Authentication and Key Agreement Protocols Preserving Anonymity

Kumar Mangipudi[1], Rajendra Katti[1], and Huirong Fu[2*]
(Corresponding author: Huirong Fu)

Department of Electrical and Computer Engineering, 1411 Centennial Blvd, [1]
North Dakota State University, Fargo, ND 58105, USA. (Email: {kumar.mangipudi, rajendra.katti}@ndsu.edu)
Department of Computer Science and Engineering, [2]
Oakland University, Rochester, MI 48309, USA. (Email: fu@oakland.edu)

(Received Sept. 7, 2005; revised and accepted Oct. 11 & Oct. 27, 2005)

Abstract

Anonymity is a very important security feature in addition to authentication and key agreement features in communication protocols. In this paper, we propose two authentication and key agreement (AKA) protocols: the AKA protocol with user anonymity (UAP) and the AKA protocol with user and server anonymity (USAP). The proposed protocols have the following advantages: first of all, they preserve anonymity, which is a security feature that was ignored in most of the previously proposed AKA protocols; secondly, they exploit the difference in capabilities between resource constrained clients and highly resourceful servers and thus are suitable for wireless applications; thirdly, they resist known attacks; and finally, they perform better in terms of the number of messages and bits exchanged and computing time as compared to the previously proposed AKA protocols. For example, USAP preserves user and server anonymity, exchanges 3 messages with 1920 bits in total, and requires only 280 msec of processing time on the user side when implemented on Mitsubishis M16C microprocessor. Similarly, the UAP is scalable, preserves user anonymity, requires 440 msec, and exchanges 2560 bits.

Keywords: Anonymity, authentication, Elliptic Curve Cryptography (ECC), key agreement

1 Introduction

Authenticated Key Agreement (AKA) protocols provide communicating parties with a random shared-key that can subsequently be used to communicate confidentially. These protocols provide an efficient means of establishing keys and therefore solve the problems associated with key management. Nonetheless, the AKA protocols are designed with an objective that the communicating parties execute a scheme, and when it is terminated, each of the parties should have certain assurance that they know the other's true identity and share a new and random session-key derived from contributions of all the parties. This objective has to be accomplished irrespective of wired or wireless media. Client-server wireless communications, where a low end user and a server authenticate each other, often demand for few message exchanges and less computational loads.

Until now, numerous public key cryptography based AKA protocols ranging from the traditional RSA to Elliptic Curve Cryptography (ECC) have been proposed. Recently, ECC has gained a lot of attention as ECC implemented devices have higher strength per key bit, lower power consumption, and smaller bandwidths as compared to RSA based cryptosystems. Hence, it is more promising to implement ECC in constrained platforms such as wireless devices, handheld computers, and smart cards.

Apart from security services like authentication and key agreement, the requirement for having anonymity is gaining a lot of attention because transmitting a user's identity in plain during the authentication process invades the user's privacy and allows unauthorized access of his personal information that may result in violation of his privacy and raise legal issues [5]. A literature survey on AKA protocols (included in Section 2 of this paper) revealed that most of the previously proposed protocols ignored anonymity. A wireless authentication protocol that supports anonymity was proposed in [2, 3]. For the rest of our discussion, we refer to this protocol as A-WAP. Despite the authors' claim, A-WAP on one hand fails to provide anonymity and on the other hand it succumbs to several attacks as shown in [17, 26].

Preserving anonymity is a very broad and relative term. In user-server applications such as accessing or requesting services from a server, user anonymity is highly appreciated as compared to server anonymity. In either case, the anonymity is defined with respect to the public. Note that in the above applications the server has to identify and verify the user for accounting and billing purposes. As such in our design, while addressing anonymity we envision two kinds of applications. The first application is a more generic one such as in ad-hoc networks that requires only user anonymity, and the server or the service provider is a public entity that provides public services. Secondly, applications where a user and a specific server communicate with each other while remaining anonymous to the public. A user accessing his/her bank account or a remote office server is an example for second application. The proposed protocols, AKA protocol with user anonymity (UAP) and AKA protocol with user and server anonymity (USAP), respectively address the above two applications.

