ROAMING AUTHENTICATION AND END-TO-END AUTHENTICATION IN WIRELESS SECURITY

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

____
Men Long

Certificate of Approval:

Darrel Hankerson Professor Mathematics	Chwan-Hwa Wu, Chair Professor Electrical and Computer Engineering
Fa Foster Dai Associate Professor Electrical and Computer Engineering	Stephen L. McFarland Dean Graduate School

ROAMING AUTHENTICATION AND END-TO-END AUTHENTICATION IN WIRELESS SECURITY

Men Long

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 8, 2005

VITA

Men Long was born in October 1978, in Chongqing, People's Republic of China. He attended Chongqing University, Chongqing, China, in September 1996, and graduated with Bachelor of Engineering degree (honors) in Electrical Engineering in July 2000. He studied in The University of Tulsa, Oklahoma, USA, from August 2000 to May 2002, and graduated with Master of Science degree in Electrical Engineering. Men Long then entered Graduate School, Auburn University, in August 2002.

DISSERTATION ABSTRACT

ROAMING AUTHENTICATION AND END-TO-END AUTHENTICATION IN WIRELESS SECURITY

Men Long

Doctor of Philosophy, August 8, 2005
(M.S., The University of Tulsa, May, 2002)
(B.S., Chongqing University, July, 2000)

Directed by Chwan-Hwa Wu

With the rapid development of wireless networks, researchers, practitioners, and end users are paying closer attention to the security issues in mobile computing. Authentication is the imperative step to establish security for wireless networking applications. In this dissertation, we propose a comprehensive solution for the authentication issue in wireless networks by tackling the two subproblems: roaming authentication for network access, and end-to-end authentication for applications.

For roaming authentication, existing protocols are inadequate for the vastly increasing scale of networks and users. We present two authentication methods that demonstrate better performance in terms of authentication latency and energy consumption of a mobile terminal, compared to the 3G cellular roaming authentication protocol. The proposed method I, referred to as the minimum home-network-intervention authentication, employs the idea that a mobile terminal and its home network use different cryptographic random number seeds to generate random nonces for authentication. The proposed method II, called localized authentication, explore a new concept of localizing roaming authentication without the intervention of the home network of a mobile terminal. The feature of both methods is that they significantly reduce the number of message transmissions between home and visited networks for roaming authentication. Analytical models and measurement results are provided to indicate the performance superiority of the proposed methods.

For end-to-end authentication, existing methods, based on the traditional adversary model, are not sufficient to defend against attackers with break-in capabilities. Furthermore, the scarcity of energy supply for wireless devices usually conflicts with expensive cryptographic computation. We present a novel authentication protocol that combines the hash chain technique with the symmetric message authentication code (MAC). Firstly, the protocol can defend against the strong adversaries who compromise a protocol participant to obtain the authentication secrets. Secondly, we propose the technique of complementary MACs to enable the intrusion detection of a strong adversary. Thirdly, we introduce the technique of piggyback hash-chain-value-update to increase the number of allowable authentication sessions after the system setup. Finally, we present the technique of optimum hash-chain-iteration-tuning to optimize power consumption of a wireless device. Both analytical and implementation results indicate that the new protocol is among the most efficient ones in terms of authentication latency and energy consumption of a mobile device.

ACKNOWLEDGMENTS

Without the following individuals, this dissertation could not be done in a timely manner. First, I would like to give my sincere thanks to my major advisor Dr. Chwan-Hwa Wu. Dr. Wu has shaped me into a more mature person, and I have benefited greatly from his invaluable advice on both technical and life topics. Second, I would like to give my warm thanks to the other members of the committee, Dr. Darrel Hankerson and Dr. Fa Dai. I will always remember the cooperation with Dr. Hankerson on the study of elliptic curve cryptography, and more importantly, his research style of emphasizing both theory and implementation. Dr. Dai has given me constant feedbacks and encouragements during the course of this dissertation. Another special person to be acknowledged is Dr. J. D. Irwin because of his support for my Ph.D. study. In addition, I would like to thank Dr. Chung-Wei Lee for serving as the outside reader. Looking back the past three years at Auburn, I am grateful to other students in our research group and the friends in the university for their help and inspiring discussions.

Last, but not least, this dissertation is dedicated to Big J, and my Dad and Mom. I thank Big J because He is life; I thank my parents for their unconditional support and love throughout the years.

Style manual or journal used IEEE Journal style. Bibliography follows van Leunen's *A Handbook for Scholars*

Computer software used: Microsoft Word 2000 SR1, Microsoft Visio 2000 SR1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The advent of pervasive computing has presented new challenges to network security. As is stated in [1], authentication will reside at the core of security in the new paradigm of ubiquitous computing. In practice, authentication applications can be roughly categorized into two classes, authentication of a mobile user for network access, and authentication of a remote user for data transactions with a private organizational server.

In this chapter, we discuss the underlying technical challenges of network authentication. Next, we explain the motivations for our work in roaming and end-to-end authentication. Finally, the roadmap for the rest of this dissertation is presented.

1.1 Challenges for Authentication Protocols

We list the following four issues that should be the design focus for any network authentication protocol.

- ✓ **Scalability**. Wireless networks have been developed and deployed at a very fast pace. From the vision of pervasive computing, there will be a huge amount of end users and considerably large number of wireless access networks. Any roaming authentication protocol will be judged according to the scalability to myriad users and networks.

- ✓ **Scarcity of energy supply for wireless devices**. Wireless devices are usually battery-powered, which have very limited energy input. On the other hand, the cryptographic primitive of an authentication protocol may be computationally expensive, or generate long messages to incur high communicational overhead.
- ✓ **Server handling many simultaneous authentication requests**. In the past, protocol designers usually focus only on the efficacy at the wireless device side and assume that a server has unlimited computational power to handle authentication handshakes. Nevertheless, the importance of the efficiency at the server side should not be ignored, as denial of service (DoS) attacks are getting increasing attentions from the society. If the authentication protocol is heavy weight for a server, computer hackers can exploit the feature to send bogus authentication requests to trap the server into the intensive computation. This kind of attack is clearly more dangerous than the simple packet flooding attacks.
- ✓ **Ever-increasing capabilities of hackers**. In the traditional adversary model, attackers can monitor/record the communication, modify the messages transmitted, and initiate authentication requests with the server or forge a response to the authentication request from a legitimate user. With the rapid development of computer and electronics technology, attackers are able to do more than those prescribed in the traditional adversary model. Thus we use the term "strong adversary model" to reflect the evolving capabilities of computer hackers. In the strong adversary model, attackers can compromise

either a user's wireless device or the private server in addition to their capabilities in the traditional adversary model. If an attacker subverts the device or the server, all the stored secrets for authentication are obtained by the adversary.

1.2 Motivation for Proposed Roaming Authentication Methods

Roaming is an inherent feature of a 3G data network and used to enlarge the service area while inter-network roaming is yet to appear for wireless LAN. Wireless LANs are popular in university campuses and hotspots such as convention centers, airport lounges, and cafés, to name just a few. In the foreseeable future a few big wireless Internet service providers may appear with a network scale comparable to that of cellular networks. Roaming supports the goal of connecting to the Internet anywhere and at any time, and users are able to receive only one bill from their service provider when they travel across different networks.

Each wireless Internet service provider may have tens or hundreds of networks, and every network will have one authentication center. This scenario, from an abstract structure point of view, will be similar to that of a 3G cellular data network. As extensions of the wireless LAN, the forthcoming IEEE 802.16 (wireless MAN) and IEEE 802.20 (wireless WAN) will perhaps have a similar structure of inter-network roaming.

Authentication protocols for 3G roaming were finalized a few years ago [2], [3], [4]. Recently several proposals which specifically target wireless LAN inter-network roaming authentication have surfaced [5], [6], [7], [8]. The issue of inter-network roaming authentication describes the way in which a visited network mutually authenticates a roaming user, and the protocol design difficulty arise because the authentication center of a visited network does not share a secret with a roaming user in advance.

Fig. 1.1 depicts this concept, in which a mobile terminal, belonging to home network A, may share a secret with the authentication center. However, when the mobile terminal travels into visited network B, the authentication center of network B does not have a shared secret with the mobile terminal. At this moment, if the mobile terminal makes an association request, a mutual authentication will be required between the authentication center of network B and the mobile terminal. This type of authentication is called inter-network roaming authentication. If the mobile terminal subsequently moves from one cluster of access points to another within network B, or is periodically reauthenticated by network B during its residence, intra-network roaming authentications occurs.

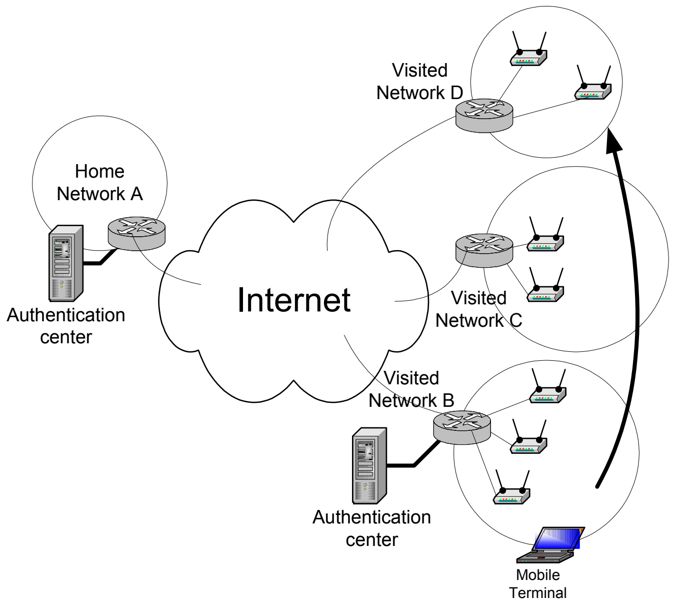


Fig. 1.1: A scenario for wireless LAN inter-network roaming

Since the 3G authentication protocol is believed to be a viable approach, a question that naturally arises is whether or not the 3G roaming authentication protocol can be directly adopted for all wireless inter-network roaming. Our work answers this question by providing two alternative authentication methods that provide a cost advantage in terms of latency and energy consumption of mobile terminals. In the meantime, the security level provided by our methods is not inferior to that of the 3G authentication protocol.

The proposed Method I is called minimum home-network-intervention authentication, and it employs symmetric key-based cryptography. We advocate that a mobile terminal and its home network authentication center have different cryptographic random number seeds, which generate the cryptographic nonces used to challenge each other. After the initial authentication, a shared key is derived for subsequent intra-network roaming authentication. This approach stands in contrast to the synchronized authentication of 3G, where a mobile terminal and its home network keep track of a sequence number and the sequence number will increase by one after each successful mutual authentication.

The proposed Method II is called localized authentication, and it exploits public key-based cryptography. The authentication traffic is completely localized without any intervention of a user's home network. This method is robust against any possible failure of a home network authentication server or any of the networks along the path between home and visited network authentication servers. One critical design issue is key storage scalability. We solve this problem by proposing a practical public key certificate structure such that the number of public/private keys stored in a network authentication server is linear with the number of service providers rather than the much larger number of networks. In addition, the protocol design is able to defend against denial of service attacks when attackers try to exploit the intensive computation required in public key schemes.

1.3 Motivation for Proposed End-to-End Authentication Method

The widespread availability of wireless devices such as PDAs and laptops are one step towards making the vision of anywhere, anytime pervasive access and computing a reality. One quintessential application is that a corporate employee travels on the road and uses a handheld device to retrieve/upload mission-critical data from/to a private server of the organization. Between a mobile device and a private server are public wireless access networks and the wired Internet. The underlying radio medium of a wireless network is vulnerable to attacks because a nearby adversary can monitor and manipulate message flows. The security concern poses a serious impediment to the adoption of wireless devices for corporate computing applications.

Entity authentication is the first step for bringing security into those applications. Fig. 1.2 illustrates end-to-end authentication under the pervasive computing paradigm. Through the authentication, a mobile device and a private server can establish the trust that the other side is the one with which it wants to communicate. The message privacy and integrity can be established after the entity authentication.

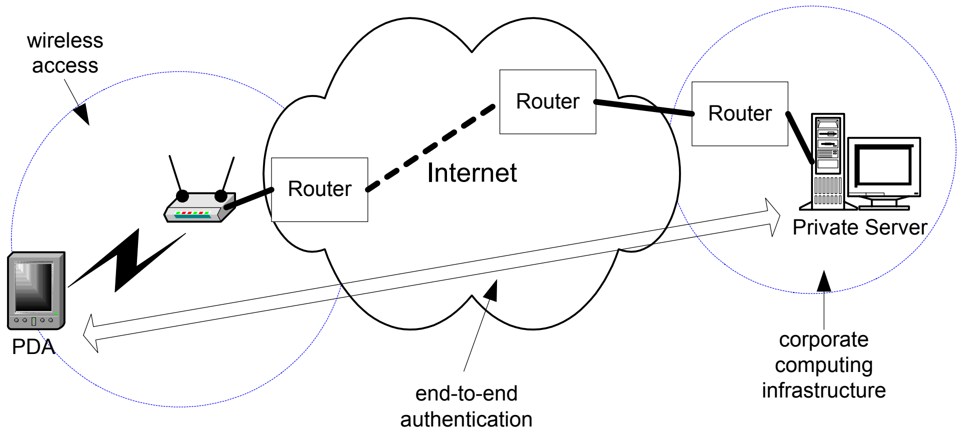


Fig. 1.2: End-to-end authentication of a mobile device

Our work focuses on designing an efficient authentication protocol for mutual entity authentication in the context of wireless security. Specifically, the protocol is robust against the strong adversaries who can compromise a participant of the authentication protocol to obtain the authentication secrets. In addition, the protocol intends to lower the power consumption of authentication handshake of a mobile device.

There have been a number of end-to-end authentication methods, but most of them do not consider the cases of adversary compromising a protocol participant. A large proportion of the protocols can be roughly classified into the family of EAP framework (extensible authentication protocol) [9], where EAP-MD5 [10] and EAP-TLS [11] are two representative methods. In EAP-MD5, a user shares a password with the private server. The wireless device and the server separately hash a random nonce concatenated with the password to authenticate each other. Since only several hash operations are used, the computational and communicational cost is one of the lowest among authentication protocols. The disadvantage is its vulnerability to password-guessing attacks in the traditional adversary model (pp. 391-393) [12]. EAP-TLS uses public key cryptography, where a wireless device and a private server have their public/private key pair. Thus, the computational and communicational cost is significantly more than that of EAP-MD5. EAP-TLS is secure in the traditional adversary model, but becomes insecure in the strong adversary model.

The feature of the proposed protocol is the novel way of combining a symmetric key, a symmetric pin, and an asymmetric hash chain. A user shares a pin and a cryptographic key with the private server. In addition, the user's wireless device has a hash chain preimage while the server stores the hash chain value. The hash chain creates an asymmetric setting; thus an adversary compromising the server to attain the authentication secrets cannot succeed in impersonating a legitimate user at the next login. As the pin is remembered in a user's brain and never stored in the mobile device, a strong adversary compromising the device does not learn the pin and hence cannot manage the next authentication attempt. Therefore, the protocol is secure in the traditional adversary model as well as in the strong adversary model. The detection of strong adversary presence is based on the technique of complementary MACs. Furthermore, the technique of piggyback hash-chain-value-update significantly reduces the cost of system configuration, and the technique of optimum hash-chain-iteration-tuning lowers the power consumption of a wireless device.

1.4 Dissertation Organization

This dissertation is divided into eight chapters. The first two chapters provide the background information that is necessary to understand the rest of the work. The remaining chapters are the main contribution of our work.

In Chapter 2, we present the related work on roaming and end-to-end authentication methods. Several representative protocols that are popular either in the real-world implementation or in the research literature are described at length.

In Chapter 3, we describe the technical details of the proposed methods for roaming authentication. We break down the protocols into a flow-by-flow description. The cryptographic primitives associate with each flow are also thoroughly discussed.

