inverse cookie based virtual password authentication protocol, the web server stores cookie on the user's computer when he has not submitted correct identity and password for his authentication to the web server. In both these protocols, the computational effort required from the attacker during login on to the web server increases exponentially with each login failure. The concept of trust has been used so that the legitimate client can easily authenticate himself to the web server from any computer irrespective of whether that computer contains cookie or not. The client generated virtual password for a user is different in each new session of SSL protocol. These concepts combine traditional password authentication with a challenge that is easy to answer by a legitimate client but the computation cost of authentication for an attacker increases with each login failure. Therefore, even automated programs can not launch online dictionary attacks on these proposed protocols. These protocols remove some of the deficiencies of previously suggested password based authentication protocols and are shown to be secure against different types of attacks that can be launched by the attacker.

Single Sign On (SSO) authentication is time efficient because it allows the user to enter his identity and password once within specific time period to login on to multiple hosts and applications within an organization. Most of the password based authentication protocols rely on a single authentication server for user's authentication. The user's password verification information stored on the single server is a main point of susceptibility and remains an attractive target for the attacker. We presented SSO password based two-server authentication protocol that issues a ticket to the user for a specific time period. This time-bound ticket mechanism allows the legitimate user to login with less computational efforts in succeeding login attempts after getting the valid ticket from the authentication and the control server. The user can use this ticket to generate dynamic ticket information to login on to the authentication server. Ticket issued for one authentication server can be used for user authentication on another authentication server that is under the control of the same control server. Our protocol uses two-server paradigm by imposing different levels of trust upon the two servers so that password verification information is distributed between two servers (an authentication server and a control server). Therefore, the proposed protocol is more resistant to dictionary attacks as compared to other existing single-server password based authentication protocols. The proposed protocol is efficient and practical for its implementation because it does not use public key that causes computation and communication burden in a resource constrained environment. Therefore, this architecture increases the overall security of the system and resiliency to dictionary attack.

Smart card based password authentication provides inherent confidentiality, portability and intelligent computing capability. A brief review of some static identity based smart card authentication protocols is presented. Cryptanalysis of these protocols is carried out for different types of attacks and improved protocols are proposed. The comparison of the cost and functionality of the proposed improved protocols with the other related protocols is also done.

We presented a cryptanalysis of Yoon et al.'s scheme [170] by showing that their scheme is susceptible to stolen smart card attack, impersonation attack, parallel session attack and man-in-the-middle attack. An improvement to Yoon et al.'s scheme [170] is proposed that inherits the merits of Yoon et al.'s scheme and enhances the security of their scheme.

In 2005, Yoon and Yoo [171] proposed a remote user authentication scheme that provides mutual authentication, secret key forward secrecy and fast detection of wrong password. In 2009, Kim and Chung [65] found that Yoon and Yoo's scheme [171] easily reveals a user's password and is susceptible to masquerading user attack, masquerading server attack and stolen verifier attack. Then, Kim and Chung [65] proposed a new remote user authentication scheme and claimed that the proposed scheme resolves all aforementioned security flaws, while keeping the merits of Yoon and Yoo's scheme [171]. However, we found that Kim and Chung's scheme [65] is susceptible to masquerading user attack, masquerading server attack, offline dictionary attack and parallel session attack. We described a modified smart card based remote user authentication protocol which resolves the above security flaws of Kim and Chung's [65] scheme, while keeping the merits of Kim and Chung's scheme [65]. The proposed protocols are simple and fast if the user possesses a valid smart card, a valid identity and correct password for its authentication. The proposed protocols are practical and efficient because only one way hash functions and XOR operations are used in its implementation.

We presented a cryptanalysis of Xu et al.'s scheme [157] and showed that their scheme is susceptible to forgery attack in insecure communication channel. An improvement to Xu et al.'s scheme is proposed that inherits the merits of Xu et al.'s scheme and resists different possible attacks. The proposed protocol not only defends forgery attack but also has less computation costs as compared to the related schemes. The security of proposed protocol depends upon the discrete logarithm problem and one way hash function.

Then, we presented a cryptanalysis of Liu et al.'s scheme [91] and showed that their scheme is vulnerable to stolen smart card attack. Also Sun et al. [140] demonstrated man-in-the-middle attack on Liu et al.'s scheme. An improved protocol is proposed that inherits the merits of Shen et al. [128] and Liu et al.'s [91] schemes and resists different possible attacks. The proposed protocol allows the user to choose and change the password at their choice and provides mutual authentication between the user and the server to protect it from forgery attack. It withstands the password guessing attack even if the attacker obtains the smart card of the user. The security of proposed protocol depends upon the discrete logarithm problem and one way hash function. Security analysis proved that the proposed protocols are more secure and practical.

Next, this thesis investigates some smart card authentication protocols for different attack scenarios and improved dynamic identity based smart card authentication protocols are proposed. We presented a cryptanalysis of Liao et al.'s scheme [81] and showed that their scheme is susceptible to malicious user attack, impersonation attack, stolen smart card attack and offline password guessing attack. Liao et al.'s scheme [81] also does not maintain the user's anonymity and its password change phase is insecure. We proposed a modified dynamic identity based smart card authentication protocol which resolves the above security flaws of Liao et al.'s [81] scheme, while keeping the merits of different dynamic identity based authentication schemes.

Then, we presented a cryptanalysis of Liou et al.'s scheme [88] and showed that their scheme is vulnerable to impersonation attack, malicious user attack, offline password guessing attack and man-in-the-middle attack. A secure dynamic identity based authentication scheme using smart cards is proposed to resolve the aforementioned problems, while keeping the merits of different dynamic identity based authentication schemes.

Afterwards, we presented a cryptanalysis of Wang et al.'s scheme [151] and showed that their scheme is vulnerable to impersonation attack, stolen smart card attack, offline password guessing attack, denial of service attack and also fails to preserve the user's anonymity. An improvement to Wang et al.'s scheme is proposed that inherits the merits of different dynamic identity based authentication schemes and resists different possible attacks.

Then, we presented a cryptanalysis of Lee et al.'s scheme [77] and showed that their scheme is susceptible to impersonation attack, malicious user attack and reflection attack. Moreover, Lee et al.'s scheme does not maintain the user's anonymity in communication channel. An improved dynamic identity based authentication protocol is proposed that inherits the merits of Lee et al.'s scheme and resists different possible attacks. The proposed protocol allows the user to choose and change the password at their choice and provides mutual authentication between the user and the server. One of the main features of the proposed protocol is that it does not allow the server to know the password of the user even during the registration phase.

Then, we presented a cryptanalysis of Hsiang and Shih's scheme [48] and showed that their scheme is vulnerable to impersonation attack and offline guessing attack. Their

scheme delays the checking of legitimacy of the user to authentication phase and fails to preserve the user anonymity. An enhancement to Hsiang and Shih's scheme is proposed that inherits the merits of Hsiang and Shih's scheme and resists different possible attacks. The proposed dynamic identity based authentication protocol is simple, fast and efficient because only one-way hash functions and XOR operations are used in its implementation. Security analysis proved that the improved proposed protocol is more secure and practical.