The following are the advantages of our protocols: first of all, they preserve anonymity, which is a security feature that was ignored in most of the previously proposed AKA protocols; secondly, they exploit the difference in capabilities between resource constrained clients and highly resourceful servers and thus are suitable for wireless applications; thirdly, they resist known attacks; and finally, they perform better in terms of the number of messages and bits exchanged and computing time. Since a user can be a low power device, as in wireless applications, we measure the performance of our protocols from user's perspective. The proposed USAP is computationally efficient with fewer message exchanges but is not scalable as the user can communicate only with a specific server. Hence, the scalability as we envision is the ability of a user to communicate with a number of specific servers while preserving both user and server anonymity. On the other hand, the proposed UAP is scalable (a user can communicate with any arbitrary server rather than a specific server) but only at the cost of server anonymity and increased computational and communicational overhead as compared to USAP. For example, USAP preserves user and server anonymity, exchanges 3 messages with 1920 bits, and requires only 280 msec of processing time on the user side when implemented on Mitsubishis M16C microprocessor. The UAP, preserves user anonymity, requires 440 msec, and exchanges 2560 bits. These timings are based on the authors' analysis of MSR-Hybrid, a fast authenticated and key establishment protocol, for sensor networks that does not support anonymity, requires 455 msec, and exchanges 4448 bits in 4 messages [13].

The rest of this paper is organized as follows. We discuss various authentication schemes and review previously proposed AKA protocols based on public-key cryptography in Section 2. Next, we introduce the design criteria for our proposed protocols in Section 3. Then, in Sections 4 and 5, we present the proposed UAP and USAP, respectively. In Sections 6 and 7, we analyze the security and compare the performance of our proposed protocols, respectively. Finally, we conclude in Section 8.

2 Related Work

There exists plethora of authentication schemes in the literature that are designed to address a variety of applications. The following is one of the many ways of classifying those schemes based on the kinds of security services they support and the underlying cryptographic functions used in their design:

1) Hash-based password authentication protocols [11, 16, 21, 24]

2) Public-key based authentication and key agreement (AKA) protocols (as discussed below)

3) Symmetric-key authentication protocols [22]

4) Authentication schemes based on key-chains [19, 27]

A complete description of the above mentioned authentication schemes are beyond the scope of this paper. As such, this paper only refers to well-known and widely-used AKA protocols based on public-key cryptography.

Diffie and Hellman first proposed the Diffie-Hellman (DH) key exchange based on the discrete logarithm problem in 1976 [10]. Since the original DH protocol is vulnerable to a man-in-the-middle attack, modifications were proposed to resist such attack [28]. Later, Bellovin and Merrit presented a password based key exchange protocol for two-party communications known as Encrypted Key Exchange (EKE) [7]. Further, an efficient and elegant scheme for EKE that was considered for standardization by the IEEE P1363 Standard working group is AuthA, which was later enhanced by Bresson et al. in [8] to resist the denial-of-service attack. In [30], Zhang showed that Strong Password only Authenticated Key Exchange (SPEKE), a password authenticated key exchange protocol defined in [15] was susceptible to password guessing attack. Wong and Chan [29] proposed a mutually authenticated key exchange protocol for low power computing devices, which was later proven insecure against unknown key-share attacks by Shim [23]. Zhu et al. presented a password based authenticated key exchange protocol based on RSA for imbalanced wireless networks in [31]. Further, protocols proposed by Beller et al. [6] and Aziz and Diffie [4] address mutual authentication and key agreement issues for low end devices. Unfortunately, none of the above protocols provides anonymity. A widely-used standard for IPSec protocol suite is the Internet Key Exchange (IKE) [12]. IKE has several drawbacks, and further, it transmits a user's identity in clear. Just Fast Keying (JFK) protocols proposed in [1] address the shortcomings of IKE. However, they too ignored anonymity of the communicating parties. In recent years, several ECC-based key agreement protocols, such as the ECMQV protocol with ECC X509 certificates [25], implicit certificates and the Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) protocol [9], A-WAP, and two fast authenticated key exchange protocols [13], were proposed. Out of these, only A-WAP claims that it addresses anonymity.

However, a direct application of this protocol is questionable as it not only fails to provide anonymity but also succumbs to the attacks listed in [12, 26].

3 Protocol Design Goals

This section presents a list of security requirements that are considered while designing AKA protocols. Also included in this section are initial assumptions and nomenclature used throughout this paper.

3.1 Protocol Requirements

The following security services are considered in the design of our protocols.