In Chapter 4, we compare the proposed methods with the 3G cellular authentication protocol in terms of latency and energy consumption. The performance evaluation includes the analytical model, network measurement, and simulation results.

In Chapter 5, we give the algorithm details of the proposed method for end-to-end authentication. The design rationales for the proposed protocol are thoroughly discussed.

In Chapter 6, we compare the proposed method with the EAP-MD5 and EAP-TLS protocols. The evaluation methodology consists of analytical models and implementation results.

In Chapter 7, we consider some extensions to the proposed authentication protocols for roaming and end-to-end authentication. The theme of this chapter is to add some important security features at the cost of increasing the algorithm and implementation complexity.

The main results of this dissertation are summarized in Chapter 8. Future work is also included in this chapter to conclude this dissertation.

CHAPTER 2

BACKGROUND AND RELATED WORK

An overview on the existing work of roaming and end-to-end authentication protocols is given in this chapter. The objective is not to exhaustively list all the existing methods, but to use the representative samples to illustrate security protocol design principles. For a coherent and concise presentation, we decide not to give a separate introduction on all the cryptographic primitives involved in the authentication protocols. Instead, we adopt the just-in-time approach to present the mathematics wherever it is necessary to be discussed.

2.1 Samples of Existing Roaming Authentication Protocols

2.1.1 3G Authentication Protocol

In the 3G authentication protocol, a mobile terminal and the authentication center of its home network share a key. Moreover, both entities keep track of a sequence number. Fig. 2.1 illustrates the entire procedure of the authentication.

When a mobile user moves into a visited network, the authentication center of the visited network does not have previously stored authentication information about the mobile user and thus sends the authentication data request message to the home network of the mobile user. Upon receiving the authentication data request, the authentication center of the home network will perform the cryptographic algorithm on the shared secret and sequence number to calculate an authentication vector AV. An AV consists of a random number RAND, expected response XRES, cipher key CK, and integrity key IK, and authentication token AUTN. Then the array of AV (several authentication vectors) is delivered to the visited network.

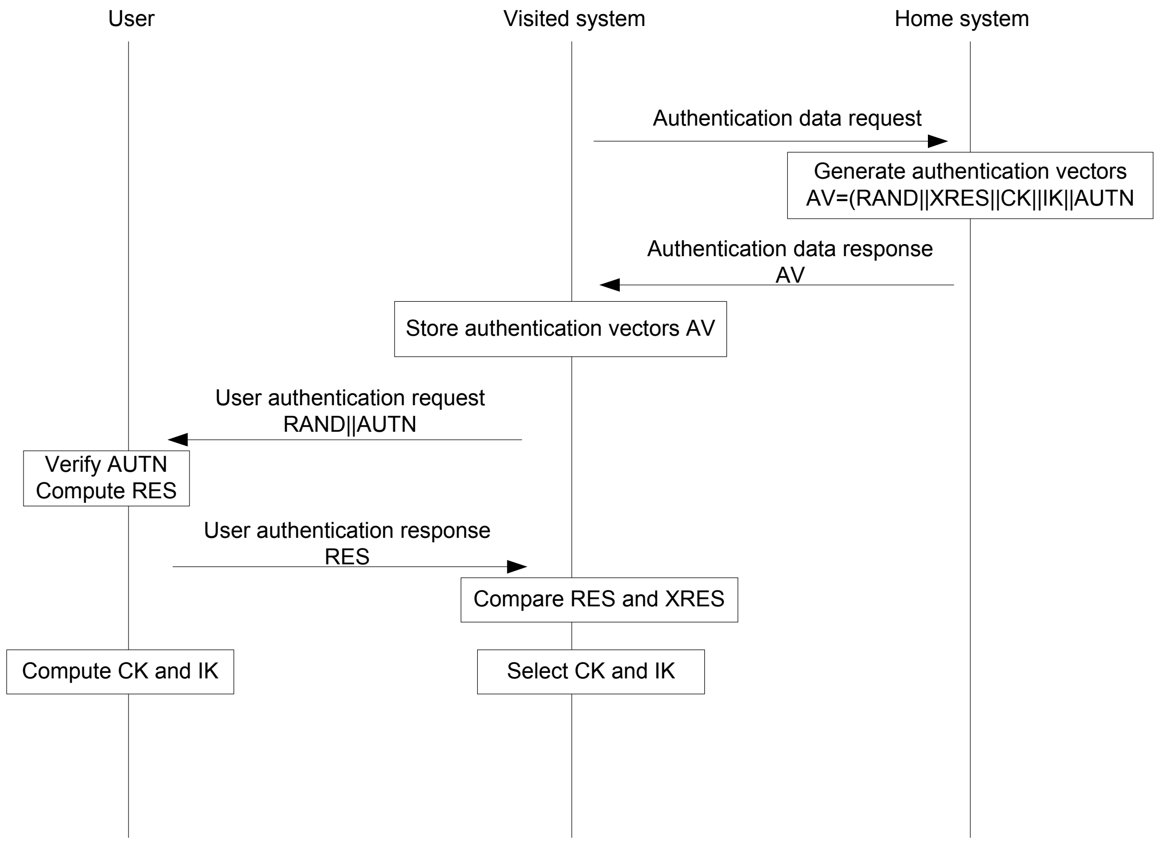

User
Visited system
Home system
Authentication data request
Generate authentication vectors
AV=(RAND||XRES||CK||IK||AUTN
Authentication data response
AV
Store authentication vectors AV
User authentication request
RAND||AUTN
Verify AUTN
Compute RES
User authentication response
RES
Compare RES and XRES
Compute CK and IK
Select CK and IK

Fig. 2.1: Message flows of the 3G cellular authentication protocol. The communication channel between visited and home networks is secure by the other means out of the scope of the authentication protocol.

The visited network authentication center selects the next unused authentication vector from the ordered AV array and sends RAND and AUTN to the mobile terminal. The roaming mobile terminal will use its key and sequence number to verify AUTN. If AUTN is valid, the mobile terminal produces a response RES that is sent back to the visited network. The visited network compares the received RES with XRES. If they match, then the authentication and key agreement exchange is successfully completed. In addition, during the handshake, the mobile terminal computes CK and IK using the received RAND and AUTN. CK and IK are used for performing encryption and integrity protection for the subsequent data delivery. The visited network simply retrieves CK and IK from the AV. The visited network will compare the received response from the mobile terminal with the expected one from the home network.

The sequence number defends against replay attacks. Since the visited network cannot derive the AV, every intra-network roaming authentication requires one transmission between the home and the visited networks. Because the transmission is usually expensive, increasing the number *L* of AVs to reduce the number of transmissions is desirable. On the other hand, if *L* is too large, the AVs may consume too much network bandwidth for every transmission from the home network to the visited network. In the 3G standard, *L* is fixed at 5. An adaptive algorithm on *L* was proposed in [13], which achieves better performance at the cost of increased implementation complexity. References [14], [15], [16], [17] have the authentication architecture is similar to that of the 3G authentication protocol, although specific algorithms are different.

2.1.2. Password-based Method

The method outlined in [5] is described as follows: a user has a password and the authentication center of its home network stores a one-way function of it. An encrypted channel will be established by SSL or TLS (public key based protocols) [18], [19] and then the user enters his/her credentials, such as name@domain and password, into the authentication portal of the visited network through the encrypted channel. Next, the visited network sends the user’s credentials to its home network through a pre-established secure channel between networks. Only the home network is capable of verifying the user’s credentials and thus sends the decision (“accept” or “reject”) back to the visited network. The network-to-user authentication is achieved during the execution of SSL handshake protocol. One weakness of the method is that the password, presumably a secret only known by the mobile terminal and its home network, is now released to visited networks. A single sign-on authentication architecture that confederates wireless LAN service providers through trusted identity providers was proposed in [6].

2.1.3 Limitations of Existing Methods

3G authentication protocol is based on the symmetric key cryptography that has the advantage of lightweight computational cost. Due to the scalability limit, however, a visited network authentication center cannot *a priori* agree on a shared key with a roaming user. In this sense, the fundamental limit is that if only symmetric key cryptography is employed, some third party (in this case, home network) has to be involved. The design rationale of such protocol is to reduce the computational cost at the expense of communicational overhead.

The password protocol also consists of three parties (roaming user, home and visited networks). The difference is that the protocol is designed for the wireless Internet applications, taking advantage of SSL or TLS protocol that has already been embedded into web programs. Nevertheless, the authentication handshake in SSL or TLS is computationally more expensive than that in the symmetric key based protocols. Another drawback of the password based protocol is that user’s password will be known by visited networks during the authentication handshake between the user and the home network.

2.2 Samples of Existing End-to-End Authentication Protocols

End-to-end authentication protocol has a long history in computer and communication security (chapter 10) [12]. In general, most of them are based on the challenge-response paradigm, where one entity (the claimant) proves its identity to another entity (the verifier) by demonstrating knowledge of a secret known to the claimant and/or the verifier. If the authentication is performed in two ways, it is mutual authentication. The challenge is typically a number chosen by one entity (randomly and secretly) at the outset of the protocol. If the communications line is monitored, the response from one execution of the authentication protocol should not provide an adversary with useful information for a subsequent authentication, as later challenges will differ. The same rationale applies to the part of response.

2.2.1 EAP-MD5 Protocol

This protocol utilizes a one-way hash function and its message flow is a 3-way handshake. The two participants of the protocol share a password. After the communication link establishment phase is complete, the authenticator sends a challenge to the peer. The peer responds with a value calculated using the one-way hash function over the challenge and the password. The authenticator checks the response against its own calculation of the expected hash value. If the values match, the authentication is acknowledged; otherwise the connection would be terminated.

2.2.2 EAP-TLS Protocol

TLS protocol has gained the substantial popularity in the web sensitive data transaction applications, and is a protocol based on the client-server model. EAP-TLS protocol can be regarded as applying the algorithm of TLS to the realm of wireless end-to-end authentication.

In the protocol, both client and sever have their own public/private key pairs. The client sends a random number to the server that also generates a random number and transmits it to the client. Next, the client uses its private key to sign the two nonces. In addition, the client generates a premaster secret and uses the server's public key to encrypt the premaster secret. Then, the client transmits the ciphertext and the signature to the server. The server decrypts the ciphertext first to obtain the premaster secret, and then verifies the authenticity of the signature. If so, the server believes that the client is legitimate, and generates a master key from the premaster secret. The master key is used to produce a message authentication code over the nonce, and the server sends the message authentication code back to the client. The client use the master key derived from the premaster secret to validate the message authentication code. If it is valid, the mutual authentication and key agreement is successfully completed.

2.2.3 Limitations of Existing Method

EAP-MD5 and the EAP-TLS are vulnerable to the strong adversaries. If an adversary compromises a protocol participant, he obtains the key and then is able to execute the protocol for subsequent authentication attempts. In other words, if a compromise occurs, the adversary is now indistinguishable with the legitimate user.

CHAPTER 3

PROPOSED ROAMING AUTHENTICATION METHODS

In this chapter, the proposed method I (minimum-home-network-intervention authentication) is discussed in depth, followed by the proposed method II (localized authentication).

3.1 Proposed Method I (Minimum-Home-Network-Intervention Authentication)

Method I, like the 3G authentication protocol, is used to authenticate a device. Under this method, a user's mobile terminal and its home network share a key, but have different cryptographic random number seeds. During inter-network roaming authentication, both entities will challenge each other using the nonce generated by the random number seed. Another feature of the scheme is that an authentication key *k_auth* is derived during the protocol execution, which is cached by the authentication center of the visited network and the mobile terminal. The derived *k_auth* will be used for subsequent intra-network roaming authentications.

3.1.1 Inter-network Roaming Authentication Scheme

Fig. 3.1 shows the message flows of the proposed protocol. The underlying authentication structure is the challenge-response paradigm. A cryptographic nonce generated by the mobile terminal is included in the first message flow. Accordingly, the

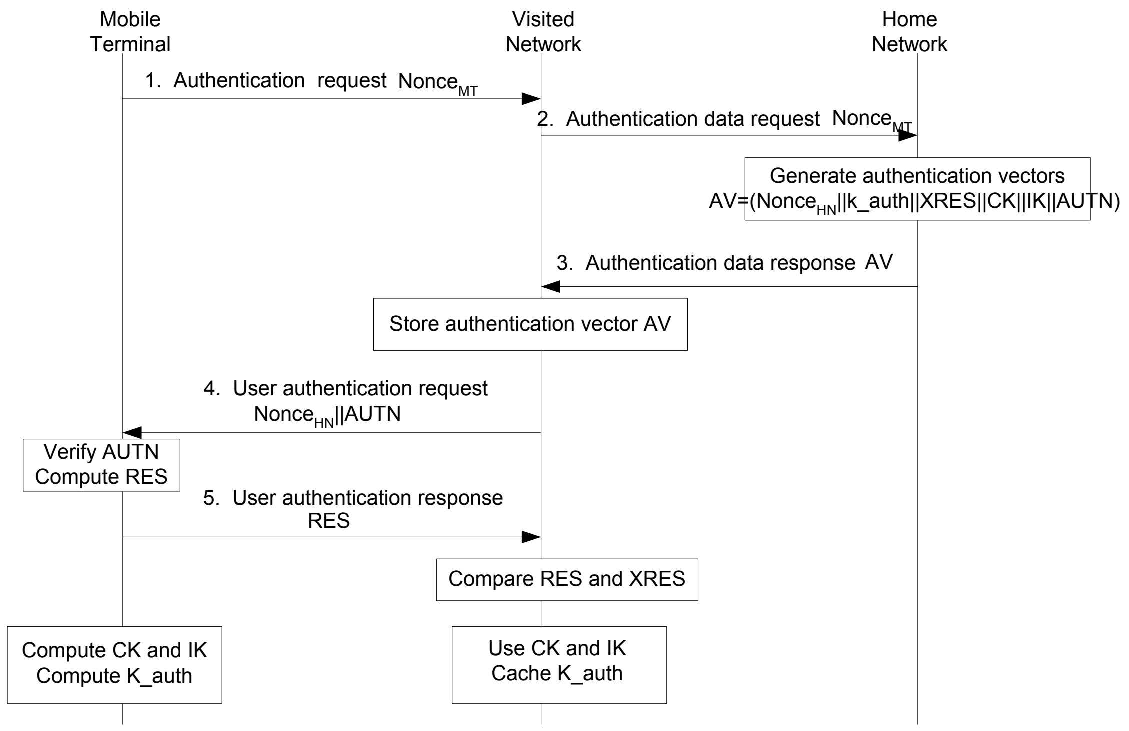

Mobile Terminal
Visited Network
Home Network
1. Authentication request Nonce_MT
2. Authentication data request Nonce_MT
Generate authentication vectors AV=(Nonce_HN||k_auth||XRES||CK||IK||AUTN)
3. Authentication data response AV
Store authentication vector AV
4. User authentication request Nonce_HN||AUTN
Verify AUTN Compute RES
5. User authentication response RES
Compare RES and XRES
Compute CK and IK Compute K_auth
Use CK and IK Cache K_auth

Fig. 3.1: Message flows in the proposed method I (minimum home-network-intervention authentication). The communication channel between visited and home networks is secure by the other means out of the scope of the proposed authentication protocol.

authentication center of the home network performs the cryptographic algorithm over the nonces of the mobile terminal as well as its own. Assume that the mobile terminal and its home network authentication center share a key *k*. The mathematical expressions for the components in the authentication vector AV are:

$$\begin{aligned} XRES &= \mathrm{HMAC}_k(Nonce_{MT}, Nonce_{HN}), \\ AUTN &= \mathrm{HMAC}_k(Nonce_{HN}, Nonce_{MT}), \\ k\_auth &= \mathrm{HMAC}_k(Nonce_{MT}, Nonce_{HN}, ID_{MT}, ID_{VN}), \\ CK &= \mathrm{HMAC}_k(Nonce_{MT}, Nonce_{HN}, padding_1), \\ IK &= \mathrm{HMAC}_k(Nonce_{MT}, Nonce_{HN}, padding_2), \end{aligned} \tag{3.1}$$

where HMAC is the standard keyed message authentication code [20]; *ID*s are the identifiers; *padding*s are the prescribed publicly known character strings. *XRES* is the expected response used by the visited network to authenticate the mobile terminal. *AUTN* is the authentication token used by the mobile terminal to authenticate the visited network. *CK* and *IK* are the cipher and integrity keys for the data protection of the session. The security level depends on the key bit-length. One recommended value in current security practice is 160 bits for the random number seed as well as the shared key *k*.