User's anonymity is an important issue in e-commerce applications. Dynamic identity based authentication protocols aim to provide privacy to the user's identity so that the users are anonymous in communication channels. Researchers have proposed different multi-server authentication protocols to eliminate main point of susceptibility of the single-server systems. In e-commerce, the number of servers providing the services to the user is usually more than one and hence secure authentication protocols for multi-server environment are required. Moreover, the multi-server architecture based authentication protocols make it difficult for the attacker to find out any significant authentication information related to the legitimate users. In 2009, Liao and Wang [83] proposed a dynamic identity based remote user authentication protocol for multi-server environment. We presented a cryptanalysis of Liao and Wang protocol [83] and showed that their protocol is susceptible to malicious server attack and malicious user attack. We proposed an improved dynamic identity based authentication protocol for multi-server architecture using smart card that resolves the aforementioned security flaws, while keeping the merits of Liao and Wang's protocol. In 2009, Hsiang and Shih [49] improved Liao and Wang's [83] dynamic identity based remote user authentication protocol for multi-server environment. However, we showed that Hsiang and Shih's protocol is susceptible to replay attack, impersonation attack and stolen smart card attack. Moreover, the password change phase of Hsiang and Shih's protocol is incorrect. We presented a dynamic identity based authentication protocol for multi-server architecture using smart card that resolves the aforementioned flaws, while keeping the merits of Hsiang and Shih's protocol. Both the proposed protocols use two-server paradigm by imposing different levels of trust upon the two servers and the user's authentication functionality is distributed between these two servers known as the service provider server and the control server. The proposed protocols help the service provider servers and the control server to authenticate the user

completely by computing their static identity and at the same time keeps the identity of the user dynamic in communication channel. The proposed protocols are simple and fast if the user possesses valid smart card, valid identity and correct password for authentication. The proposed protocols are practical and efficient because only one-way hash functions and XOR operations are used in its implementation. Security analysis proved that the both proposed protocols are more secure and practical.

Finally to sum up, in our proposed single password based anti-phishing protocol, client can use a single password for different online accounts and that password can not be detected by any of the malicious servers or the attacker. This protocol is equally secure for security ignorant users, who are not very conversant with the browser's security indicators. The protocol does not allow the server to know the client's password at any time. The proposed cookies based and an inverse cookie based virtual password authentication protocols are very effective to thwart online dictionary attacks because the computation cost of login on to the web server increases exponentially with each login failure for an attacker. The legitimate client can easily authenticate himself to the web server from any computer irrespective of whether that computer contains cookie or not. Most of the existing SSO password based authentication protocols are designed for single-server environment. We proposed an efficient SSO password based two-server architecture in which the user has to login once to get a valid ticket. Smart card based password authentication is one of the most convenient ways to provide multi-factor authentication by acquiring the smart card and knowing the identity and password for the communication between a client and a server. Improvements to several static and dynamic identity based authentication protocols have also been suggested. User's privacy is an important issue in e-commerce applications. The proposed dynamic identity based authentication protocols aim to provide the privacy to the user's identity so that users are anonymous in communication channel. Also the concept of two-tier authentication for the client makes it difficult for an attacker to guess out the information pertaining to password and ticket. Confidence of clients in e-commerce and other online transactions can be enhanced by negating phishing, dictionary and other possible attacks. The work presented in this thesis is a step toward making e-commerce transactions more reliable and secure.

## 8.2 RECOMMENDATIONS AND FUTURE SCOPE OF WORK

Research is a continuous process. An end of a research project is in fact a beginning of a lot of other avenues for future work. A door to new research issues is opened upon the end of a research project. Following aspects are identified for future research work in this area:

1. One interesting research area is the effectiveness of blacklist based solutions in mitigating the phishing attacks. The effectiveness of blacklist based solution depends upon how much regularly the database of blacklisted sites is updated. Moreover, what techniques are used to find out the phishing sites. The effective page analysis techniques are required that can distinguish phishing sites from legitimate sites.
2. In future, more computation and communication efficient password authentication schemes should be developed which can resist different attacks in a better way.
3. Elliptical curve cryptography is a current area of research especially for power constraint and integrated circuit space limited devices (like smart card) that results in secure system even with smaller key size.
4. Biometric authentication is a very secure method for user authentication. Researchers are putting efforts in developing computation and communication efficient biometric authentication schemes.
5. Quantum cryptography is one of the future research areas which assert that shared secret can be established over public communication channels in such a way that the total information of an eavesdropper can be made arbitrarily small with probability arbitrarily close to 1.
6. A lot of two and three party key exchange protocols are proposed by researchers but most of them suffer from different types of attacks. Threatening attack on two and three party key exchange protocols is undetectable online dictionary attack. The computation and communication efficiency is a crucial criteria for designing and evaluating the efficiency of such new proposed schemes.
7. Researchers have developed different group key management protocols based on methods like chain tree, dual encryption protocols, group Diffie-Hellman key exchange and logical tree hierarchy. They have suggested a lot of solutions to handle dynamic leave and join of members, still rekeying a group is complex problem. Researchers are putting efforts in developing better techniques to handle dynamic leave and join of members. Group key management protocols should provide forward secrecy, backward secrecy and should achieve collusion freedom.



## REFERENCES

---

- [1] **Abadi M., Mark T., Lomas A. and Needham R.**, "Strengthening Passwords," Technical Report-033, September 1997.
- [2] **Abdalla M. and Pointcheval D.**, "Simple Password Based Encrypted Key Exchange Protocols," Springer-Verlag, LNCS, vol. 3376, pp. 191-208, February 2005.
- [3] **Adelsbach A., Gajek S. and Schwenk J.**, "Visual Spoofing of SSL Protected Web Sites and Effective Countermeasures," Information Security Practice and Experience, Springer-Verlag, LNCS, vol. 3469, pp. 204-216, September 2005.
- [4] **Adida B.**, "BeamAuth: Two-Factor Web Authentication With a Bookmark," Proc. of 14<sup>th</sup> ACM Conference on Computer and Communications Security, Alexandria, USA, pp. 48-57, October 2007.
- [5] **AlZomai M., AlFayyadh B., Josang A. and McCullagh A.**, "An Experimental Investigation of the Usability of Transaction Authorization in Online Bank Security Systems," Proc. of 6<sup>th</sup> ACS Australasian Information Security Conference (AISC2008), Wollongong, Australia, pp. 65-73, January 2008.
- [6] **Andreys G., Jain A. and Sivakumar G.**, "Intelligent Real-Time Reactive Network Management," Proc. of the 1<sup>st</sup> European Conference on Computer Network Defence, Glamorgan, Wales, UK, EC2ND, Springer-Verlag, ISBN 1846283116, 2006.
- [7] **Anti-Phishing Working Group**, "<http://www.antiphishing.org/>.", Accessed: March 3, 2009.
- [8] **Awasthi A. K.**, "Comment on a Dynamic ID-Based Remote User Authentication Scheme," Transaction on Cryptology, vol. 1, no. 2, pp. 15-16, May 2004.
- [9] **Bank of America SiteKey**, "<http://www.bankofamerica.com/privacy/sitekey/>", Accessed: May 2, 2009.
- [10] **Bellare M. and Rogaway P.**, "Provably Secure Session Key Distribution-The Three Party Case," Proc. of the 27<sup>th</sup> Annual Symposium on the Theory of Computing, ACM, pp. 57-66, 1995.