1) Mutual authentication assures the communicating parties that the message has originated from the intended party and that it has not been tampered with.

2) Key agreement ensures that the communicating parties agree upon a random shared session-key that is independent of the previous communications. Further, the agreed session-key is derived from the contribution of all the parties and thus no single party is able to exercise control on the selection of the key.

3) Anonymity in communication ensures that no one else other than the intended parties be able to figure out who is communicating with whom.

4) Confidentiality protects the transmitted messages from eavesdropping.

5) Non-repudiation ensures that no parties can deny their actions after the completion of their communication.

In addition to supporting the above security services, the designed protocols should not only resist various attacks but also should satisfy the performance requirements. Some of the attacks on AKA protocols are replay, man-in-the-middle, denial-of-service, impersonation, known-key, and unknown-key share attacks. Furthermore, the designed protocols should provide forward secrecy. A full description of these terms can be obtained from [17, 26, 28]. The authors in [28] suggest the following performance requirements:

1) Minimum number of messages exchanged,

2) Low communication overhead (total number of bits transmitted),

3) Low computation overhead (total number of arithmetic operation involved), and

4) Possibility of pre-computations that reduce the protocol execution timings.

3.2 Assumptions, Nomenclature, and Functions Used

As with the other public key cryptosystems, we consider that a trusted third party known as Certificating Authority (CA) issues a certificate (r, s) and its expiration date (t) to the communicating parties (users/servers) during the initialization phase. The certificate (r, s) is computed by using Elliptic Curve Digital Signature Algorithm (ECDSA). More details on the ECDSA can be found in [14, 18]. In addition to issuing certificates, we also assume that CA can assign unique identities, store, and distribute any other required information to the communicating parties. In real world, the latter functions of the CA can be performed by some independent body.

First, an elliptic curve over $GF(p)$ (where p is a prime number of length greater than 160 bits) with suitable coefficients is defined. Note that the elliptic curve can also be defined over $GF(2^m)$. Then a base point $P = (P.x, P.y)$(where $P.x$ and $P.y$ are the x and y coordinates of the point P, respectively) of large order n, belonging to this elliptic curve group is selected and made public to all users/servers. The CA selects a random number d_{CA} as its private key and performs the point multiplication $d_{CA} \times P$ to obtain its public key Q_{CA}, i.e., $Q_{CA} = d_{CA} \times P$. We shall denote the randomly generated private and public key pair as (d, Q) with appropriate subscripts. Thus, CA's randomly generated private and public key pair is (d_{CA}, Q_{CA}). We also employ a symmetric encryption and decryption algorithm, such as Advanced Encryption Standard (AES) and a hash function, such as SHA-1 or MD-5 [22]. The encryption, decryption, and hash function are denoted as E, D, and H, respectively. Finally, our proposed protocols consist primarily of two phases: an initialization phase, an offline process, which is executed between a user (or a server) and the CA through a secure channel; and a mutual authentication phase, an online process that is executed in real time whenever the parties (a user and the server) want to mutually authenticate and agree upon a session-key. The users and server execute the initialization phase at the end of their expiration dates (t) to obtain another valid certificate and expiration date. In addition to the subscript CA, we will also use the subscripts U and S to indicate to a user and the server, respectively.

4 AKA Protocol with User Anonymity (UAP)

In this section, we propose an AKA protocol with user anonymity (UAP) that preserves user anonymity during the mutual authentication phase. Note that in UAP, we do not consider server anonymity as the server and its services are public. This protocol addresses situations similar to ad-hoc networks, where a user and any server need to authenticate each other before establishing secure communication. We describe UAP in two phases:

User		**CA**
1. Chooses $d_U \in \{2, n-2\}$		Chooses $k_U \in \{2, n-2\}$
2. $Q_U = d_U \times P$		$R_U = k_U \times P$
3. Sends Q_U	→	Receives Q_U
4.		Chooses a unique I_U and t_U
5.		$r_U = R_U.x$, $e_U = H(Q_U.x, I_U, t_U)$
6.		$s_U = k_U^{-1}(e_U + d_{CA} r_U)$
7. Receives $Q_{CA}, I_U, (r_U, s_U), t_U$	←	Sends $Q_{CA}, I_U, (r_U, s_U), t_U$
8. Stores $d_U, Q_U, Q_{CA}, I_U, (r_U, s_U), t_U$		

Figure 1: Initialization phase in UAP

the initialization phase, where all users and the servers in the network obtain their respective certificates and expiration dates from a CA through a secure channel and the mutual authentication phase that is executed in real time for mutual authentication and key agreement between a user and the server. The rest of this section contains the details of UAP.