The authentication request includes $Nonce_{MT}$ generated by the mobile terminal, which is an unpredictable random number. The home network also produces $Nonce_{HN}$. Both nonces appearing in the HMAC function can ensure freshness of the AV components. In addition, both sides contribute to the input of the HMAC function, which can defend against chosen text attacks if one participant is malicious. Most importantly, these random nonces $Nonce_{MT}$ and $Nonce_{HN}$ make the synchronized sequence number between a roaming user and its home network unnecessary.

3.1.2 Security Analysis

The mobile terminal can use the nonce and the shared key to verify the authenticity of the message flow 4. The correctness of flow 4 implies that it is from the home network and the home network trusts the visited network. Thus, if flow 4 is valid, the mobile terminal believes that the visited network is authentic. The visited network compares the flow 5 with the expected response. If they are equal, the visited network believes that the mobile terminal is the legitimate one from the corresponding home network. Notice that the bogus network cannot establish itself as the man-in-the-middle because it cannot setup the security association in the secure channel with an actual home network. Hence, the bogus network does not get the correct message flow 4, and the authentication protocol will abort upon the invalidity of flow 4.

3.1.3 Intra-Network Roaming Authentication

The number of AV in the proposed method is one. After the inter-network roaming authentication, the visited network authentication center caches *k_auth*. For instance, *k_auth* will be cached from the time of creation to 23:59:59 of the same day or even longer. Similarly, the client program in the user's terminal also caches *k_auth*. During the cache valid period, *k_auth* will be used for all intra-network roaming authentications. Consequently, no authentication transmission between the home and visited networks is needed for intra-network roaming authentications. Since only one transmission between the home and visited networks is needed for roaming user authentication, we coin the term "minimum home-network-intervention authentication" for method I.

3.1.4 Summary of Proposed Method I

The architecture of the proposed method I, as is shown in Fig. 3.1, is similar to that of the 3G authentication protocol. Nevertheless, there are two important improvements of method I. The first one is that we apply the cryptographic operations over the random nonces from the mobile terminal and home network, which eliminates the need for the synchronized sequence number. The second improvement is that we use a derived key from the inter-network roaming authentication handshake for the subsequent intra-network roaming authentication.

3.2 Localized Authentication

Method II is based on public key cryptography. The nodes involved in the system, i.e. authentication centers and mobile terminals, have their public/private key pairs. Since this is a closed system of wireless Internet service providers and their subscribers, we advocate that the service provider be the key issuer or certification authority. No third-party certificate authority is needed; thus the operation cost of the scheme can be reduced. Because the number of participants (users and networks) is huge, the public key certificate structure must be carefully designed.

3.2.1 Public key Certificate Structure

Fig. 3.2 illustrates the public key certificate structure. For simplicity of presentation, we assume an imaginary user Bob belonging to WISP I (Wireless Internet Service Provider). His home network is network I. Under the proposed public key certificate structure, service providers are in the top of the hierarchy, which have the master public/private key pairs. WISPs I and II have the public/private key pairs PK_1 / SK_1 and PK_2 / SK_2, respectively. The valid period for these keys may be relatively long (half year or more). Service providers will also cross-certify each other's public key, as shown by $SK_1 << PK_2 >>$ (WISP I certifying the public key of WISP 2) in the notation of X.509 [17]. Each network of a service provider will have a public/private key pair, e.g. $SK_1 << pk_{11} >> / sk_{11}$ in the case of network I of WISP I. The network's private/public key pair may have a shorter valid period (one month). Additionally, the service provider distributes its public key as well as other service providers' public keys to its networks. A subscriber of WISP I has their public/private key pairs certified by the service provider, as denoted by $SK_1 << PK_{MT} >> / SK_{MT}$. The user will also obtain the public key of its service provider. The valid period of the public/private key pair of the user may be short as well (one month). For instance, Bob's home network is network I of WISP I. When Bob moves to network II of WISP I, Bob is able to verify the visited network's public key $SK_1 << pk_{12} >>$ via PK_1. Meanwhile, the visited network can verify Bob's public key $SK_1 << PK_{MT} >>$ via PK_1. If Bob moves to network I of WISP II, Bob can chain-verify the visited network's public key $SK_2 << pk_{21} >>$ and $SK_1 << PK_2 >>$ via PK_1, and the visited network can verify Bob's public key $SK_1 << PK_{MT} >>$ via PK_1. We have chosen a short valid period for user's and network's public key so that the certificate revocation issue may be avoided in our scheme.

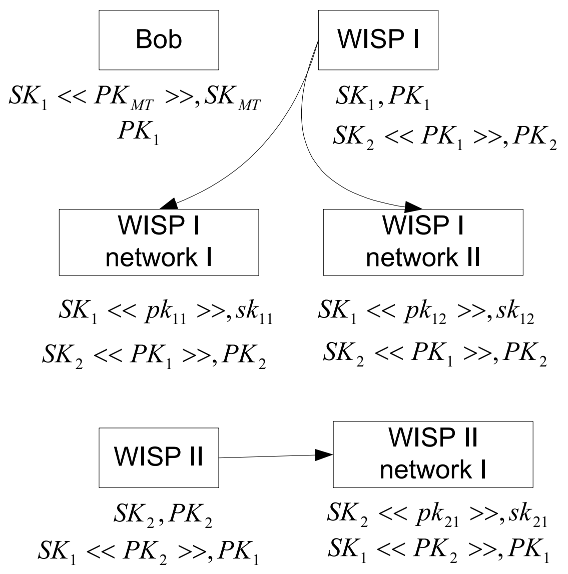


Fig. 3.2: Proposed public key certificate structure for method II.

The security level of the authentication depends on the bit-length of the public/private keys. Under the current guideline, one example may be that the public/private keys of the service provider are 2048–bit RSA keys while 1024-bit RSA keys are used for user and network public/private keys.

To summarize, suppose that in the roaming agreement there are a total of *a* service providers, *b* networks and *c* subscribers. One realistic estimate is that *a*, *b*, and *c* are equal to 6, 2000, and 1,000,000, respectively. Under our scheme, as is depicted in Fig. 3, each user stores 3 keys (its public/private key pair and the service provider's public key); each network stores 2(*a*-1)+2 keys (its public/private key pair, and other service providers' public keys and the cross-certified public key of its service provider); each service provider stores 2(*a*-1)+2 keys as well. As a result, the number of keys stored is linear with the number *a* of service providers, which is relatively a small number.

3.2.2 Message Flows of Proposed Method II

Fig. 3.3 shows the message flows of the authentication protocol, which share some similarities with the SSL or TLS handshake algorithms. After verifying the authenticity of the visited network's public key, the mobile terminal will encrypt a secret random number *premaster_key* using the visited network's public key. Since only the visited network has the corresponding private key, the visited network can decrypt the cipher text to obtain the *premaster_key*. Thus both sides establish a shared secret.

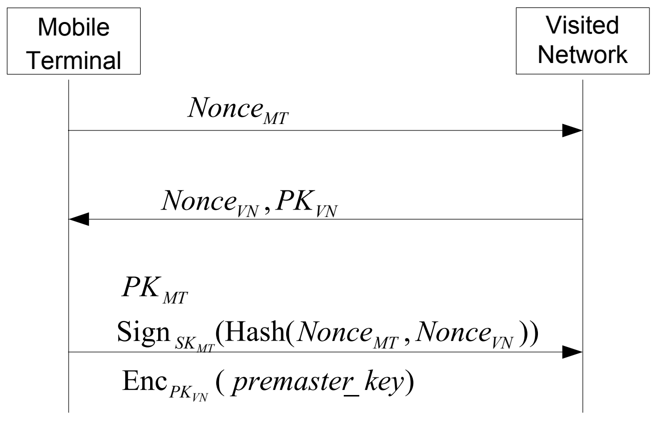

Mobile Terminal
Visited Network
$Nonce_{MT}$
$Nonce_{VN}, PK_{VN}$
PK_{MT}
$\text{Sign}_{SK_{MT}}(\text{Hash}(Nonce_{MT}, Nonce_{VN}))$
$\text{Enc}_{PK_{VN}}(premaster\_key)$

Fig. 3.3: Message flows in method II (localized authentication)

3.2.3 Summary of Proposed Method II

One difference from the proposed scheme is that we derive *k_auth*, which is used for all subsequent intra-network roaming authentication. One way to derive *k_auth* is

$$k\_auth = \text{HMAC}_{premaster\_key}(Nonce_{VN}, PK_{VN}, Nonce_{MT}, PK_{MT}) \qquad (3.2)$$

Another major modification is that the mobile terminal signs first and then does the encryption. Consequently, the visited network will verify the user's signature first. If it is authentic, the visited network then decrypts the cipher text. If not, the visited network simply drops the session. Since the signature appears earlier than the cipher text, we modify the signing step as $\text{Sign}_{SK_{MT}}(\text{Hash}(Nonce_{MT}, Nonce_{VN}))$, where the mobile terminal uses its private key to sign over the hash result of two nonces. The computational savings in case of bogus request by our scheme is significant, if the RSA encryption and signature [21] are used. We did the benchmark using the well-regarded Crypto++ library [22] on a Pentium 4 machine with 1.9GHz CPU and 512 MB memory. The result is shown in Table 3.1. The exponents for encryption and signature verification are 17.

Table 3.1. Computational time of crypto operations on a 1.9 GHz Intel Pentium 4 (RSA: 1024 bits, HMAC: 160 bits using SHA-1)

	Computational time (ms)
RSA encryption (*e*=17)	0.19
RSA decryption	4.65
RSA signature generation	4.65
RSA signature verification (*e*=17)	0.19
HMAC	0.002

Assume that an attacker, who does not have the legitimate public/private key pair, sends an authentication request. Under our scheme, the visited network consumes 0.19ms of CPU time to verify the signature, while in the original SSL protocol, the visited network has to consume 4.65ms+0.19ms to get to determine that the other end is not a legitimate user. As a result, the devastating effect of floods of authentication requests trying to overwhelm the authentication center of a visited network can be mitigated by our scheme. It is clear that no home network is involved for roaming authentication under method II; therefore, we coin the term "localized authentication" for method II.

3.2.4 Security Analysis

The bogus network does not have the private key of a legitimate network, so it cannot decrypt the message flow 3 to obtain the proper *premaster_key*. Consequently, it is not able to establish the security association with the mobile terminal. An illegitimate user does not have the appropriate public key certified by a wireless Internet service provider. Therefore, the visited network will surely reject the signature in flow 3.

CHAPTER 4

PERFORMANCE EVALUATION OF PROPOSED ROAMING AUTHENTICATION METHODS

In this chapter, we compare the performance of the proposed roaming authentication methods to some well known existing authentication approaches in terms of latency and energy consumption. First, we give the analytical model on the authentication transmission overhead between home and visited networks. Second, we present the results on the measured authentication latency in the perspective of a roaming user. Third, we provide simulation results based on a simplified queueing model. Finally, we give the energy measurement results on the energy consumption of the authentication protocol over the mobile devices.

4.1 Analytical Model: Authentication Transmission Overhead between Home and Visited Networks

It is well known in the wireless networking community that, for roaming authentication purposes, transmission between the home and visited networks is usually expensive. In this section, we evaluate the performance of method I, the 3G authentication protocol, and the adaptive scheme for 3G authentication (method of Lin and Chen) [13] in terms of the number of authentication transmissions between the home and visited networks. The evaluation methodology is drawn from [13].

Let the total transmission cost per user be $J = nC$, where n is the number of transmissions and C is the normalized cost per transmission. Assume that a roaming user makes a number of authentication requests, which satisfy a Poisson distribution with mean λ in a unit time. According to the probability mass function of the Poisson distribution,

$$\Theta(n,\tau) = \sum_{i=1}^{L} \left\{ \frac{(\lambda\tau)^{(n-1)L+i}}{[(n-1)L+i]!} \right\} e^{-\lambda\tau} \tag{4.1}$$

where $\Theta(n,\tau)$ is the probability that there are n transmissions between the visited and home networks in a specific period τ, and L is the number of AVs transmitted each time. The residence time period is assumed to be exponentially distributed with mean $1/\mu$ and the probability density function is

$$f(t) = \mu e^{-\mu t} \tag{4.2}$$

Then the probability that there are n authentication transmissions between the visited and home networks during the mobile terminal's residence in the visited network is

$$p(n) = \int_{t=0}^{\infty} \Theta(n,t) f(t) dt = \left(\frac{\lambda}{\lambda+\mu} \right)^{(n-1)L} \left[1 - \left(\frac{\lambda}{\lambda+\mu} \right)^{L} \right] \tag{4.3}$$

Thus we obtain the average number of transmissions by substituting $v = (\lambda/(\lambda+\mu))^{L}$

$$\mathrm{E}[N] = \sum_{n=1}^{\infty} np(n) = \sum_{n=1}^{\infty} n(v^{n-1} - v^{n}) = \frac{1}{1 - \left(\frac{\lambda}{\lambda+\mu} \right)^{L}} \tag{4.4}$$

Define the cost of transmitting one AV is a unit cost. Therefore, the normalized transmission overhead per user in its residence at a visited network for both the 3G method and the method of Lin and Chen is

$$J_1(L) = \mathrm{E}[N](L + 2\alpha) \tag{4.5}$$

where 2α is the fixed cost of a round-trip transmission from the home network to the visited network, and L represents the cost of the AV transmission from the home network to the visited network. The 3G standard fixes L=5, whereas L is adaptive in the method of Lin and Chen. The transmission cost of the proposed method I will be

$$J_2 = 1 \times (1 + 2\alpha) \tag{4.6}$$

We are able to estimate the normalized total costs when adjusting the parameter $2\alpha/1$, i.e. the ratio of fixed cost to one AV transmission cost. Fig. 4.1 shows the results of the normalized overhead per user under the three methods with different values of α and ratio λ/μ. A larger α means the fixed transmission has a higher share of the transmission cost; and a larger λ/μ indicates a roaming user makes more frequent authentication requests during its residence in a visited network.

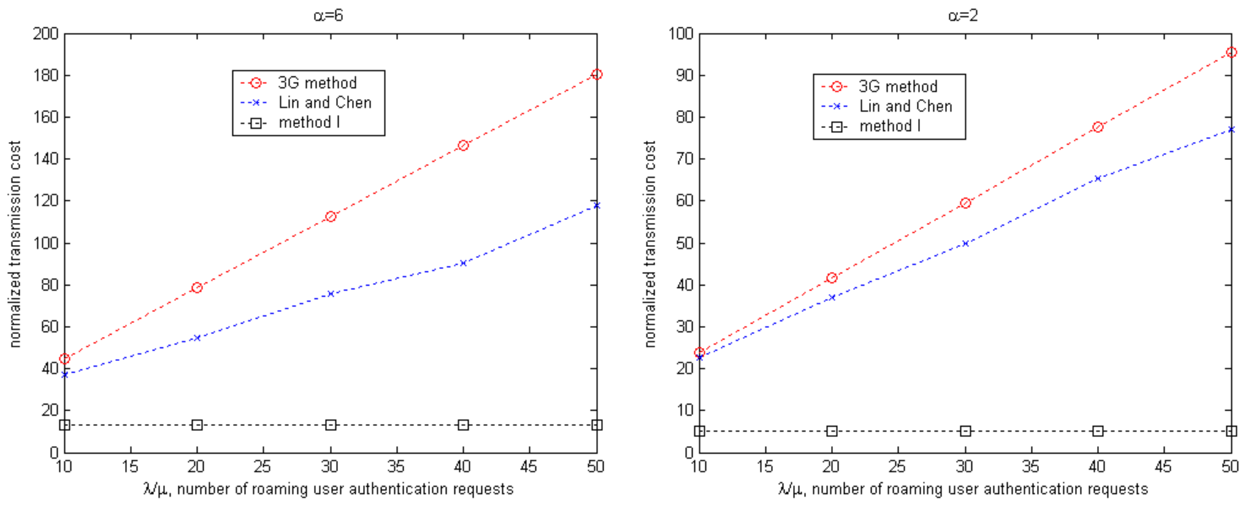


Fig. 4.1: Normalized cost for roaming authentication transmissions between visited and home networks.