- [11] **Bellare M., Pointcheval D. and Rogaway P.**, “Authenticated Key Exchange Secure Against Dictionary Attacks,” *Advances in Cryptology, EUROCRYPT 2000*, Springer-Verlag, LNCS, vol. 1807, pp. 140-155, 2000.
- [12] **Bellovin S.M. and Merrit M.**, “Encrypted Key Exchange: Password Based Protocols Secure Against Dictionary Attacks,” *IEEE Computer Society Symposium on Research in Security and Privacy, California, USA*, pp. 72-84, May 1992.
- [13] **Bhattacharyya D.K. and Nandi S.**, “CA Based Password-Only Authenticated Key Exchange,” *IEEE Workshop on Signal Processing Systems*, pp. 820-827, 2000.
- [14] **Bird R., Gopal I., Herzberg A., Janson P., Kuttan S., Molva R. and Yung M.**, “The KryptoKnight Family of Light Weight Protocols for Authentication and Key Distribution,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 1, pp. 31-41, February 1995.
- [15] **Blundo C., Cimato S. and Prisco R.D.**, “A Lightweight Approach to Authenticated Web Caching,” *Proc. of IEEE International Symposium on Applications and the Internet (SAINT 2005)*, pp. 157-163, February 2005.
- [16] **Brainard J., Juels A., Kaliski B. and Szydlo M.**, “A New Two-Server Approach for Authentication With Short Secrets,” *Proc. of 12<sup>th</sup> USENIX Security Symposium, Washington D.C., USA*, pp. 201-214, August 2003.
- [17] **Chakrabarti S. and Singhal M.**, “Password based Authentication: Preventing Dictionary Attacks,” *IEEE Computer Society*, vol. 40, no. 6, pp. 68-74, June 2007.
- [18] **Chakraborty D. and Sarkar P.**, “A General Construction of Tweakable Block Ciphers and Different Modes of Operations,” *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 1991-2006, 2008.
- [19] **Chan C.K. and Cheng L.M.**, “Cryptanalysis of Timestamp-Based Password Authentication Scheme,” *Computers & Security*, vol. 21, no. 1, pp. 74-76, 2002.
- [20] **Chang C.C. and Chang Y.F.**, “A Novel Three Party Encrypted Key Exchange Protocol,” *Computer Standards & Interfaces*, vol. 26, no. 5, pp. 472-476, January 2004.

- [21] **Chang C.C. and Lee J.S.**, "An Efficient and Secure Multi-Server Password Authentication Scheme Using Smart Cards," Proc. of International Conference on Cyber Worlds, Taiwan, pp. 417-422, November 2004.
- [22] **Chatterjee S. and Sarkar P.**, "Identity-Based Encryption and Hierarchical Identity-Based Encryption," In Identity-Based Cryptography, An Edited Volume of IOS Press Cryptology and Information Security Series, 2009.
- [23] **Chen C.M. and Ku W.C.**, "Stolen Verifier Attack on Two New Strong Password Authentication Protocols," IEICE Transactions on Communications, vol. E85-B, pp. 2519-2521, November 2002.
- [24] **Chen T.H., Lee W.B. and Chen H.B.**, "A Round and Computation Efficient Three Party Authenticated Key Exchange Protocol," Journal of Systems and Software, vol. 81, no. 9, pp. 1581-1590, September 2008.
- [25] **Chien H.Y., Jan J.K. and Tseng Y.M.**, "An Efficient and Practical Solution to Remote Authentication: Smart Card," Computers & Security, vol. 21, no. 4, pp. 372-375, November 2002.
- [26] **Chien H.Y. and Chen C.H.**, "A Remote Authentication Scheme Preserving User Anonymity," Proc. of Advanced Information Networking and Applications, vol. 2, pp. 245-248, March 2005.
- [27] **Chung H.R. and Ku W.C.**, "Three Weaknesses in a Simple Three Party Key Exchange Protocol," Information Sciences, vol. 178, no. 1, pp. 220-229, January 2008.
- [28] **Das M.L., Saxena A. and Gulati V.P.**, "A Dynamic ID-Based Remote User Authentication Scheme," IEEE Transactions on Consumer Electronics, vol. 50, no. 2, pp. 629-631, May 2004.
- [29] **Dhamija R. and Tygar J.D.**, "The Battle Against Phishing: Dynamic Security Skins," Symposium on Usable Privacy and Security (SOUPS), pp. 77-88, May 2005.
- [30] **Ding Y. and Horster P.**, "Undetectable Online Password Guessing Attacks," ACM Operating Systems Review, vol. 29, no. 4, pp. 77-86, October 1995.

- [31] **eBay Toolbar**, "[http://pages.ebay.com/ebay\\_toolbar](http://pages.ebay.com/ebay_toolbar)", Accessed: January 2, 2009.
- [32] **Fan L., Li J.H. and Zhu H.W.**, "An Enhancement of Timestamp-Based Password Authentication Scheme," *Computers & Security*, vol. 21, no. 7, pp. 665-667, 2002.
- [33] **Ford W. and Kaliski B.S.**, "Server-Assisted Generation of a Strong Secret from a Password," *Proc. of IEEE 9<sup>th</sup> International Workshop Enabling Technologies*, pp. 176-180, June 2000.
- [34] **Freier A.O., Karlton P. and Kocher P.C.**, "SSL Protocol Version 3.0 Internet Draft," IETF, November 1996.
- [35] **Fu K., Sit E., Smith K. and Feamster N.**, "Dos and Don'ts of Client Authentication on the Web," *Proc. of 10<sup>th</sup> USENIX Security Symposium*, pp. 1-16, August 2001.
- [36] **Gaurav A., Sharma A., Gelara V. and Moona R.**, "Using Personal Electronic Device For Authentication-Based Service Access," *Proc. of IEEE International Conference on Communications (ICC2008)*, Beijing, May 2008.
- [37] **Gaw S. and Felten E.W.**, "Password Management Strategies For Online Accounts," *Symposium on Usable Privacy and Security (SOUPS) 2006*, USA, pp. 44-55, July 2006.
- [38] **Gong L., Lomas M., Needham R. and Saltzer J.**, "Protecting Poorly Chosen Secrets From Guessing Attacks," *IEEE Journal of Selected Areas Communication*, vol. 11, no. 5, pp. 648-656, June 1993.
- [39] **Google Safe Browsing**, "[http://www.google.com/tools/firefox/safebrowsing/.](http://www.google.com/tools/firefox/safebrowsing/)", Accessed: March 5, 2009.
- [40] **Goyal V., Kumar V., Singh M., Abraham A. and Sanyal S.**, "A New Protocol to Counter Online Dictionary Attacks," *Computers & Security*, vol. 25, no. 2, pp. 114-120, March 2006.
- [41] **Goyal V., Gupta S.K. and Gupta A.**, "Malafide Intension and Its Mapping to Privacy Policy Purposes for Masquerading," *Proc. of the 10<sup>th</sup> IEEE Computer Society International Database Engineering & Application Symposium (IDEAS'06)*, December 2006.