4.1 Initialization Phase in UAP

During the initialization phase, all the users and servers in a network execute an initialization protocol similar to the one that is shown in Figure 1 to obtain their respective certificates and expiration dates from CA. Note that this initialization protocol is executed via a secure channel. In Steps 1 through 3 of Figure 1, a user, U generates a random private and public key pair (d_U, Q_U) and sends Q_U to CA to obtain a certificate for his/her public key. Then CA first assigns an identity I_U and an expiration date t_U, calculates hash value $\mathrm{e}_U = H(Q_U.x, I_U, t_U)$, generates a certificate (r_U, s_U) and then sends $I_U, (r_U, s_U), t_U$ along with its public key Q_{CA} to U. Finally, U, stores $d_U, Q_U, Q_{CA}, I_U, (r_U, s_U), t_U$. A server, S, executes a similar initialization protocol with CA and obtains $Q_{CA}, I_S, (r_S, s_S)$, and t_S.

4.2 Mutual Authentication Phase in UAP

Before establishing a secure communication channel, a user and the server execute the mutual authentication phase, as shown in Figure 2, the details of which are given below. To make the understanding of the protocol simple, we divide this mutual authentication phase into three sub-phases: the temporary-key generation, the certificate verification, and the session-key generation. In temporary-key generation sub-phase, i.e., Steps 1 through 6, the communicating parties set up a temporary-key for encrypting the messages that contain sensitive information such as certificates, expiration dates, random numbers, identities, etc. Steps 7 through 19 define the certificate verification sub-phase, where both parties decrypt the received messages and verify each other's certificates. A new and random session-key is generated in the final stages of the mutual authentication phase, which is the session-key generation phase (Steps 20 and 21). What follows next is the mutual authentication phase between a user, U and the server, S in detail.

Initially, S sends its public key Q_S and identity I_S to U. Note that we do not consider the anonymity of the server as it is a public entity. The user anonymity is a required security service in order to keep his/her actions untraceable. Upon receiving the server's public key Q_S, U then calculates $Q_R = g_U \times Q_S$ (g_U, a random number) and transmits Q_R to S. Transmitting Q_R is not of security concern as will be explained in Section 6. U also calculates $Q_K = g_U \times P$ to obtain the temporary-key $Q_K.x$ and g_{US}. Note g_{US} is the same as $Q_K.x$. First, S performs $d_S^{-1} \times Q_R$ to obtain the temporary-key $Q_K.x$ and g_{US} and then it encrypts $(r_S, s_S), t_S$, g_{US} and its generated random number g_S using $Q_K.x$ and sends this encrypted message to U as C_0, which is the Step 8 of Figure 2. Upon receiving C_0, U decrypts C_0, checks the validity of g_{US} and t_S, terminates the protocol upon a failure; otherwise, encrypts (r_U, s_U), t_U, I_U, $Q_U.x$ and the received random value g_S using $Q_K.x$, sends C_1 (encrypted message, i.e., $E(Q_K.x,\ (r_U, s_U), I_U, t_U, Q_U.x, g_S)$)) to S, and then calculates $e_S = H(Q_S.x,\ I_S,\ t_S)$ before verifying the received server's certificates as shown in Steps 14 through 19 of Figure 2. Similarly, S verifies the received user's certificates. If the certificates are valid, both parties generate a unique session-key $k_M = H(g_{US}, g_{S_,}, e_S, e_U)$ and destroy g_{US}, and g_S from their memory.