For 3G authentication and the method of Lin and Chen, the transmission cost between the home and visited networks increases with the number of authentication requests, whereas in method I the cost curve remains flat. For example, when λ / μ=50 and $\alpha = 6$, the cost of method I is 1/14 of that in 3G authentication and 1/9.5 of that in the method of Lin and Chen. When λ / μ=50 and $\alpha = 2$, the cost of method I is 1/19 of that in 3G authentication and 1/13.6 of that in the method of Lin and Chen.

4.2 Measured Authentication Latency

Method II does not contain the transmission cost between the home and visited networks for authentication; however, the cryptographic computation involved in method II is more expensive than that in method I and the 3G authentication protocol. In this section, we use the measured data to compare the authentication latency of methods I and II and the 3G authentication protocol.

A laptop in the authors' lab within Auburn University sent requests to three SSL or TLS-enabled webmail servers: Auburn University (https://tigermail.auburn.edu/), The Georgia Institute of Technology (https://webmail.mail.gatech.edu/), and The University of Washington at Seattle (https://weblogin.washington.edu/). The laptop, through layer 2 and layer 3 switches, connects to the border router at Auburn University. The rationale for performing this study is contained in the fact that the cost of a single roaming transmission for the 3G authentication protocol and method I is comparable to the SSL session reuse case while the computational cost of method II is comparable to that of the SSL handshake protocol. The laptop made repeated connections within the first five-minute span of every hour from 8:00 until 22:00 (US central time) on February 17, 2004.

The `tcpdump` tool records the time of transmission from the authentication request leaving the laptop Ethernet interface to the web servers' authentication response coming back to the Ethernet interface.

The session reuse case between the laptop and the Auburn webmail server (labeled "AU" in Fig. 4.2) estimates the cost of intra-network roaming authentication. The session reuse case between the laptop and The Georgia Institute of Technology and The University of Washington at Seattle (labeled "GT" and "UW" in Fig. 4.2) approximates the latency of authentication transmission between home and visited networks. The SSL handshake protocol of Auburn University's webmail server (labeled "AU public key" in Fig. 4.2) estimates the latency of method II. Notice that this cost will be a slight underestimate since the implementation of the SSL handshake protocol does not include client signature (computational time of signature generation listed in Table 3.1).

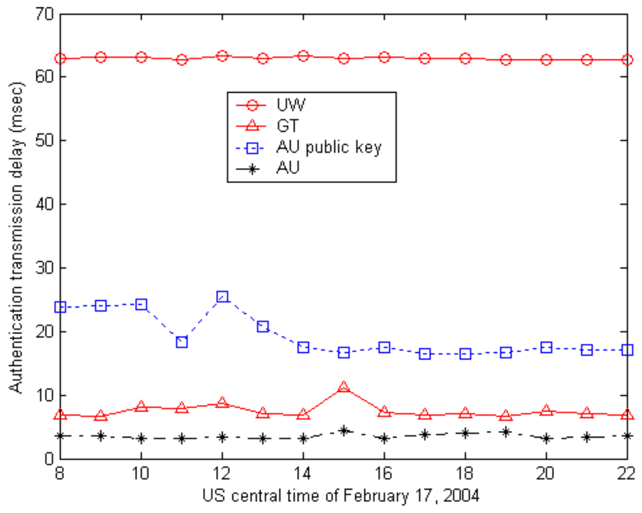


Fig. 4.2. Measured data of authentication transmission delay. "UW" and "GT": transmission between home and visited networks, "AU": intra-network authentication, "AU public key": latency of method II.

The average transmission delays from UW and GT are 62ms and 8 ms, respectively. The data clearly demonstrate the effect of different geographic distances. GT is about 169 km away from Auburn whereas UW is at a distance of about 6400 km. The AU case is roughly 3 ms, which shows that intra-network authentication can be quite cheap because there is no round-trip between the home and visited networks. All in all, these data indicate that for roaming authentication purposes the transmission between the home and visited networks is expensive, especially if they are far apart. Therefore, a roaming authentication protocol should minimize transmission between the home and visited networks.

An average of 20ms is observed for the SSL handshake protocol in Fig. 4.2. If a home network is far away from a visited network (as is the case with the UW), method II will probably have the lowest latency. In all cases, Method I is better than the 3G authentication method in that there is only one transmission between the home and visited networks in method I but possibly many transmissions in the 3G approach.

To further support the fact that transmission cost is a dominant factor in authentication delay, data from the National Laboratory for Applied Network Research at San Diego Supercomputer Center [23] are reported. They are conducting site-to-site round trip time measurements through what is known as the active measurement project (AMP), where round trip time is measured once per minute using the `fping` program. The program sends an ICMP echo packet to a border router of each site and waits for an ICMP reply packet. The authors' school does not participate in the AMP project, so we use data collected from The University of Alabama (225 km away from Auburn) on March 31, 2004. Table 4.1 shows the round trip time from The University of Alabama to various US universities. We observe that the transmission cost between home and visited networks ranges from tens of milliseconds to hundreds of milliseconds. In contrast, for a mobile terminal, the computational cost of method I is in the microsecond range; the computational cost of method II is in the millisecond range (shown in Table 3.1).

TABLE 4.1Round trip time measured using ICMP packets from Univ. of Alabama, Mar. 31, 2004

	Min (ms)	Mean (ms)	Max (ms)	Stddev (ms)
Univ. of Washington, Seattle	62	62.24	78	1.03
Dartmouth College	38.00	43.37	222.00	11.10
Univ. of California, Berkeley	66.00	66.06	68.00	0.24
Columbia Univ.	42.00	43.42	196.00	7.41
Georgia Institute of Technology	5.00	5.19	7.00	0.40
Cornell Univ.	37	38.19	307.00	7.19
Washington State Univ.	79.00	115.27	6637.00	209.07

4.3. Simulation Result on Average Authentication Latency

This section complements the previous sections by studying the transmission cost under the influence of QoS factors for wireless access networks. The case to be investigated is that in which many roaming users come to a visited network and make simultaneous authentication requests. We neglect the case that a user makes frequent authentication requests in a very short time span, since this latter situation only makes the delay worse.

We divide the latency into two parts: the stochastic delay caused by routing device queues and channel contention, and the deterministic delay due to propagation latency and limited bandwidth. Since it is hard to precisely model the various packet paths and the different home networks for roaming users, we use a lumped model (M/G/1 queue) to do a first order estimation of the stochastic delay. For simplicity, we classify the roaming users into three classes depending on the distance to their home networks (short, medium, and long). Each class has its own M/G/1 queue, independent from each other, and the model is shown in Fig. 4.3. Two parameters are of interest: the server service time *x*, and the interarrival time ϕ of roaming users.

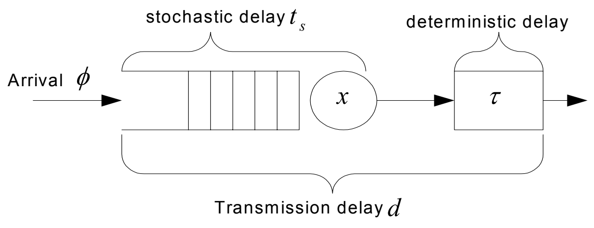


Fig. 4.3: Queueing model for transmission delay

The closed-form expression for the stochastic delay is given first. Using the Pollaczek-Khinchin mean value formula and Little's Law [24], we obtain the average stochastic delay t_s (waiting delay + service delay) in steady-state as

$$t_s = \bar{x} + \frac{\rho\bar{x}(1+\sigma^2/\bar{x}^2)}{2(1-\rho)} \tag{4.7}$$

where the utilization factor $\rho = \omega\bar{x}$; ω is the mean of the interarrival rate ($1/\bar{\phi}$); $\bar{x}$ is the average service time; and σ^2 is the variance of the service time. The transmission delay in the model will be $d = t_s + \tau$, where τ is the deterministic delay, and *d* approximates the latency from a home network to a visited network.

Formula (4.7) indicates the factors that could significantly impact the delay. In practice, we can estimate the above variables through some site-to-site time measurements. For instance, the average service time $\bar{x}$ could be the difference between mean and minimum delays, in which the latter represents the deterministic delay τ in the model.

We simulate the model by an event-driven simulator. In light of (4.7), the factors affecting the delay are ϕ and x. The parameters for simulation are: a queue buffer size of 20 and a simulation time of 15 seconds. The service time is derived from the Gamma distribution with mean $\bar{x}$ and variance σ^2. The arrival of roaming users in the visited network is a Poisson process with mean $1/\bar{\phi}$ or ω.

The first experiment observes the delay t_s under different values of ω. For each ω under the given $\bar{x}/\sigma^2$ pair, we run 10 simulations. Fig. 4.4 reports the mean value of the average delay t_s of these 10 simulations. The trend is that when ω increases t_s will increment accordingly. If ω is above a certain threshold, the increase of t_s will be significant. The second experiment studies the delay under different values of σ^2. It shows that the increase of σ^2 will certainly make the delay worse. Fig. 4.4 illustrates that the transmission delay between the home and visited networks will increase significantly if a large number of roaming users are present or the wireless access network QoS deteriorates. This, in turn, implies that the number of transmissions between the home and visited networks should be minimized for roaming authentication purposes.

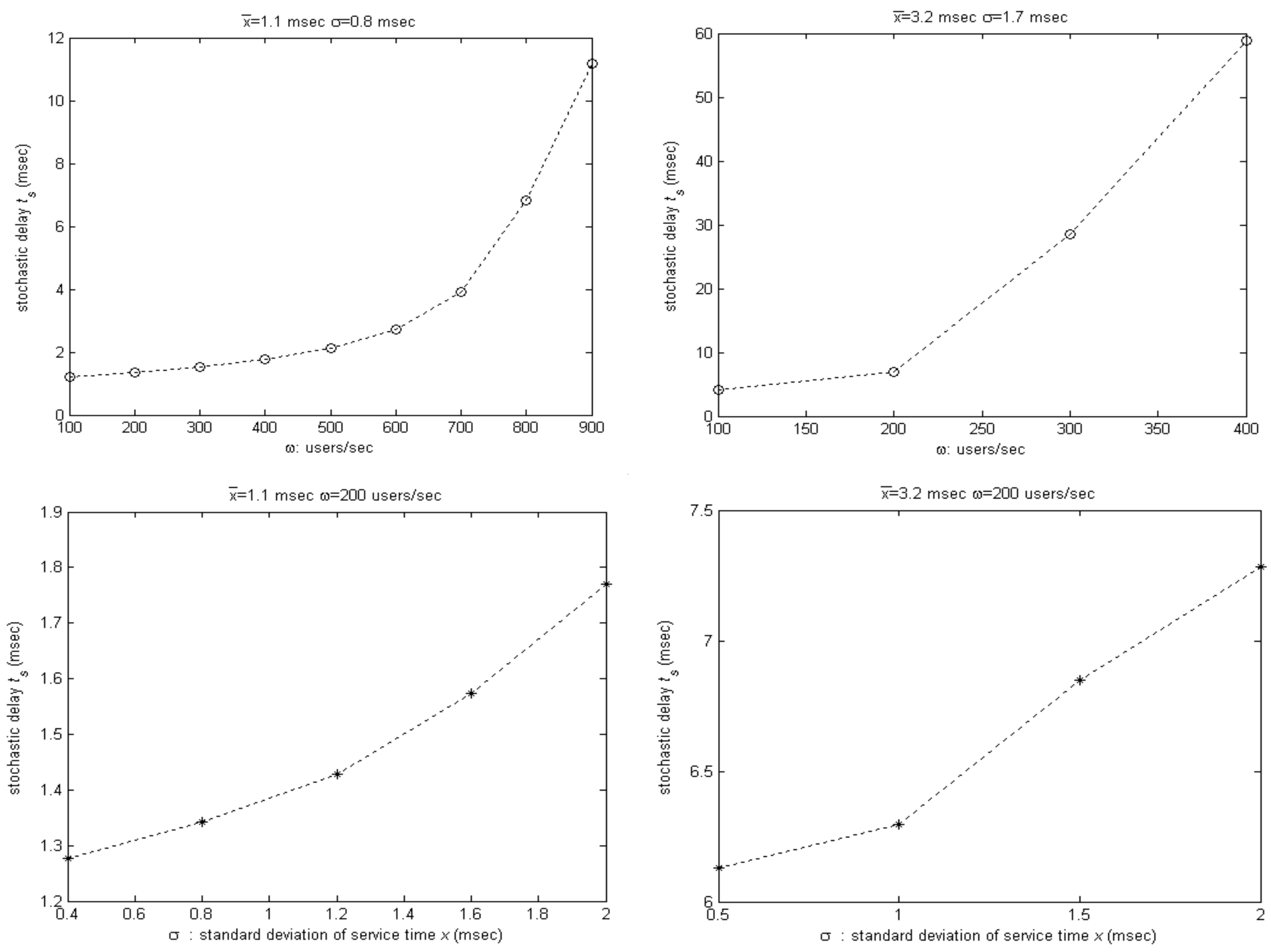


Fig. 4.4: Simulated stochastic delay for transmission cost between home and visited networks

4.4 Energy Consumption Measurement

Energy consumption is an important issue in wireless roaming authentication because of the nature of mobile terminals. The methodology for energy measurement is derived from that reported in [25]. A pocket PC (266 MHz CPU), through an IEEE 802.11b interface, connects to an access point, which in turn connects to an authentication server (Pentium 4, 1.9 GHz CPU, 512 MB memory). The experimental setup is shown in Fig. 4.5.

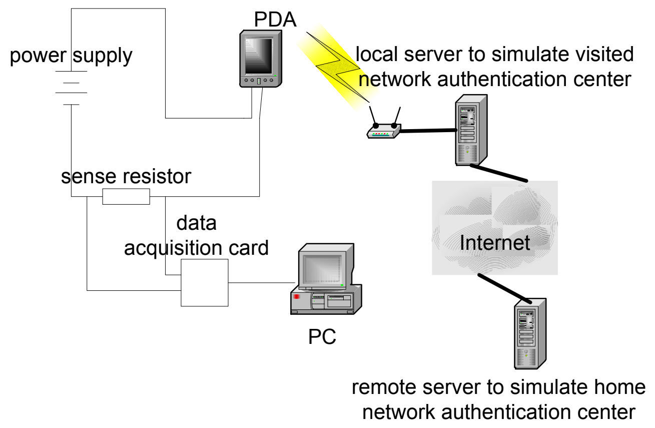

PDA
power supply
local server to simulate visited network authentication center
sense resistor
data acquisition card
Internet
PC
remote server to simulate home network authentication center

Fig. 4.5: Energy measurement testbed.