- [42] **Gouda M.G., Liu A.X., Leung L.M. and Alam M.A.**, "SPP: An Anti-Phishing Single Password Protocol," *Computer Networks*, vol. 51, no. 13, pp. 3715-3726, April 2007.
- [43] **Guan Z.**, "Invited Talk," International ICT Security Exhibition and Conference, Guangzhou, China, November, 2007.
- [44] **Guo H., Li Z., Mu Y. and Zhang X.**, "Cryptanalysis of Simple Three Party Key Exchange Protocol," *Computers & Security*, vol. 27, no. 2, pp. 16-21, May 2008.
- [45] **Halderman J.A., Waters B. and Felten E.W.**, "A Convenient Method for Securely Managing Passwords," *Proc. of 14<sup>th</sup> ACM International World Wide Web Conference*, Chiba, Japan, pp. 471-479, May 2005.
- [46] **Haller N.**, "The S/KEY One Time Password System," RFC 1760, February 1995.
- [47] **Herzberg A. and Gbara A.**, "TrustBar: Protecting (Even) Naive Users from Spoofing and Phishing Attacks," *Cryptology e-print Archive*, Report 2004/155, February 2004.
- [48] **Hsiang H.C. and Shih W.K.**, "Weaknesses and Improvements of the Yoon-Ryu-Yoo Remote User Authentication Scheme Using Smart Cards," *Computer Communications*, vol. 32, no. 4, pp. 649-652, March 2009.
- [49] **Hsiang H.C. and Shih W.K.**, "Improvement of the Secure Dynamic ID Based Remote User Authentication Scheme for Multi-Server Environment," *Computer Standards & Interface*, vol. 31, no. 6, pp. 1118-1123, November 2009.
- [50] **Hsu C.L.**, "Security of Chien et al.'s Remote User Authentication Scheme Using Smart Cards," *Computer Standards & Interfaces*, vol. 26, pp. 167-169, March 2004.
- [51] **Hu L., Niu X. and Yang Y.**, "An Efficient Multi-Server Password Authenticated Key Agreement Scheme Using Smart Cards," *Proc. of International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pp. 903-907, April 2007.
- [52] **Hwang M.S. and Li L.H.**, "A New Remote User Authentication Scheme Using Smart Cards," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 28-30, March 2000.

- [53] **Hwang M.S., Lee C.C. and Tang Y.L.**, "A Simple Remote User Authentication," *Mathematical and Computer Modelling*, vol. 36, pp. 103-107, October 2002.
- [54] **Imamoto K. and Sakurai K.**, "A Design of Diffie-Hellman Based Key Exchange Using One-Time ID in Pre-Shared Key Model," 18<sup>th</sup> International Conference on Advanced Information Networking and Applications (AINA 2004), March 2004.
- [55] **Imamoto K. and Sakurai K.**, "Design and Analysis of Diffie-Hellman Based Key Exchange Using One-Time ID by SVO Logic," *Electronic Notes in Theoretical Computer Science*, vol. 135, pp. 79-94, June 2005.
- [56] **Izumi A., Ueshige Y. and Sakurai K.**, "A Proposal of Key Management Scheme and Its Operation Using Anonymous Biometrics on ID-based Infrastructure," *International Journal of Security and Its Applications*, vol. 1, no. 1, pp. 83-94, July 2007.
- [57] **Jablon D.P.**, "Password Authentication Using Multiple Servers," *Proc. of RSA Security Conference*, pp. 344 -360, April 2001.
- [58] **Jan J.K. and Chen Y.Y.**, "'Paramita Wisdom' Password Authentication Scheme Without Verification Tables," *The Journal of Systems and Software*, vol. 42, no. 1, pp. 45-57, July 1998.
- [59] **Jaung W.S.**, "Efficient Three Party Key Exchange Using Smart Cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 619-624, May 2004.
- [60] **Johns M.**, "Using Java in Anti DNS-Pinning Attacks," <http://shampoo.antville.org/stories/1566124>, February 2007.
- [61] **Juang W.S.**, "Efficient Multi-Server Password Authenticated Key Agreement Using Smart Cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 1, pp. 251-255, November 2004.
- [62] **Juels A., Jakobsson M. and Jagatic T.N.**, "Cache Cookies For Browser Authentication," *IEEE Symposium on Security and Privacy*, pp. 301-305, May 2006.
- [63] **Karlof C., Shankar U., Tygar J.D. and Wagner D.**, "Dynamic Pharming Attacks and the Locked Same Origin Policies For Web Browsers," *Proc. of ACM Conference on Computer and Communications Security*, pp. 58-71, November 2007.

- [64] **Kelsey J., Schneier B., Hall C. and Wagner D.**, "Secure Applications of Low Entropy Keys," Springer-Verlag, LNCS, vol. 1396, pp. 121-134, September 1998.
- [65] **Kim S.K. and Chung M.G.**, "More Secure Remote User Authentication Scheme," Computer Communications, vol. 32, no. 6, pp. 1018-1021, April 2009.
- [66] **Kirda E. and Kruegel C.**, "Protecting Users Against Phishing Attacks," Computer Journal, vol. 49, no. 5, pp. 554-561, January 2006.
- [67] **Kocher P., Jaffe J. and Jun B.**, "Differential Power Analysis," Proc. of CRYPTO 99, Springer-Verlag, LNCS, vol. 1666, pp. 388-397, August 1999.
- [68] **Kormann D.P. and Rubin A.D.**, "Risks of the Passport Single Sign On Protocol," Computer Networks, vol. 33, pp. 51-58, June 2000.
- [69] **Ku W.C. and Chen S.M.**, "Weaknesses and Improvements of an Efficient Password Based Remote User Authentication Scheme Using Smart Cards," IEEE Transactions on Consumer Electronics, vol. 50, no. 1, pp. 204-207, February 2004.
- [70] **Ku W.C. and Chang S.T.**, "Impersonation Attack on a Dynamic ID-Based Remote User Authentication Scheme Using Smart Cards," IEICE Transactions on Communications, vol. E88-B, no. 5, pp. 2165-2167, May 2005.
- [71] **Lamberger M., Mendel F., Rechberger C., Rijmen V. and Schlaffer M.**, "Rebound Distinguishers: Results on the Full Whirlpool Compression Function," Advances in Cryptology, ASIACRYPT 2009, pp. 1-18, 2009.
- [72] **Lamport L.**, "Password Authentication With Insecure Communication," Communications of the ACM, vol. 24, no. 11, pp. 770-772, November 1981.
- [73] **Lee W.B. and Chang C.C.**, "User Identification and Key Distribution Maintaining Anonymity for Distributed Computer Network," Computer System Science, vol. 15, no. 4, pp. 211-214, July 2000.
- [74] **Lee S., Kim H. and Yoo K.**, "Improved Efficient Remote User Authentication Scheme Using Smart Cards," IEEE Transactions on Consumer Electronics, vol. 50, no. 2, pp. 565-567, May 2004.
- [75] **Lee T.F., Hwang T. and Lin C.L.**, "Enhanced Three Party Encrypted Key Exchange without Server Public Keys," Computers & Security, vol. 23, no. 7, pp. 571-577, October 2004.