5 AKA Protocol with User And Server Anontmity (USAP)

In this section, we propose an AKA protocol with user and server anonymity (USAP) that allows a user and a specific server to mutually authenticate each other before establishing a secret session, while remaining anonymous to the public. A real world example where USAP can be applied is when a user wants to access services from a specific server such as his/her bank account or a remote office server. Anonymity is desirable in the interests of both communicating parties. USAP is almost similar to UAP, except for few modifications to support server anonymity in addition to user anonymity. A user

User		**Server**
1. Generates a random number $g_U \in \{2, n-2\}$		
2. Receives Q_S, I_S	←	Sends Q_S, I_S
3. $Q_R = g_U \times Q_S = (g_U d_S) \times P$		Generates a random number $g_S \in \{2, n-2\}$
4. Sends Q_R	→	Receives Q_R
5. $Q_K = g_U \times P$		$Q_K = d_S^{-1} \times Q_R = (d_S^{-1} g_U d_S) \times P = g_U \times P$
6. $g_{US} = Q_K.x$: Temporary-key		$g_{US} = Q_K.x$: Temporary-key
7.		$C_0 = E(Q_K.x, (r_S, s_S), t_S, g_{US}, g_S)$
8. Receives C_0	←	Sends C_0
9. $D(Q_K.x, C_0)$: Valid g_{US}, t_S?		
10. $C_1 = E(Q_K.x, (r_U, s_U), I_U, t_U, Q_U.x, g_S)$		
11. Sends C_1	→	Receives C_1
12. Calculates $e_S = H(Q_S.x, I_S, t_S)$		$D(Q_K.x, C_1)$: Valid g_S, t_U?
13.		$e_U = H(Q_U.x, I_U, t_U)$
14. $c = s_S^{-1}$		$c = s_U^{-1}$
15. $u_1 = ce_S$		$u_1 = ce_U$
16. $u_2 = cr_S$		$u_2 = cr_U$
17. $R = u_1 \times P + u_2 \times Q_{CA}$		$R = u_1 \times P + u_2 \times Q_{CA}$
18. $v = R.x$		$v = R.x$
19. if $v \neq r_S$, then abort		if $v \neq r_U$, then abort
20. $k_M = H(g_{US}, g_S, e_S, e_U)$: Unique session-key		$k_M = H(g_{US}, g_S, e_S, e_U)$: Unique session-key
21. Destroys g_{US}, g_S		Destroys g_{US}, g_S

Figure 2: Mutual authentication phase in UAP

in USAP stores some server specific information to support server anonymity. Hence, the scalability as we define is the ability of the user to store servers' specific information to communicate securely with several servers while preserving both user and server anonymity. The larger is the number of servers that a user can communicate with, the higher is the memory required to store the specific information pertaining to each of the servers.

First, USAP has two different initialization phases: the server initialization phase defined for the server and the user initialization phase defined for a user. Second, all users obtain the server's public key Q_S and hashed value e_S from CA during the user initialization phase, which will then be used in preserving server anonymity during the mutual authentication phase. Note that in mutual authenticating phase of both UAP and USAP, the communicating parties (a user and the server) first authenticate each other and then agree upon a session-key to secure the subsequent communication. Finally, USAP relieves the user from verifying server's certificates and thereby lessens the computational load during the mutual authentication phase. However, the user validates the server's public key, identity and expiration date by calculating $e'_S = H(Q_S.x, I_S, t_S)$ and comparing it with the e_S received from CA during the initialization phase.

The rest of this section contains the details of USAP and its two phases: the initialization phase and the mutual authentication phase. Again, the initialization phase is executed between a user (or a server) and the CA through a secure channel to obtain their respective certificates and expiration dates and the mutual authentication phase is performed between a user and the server to first authenticate each other and then establish a secure communication channel by agreeing upon a new random session-key.

5.1 Initialization Phase in USAP

In USAP, a user and the server execute two different initialization phases whose details are given below.

1) Server Initialization Phase: Figure 3 shows the server initialization phase. First, the server, S, generates a random public and private key pair (d_S, Q_S), then sends Q_S to the CA and calculates d_S^{-1}. Upon receiving Q_S, the CA assigns a unique identity I_S and an expiration date t_S, calculates hashed value $e_S = H(Q_S.x, I_S, t_S)$, and sends I_S, t_S and its public key Q_{CA} to S. The CA then stores the server's public key Q_S and e_S. Finally, S receives Q_{CA}, I_S and t_S and stores d_S^{-1}, Q_{CA}, I_S and t_S.