Three tests were performed: the first is the SSL protocol with client authentication (local authentication server); the second is the implementation of a shared-key protocol (local authentication server); and the third is a shared-key protocol- implementation (remote authentication server, 1280 km away). The first approximates the energy consumption of method II; the second approximates the energy consumption for intra-network roaming authentication; and the third approximates the energy consumption for roaming authentication with home network intervention (3G authentication protocol and the proposed method I). The tests show that, on average, the SSL protocol costs 710 mJ, the second one 67 mJ and the third one 117 mJ. The difference between the last two might be due to the transmission between the remote and local servers that increases the energy consumption of the wireless transceiver in the mobile terminal by elongating the duration of authentication. It is worth pointing out that the results are dependent on the protocol implementations as well. To quantitatively compare the energy consumption of a mobile terminal, we assume that a user makes z authentication requests during its residence in a visited network. Under the 3G method, the energy cost is $117 \times (\lfloor z/5 \rfloor + 1) + 67 \times (z - \lfloor z/5 \rfloor - 1)$ (mJ); under the proposed method II, the energy cost is 710+67(z-1) (mJ); under the proposed method I, the energy cost is 117+67(z-1) (mJ). Note that $\lfloor x \rfloor$ denotes the largest integer that is less than or equal to x. Fig. 4.6 shows the energy cost with respect to the number of authentication requests.

In terms of energy consumption, method I generally has the best performance in that it only incurs one transmission between the home and visited networks. If a roaming user has many authentication requests during residence, both proposed methods have lower energy consumption than the 3G authentication protocol.

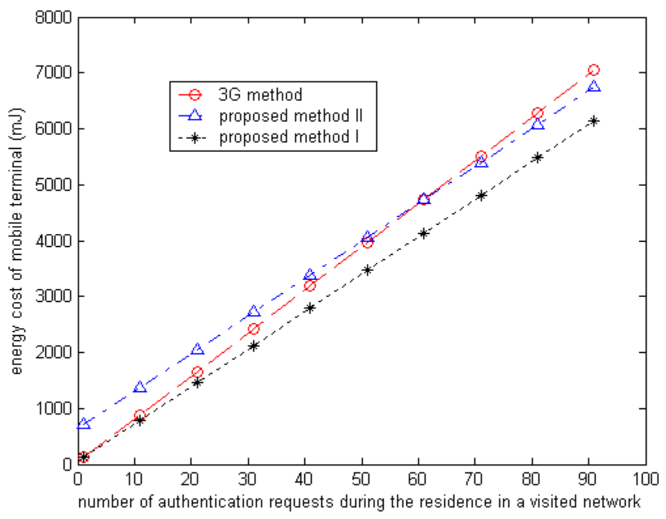


Fig. 4.6: Comparison of energy consumption of the mobile terminal under three authentication methods.

CHAPTER 5

PROPOSED END-TO-END AUTHENTICATION METHOD

In this chapter, we first introduce the cryptographic blocks for the proposed protocol, which comprise nonce generation, hash chain operation, and update hash chain preimage. Then the algorithm details are presented, followed by the security analysis of the proposed protocol.

The basic idea of the new authentication method is the combination of pin, MAC key, and hash chain. We use hash chain to imitate the asymmetric setting so that attackers compromising the server cannot succeed in next authentication attempt. We use pin to protect the user so that intruders subverting the device cannot impersonate the user in next authentication handshake.

5.1 CRYPTOGRAPHIC BLOCKS FOR PROPOSED PROTOCOL

The main primitive used for the protocol is the cryptographic hash function, which is employed for the nonce generation and the hash chain operations. As for the nonce generation, we use the algorithm specified in the NIST digital signature standard [26]. The hash chain in its original paper [27] is the one-way function iterations over a password. Because the password is usually a low-grade secret (pp. 392) [12], the scheme (later evolve into the one-time password method [28]) may be vulnerable to password-guessing attacks. We propose to use hash chain iterations over a high-grade random number such as a 160-bit random number to enhance the security. Moreover, the automatic update of hash chain preimage is presented in this section.

5.1.1. Nonce Generation

We use a hash function over a random number seed to generate nonces. A nonce is a value used no more than once, which serves in authentication protocols to counteract replay and interleaving attacks, to provide uniqueness or timeliness guarantees, and to prevent chosen-text attacks (pp. 397-399) [12]. The unpredictability of a nonce is crucial to the security in an authentication protocol.

The hash function can be understood as a one-way function h such that for each preimage x in the domain of h, it is easy to compute the value $h(x)$; but for essentially all y in the range of h, it is computationally infeasible to find any x such that $y=h(x)$. Pseudorandom numbers are generated by applying the one-way function h to a random number seed. Notice that good pseudorandom numbers are sufficient for most security applications as the substitute of true random numbers that requires a naturally occurring source of randomness (pp. 169-173) [12].

The nonce generation algorithm is listed in the following box. A random number seed, generated by a hardware or software solution [29], is supplied to a protocol participant in the initial configuration. Step 2 is accountable to the generation of a sequence of nonces by updating the random number seed on every occasion. We assume the bit-length of a random number seed is 160 throughout this paper.

```
Algorithm 5.1: Nonce generation
Input:   $seed\_x$   //  random number seed
Output:  $nonce$
Step 1: $w=h(seed\_x)$    // $h(\ )$: hash function
Step 2: $seed\_x=1+seed\_x+w \bmod 2^{160}$   // update the random number seed
Step 3: $nonce=h(w)$
```

Our modification lies in the addition of step 3. Without step 3, an attacker monitoring the wireless channel knows the nonce *w*. So the partial information of *seed_x* (seed of pseudorandom number generator) is leaked to the adversary. If the adversary is able to record many authentication runs of a particular device, then it may facilitate the attacker's cryptanalysis over the nonce seed evolution. Notice that, in the revised random number generation algorithm of the NIST digital signature standard [26], two hash operations are required and the second one is performed over *seed_x*. In step 3, one more hash operation is performed so that it may add difficulty for attackers to learn the evolution of *seed_x*. This addition may have the positive impact to make the protocol more secure. As will be demonstrated in later sections, one more hash operation incurs negligible computational penalty.

5.1.2 Hash Chain Operation

A user begins with a secret *s*. A one-way function is employed to generate the secret sequence: s, $h(s), h(h(s)),..., h^N(s)$. *N* is defined as the maximum number of authentication sessions to be allowed. After *N* times of authentication the system is manually restarted with a new *s*. The private server will store the initial value $u_0 = h^N(s)$. For the *i*th authentication session ($1 \le i \le N$), the user's device computes $u_i = h^{N-i}(s)$ and transmits it to the private server. The private server checks whether the received u_i satisfies $h(u_i) = u_{i-1}$. If so, the private server will save u_i for the next session verification.

The feature of the hash chain is the asymmetry that one side knows the preimage *s* and the other side has the value of the hash iterations over the preimage. If an adversary compromises the server to obtain the stored u_{i-1}, he cannot derive u_i required for the next authentication session because of the one-way function *h*.

5.1.3 Update Hash-Chain-Preimage

From a pragmatic perspective, the iteration times *N* of a hash chain cannot be high for mobile devices, even if a hash operation is lightweight. It is intuitively clear that *N* is less than several hundred under the context of battery-powered wireless devices for protocol efficiency. Furthermore, adopting an optimal value of *N* in terms of some performance metrics is highly desirable in applications. In later section, we will obtain such optimal value of *N* based on the criterion of energy consumption.

Since *N* is small, the above situation leads to the frequent manual configurations for replenishing hash chain seed and value, which significantly increases the IT cost of a corporate. To solve the problem, we propose an automatic update of hash chain preimage. The algorithm is listed in the following box.

Algorithm 5.2: Update one-way hash chain preimage

```
Input:  seed_y  //  random number seed
Output: s   //preimage used for hash chain
Step 1: s=h(seed_y)  // h( ): hash function
Step 2: seed_y=1+seed_y+s mod $2^{160}$
```

We elaborate the difference between the update of the preimage and the hash chain iteration. The update of preimage is the hash operation over *seed_y* to generate *s*, whereas the hash chain iteration is the hash operation over *s*. These two processes will generate different sequences because step 2 in Algorithm 5.2 decouples *seed_y* with *s*. The remaining issue is to let the server know the new hash chain value $h^N(s)$.

5.2 Proposed Authentication Protocol

First, we discuss the authentication system initialization. Second, the message flows of the authentication protocol are described. Third, we present a way to piggyback the hash chain value update message on the regular authentication handshake. Our contributions include: a secure and efficient combination of hash chain and shared key techniques, complementary MACs for detecting online password-guessing attacks, and automatic update of hash chain value.

5.2.1. System Initialization

A user shares a pin and a cryptographic key with the private server. The wireless device will be provided with two random number seeds: one is for the nonce generation, and the other for the hash chain preimage. The device then generates the first hash chain preimage s_0 and the initial hash chain value $h^N(s_0)$. The private server stores $h^N(s_0)$.

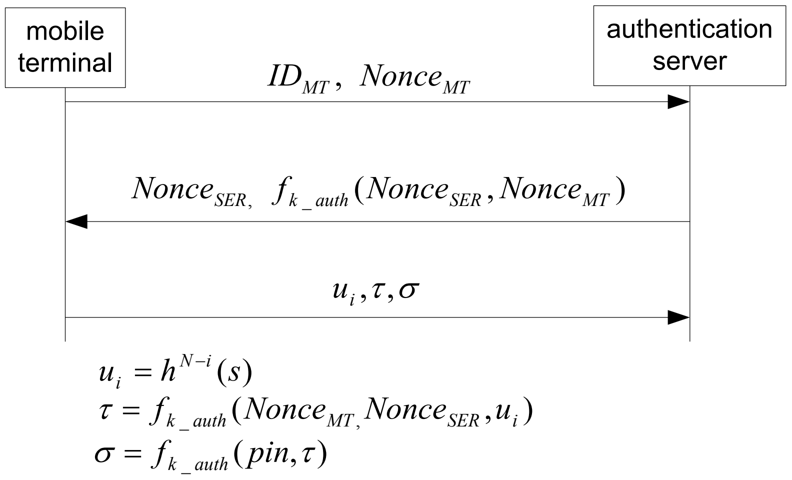
5.2.2. Message Flows of Authentication Handshake

Fig. 5.1 depicts the message flows of the authentication protocol, where the cryptographic core is described. Message authentication code (MAC) is another cryptographic primitive that is employed in the protocol. MAC can be viewed as hash functions, which take two inputs (a message and a secret key) and produce a fixed-size output, so it is infeasible to produce the same output without knowledge of the key.

Flow 1: The mobile terminal will send its identifier ID_{MT} and the nonce $Nonce_{MT}$ to the server. Then the server will use ID_{MT} to fetch the private credentials of the user (authentication key *k_auth* and the pin).

Flow 2: The server will generate a nonce $Nonce_{SER}$ and the keyed MAC $f_{k\_auth}(Nonce_{SER}, Nonce_{MT})$, and then send its nonce and the MAC to the device. Because the device shares *k_auth* and knows the two nonces, it is able to check the received MAC. If the check succeeds, the device will continue the authentication handshake. Otherwise the device will terminate the session immediately.

Flow 3: The device will generate the hash chain value $u_i = h^{N-i}(s)$ for the *i*th authentication session. Then it produces the first MAC $\tau = f_{k\_auth}(Nonce_{MT}, Nonce_{SER}, u_i)$ and the second MAC $\sigma = f_{k\_auth}(pin, \tau)$. Notice that the pin is never sent out onto the channel. Upon receiving flow 3, the server will perform the hash chain verification on $h^{N-i}(s)$, then the first MAC τ, and the second MAC σ. If any of them fails, the handshake session will be dropped immediately.


mobile terminal
authentication server
ID_{MT}, $Nonce_{MT}$
$Nonce_{SER}$, $f_{k\_auth}(Nonce_{SER}, Nonce_{MT})$
u_i, τ, σ
$u_i = h^{N-i}(s)$
$\tau = f_{k\_auth}(Nonce_{MT}, Nonce_{SER}, u_i)$
$\sigma = f_{k\_auth}(pin, \tau)$
Fig. 5.1: Message flows for the *i*th authentication session in the proposed authentication protocol.

5.2.3 Piggyback Hash Chain Value Update

Our protocol prescribes *N*-1 authentication sessions within one hash chain cycle of *N*. As described in Section III.C, there exists the conflict between the small value of *N* and the larger number of possible authentication sessions. Algorithm 2 in Section III.C solves the hash chain preimage update at the device side. The other half problem is how to let the server know the new hash chain value generated by the new preimage. The design consideration is that manual registering a new hash chain value to a server is costly.

We solve this problem by automatically updating the hash chain value at the (*N*-1)st authentication attempt. The new hash chain value $h^N(s_{j+1})$ is piggybacked in flow 3, as shown by

$$h^{N-(N-1)}(s_j), h^N(s_{j+1}), \tau', f_{k\_auth}(pin, \tau') \tag{5.1}$$

$$\tau' = f_{k\_auth}(Nonce_{MT}, Nonce_{SER}, h^{N-(N-1)}(s_j), h^N(s_{j+1})) \text{ and } s_{j+1} = h(seed\_y). \tag{5.2}$$

The input of τ' includes the new hash chain value $h^N(s_{j+1})$. The authentication key *k_auth* is crucial for the authenticity of the update. Upon checking τ', the server is able to discern whether the new hash chain value $h^N(s_{j+1})$ is intact and from the legitimate wireless device. Flow 3 in the (*N*-1)st authentication session achieves two purposes: authentication for this session and updating the hash chain value for the next (*N*-1) authentication sessions.

5.2.4. Mathematical Expression of Hash Chain Process

From the results of Subsection C, we give a more formal expression of the hash chain u_i in flow 3 of the proposed protocol. u_i can be written as

$$u_i = \begin{cases} h^{N-i}(s_0) & i = 1, \cdots, N-1 \\ h^{2N-1-i}(s_1) & i = N, \cdots, 2N-1 \\ h^{3N-1-i}(s_2) & i = 2N, \cdots, 3N-1 \\ \cdots & \cdots \end{cases} \tag{5.3}$$

where i is the index of authentication sessions and $\{s_m \mid m = 0, 1, \cdots\}$ are the hash chain preimage discussed in Section III.C. Notice that the expression can be simplified if we consider *i* wraps around *N*-1. In this way, it is $u_i = h^{N-i}(s_m)$.

5.3 Security Analysis

First, we explain why *N*-1 authentication sessions are prescribed in the hash chain cycle with the upper limit of *N*. Second, we show that EAP-MD5 is susceptible in the traditional adversary model and EAP-TLS is vulnerable in the strong adversary model. Third, we indicate that proposed protocol is secure in the strong adversary model.

5.3.1 Hash Chain Cycle

Algorithm 1 of nonce generation takes into account hiding the random number seed for the one-way function. The same rationale extends to the hash chain cycle, which has the upper limit of *N* authentication sessions. If the device only performs *N*-1 authentication sessions, according to Algorithm 2, the hash chain seed *seed_y* will not be seen by the adversary who monitors the communication. In this manner, the security will be enhanced.

5.3.2 Security Analysis of EAP-MD5 and EAP-TLS

In EAP-MD5, an adversary with the knowledge of MD5 algorithm can do an exhaustive search on the set of possible passwords by plugging the possible values into the one-way function and then comparing the results with the captured handshake message. In general, a password has low entropy, so the space of possible passwords is relatively small. As a result, a determined adversary can find the correct password with a very high probability. The vulnerability lies in the small space of the input to the one-way function.

For EAP-TLS, if a hacker compromises the device, all the stored secrets are available to the adversary. Since EAP-TLS uses only the secrets stored in the device, the attacker can impersonate the legitimate user and succeed in next authentication attempt after compromising the device.

5.3.3 Intrusion Detection of Proposed Protocol (Complementary MACs)

For the proposed protocol, if an adversary compromises the device, he does not know the pin. If the attacker initiates the authentication request to the server, the first MAC τ will be correct since the code only requires the secrets stored in the device. Nevertheless, the second MAC σ will fail because the correct MAC requires the pin. The possible choice for the attacker is to perform an online password-guessing attack, which is easily detectable by the server since too many authentication requests originate from the same device in a short time span. Note that an attacker without compromising the device cannot get the first MAC τ correct because of the enormous space of *k_auth* (high-grade secret). As a result, when the server detects that the first MAC is correct and the second one is wrong, the server can infer that the device has been compromised. Since the technique involves two MACs that need different secrets, we name it complementary MACs.