- [76] **Lee N.Y. and Chiu Y.C.**, "Improved Remote Authentication Scheme With Smart Card," *Computer Standards & Interfaces*, vol. 27, no. 2, pp. 177-180, January 2005.
- [77] **Lee S.W., Kim H.S. and Yoo K.Y.**, "Improvement of Chien et al.'s Remote User Authentication Scheme Using Smart Cards," *Computer Standards & Interfaces*, vol. 27, no. 2, pp. 181-183, January 2005.
- [78] **Lee S. and Sivalingam K.M.**, "An Efficient One-Time Password Authentication Scheme Using a Smart Card," *International Journal of Security and Networks*, Inderscience, vol. 4, no. 3, pp. 145-152, 2009.
- [79] **Lei M., Xiao Y., Vrbsky S.V. and Li C.C.**, "Virtual Password Using Random Linear Functions For On-Line Services, ATM Machines, and Pervasive Computing," *Computer Communications*, vol. 31, no. 18, pp. 4367-4375, December 2008.
- [80] **Li L.H., Lin I.C. and Hwang M.S.**, "A Remote Password Authentication Scheme for Multi-Server Architecture Using Neural Networks," *IEEE Transactions on Neural Network*, vol. 12, no. 6, pp. 1498-1504, November 2001.
- [81] **Liao I.E., Lee C.C. and Hwang M.S.**, "Security Enhancement For a Dynamic ID-Based Remote User Authentication Scheme," *Proc. of Conference on Next Generation Web Services Practice*, pp. 437-440, July 2005.
- [82] **Liao I.E., Lee C.C. and Hwang M.S.**, "A Password Authentication Scheme Over Insecure Networks," *Journal of Computer and System Sciences*, vol. 72, no. 4, pp. 727-740, June 2006.
- [83] **Liao Y.P. and Wang S.S.**, "A Secure Dynamic ID Based Remote User Authentication Scheme For Multi-Server Environment," *Computer Standards & Interface*, vol. 31, no. 1, pp. 24-29, January 2009.
- [84] **Lin C.L., Sun H.M. and Hwang T.**, "Three Party Encrypted Key Exchange: Attacks and a Solution," *ACM Operating Systems Review*, vol. 34, no. 4, pp. 12-20, May 2000.
- [85] **Lin C.L., Sun H.M. and Hwang T.**, "Attacks and Solutions on Strong Password Authentication," *IEICE Transactions on Communications*, vol. E84-B, pp. 2622-2627, September 2001.



- [86] **Lin C.L., Sun H.M., Steiner M. and Hwang T.**, "Three Party Encrypted Key Exchange Without Server Public Keys," *IEEE Communication Letter*, vol. 5, no. 12, pp. 497-499, December 2001.
- [87] **Lin I.C., Hwang M.S. and Li L.H.**, "A New Remote User Authentication Scheme for Multi-Server Architecture," *Future Generation Computer System*, vol. 19, no. 1, pp. 13-22, January 2003.
- [88] **Liou Y.P., Lin J. and Wang S.S.**, "A New Dynamic ID-Based Remote User Authentication Scheme Using Smart Cards," *Proc. of 16<sup>th</sup> Information Security Conference*, Taiwan, pp. 198-205, July 2006.
- [89] **Litan A.**, "Phishing Attack Victims Likely Targets for Identity Theft," *Gartner First Take FT-22-8873*, Gartner Research, May 2004.
- [90] **Liu A.X., Kovacs J.M., Huang C.T. and Gouda M.G.**, "A Secure Cookie Protocol," *Proc of 14<sup>th</sup> IEEE International Conference on Computer Communications and Networks*, Austin, USA, pp. 333-338, October 2005.
- [91] **Liu J.Y., Zhou A.M. and Gao M.X.**, "A New Mutual Authentication Scheme based on Nonce and Smart Cards," *Computer Communications*, vol. 31, no. 10, pp. 2205-2209, 2008.
- [92] **Lu R.X. and Cao Z.F.**, "Simple Three Party Key Exchange Protocol," *Computers & Security*, vol. 26, no. 1, pp. 94-97, February 2007.
- [93] **Ludl C., McAllister S., Kirda E. and Kruegel C.**, "On the Effectiveness of Techniques to Detect Phishing Sites," *Springer-Verlag, LNCS*, vol. 4579, pp. 20-39, May 2007.
- [94] **Mackenzie P., Shrimpton T. and Jakobsson M.**, "Threshold Password-Authenticated Key Exchange," *Proc. of Advances in Cryptology (Eurocrypt '02)*, pp. 385-400, August 2002.
- [95] **Mackenzie P., Shrimpton T. and Jakobsson M.**, "Threshold Password-Authenticated Key Exchange," *Journal of Cryptology*, vol. 19, no. 1, pp. 27-66, January 2006.
- [96] **McAfee SiteAdvisor**, "<http://www.us.mcafee.com/root/catalog.asp?catid=free/>.", Accessed: April 12, 2009.

- [97] **Manber U.**, "A Simple Scheme to Make Passwords Based on One Way Functions Much Harder to Crack," *Computers & Security*, vol. 15, no. 2, pp. 171-176, November 1996.
- [98] **Messerges T.S., Dabbish E.A. and Sloan R.H.**, "Examining Smart-Card Security Under the Threat of Power Analysis Attacks," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 541-552, May 2002.
- [99] **Microsoft Sender ID home page**,  
"http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx/.",  
Accessed: November 2, 2008.
- [100] **Microsoft Passport**, "http://www.passport.net/.", Accessed: December 10, 2008.
- [101] **Mitchell C.J. and Chen L.**, "Comments on the S/KEY User Authentication Scheme," *ACM Operating Systems Review*, vol. 30, no. 4, pp. 12-16, October 1996.
- [102] **Mittal S., Garg R., Mohania M., Gupta S.K. and Iwaihara M.**, "Detecting Frauds in Online Advertising Systems," *EC-Web 2006*, pp. 222-231, 2006.
- [103] **Molva R., Tsudik G., Herreweghen E.V. and Zatti S.**, "KryptoKnight Authentication and Key Distribution System," *European Symposium on Research in Computer Security- ESORICS*, pp. 1-16, April 1992.
- [104] **Mozilla Firefox Phishing Protection**,  
"http://en.www.mozilla.com/en/firefox/phishing-protection/.", Accessed:  
December 3, 2008.
- [105] **Naik P., Ravichandran K. and Sivalingam K.M.**, "Cryptographic Key Exchange Based on Locationing Information," *Pervasive and Mobile Computing*, vol. 3 pp. 15-35, 2007.
- [106] **Netcraft Anti-Phishing Toolbar**, "http://www.toolbar.netcraft.com/.", Accessed:  
December 16, 2008.
- [107] **Neuman B.C. and Ts'o T.**, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications*, vol. 32, no. 9, pp. 33-38, September 1994.
- [108] **Nikova S., Rijmen V. and Martin Schlaffer**, "Using Normal Bases for Compact Hardware Implementations of the AES S-box," *Security and Cryptography for Networks*, Springer-Verlag, LNCS, vol. 5229, pp. 236-246, 2008.