2) User Initialization Phase: All users execute the user initialization protocol with CA as shown in Figure 4 when they first subscribe. First, a user, U, generates a random private and public key pair (d_U, Q_U) and sends Q_U to CA to obtain a certificate and an expiration date for its public key. Then CA first assigns an identity I_U, an expiration date t_U, calculates hash value $e_U = H(Q_U.x, I_U, t_U)$, generates a certificate pair (r_U, s_U) and then sends $I_U, t_U, (r_U, s_U)$ along with the server's public key Q_S and hash value e_S to the user. Distributing the server's public key to all

Server		**CA**
1. Chooses $d_S \in \{2, n-2\}$		
2. $Q_S = d_S \times P$		
3. Sends Q_S	→	Receives Q_S
4. Calculates d_S^{-1}		Chooses a unique I_S and t_S
5.		$e_S = H(Q_S.x, I_S, t_S)$
6. Receives Q_{CA}, I_S, t_S	←	Sends Q_{CA}, I_S, t_S
7. Stores $d_S^{-1}, Q_{CA}, I_S,\ t_S$		Stores Q_S, e_S

Figure 3: Server initialization phase in USAP

the users ensures that every user knows the server's public key prior to the authentication scheme.

5.2 Mutual Authentication Phase in USAP

The mutual authentication phase shown in Figure 5 is executed in real time, i.e., whenever a user, U, wants to set up a secure communication with the server, S. Again, the mutual authentication phase is divided into three sub-phases, the details of which are included next. Steps 1 through 5 of Figure 5 indicate the temporary-key agreement sub-phase, where U generates a random number g_U, calculates $Q_R = g_U \times Q_S$, and transmits Q_R to S. Note that the user obtains the server's public key Q_S during the user initialization phase (Figure 4). U also calculates $Q_K = g_U \times P$ to obtain the temporary-key $Q_K.x$. Upon receiving Q_R, S first performs $d_S^{-1} \times Q_R$ to obtain the temporary-key $Q_K.x$ and then encrypts the random numbers g_S, g_{US} (g_{US} is the same as the key, required for data freshness) and I_S, t_S with $Q_K.x$ and finally sends this encrypted value C_0 to U. Next is the certificate verification sub-phase (Steps 6 through 19 in Figure 5). In this sub-phase, U is not required to verify the server's certificate as the server's public key was obtained from CA. However, U decrypts C_0, checks for the presence of g_{US} and validity of the expiration date t_S, computes $e'_S = H(Q_S.x, I_S, t_S)$ and then compares e'_S with the e_S obtained from CA during the user initialization phase. Comparing e'_S with e_S is necessary because e_S binds a server's public key Q_S with its identity I_S and expiration date t_S. The process of verifying the hash value (Step 9 of Figure 5) instead of verifying the server's certificates relieves the user from the computationally intensive point multiplications. If valid, U obtains C_1 by encrypting its certificate (r_U, s_U), identity I_U, x-coordinate of its public key $Q_U.x$, the expiration date t_U, and the server's random number g_S with the temporary-key $Q_K.x$ and sends C_1 to S. Upon receiving C_1, S decrypts it, checks for the presence of g_S and user's expiration date t_U. If the check fails, S aborts the protocol; otherwise, it calculates hash value on the received $Q_U.x$, I_U, and t_U, i.e., $e_U = H(Q_U.x,\ I_U,\ t_U)$ and verifies U's certificate (r_U, s_U) using ECDSA as shown in Steps 13 through 19 of Figure 5. If valid, both parties generate the unique session-key $k_M = H(g_{US},\ g_{S_,},\ e_S,\ e_U)$ in the session-key generation sub-phase (Steps 20 and 21) and destroy g_{US}, and g_S from their memory.

6 Security Analysis

In this section, we provide a provably secured analysis for our protocols. The following is the roadmap. First, we define ECDLP and then prove two theorems. Based on these theorems, we show how our protocols satisfy the design goals (security services and attack resistant) set forth in Section 3 of this paper.

The security of ECC based public cryptosystems largely depends upon solving a widely studied hard problem known as elliptic curve discrete logarithm problem (ECDLP).

Definition 1. *In an elliptic curve group, the ECDLP is defined as finding d given Q and P, where $Q = d \times P$. Here Q and P are the points on the elliptic curve, $\times$ represents point multiplication (i.e., addition of point P, d times), and d is any integer between 2 and $n-2$, where n is the order of the points P and Q.*

It is believed that ECDLP is intractable. More details on ECC, ECDLP, and point multiplication can be found in [9, 14].