5.3.4 Intrusion Resilience of Proposed Protocol

If an adversary compromises the server, he can learn the pin and *k_auth*. Since the hash chain uses a one-way function, the attacker cannot derive the hash chain value for the next authentication attempt.

The trust relationship of sharing a pin and a secret is feasible to implement for the corporate computing applications. The proposed protocol combines the usage of a pin, a key, and a hash chain in an efficient manner to achieve a strong security level. If the pin is not used in the protocol, when an adversary compromises the device, he can succeed in impersonating the user. If the MAC key is not used in the protocol, the update of hash chain value might be tampered by the adversary. So the server and the device will be out of synchronization, which causes denial of service to legitimate users. If the hash chain is not used in the protocol, the adversary compromising the server can succeed in impersonating the legitimate user in the next authentication attempt because the adversary now learns the pin and MAC key of the user.

CHAPTER 6

PERFORMANCE ANALYSIS OF PROPOSED END-TO-END AUTHENTICATION PROTOCOL

In this chapter, we first evaluate the authentication latency of the proposed protocol through the analysis, simulation, and implementation. Then we thoroughly discuss some energy optimization techniques.

6.1 AUTHENTICATION LATENCY OF PROPOSED PROTOCOL

The communicational and computational cost of an authentication protocol boils down to the authentication latency. In the study, the basic computational unit is one hash function with the input of one standard block length. For a fair comparison, we count the raw computational and communicational cost and do not consider the possible implementation improvement. We first calculate the computational cost of one authentication handshake execution at the device side. This is of particular interest due to the relatively limited resources of wireless devices. Then the computational cost at the server side is reported. The communicational cost and the secret storage are also discussed. Finally, the implementation results are provided to support the analytical results.

6.1.1 Computational Cost at Mobile Terminal

We choose SHA-1 [30] as the hash function and HMAC-SHA1 [20] as the MAC method since they are standardized and computationally efficient. One version of HMAC implementation requires only two hash operations assuming the input of one standard block length [20]. For the fixed cost, two hash operations are required to generate the nonce in flow 1 (Fig. 5.1) according to Algorithm 1 in Section 5.1; two hash operations to check the HMAC in flow 2, four hash operations are required to generate the two HMACs in flow 3. Thus, the total fixed cost is 8 hash operations. The variable cost of a protocol execution is caused by the hash chain in flow 3. During one hash chain cycle, calculating the initial hash chain value, $h^N(s_m)$, requires N hash operations; performing hashing chain from 1st to the (N-1)st authentication sessions requires the number of hash operations N-1+(N-2)+…+1. In addition, according to Algorithm 2, one hash operation for updating the hash chain preimage s is required. Thus the average hash operations for one execution of the authentication protocol at the device side is

$$C_1(N) = 8 + \frac{N + N - 1 + \cdots 1 + 1}{N - 1} = 8 + \frac{1 + N/2 + N^2/2}{N - 1} \qquad (6.1)$$

If we consider the potential implementation improvement such as saving the intermediate hash chain results in storage, (6.1) still holds and can be regarded as an upper bound.

6.1.2 Computational Cost at Server Side

The server side incurs only a fixed cost. Two hash operations are required to generate nonce and two hash operations to generate the HMAC in flow 2. One hash operation is required to verify the hash chain, and four hash operations are needed to verify the two MACs in flow 3. Thus the total is nine hash operations for one execution of the authentication protocol at the server side.

6.1.3 Message Transmission Cost

The bit-length of hash output is defined as l_h and the bit-length of identifier is defined as l_{id}. So the fixed cost of message transmission is $l_{id} + 6l_h$. The variable cost, due to the piggyback hash chain update in (1), averages $l_h/(N-1)$. For one authentication execution, the device transmits the message with average length

$$L_1(N) = 4l_h + l_{id} + \frac{l_h}{N-1} \tag{6.2}$$

The server transmits the message with length $2l_h$.

6.1.4 Secrete Storage

For the device, the secret storage consists of the shared MAC key *k_auth* and the two random number seeds *seed_x* and *seed_y*. Since SHA-1 is used, the above three secrets costs 60 bytes. The server will store its random number seed (20 bytes), *k_auth* (20 bytes per user), and pin (i.e., 4 bytes per user).

6.1.5 Comparison with EAP-MD5 and EAP-TLS

For EAP-MD5, there are four hash operations in total at either side (two for generating nonces according to Algorithm 1, one for verifying the other side's response on the nonce, and one for generating its own response). In EAP-TLS, the device performs one signature generation, the public key verification, and one public key encryption. The server performs one public key decryption, the public key verification, and one signature verification.

It is noticeable that the message length of the public key protocol is comparatively high. For example, the public key certificate length usually ranges from 500 bytes to 1000 bytes, which significantly surpasses that of EAP-MD5 or the proposed protocol, given that the output of SHA-1 is 20 bytes in length.

6.1.6 Implementation Result

The authentication latency can be roughly divided into three parts: operating system overhead, cryptographic operation, and transmission cost. In Phase I, we discuss the implementation of the cryptographic primitives. We implement these primitives based on the well-regarded Crypto++ library [22] in the two platforms: a PDA with an Intel Xscale 400 MHz CPU, 64MB SDRAM, and 32MB Flash ROM; a laptop with a 2.1 GHz Intel Pentium 4 CPU and 256 MB RAM. The primitives include SHA-1, HMAC-SHA-1, RSA [21], and ECDSA (pp. 184-186) [31]. Tables 6.1-2 list the timing results that clearly convey the public key operation is much slower (1000 folds on average) than the hash function execution.

Table 6.1. Computational cost of crypto operations on a 2.1 GHz Intel Pentium 4 CPU
(RSA: 1024 bits, HMAC-SHA-1: one input block, ECDSA: 163-bits, SHA-1: one input block)

	Computational time (ms)
RSA encryption (e=17)	0.19
RSA decryption	4.65
RSA signature generation	4.65
RSA signature verification (e=17)	0.19
HMAC	0.002
ECDSA signature generation	1.46
ECDSA signature verification	1.95
SHA-1	0.001

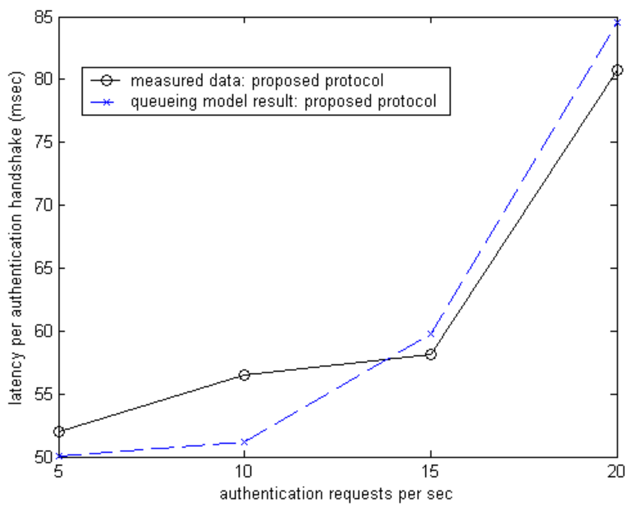
Table 6.2. Computational cost of crypto operations on a 400MHz Intel Xscale CPU
(RSA: 1024 bits, HMAC-SHA-1: one input block, ECDSA: 163-bits, SHA-1: one input block)

	Computational time (ms)
RSA encryption (*e*=17)	1.42
RSA decryption	33.3
RSA signature generation	33.3
RSA signature verification (*e*=17)	1.42
HMAC	0.015
ECDSA signature generation	11.6
ECDSA signature verification	17.2
SHA-1	0.009

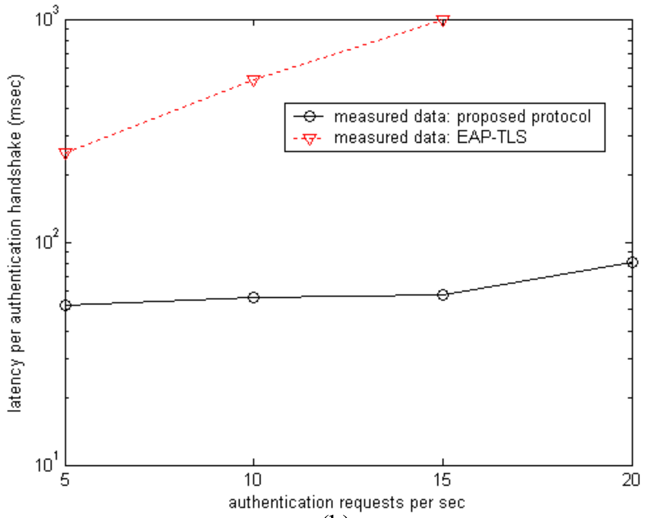
In Phase II, we run the prototype implementations of EAP-TLS and the proposed protocol (*N*=10) on the client-server platform, respectively. The client device is the PDA with the integrated WLAN, and it connects the server machine through an access point. The server's configuration is 2.1 GHz Intel Pentium 4 CPU and 1 GB RAM. In this phase, the server's load is very lightweight. We send one authentication request per second to measure the latency of the authentication handshake. The `tcpdump` tool (a packet capture program) is used to record the timing of packet arrivals and departures, which then derives the protocol timing. The average results of 20 repetitive experiments are 176.6 msec for EAP-TLS and 35.1 msec for the proposed protocol. The experimental data suggest that the performance of proposed protocol is much more efficient than the public key based schemes in terms of authentication time latency (5 folds faster).

The experiment in Phase III is similar to that of Phase II except that we increase the load of the server, which is closer to the reality. We define λ as the number of authentication requests per second. The interval time of two consecutive authentication requests is $1/\lambda$ sec in our settings. In the experiment, λ takes on values of 5, 10, 15, and

20, respectively. We then measure the average latency of one authentication request. The result is shown in the solid line of Fig. 6.1.



(a)



(b)

Fig. 6.1. Authentication latency of the proposed protocol and EAP-TLS. Test platform: PDA with a 400MHz Intel Xscale CPU. (a) simulated and measured latency of the proposed protocol (b) latency comparison between the proposed protocol and EAP-TLS.

We propose a simple queueing model to fit the measured data. The model contains one queue with the arrival process of authentication requests. The service time μ for the queue is

$$\mu = \mu_{\text{fix}} + \mu_{\text{var}} \tag{6.3}$$

where μ_{fix} is the fixed time and μ_{var} is the perturbation. We assume that average authentication latency measured in Phase II is μ_{fix} because the server is under very low load in that case. Thus, we have μ_{fix}=35.1 msec. The variable part is due to the multitasking nature of computer operating system. In our example, we assume that μ_{var} observes the gamma distribution with mean 15 msec and standard deviation 25 msec--the parameters are obtained by experiments. We have 20 repetitive runs of a simulation under each value of λ. The result is shown in the dashed line of Fig. 6.1. It can be seen that the simulation results and the measured data are quite close. The performance of TLS has been quantitatively investigated in [32]. In this paper, we reproduce the experiment under our settings. Fig. 6.1 shows the result in the dotted line. It is clear that the proposed protocol has much better performance than that of public key schemes when under multiple authentication requests.

6.2 ENERGY CONSUMPTION OPTIMIZATION

The choice of N entails the tradeoff between communicational and computational cost. When N increases, the communicational cost will decrease, as indicated by (5), though the computational cost increases according to (4). We try to find a value of N which minimizes the energy consumption since this is an important factor for authentication protocol design in the context of wireless security.

6.2.1 Analytic Results

The framework of a battery recharge cycle is assumed first. For example, it can be one day or a few days depending on the environment. In one recharge cycle, multiple cycles of the hash chain might be performed--one hash chain cycle contains N-1 authentication sessions. The cost of energy consumption at the device associated with the choice of N is mainly due to three parts: hash operations, radio transmission of hash chain value update, and the hash chain preimage update (writing to the EEPROM or any other form of non-volatile memory).

We assume that the authentication requests satisfy a Poisson distribution with mean φ during one recharge cycle. Then there are on average φ/(N-1) hash chain cycles in one recharge cycle. We further define the energy consumption of one hash operation is α; energy consumption of writing to the EEPROM for the new hash chain preimage is β; energy consumption of transmitting the hash chain value update is γ. *P*(*N*) represents the variable cost of energy consumption at the device for one recharge cycle, and minimizing *P*(*N*) will result in the least energy consumption for the wireless device since the fixed cost is independent with *N*. Thus, we have

$$P(N) = \frac{\varphi}{N-1}[\alpha(\frac{N^2+N+2}{2}) + \beta + \gamma] \tag{6.4}$$

To minimize *P*(*N*), we set

$$\frac{d}{dN}P(N) = \frac{\varphi}{(N-1)^2}(\alpha\frac{N^2}{2} - \alpha N - \frac{3\alpha}{2} - \beta - \gamma) = 0 \tag{6.5}$$

Simplifying (8), we get

$$N^2 - 2N - (3 + 2\frac{\beta+\gamma}{\alpha}) = 0 \tag{6.6}$$

Then we finally obtain

$$N = 1 + \frac{\sqrt{16 + \frac{8(\beta+\gamma)}{\alpha}}}{2} = 1 + 2\sqrt{1 + \frac{\beta+\gamma}{2\alpha}} \tag{6.7}$$

Equation (6.7) is in accordance with one's intuition. If the transmission is expensive (large γ), we expect to increase *N* to reduce the communicational overhead. From the controlled experiments in our test bed, we obtain that α=0.014 mJ (one hash operation), β=0.39 mJ (write new hash chain preimage to EEPROM), and γ=0.15 mJ (piggyback hash chain value update). Substituting these values into (8), we then get *N*=10, which confirms that *N* should not be very high for the efficiency purpose. Although α, β, and γ are device-dependent values, we conjecture that the optimum value of *N* ranges from 10 to 30 for generic PDAs.

6.2.2 Implementation Results

The networking setting is the same as the one in the authentication latency experiment. We use the methodology for energy consumption measurement from the one reported in [25]. The results are obtained by measuring the current drawn from the power supply (connect a sense resistor in series between the PDA and the external energy supply) during authentication handshake. The experimental results show that EAP-TLS protocol costs 710 mJ while the proposed protocol 78 mJ (*N*=10). The data suggest that the new protocol is power efficient and suitable for the wireless devices to perform frequent authentication handshakes. In addition, we adjust the values of *N* from 6 to 15 during the experiment. Due to the limitation of our experiments, we did not test the cases where *N* is greater than 15. Nevertheless, the data could support the result of (10)-- *N*=10 is the optimum value under our test platform--as shown in Fig. 6.2.

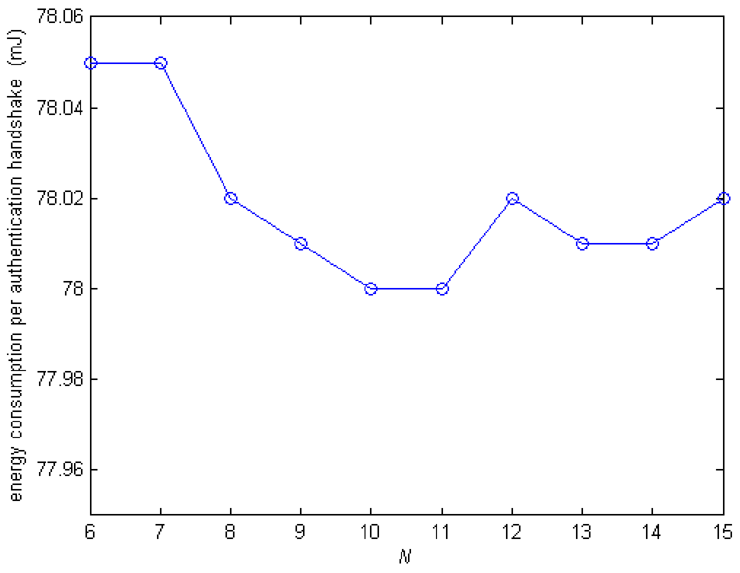


Fig. 6.2: Energy consumption of the proposed protocol with respect to *N*. Test platform: PDA with a 400MHz Intel Xscale CPU.