- [109] **Nishi R., Hori Y. and Sakurai K.**, "Key Distribution Scheme Using Matched Filter Resistant Against DoS Attack," 22<sup>nd</sup> International Conference on Advanced Information Networking and Applications - Workshops (AINA 2008), ISBN: 978-0-7695-3096-3, March 2008.
- [110] **Nishi R., Hori Y. and Sakurai K.**, "Reliable Key Distribution Scheme For Lossy Channels," IEICE Transactions on Information and Systems, vol. E91-D, no. 5, pp. 1485-1488, May 2008.
- [111] **Oppliger R., Hauser R. and Basin D.**, "SSL/TLS Session-Aware User Authentication Revisited," Computers & Security, vol. 27, pp. 64-70, June 2008.
- [112] **Park J.S. and Sandhu R.**, "Secure Cookies on the Web," IEEE Internet Computing, vol. 4, no. 4, pp. 36-44, August 2000.
- [113] **Phan R.C.W., Yau W.C. and Goi B.M.**, "Cryptanalysis of Simple Three Party Key Exchange Protocol," Information Sciences, vol. 178, no. 13, pp. 2849-2856, July 2008.
- [114] **Phishing Filter, Microsoft Phishing Filter FAQ**,  
"<https://phishingfilter.microsoft.com/faq.aspx/>.", Accessed: April 4, 2009.
- [115] **Pinkas B. and Sander T.**, "Securing Passwords Against Dictionary Attacks," 9<sup>th</sup> ACM Conference on Computer and Communication Security, USA, pp. 161-170, November 2002.
- [116] **Pramstaller N., Lamberger M. and Rijmen V.**, "Second Preimages for Iterated Hash Functions and Their Implications on MACs," Proc. of the 12<sup>th</sup> Australasian Conference on Information Security and Privacy, ACISP 2007, Springer-Verlag, LNCS, vol. 4586, pp. 68-81, July 2007.
- [117] **Qi F., Li T., Bao F. and Wu Y.**, "Preventing Web-Spoofing With Automatic Detecting Security Indicator," ISPEC, Springer-Verlag, LNCS, vol. 3903, pp. 112-122, April 2006.
- [118] **Raimondo M.D. and Gennaro R.**, "Provably Secure Threshold Password-Authenticated Key Exchange," Proc. of Advances in Cryptology (Eurocrypt '03), pp. 507-523, May 2003.

- [119] **RFC 2617**, "HTTP Authentication: Basic and Digest Access Authentication," June 1999.
- [120] **Ross B., Jackson C., Miyake N., Boneh D. and Mitchell J.C.**, "A Browser Plug-in Solution to the Unique Password Problem," Technical Report, Stanford-SecLab, June 2005.
- [121] **RSA**, "RSA Security: Protecting Against Phishing by Implementing Strong Two-factor Authentication," [https://www.rsasecurity.com/secured/PHISH\\_WP\\_0904.pdf](https://www.rsasecurity.com/secured/PHISH_WP_0904.pdf)," June 2004.
- [122] **Sakai A., Hori Y. and Sakurai K.**, "Formal Verification for Access Control in Web Information Sharing System," Springer-Verlag, LNCS, vol. 5576, pp. 80-89, 2009.
- [123] **Samar V.**, "Single Sign-On Using Cookies for Web Applications," Proc. of 8<sup>th</sup> Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises, IEEE Computer Society, pp. 158-163, June 1999.
- [124] **Sandhu R., Bellare M. and Ganesan R.**, "Password-Enabled PKI: Virtual Smart-Cards Versus Virtual Soft Tokens," Proc. of 1<sup>st</sup> Annual PKI Research Workshop, pp. 89-96, 2002.
- [125] **Sandhu R., Schoppert B., Ganesan R., Bellare M. and Desa C.**, "Authentication Protocol Using a Multi-Factor Asymmetric Key Pair," US Patent 7386720, June 10, 2008.
- [126] **Sandirigama M., Shimizu A. and Noda M.T.**, "Simple and Secure Password Authentication Protocol," IEICE Transactions on Communications, vol. E83-B, pp. 1363-1365, June 2000.
- [127] **Sharma S., Moona R. and Sanghi D.**, "TransCrypt: A Secure and Transparent Encrypting File System For Enterprises", 8<sup>th</sup> International Symposium on Systems and Information Security (SSI 2006), Sao Paulo, Brazil, November, 2006.
- [128] **Shen J.J., Lin C.W. and Hwang M.S.**, "Security Enhancement for the Timestamp-Based Password Authentication Scheme Using Smart Cards," Computers & Security, vol. 22, no. 7, pp. 591-595, 2003.

- [129] **Shih H.C.**, "Cryptanalysis on Two Password Authentication Schemes," Laboratory of Cryptography and Information Security, National Central University, Taiwan, July 2008.
- [130] **Shoup V. and Rubin A.**, "Session Key Distribution Using Smart Cards," EUROCRYPT 96, Springer-Verlag, LNCS, vol. 1070, pp. 321-331, May 1996.
- [131] **Sivakumar G.**, "Cryptographic Protocols and Network Security," Available at <http://www.cse.iitb.ac.in/~siva/talks/crypto.pdf>, Accessed: March 2, 2009.
- [132] **Sivakumar G.**, "Internet Security and Cryptographic Protocols," Available at <http://www.cse.iitb.ac.in/~siva>, Accessed: March 16, 2009.
- [133] **SpoofStick**, "<http://www.spoofstick.com> (2005)", Accessed: April 15, 2009.
- [134] **Stanford, SpoofGuard Home Page**, "[http://crypto.stanford.edu/SpoofGuard/.](http://crypto.stanford.edu/SpoofGuard/)", Accessed: March 11, 2009.
- [135] **Steiner M., Tsudik G. and Waidner M.**, "Refinement and Extension of Encrypted Key Exchange," ACM Operating Systems Review, vol. 29, no. 3, pp. 22-30, July 1995.
- [136] **Stubblebine S.G. and Oorschot P.C.V.**, "Addressing Online Dictionary Attacks With Login Histories and Humans in the Loop," Financial Cryptography, Springer-Verlag, LNCS, vol. 3110, pp. 39-53, January 2004.
- [137] **Suckers for Spam (2005)**, "<http://www.Internetnews.com>", Accessed: May 4, 2009.
- [138] **Sun H.M.**, "An Efficient Remote User Authentication Scheme Using Smart Cards," IEEE Transactions on Consumer Electronics, vol. 46, no. 4, pp. 958-961, October 2000.
- [139] **Sun H.M., Chen B.C. and Hwang T.**, "Secure Key Agreement Protocols For Three-Party Against Guessing Attacks," The Journal of Systems and Software, vol. 75, pp. 63-68, May 2005.
- [140] **Sun D.Z., Huai J.P., Sun J.Z. and Li J.X.**, "Cryptanalysis of a Mutual Authentication Scheme Based on Nonce and Smart Cards," Computer Communications, vol. 32, no. 6, pp. 1015-1017, 2009.