CHAPTER 7

SPATIAL-TEMPORAL ATTRIBUTE TO END-TO-END AUHTNENTICATION

In chapter 6, we have proposed a very efficient end-to-end authentication protocol that is based on symmetric key cryptography. In this chapter, we give a new protocol that is more heavy weight than the one in chapter 6; however, the niche of this new protocol is to offer the spatial and temporal attribute that provides strong security and significantly helps computer forensics.

7.1 Motivation for Spatial-Temporal Attributes in End-to-End Authentication

Two adversarial phenomena concerning client/server authentication are under development. One phenomenon deals with outside adversaries, where the problem of mobile device theft is on the rise. An adversary is able to employ the stolen device to impersonate a legitimate user. In addition, cases of authentication server compromise are increasing dramatically. Thus, adversaries exploit the operating system vulnerabilities to corrupt servers, obtain the secrets, and then impersonate a legitimate user.

Another phenomenon deals with dishonest insiders. Both server and client need to protect their own interests if any dispute occurs after the transaction. For example, a dishonest legitimate user might deny her login records while an innocent legitimate user might be accused as a result of forged login records. Two recent papers [33], [34] have challenged the assumption that the two sides, involved in authentication, have an honest collaborative relationship.

In the face of these new challenges, some existing authentication protocols, i.e. IPsec [35] and authenticated key exchange [36], are not effective for some applications involving high-value data and strong adversary presence, because these solutions are often based on the assumption that the cryptographic key is not used by an unauthorized attacker. If a compromise occurs, between the time when the key is actually leaked to an adversary and the leak is detected, the adversary becomes a “legitimate” user. Furthermore, after detection, both server and user have to revoke their old keys through out of band approaches, thus increasing the administrative burden and cost. In addition, existing solutions rarely address the issue regarding dishonest insiders.

In this dissertation, we propose a new password-based authentication protocol with time and location attributes. The two underlying assumptions are as follows:

Assumption 1: Adversarial compromise is a real threat. If an adversary compromises a protocol participant, the secret owned by the participant will be available to the adversary.

Assumption 2: The relationship between a server and a user is not absolutely collaborative. Their interests are different, and the potential for adversarial compromise will complicate their relationship.

Notice that the efficient protocol in Chapter 6 can accommodate assumption 1, but it is the assumption 2 that distinguishes this protocol from the one in Chapter 6. Three main technical merits of the proposed protocol are the following:

- When an adversary compromises either the server or the device, she cannot do an offline dictionary attack to recover the password. Thus she cannot impersonate the user. Even if a compromise occurs, no keys need to be revoked. The security is automatically self-recoverable.
- The crypto scheme is constructed such that time and location attributes are an inherent part of the scheme, and thus a server in the authentication process will have more confidence that a user at a remote site is genuine, i.e. if a server knows a priori the location of a user, the server can verify whether the location attribute passed from the protocol execution matches the expected one.
- The attributes have a strong non-repudiation property, even in the compromise situation. The signature containing the attributes consists of one message flow of the protocol, which could be placed in log files to provide neutral evidence in case of disputes or forensics.

The first step of our solution exploits the fact that, in many cases, a mobile device, e.g. laptop or PDA, is employed for remote access to a server. Therefore, we assume an asymmetric crypto model in which client *C* has a password; Client *C*'s mobile device *Dev* has a secret; and server *S* holds a one-way function of the password and the device secret. This model paves a way to prevent an adversary from doing offline dictionary attacks to recover the password if she compromises either the device or the server.

The second step of our solution involves the selection of a suitable crypto scheme for realizing the above asymmetric model. Server, without knowing the device secret, should be convinced that client's device possesses the secret. Another criterion to counter against the compromise is that the scheme should have a time-dependent property.

Consequently, a forward secure signature [37], based on the zero-knowledge paradigm, is employed in the protocol. The secret key of the signature will evolve with time while the corresponding public key remains the same. The superiority of this scheme results from the fact that even if the secrets of current time are leaked, the past secrets cannot be recovered. Another advantage is storage efficiency, since the server only stores the fixed public key.

The third step is to embed location when constructing the signature. A noticeable phenomenon in the mobile/wireless network infrastructure is the existence of a foreign agent or visited network. The visited network, through an access point or base station, provides the network service to users. When users do online banking or access their private organization server, the packet traffic passes through the visited network. Since the range of the visited network covers a user's mobile device, we envision that the visited network provides the location attribute to assist in client/server authentication. To achieve this goal, a visited network has its own time-dependent private key and registers the fixed public key with a server. Then a visited network can simply superpose its signature onto that of the user. This superposition of signatures by mobile user and foreign agents is achieved using only one modular multiplication. The final signature in the authentication protocol can be placed in the log files, which are useful for intrusion detection and forensics because of the strong non-repudiation property. In this case, a user's login activity is framed at a particular time plus a particular location that he cannot deny at later time. One point we elaborate is that the proposed protocol also ensures a user's location information is not leaked to any other party except the user himself and the server.

A big picture of our protocol can be viewed in the following manner. Based on some domain parameters, a legitimate user chooses a password and obtains a key stored in his mobile device. The password verifier, a one-way function of the password and the key, is stored in the server. The organization has contracts with service providers, or it may deploy its own foreign networks. The verifier for the time-dependent secret key of a visitor network and its corresponding geographical location are also stored in the server.

With respect to efficiency, our protocol assumes the minimal hardware requirement. No smart card or other wearable tokens are employed. The computing interface is a familiar one in which a user simply enters their password. The device key stored in his computing platforms is totally transparent to the user.

7.2 INTRUSION RESILIENT AUTHENTICATION PROTOCOL

7.2.1 System Initial Setup

We denote the principals of the proposed protocol as follows: client *C*, client's device *Dev*, Foreign agent *FA*, and server *S*. For clarity of presentation, the protocol is described in a one-server-one-*FA*-one-client manner.

- *Publicly Known Values to All Parties*

We require a designated object, trusted by all parties, to generate two safe prime numbers ($p_1 = 2q_1 + 1, p_2 = 2q_2 + 1$), then an RSA modulus $n = p_1 \times p_2$ is derived (*l*, the number of bits of *n*, is the security parameter, e.g. 1024 bits). Then the object immediately destroys p_1, p_2, q_1 and q_2. All the parties follow a predefined setup of *T* time periods. The object generates a series of prime numbers $e_1, e_2, ..., e_T$ (l_1 is the bit length, e.g. l_1=160). The subscripts correspond to the time periods 1, 2, …, *T*, respectively. In this case, the *e*'s are automatically relatively prime to $\phi(n)$ (the number of integers less than and relatively prime to *n*) since the *e*'s are smaller than q_1 and q_2. It also generates a prime number *p* (l_2 is the bit length, e.g. 1024 bits) and its generator *g* of a big subgroup.

$e_1, e_2, ..., e_T$, *n*, *p* and *g* are the publicly known parameters or domain parameters shared by the whole system, though *p* and *g* are not needed by *FA*. At this point, there is no trapdoor information (p_1, p_2, q_1 and q_2) stored within the system.

- *Password, Device key and Password Verifier*

Every client chooses a fixed password *pw* and their device chooses a random number *v* from Z_n^* (the set of positive integers less than and relatively prime to *n*) as the device key. Then the device calculates a password verifier *J* as

$$J = \frac{1}{[vf(pw)]^{e_1 e_2 \cdots e_T}} \bmod n, \tag{7.1}$$

where *f* is a one-way function and its output is also in Z_n^*. *J* will be stored only in a server, e.g. in the shadow file as a password verifier. A server will use *J* to verify a particular user. On the other hand, because only the server and this client's device know *J*, *J* can be used by the client device to authenticate this server as well. It is worth noting that the client device never statically stores *J* in order to thwart adversarial offline password-guessing attacks.

v is the secret value of the device known only to the device, and changes with every time period. The device will update *v* at the beginning of every time period and then delete the previous one. For example, we let $v_1 = v$ for the 1st time period. The formula for updating v is

$$v_i = v_{i-1}^{e_{i-1}} \bmod n,\ i \geq 2. \tag{7.2}$$

At time period 2, the device derives v_2 and then immediately deletes v_1.

- *FA Signature Private Key and its Verifier*

Similarly, a *FA* will choose a random number u from Z_n^* and then enroll its public key J_{FA} (publicly known) with the corresponding geographical location to a server,

$$J_{FA} = \frac{1}{u^{e_1 e_2 \cdots e_T}} \bmod n. \tag{7.3}$$

And its private key will evolve as

$$u_i = u_{i-1}^{e_{i-1}} \bmod n,\ i \geq 2. \tag{7.4}$$

7.2.2 Protocol Flows

The basic form of our protocol is depicted in Fig. 7.1. For clarity of presentation, we will employ the compact notations such as g^X instead of $g^X \bmod p$. In Fig. 7.1, we explicitly mention the time period i since the crypto primitive that we use has a time dependent property.

Each time a user wants to access a server, the protocol is executed. A user can log in and out multiple times in one period, where the number of total time periods is T. While the signature key formed by the password and time-dependent device key is the same in one period, it must be deleted and regenerated at the device for every login attempt. In other words, a user has to type in his password for each login attempt.

There are basically four message flows in the protocol. We also propose that flow 3 be written to log files as a login record. Notice that the messages in Fig. 7.1 construct the essential part of the protocol. For the implementation requirements, more information may be needed.

- *Flow 1 (Client to Server)*

Flow 1 is an authentication request, which is sent in clear text. The identity of client *C* is included in flow 1. The identifier is sufficient for flow 1 as the result of the "hello" nature of the first message in every authentication protocol. Location privacy can be achieved by changing the identifier or using pseudonym every login. The highlight of this approach is that *FA* will check to see if the request is from its vicinity. If a user is not local, the *FA* will drop the session. Otherwise, the *FA* appends its identifier to the message and forwards the message to the server.

- *Flow 2 (Server to Client)*

Flow 2 is also sent in clear text. The server will pick up a nonce *Ns* to challenge the user. *FA* will append its public key J_{FA}, which later will be signed by the user. J_{FA} can be understood as a location attribute since different *FA*s will have different J_{FA}. For the signature operation, the FA will generate a random number *k* from Z_n^*.

- *Flow 3 (Client to Server)*

Flow 3 is also sent in clear text. After receiving the challenge from the server, a user will input a password *pw*. The device, aware of the time period *i*, picks up v_i and calculates the signature key SK_i for time period *i*

$$SK_i = v_i^{e_{i+1}e_{i+2}\cdots e_T} \times f(pw)^{e_1 e_2 \cdots e_{i-1} e_{i+1} \cdots e_T} \bmod n \qquad (7.5)$$

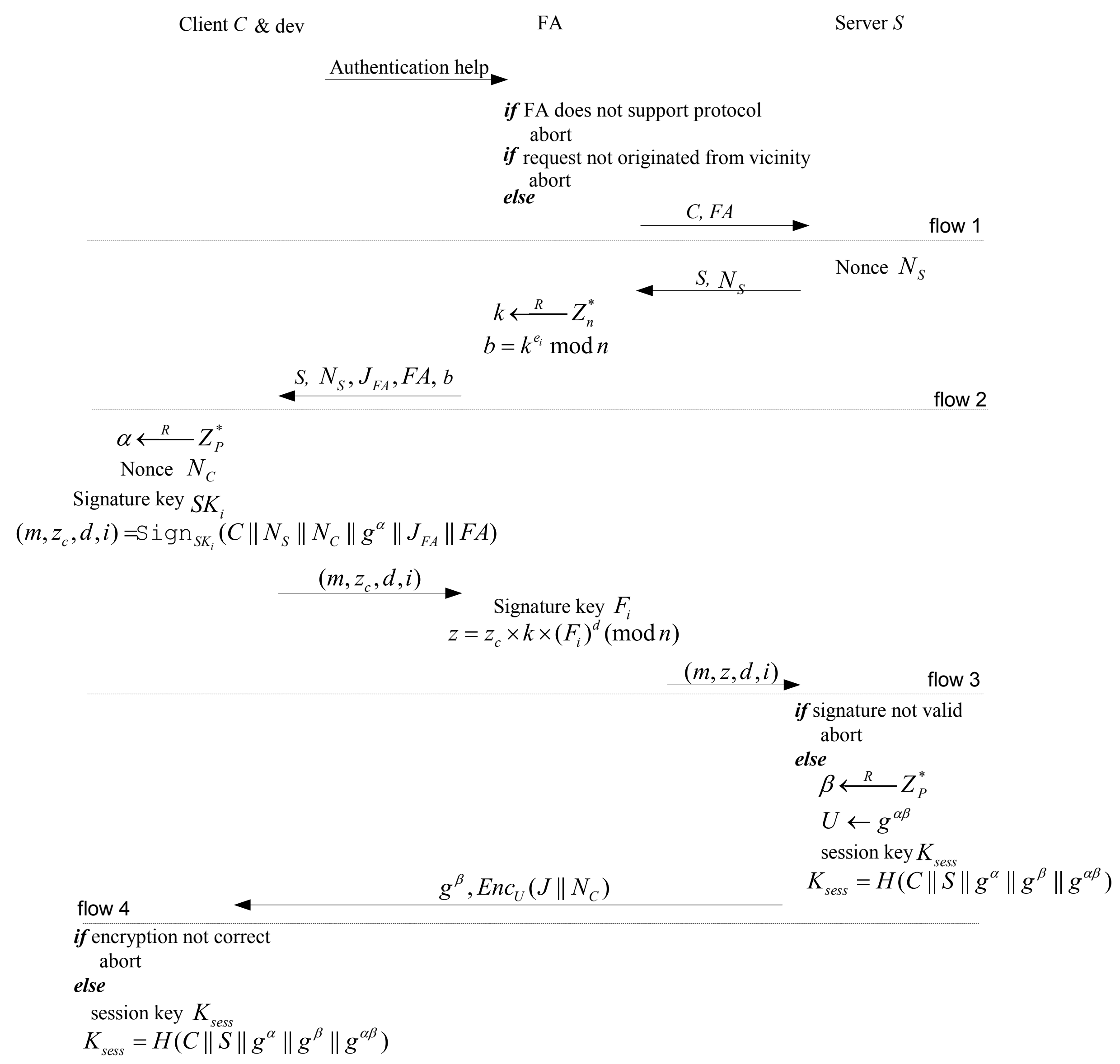

Client C & dev
FA
Server S
Authentication help
if FA does not support protocol abort
if request not originated from vicinity abort
else
C, FA
flow 1
Nonce N_S
S, N_S
$k \xleftarrow{R} Z_n^*$
$b = k^{e_i} \bmod n$
S, N_S, J_{FA}, FA, b
flow 2
$\alpha \xleftarrow{R} Z_P^*$
Nonce N_C
Signature key SK_i
$(m, z_c, d, i) = \mathtt{Sign}_{SK_i}(C \| N_S \| N_C \| g^\alpha \| J_{FA} \| FA)$
(m, z_c, d, i)
Signature key F_i
$z = z_c \times k \times (F_i)^d (\bmod n)$
(m, z, d, i)
flow 3
if signature not valid abort
else
$\beta \xleftarrow{R} Z_P^*$
$U \leftarrow g^{\alpha\beta}$
session key K_{sess}
$K_{sess} = H(C \| S \| g^\alpha \| g^\beta \| g^{\alpha\beta})$
$g^\beta, Enc_U(J \| N_C)$
flow 4
if encryption not correct abort
else
session key K_{sess}
$K_{sess} = H(C \| S \| g^\alpha \| g^\beta \| g^{\alpha\beta})$

Fig. 7.1. Protocol flows in time period *i*

The device also produces a fresh nonce *Nc* for authenticating the server, a partial Diffie-Hellman (DH) key α, and a random number from Z_p^* [38]. SK_i is used to sign its nonce *Nc*, partial DH key, the server's challenge *Ns*, and the location attribute J_{FA}. After flow 3, the device deletes from memory any data related to the password, i.e. $pw, f(pw), f(pw)^{e_1 e_2 \cdots e_{i-1} e_{i+1} \cdots e_T} \bmod n$.