- [141] **Tripathy S. and Nandi S.**, "Efficient Remote User Authentication and Key Establishment For Multi-Server Environment," Proc. of 3<sup>rd</sup> International Conference on Distributed Computing & Internet Technology, Springer-Verlag, LNCS, vol. 4317, pp. 333-346, December 2006.
- [142] **Tripathy S. and Nandi S.**, "Secure User-Identification and Key Distribution Scheme Preserving Anonymity," International Journal of Security and Networks, vol. 3, no. 3, pp. 201-205, 2008.
- [143] **Tsai C.S., Lee C.C. and Hwang M.S.**, "Password Authentication Schemes: Current Status and Key Issues," International Journal of Network Security, vol. 3, no. 2, pp. 101-115, September 2006.
- [144] **Tsai J.L.**, "Efficient Multi-Server Authentication Scheme Based on One-Way Hash Function Without Verification Table," Computers & Security, vol. 27, no. 3-4, pp. 115-121, June 2008.
- [145] **Tsaur W.J., Wu C.C. and Lee W.B.**, "A Smart Card-Based Remote Scheme For Password Authentication in Multi-Server Internet Services," Computer Standard & Interfaces, vol. 27, no. 1, pp. 39-51, November 2004.
- [146] **VeriSign Messaging Security**, "<http://www.verisign.comx>", Accessed: June 23, 2009.
- [147] **Wang P., Kim Y., Kher V. and Kwon T.**, "Strengthening Password Based Authentication Protocols Against Online Dictionary Attacks," ACNS 2005, Springer-Verlag, LNCS, vol. 3531, pp. 17-32, May 2005.
- [148] **Wang W.J. and Hu L.**, "Efficient and Provably Secure Generic Construction of Three Party Password based Authenticated Key Exchange Protocols," INDOCRYPT'06, Springer-Verlag, LNCS, vol. 4329, pp. 118-132, August 2006.
- [149] **Wang Y. and Desmedt Y.**, "Perfectly Secure Message Transmission Revisited," IEEE Transaction on Information Theory, vol. 54, no. 6, pp. 2582-2595, June 2008.
- [150] **Wang Y.**, "Identity Based Authenticated Key Agreement Protocol IDAK," Submitted to IEEE 1363.3 Standards.

- [151] **Wang Y., Liu J., Xiao F. and Dan J.**, "A More Efficient and Secure Dynamic ID-Based Remote User Authentication Scheme," *Computer Communications*, vol. 32, no. 4, pp. 583-585, March 2009.
- [152] **Wang Y.**, "Efficient Identity-Based and Authenticated Key Agreement Protocol," *Cryptology ePrint Archive* 2005/108.
- [153] **Wu T.C.**, "Remote Login Authentication Scheme Based on a Geometric Approach," *Computer Communications*, vol. 18, no. 12, pp. 959-963, December 1995.
- [154] **Wu S.T. and Chieu B.C.**, "A User Friendly Remote Authentication Scheme With Smart Cards," *Computer & Security*, vol. 22, no. 6, pp. 547-550, September 2003.
- [155] **Wu M., Miller R.C. and Garfinkel S.**, "Do Security Toolbars Actually Prevent Phishing Attacks," *Proc. of ACM Computer/Human Interaction (CHI)*, pp. 601-610, April 2006.
- [156] **Xu D., Lu C. and Santos A.D.**, "Protecting Web Usage of Credit Cards Using One-Time Pad Cookie Encryption," *Proc. of 18<sup>th</sup> Annual Computer Security Applications Conference*, pp. 51-58, December 2002.
- [157] **Xu J., Zhu W.T. and Feng D.G.**, "An Improved Smart Card based Password Authentication Scheme With Provable Security," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 723-728, June 2009.
- [158] **Yahoo**, "What is a Sign-in Seal?"  
[http://security.yahoo.com/article.html?aid=2006102507/.](http://security.yahoo.com/article.html?aid=2006102507/), Accessed: November 15, 2008.
- [159] **Yang G., Wong D.S., Wang H. and Deng X.**, "Two-Factor Mutual Authentication Based on Smart Cards and Passwords," *Journal of Computer and System Sciences*, vol. 74, no. 7, pp. 1160-1172, November 2008.
- [160] **Yang W.H. and Shieh S.P.**, "Password Authentication Schemes With Smart Card," *Computers & Security*, vol. 18, no. 8, pp. 727-733, February 1999.
- [161] **Yang Y., Bao F. and Deng R.H.**, "A New Architecture for Authentication and Key Exchange Using Password For Federated Enterprises," *Proc. of 20<sup>th</sup> International Federation for Information Processing Information Security Conference (SEC '05)*, pp. 95-112, March 2005.

- [162] **Yang Y., Deng R.H. and Bao F.**, "A Practical Password-Based Two-Server Authentication and Key Exchange System," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 2, pp. 105-114, June 2006.
- [163] **Yang Y., Bao F. and Deng R.H.**, "Efficient Client-to-Client Password Authenticated Key Exchange," *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp. 202-207, December 2008.
- [164] **Yang Y., Zhou J., Weng J. and Bao F.**, "A New Approach For Anonymous Password Authentication," *Annual Computer Security Applications Conference*, pp. 199-208, December 2009.
- [165] **Ye E.Z. and Smith S.**, "Trusted Paths For Browsers," *ACM Transactions on Information and System Security*, vol. 8, no. 2, pp. 153-186, August 2005.
- [166] **Yeh T.C., Shen H.Y. and Hwang J.J.**, "A Secure One Time Password Authentication Scheme Using Smart Cards," *IEICE Transaction on Communications*, vol. E85-B, pp. 2515-2518, November 2002.
- [167] **Yongdong W.U., Yao H. and Bao F.**, "Minimizing SSO Effort in Verifying SSL Anti-phishing Indicators," *Proc. of 23<sup>rd</sup> International Information Security Conference IFIP TC 11*, Springer, vol. 278, pp. 47-61, September 2008.
- [168] **Yoon E.J., Ryu E.K. and Yoo K.Y.**, "Further Improvement of an Efficient Password Based Remote User Authentication Scheme Using Smart Cards," *IEEE Transactions on Consumer Electronics*, vol. 50, no. 2, pp. 612-614, August 2004.
- [169] **Yoon E.J., Ryu E.K. and Yoo K.Y.**, "Attacks on the Shen et al.'s Timestamp-Based Password Authentication Scheme Using Smart Cards," *IEICE Transactions on Fundamentals*, vol. E88-A, no. 1, pp. 319-321, 2005.
- [170] **Yoon E.J., Ryu E.K. and Yoo K.Y.**, "An Improvement of Hwang-Lee-Tang's Simple Remote User Authentication Scheme," *Computers & Security*, vol. 24, no. 1, pp. 50-56, February 2005.
- [171] **Yoon E.J. and Yoo K.Y.**, "More Efficient and Secure Remote User Authentication Scheme Using Smart Cards," *Proc. of 11<sup>th</sup> International Conference on Parallel and Distributed System*, vol. 2, pp. 73-77, July 2005.



- [172] **Yoon E.J. and Yoo K.Y.**, "Improving the Dynamic ID-Based Remote Mutual Authentication Scheme," Proc. of OTM Workshops 2006, Springer-Verlag, LNCS, vol. 4277, pp. 499-507, July 2006.
- [173] **Zhang Y., Egelman S., Cranor L. and Hong J.**, "Phinding Phish: Evaluating Anti-Phishing Tools," Proc. of 14<sup>th</sup> Annual Network & Distributed System Security Symposium (NDSS 2007), California, USA, March 2007.

## AUTHOR'S PUBLICATIONS

---

### **International Journals (Published/Accepted):**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Dynamic Identity-based Single Password Anti-Phishing Protocol", Security and Communication Networks, John Wiley and Sons (Interscience), DOI: wiley.com/10.1002/sec.169, In Press, 2010.
- [2] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Secure Dynamic Identity based Authentication Scheme Using Smart Cards", Accepted in Information Security Journal: A Global Perspective, Taylor & Francis, 2010.
- [3] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Cookie based Virtual Password Authentication Protocol", Accepted in International Journal of Information and Computer Security, Interscience. 2010.
- [4] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Inverse Cookie based Virtual Password Authentication Protocol", Accepted in International Journal of Network Security, 2010.
- [5] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Liou et al.'s Authentication Scheme Using Smart Cards", International Journal of Computer Application, vol. 1, no. 8, pp. 17-24, ISBN: 978-93-80746-07-4, (Through IISc Bangalore), India, 2010.

### **Book Chapter:**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Secure Dynamic Identity-Based Remote User Authentication Scheme", Springer-Verlag, LNCS, vol. 5966, pp. 224-235, 2010.

### **International/National Conferences:**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Xu et al.'s Authentication Scheme Using Smart Cards", Published in 3<sup>rd</sup> Compute 2010 Conference of ACM, ACM Digital Library, no. 15, pp. 1-5, ISBN: 978-1-4503-0001-8, Bangalore, India, 2010.
- [2] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Hsiang-Shih's Authentication Scheme Using Smart Cards", Published in ICWET 2010

## AUTHOR'S PUBLICATIONS

---

### **International Journals (Published/Accepted):**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Dynamic Identity-based Single Password Anti-Phishing Protocol", Security and Communication Networks, John Wiley and Sons (Interscience), DOI: wiley.com/10.1002/sec.169, In Press, 2010.
- [2] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Secure Dynamic Identity based Authentication Scheme Using Smart Cards", Accepted in Information Security Journal: A Global Perspective, Taylor & Francis, 2010.
- [3] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Cookie based Virtual Password Authentication Protocol", Accepted in International Journal of Information and Computer Security, Interscience. 2010.
- [4] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Inverse Cookie based Virtual Password Authentication Protocol", Accepted in International Journal of Network Security, 2010.
- [5] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Liou et al.'s Authentication Scheme Using Smart Cards", International Journal of Computer Application, vol. 1, no. 8, pp. 17-24, ISBN: 978-93-80746-07-4, (Through IISc Bangalore), India, 2010.

### **Book Chapter:**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Secure Dynamic Identity-Based Remote User Authentication Scheme", Springer-Verlag, LNCS, vol. 5966, pp. 224-235, 2010.

### **International/National Conferences:**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Xu et al.'s Authentication Scheme Using Smart Cards", Published in 3<sup>rd</sup> Compute 2010 Conference of ACM, ACM Digital Library, no. 15, pp. 1-5, ISBN: 978-1-4503-0001-8, Bangalore, India, 2010.
- [2] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improvement of Hsiang-Shih's Authentication Scheme Using Smart Cards", Published in ICWET 2010

- Conference of ACM, ACM Digital Library, pp. 19-25, ISBN:978-1-60558-812-4, Mumbai, India, 2010.
- [3] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "An Improvement of Wang et al.'s Authentication Scheme Using Smart Cards", Published in 16<sup>th</sup> IEEE National Conference on Communications, IEEE Explore, pp. 1-5, DOI: 10.1109/NCC.2010.5430153, I.I.T Madras, India, 2010.
- [4] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "Cryptanalysis of Password Authentication Schemes: Current Status and Key Issues", Published in IEEE International Conference on Methods and Models in Computer Science (ICM2CS), IEEE Explore, pp. 1-7, ISBN: 978-1-4244-5051-0, JNU, Delhi, India, 2009.
- [5] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "An Improvement of Liao et al.'s Authentication Scheme Using Smart Cards", Published in 2<sup>nd</sup> IEEE International Advance Computing Conference, IEEE Explore, pp. 240-245, DOI: 10.1109/IADCC.2010.5423004, TIET, Patiala, India, 2010.
- [6] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "Countermeasures to Thwart Phishing Attacks", Published in 1<sup>st</sup> IEEE International Advance Computing Conference, TIET, Patiala, India, 2009.
- [7] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "Secure Single Password Anti-Phishing Protocol", Accepted in 3<sup>rd</sup> IEEE International Conference on Internet Multimedia Systems Architecture and Application (IMSAA-09), IIIT, Bangalore, India, 2009.
- [8] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "An Improvement of Yoon-Ryu-Yoo's Simple Remote User Authentication Scheme", Accepted in 4<sup>th</sup> International Conference, CERA, IIT, Roorkee, India, 2009.

#### **International Journals (Communicated):**

- [1] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "Single Password Anti-Phishing Authentication Protocol to Thwart Online Dictionary Attacks", IEEE Transactions on Dependable and Secure Computing (To be Submitted), 2010.
- [2] Sandeep K. Sood, Anil K. Sarje and Kuldeep Singh, "Dynamic Identity based Authentication Protocol for Multi-Server Architecture", Communicated in Journal of Systems and Software, Elsevier Ltd, 2010.

- [3] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "A Secure Dynamic Identity based Authentication Protocol for Multi-Server Architecture", Communicated in Journal of Network and Computer Applications, Elsevier Ltd, 2010.
- [4] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "SSO Password based Two-Server Authentication Protocol", Communicated in Journal of Computer Security, I/O Press, 2010.
- [5] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improved and Secure Smart Card based Dynamic Identity Authentication Protocol", Communicated in International Journal of Multimedia Intelligence and Security, Inderscience (Special Issue on Multimedia Information and Communication Security), 2010.
- [6] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "An Improved and Secure Smart Card based Authentication Scheme", Communicated in International Journal of Multimedia Intelligence and Security, Inderscience (Special Issue on Multimedia Signal Processing and Communications), 2010.
- [7] Sandeep K. Sood, Anil K. Sarje and Kuldip Singh, "Smart Card based Secure Authentication and Key Agreement Protocol", Communicated in EURASIP Journal on Wireless Communications and Networking, (Special Issue on Security and Resilience for Smart Devices and Applications), 2010.