For the next login attempt at time period *i*, the device will calculate the signature key SK_i again using Equation (7.5). When an adversary breaks into the device, no values related to the password are present so that she has no opportunity to perform an offline dictionary attack.

FA will generate its signing key

$$F_i = u_i^{e_{i+1}e_{i+2}\cdots e_T} \bmod n . \tag{7.6}$$

Then it multiplies z_C, the core part of a user's signature, by $k \times F_i^d \bmod n$. The advantage is that the signature length is not expanded. More importantly, the final value of *z* contains the contribution from the *FA*.

Upon receiving flow 3, the server will check the validity of the signature. From the user and the *FA* identities in flow 1, the server retrieves the corresponding *J* and J_{FA}. The server will also choose the appropriate e_i, where *i* is the current time period of the server. The server verifies the nonce *Ns,* the user's identity *C* and the location attribute J_{FA}.

If the signature check fails, the server will abort the session. Otherwise, the server will choose a random number β from Z_p^* and complete the DH key generation to obtain key *U.* Also the server will generate the session key K_{sess} using $K_{sess} = H(C \| S \| g^{\alpha} \| g^{\beta} \| g^{\alpha\beta})$, where *H* is a hash function and $\|$ represents concatenation. The session key generation method is inspired by [39]. Since the three parts g^{α}, g^{β} and $g^{\alpha\beta}$ are included in the hash function, both sides have confidence the session key is fresh. This confidence stems from the fact that g^{α} and g^{β} are under the control of the device and server, respectively. A good hash function will ensure that the outputs are different if the inputs are different. As a result, the session key will not be reused by later sessions.

- *Flow 4 (server to client)*

Part of flow 4 is encrypted and flow 4 is directly "tunneled" to the user. Since *J* is known only to the user and server, the server demonstrates its knowledge of *J* to authenticate itself to the client. The server uses the complete DH key *U* to encrypt *J* as well as the client's nonce N_C. *J* must be encrypted for protection against an eavesdropper. The encryption can be done using a block cipher, e.g. AES-128 cipher. Upon receiving flow 4, the device first recovers the DH key and then decrypts the encrypted data to obtain *J* and nonce *Nc*. The device checks if

$$SK_i^{e_i} = 1/J \bmod n . \qquad (7.7)$$

If the above verifications are satisfied, the device will be convinced that the other party is the legitimate server and flow 4 is not a replay. Then the device will generate the session key K_{sess}.

At this point, the DH keys $g^{\alpha}, g^{\beta}, g^{\alpha\beta}, \beta$ and α will be deleted from the memory of the server and the device. Also deleted from the device's memory are SK_i and *J*.

7.2.3 Underlying Signature Scheme

The underlying signature scheme is called the IR signature scheme, which is provable secure in the random oracle model based on the strong RSA assumption [37]. We make a few important changes to the IR signature scheme in order to match the purpose of our authentication protocol, i.e. the construction of a signature private key based on a password and an evolving device key, the multiplication by FA. The signature scheme outlined in this paper is presented in Fig. 7.2.

The theme of the signature scheme is simply this: when an adversary breaks in at time period j, she will know all SK's and/or F's for time periods $\geq j$, but with only negligible probability, will she be able to compute the signature key for time periods $<j$ or achieve a minimal goal of forging valid signatures for time periods$<j$.

The purpose of r in the signature is to achieve semantic security. We stress that the random number r must be deleted after the signature generation in order to protect the signature scheme in the case of compromise.

To recap some highlights of our protocol, the forward secure property of the signature scheme brings a strong non-repudiated time attribute. Step 5 in Fig. 7.2 shows the signature contribution of FA. And the pre-image of d consists of J_{FA} and i, which corresponds to the location and time attributes, respectively. Thus the signature, saved in the log files, can testify that at a particular time period and in a particular location, a particular user did log into the server.

Sign algorithm: message *m*, signature key SK_i, time period *i*, public value e_i

1. The user chooses a random number $r \in Z_n^*$, then calculate $r^{e_i} \bmod n$

2. FA chooses a random number $k \in Z_n^*$, then sends $b = k^{e_i} \bmod n$ to the user

3. The user generates $d \leftarrow h(m \,||\, (r^{e_i} b \bmod n) \,||\, i)$ $// h(\cdot)$ a collision resistant hash function

4. The user calculates $z_C = r \times SK_i^d \bmod n$

5. FA augments the signature $z = z_C \times k \times F_i^d \bmod n$

6. The signature is (m, d, z, i)

Verifying algorithm: public key n, J, J_{FA}, e_i, signature (m, d, z, i)

verifier checks if $d = h(m \,||\, ((JJ_{FA})^d z^{e_i} \bmod n) \,||\, i)$ then accept

Signature key update: device initial key v_1, password *pw*, signature key SK_i

$v_i = v_{i-1}^{e_{i-1}} \bmod n, i \geq 2$

$SK_i = v_i^{e_{i+1}e_{i+2}\cdots e_T} \times f(pw)^{e_1 e_2 \cdots e_{i-1} e_{i+1} \cdots e_T} \bmod n$ $// f(\cdot)$ a one way function

$F_i = F_{i-1}^{e_{i-1}} \bmod n$

message *m*: $C \,||\, N_S \,||\, N_C \,||\, g^{\alpha} \,||\, J_{FA} \,||\, FA$

Fig. 7.2 The Underlying Signature Scheme

7.3 PROTOCOL ENHANCEMENTS

7.3.1 Login Records with Explicit Location Knowledge

We propose that message flow 3 (*m*, *z*, *d*, *i*) be saved in log files. Following the property of a forward secure signature, even if both the user's password and device secret are leaked, an adversary is still unable to forge the login records prior to the time period when she breaks into the device. Consequently this negates the action of a legitimate user who acts dishonestly to leak his secret on purpose in order to deny his login history.

In order to better defend the interests of the user against the collusion of a *FA* and server, or multiple *FA*'s, we want explicit geographic location information to be included in *d*. The reason is that the server knows the location information of J_{FA} because the server stores a table (relation of J_{FA} and its location). However, the client does not have such tables. The client thus has no assurance concerning the real geographical location associated with J_{FA}. In this case, the location attribute is sacrificed. To maintain the faithfulness of the location attribute, we require the *FA* to pass its explicit location in flow 2. We assume that the mobile devices have GPS-support. So the authentication program in the device can compare its location and the *FA*'s location. If they are too far away, the mobile device will suspect that the *FA* has purposely passed the wrong information and then abort this session. If the *FA*'s location is genuine, the user puts the *FA*'s location into the pre-image of *d*. Thus the user has the assurance that the signature contains the correct location and time information, which is the information both users and servers really care about. Note that this enhancement is optional, depending upon the possibility of collusion and the system security requirements.

7.3.2 Self-healed Security

With regard to the key exposure problem, the traditional de facto countermeasure, as stated in [40], is that after intrusion is detected both the server and client change keys. Revoking an old key is usually done through a non-automatic manner, which increases the administrative cost.

Our protocol provides an approach for automatically self-healing security, even if the intrusion is undetected. The method is to generate a new random number seed for the pseudorandom number generator, using affordable hardware methods [29], immediately prior to login for every login attempt. This seed then generates the required pseudorandom numbers for each login. Consequently, the old secret leaked to an adversary cannot help the adversary derive the device randomness of the future. The noticeable advantage of this approach is that the server need not know the change in the device’s pseudorandom number generator seed and the device does not have to detect the intrusion. In the traditional approach, if a user on the road is suspicious that key leakage has occurred, he usually calls the remote system--an out of band approach--to revoke the old key, and then waits for the new key to be delivered out of band. Using our protocol, the user simply continues the normal login process.

CHAPTER 8

CONCLUSIONS

With the proliferation of wireless access and computing, roaming and end-to-end authentication has emerged as an important issue for network operators and end-users. The design difficulty for the authentication protocol is the scalability, limited energy supply to the wireless devices, and ever-increasing capabilities of adversaries. The first contribution of our work is to propose a new roaming authentication protocol that significantly lowers the amount of the authentication traffic overhead between the home and visited network. As a result, both the authentication latency and the energy consumption of a mobile device are greatly reduced. The second contribution of our work is to propose a new end-to-end authentication protocol based on the novel technique of combining the symmetric key, pin number, and one-way hash chain. The resultant protocol is resilient to strong adversaries with break-in capabilities. The technique of automatic hash-chain-value-update can reduce the number of manual system restart; the technique of complementary MACs can detect the intrusion of a strong adversary; the technique of tuning hash chain iteration can optimize energy consumption for a mobile terminal. The third contribution of our work is to add the spatial-temporal attribute to the authentication protocol. This new idea has the potential to solve some complex problems in the authentication protocol. The forth contribution of our work is the systematic methodology to evaluate the performance of the proposed protocols.

Nevertheless, there are several limitations in our investigation on authentication protocols, and we wish to further explore the problems in the future.

- ✓ The simulation is based on a first-order queueing model. We have use a simple queueing mode for the simulation purpose. Although the simple model can capture the general trend, it may overlook some details.
- ✓ The time duration for performance evaluation is relatively long. We have spent much time on programming debugging (TCP/IP socket programming, kernel programming of operation systems). It will be an interesting alley to consider the simulation of an authentication protocol using some simulation tools such as ns-2 (network simulator, version 2). This may shorten the cycle and enable us to study more cases.

BIBLIOGRAPHY

[1] S. Greese, M. Goldsmith, B. Roscoe and I. Zakiuddin, "Authentication for pervasive computing," in *Proc. 1st Intl. Conf. on Security in Pervasive Computing*, Mar. 2003, Boppard, Germany.

[2] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Security Architecture (Release 4)," *TS 33.102 V4.3.0*, December, 2001.

[3] 3GPP2, "Enhanced Cryptographic Algorithm," *S. S0055 V1.0*, Jan. 2002.

[4] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Cryptographic Algorithm Requirements (Release 4)," *TS 33.105 V4.1.0*, June 2001.

[5] B. Anton, B. Bullock, and J. Short, "Best current practices for wireless Internet service provider (WISP) roaming," *Wi-Fi Alliance*, Feb. 2003, http://www.weca.net/OpenSection/wispr.asp.

[6] Y. Matsunaga, A. Merino, T. Suzuki, and R. Katz, "Secure authentication system for public WLAN roaming," in *Proc. ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, Sept. 2003, pp. 113-121.

[7] M. Long, C.-H. Wu, and J. Irwin, "Localized authentication for wireless LAN inter-network roaming," in *Proc. IEEE Wireless Communications and Networking*, Mar. 2004.

[8] M. Long, C.-H. Wu, and J. Irwin, “Localised authentication for inter-network roaming across wireless LANs,” *IEE Proceedings Communications,* vol. 151, no. 5, Oct. 2004, pp. 496-500.

[9] L. Blunk and J. Vollbrecht, “PPP Extensible Authentication Protocol (EAP),” *Request for Comments 2284*, Mar. 1998.

[10] W. Simpson, “PPP Challenge Handshake Authentication Protocol (CHAP),” *Request for Comments 1994*, Aug. 1996.

[11] B. Aboba and D. Simon, “PPP EAP TLS Authentication Protocol,” *Requests for Comments 2716*, Oct. 1999.

[12] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton: FL, 1996.

[13] Y.-B. Lin and Y.-K. Chen, “Reducing authentication signaling traffic in third-generation mobile network,” *IEEE Trans. Wireless Communications*, vol. 2, no. 3, 2003, pp. 493-501.

[14] H. Kim and H. Afifi, “Improving mobile authentication with new AAA protocols,” in *Proc. IEEE Conference on Communications*, vol. 1, May 2003, pp. 497-501.

[15] S. Suzuki and K. Nakada, “An authentication technique based on distributed security management for the global mobility network,” *IEEE J. Selected Areas in Communications*, vol. 15, no. 8, 1997, pp. 1608-1617.

[16] K. Hwang and C. Chang, “A self-encryption mechanism for authentication of roaming and teleconference services,” *IEEE Trans. Wireless Communications*, vol. 2, no. 2, 2003, pp. 400-407.

[17] L. Buttyán, C. Gbaguidi, S. Staamann, and U. Wilhelm, "Extensions to an authentication technique proposed for the global mobility network," *IEEE Trans. Communications*, vol. 48, no. 3, 2000, pp. 373-376.

[18] A. Freier, P. Karlton, and P. Kocher, "The SSL protocol version 3.0," Nov. 1996, http://wp.netscape.com/eng/ssl3/draft302.txt.

[19] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," *RFC 2246*, Jan. 1999.

[20] NIST, "The keyed-hash message authentication code (HMAC)," *FIPS PUB 198*, Mar. 2002.

[21] R. L. Rivest, A. Shamir, and L. A. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, 1978, pp. 120-126.

[22] Wei Dai, *Crypto++ Library 5.1*, http://www.eskimo.com/~weidai/cryptlib.html.

[23] Active Measurement Project, *National Laboratory for Applied Network Research*, http://www.nlanr.net/.

[24] L. Kleinrock, *Queueing Systems: Volume I—Theory*. New York: Wiley, 1976, pp. 187-190.

[25] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha, "Analyzing the energy consumption of security protocols," in *Proc. ACM International Symposium on Low Power Electronics and Design*, 2003, pp. 30-35.

[26] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)," *FIPS-PUB 186-2*, Jan. 2000.

[27] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, Nov. 1981, pp. 770-772.

[28] N. Haller, C. Metz, P. Nesser, M. Straw, "A one-time password system," *Request for Comments (RFC) 2289*, Feb. 1998.

[29] D. Eastlake, S. Crocker, and J. Schiller, "Randomness Recommendations for Security," *Request for Comments 1750*, Dec. 1994.

[30] National Institute of Standards and Technology (NIST), "Secure Hash Standard," *FIPS-PUB 180-2*, Aug. 2002.

[31] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York: NY, 2003

[32] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, "Securing electronic commerce: reducing the SSL overhead," *IEEE Network Magazine*, July/August, 2000, pp. 8-15.

[33] M. Satyanarayanan, "Pervasive computing: vision and challenges," *IEEE Personal Communication*, pp. 10-17, Aug. 2001.

[34] D. Clark, J. Wroclawski, K. Sollins and R. Braden, "Tussle in cyberspace: defining tomorrow's Internet," *In Proc. of ACM SIGCOMM 2002*, pp. 347-356, Aug. 2002.

[35] W. Stallings. *Cryptography and network security principles and practice (3rd edition),* Upper Saddle River, NJ: Prentice Hall, 2003.

[36] http://grouper.ieee.org/groups/1363/passwdPK/index.html.

[37] G. Itkis and L. Reyzin. "Forward-secure signatures with optimal signing and verifying,*" in Advances in Cryptology-Crypto '01*, J. Kilian (Ed.), LNCS 2139, Springer-Verlag, pp. 332-354, Aug. 2001.

[38] W. Diffie and M. Hellman. "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. 22, pp. 644-654, 1976.

[39] M. Bellare, D. Pointcheval and P. Rogaway. “Authenticated key exchange secure against dictionary attacks,” in *Advances in Cryptology-Eurocrypt 2000*, B. Preneel (Ed.), LNCS 1807, Springer-Verlag, http://www.cs.ucsd.edu/users/mihir/papers/key-distribution.html.

[40] M. Bellare and B. Yee, “Forward-Security in Private-Key Cryptography,” in *Topics in Cryptology: CT-RSA 2003*, M. Joye (Ed.), LNCS 2612, Springer-Verlag, http://www.cs.ucsd.edu/users/mihir/papers/fspkc.html.