The public-key strategy of binding a user or server's public key with its unique identity and expiration date eliminates the requirement for having a large online database. For a successful secured communication between a user and the server, it is required for each of the parties to register with CA by executing the initialization phase to obtain their respective certificates, identities, and expiration dates. It is to be noted that the initialization phase is executed through a secured channel. As such, we shall analyze the security features of our proposed protocols UAP and USAP in the mutual authentication phase which in real time will be executed via an insecure channel. Unless otherwise explicitly stated, the security analysis pertains to both protocols.

Theorem 1. *The security of our protocols depends upon the security of the temporary-key $Q_{K.x}$.*

Proof. Recall the mutual authentication phase and its three sub-phases: the temporary-key generation, the certificate verification, and the session-key generation in UAP and USAP. The temporary-key $Q_{K.x}$ generated in

Server		**CA**
1. Chooses $d_U \in \{2, n-2\}$		Chooses $k_U \in \{2, n-2\}$
2. $Q_U = d_U \times P$		$R_U = k_U \times P$
3. Sends Q_U	→	Receives Q_U
4.		Chooses a unique I_U and t_U
5.		$r_U = R_U.x$; $e_U = H(Q_U.x, I_U, t_U)$
6.		$s_U = k_U^{-1}(H(Q_U.x, I_U, t_U) + d_{CA} r_U)$
7. Receives $Q_S, I_U, (r_U, s_U), t_U, e_S$	←	Sends $Q_S, I_U, (r_U, s_U), t_U, e_S$
8. Stores $Q_U, Q_S, I_U, (r_U, s_U), t_U, e_S$		

Figure 4: User initialization phase in USAP

the temporary-key generation sub-phase (Step 6 and 5 of Figure 2 and 5, respectively) is used to encrypt the exchanged messages C_0 and C_1. Note that C_0 and C_1 contain sensitive information including the random numbers and hashed values, which will be used in deriving the unique session-key K_M. If an adversary (by some means) were able to determine $Q_{K.x}$, then he / she could easily decrypt the messages C_0 and C_1 and calculate the current session-key K_M. Hence, the security of our protocols depends upon the security of the temporary-key $Q_{K.x}$ □

Theorem 2. *Finding out the temporary-key $Q_K.x$ is as hard as solving the ECDLP.*

Proof. The following are the ways to calculate the temporary-key $Q_K.x$.

1) $Q_K = g_U \times P = (Q_K.x, Q_K.y)$ (on the user's side)

2) $Q_K = d_S^{-1} \times Q_R = (d_S^{-1} g_U d_S) \times P = g_U \times P = (Q_K.x, Q_K.y)$ (on the server's side)

The point $Q_R = g_U \times Q_S$ is a random point as it depends upon the random number g_U. For any adversary A to calculate the temporary-key $Q_K.x$, it is required for him to derive the server's private key d_S from public key Q_S. Deriving d_S knowing $Q_S(d_S \times P)$ and the base point P is the problem of ECDLP. Similarly, deriving g_U knowing Q_R and Q_S is again the problem of ECDLP. Hence, calculating $Q_{K.x}$ generated during the mutual authentication protocol is as hard as solving the ECDLP. Further, in USAP, because of server anonymity, an adversary does not have any additional information about Q_S. □

Based on the assumption that an adversary cannot obtain the current session-key K_M without solving for $Q_{K.x}$, we analyze the security of our proposed protocols in the next two sub-sections: security services and attack resistance analysis.

6.1 Security Services

In this sub-section, we prove how our protocols support the security services like mutual authentication, key agreement, key confirmation, anonymity, confidentiality, and non-repudiation.

1) Mutual Authentication:

   Lemma 1. *Our protocols provide mutual authentication of the communicating parties.*

   Proof. The communicating parties achieve mutual authentication by executing a challenge-response sequence and it is described as follows. Here the challenge is sending 'Q_R' and random numbers and the response is to verify the presence of expected random numbers in the encrypted messages during the run of the mutual authentication phase in UAP and USAP. Given Q_R, it is infeasible for an adversary to compute the correct response g_{US} without the knowledge of the server's private key. Note that solving for a server's private key is the problem of ECDLP. If U is able to decrypt C_0 and check the presence of g_{US} (i.e., Step 9 and 8 of Figure 2 and 5, respectively) then U is assured that the server, S, has the knowledge of the private key corresponding to its public key and is able to derive the correct temporary-key $Q_K.x$, which is also the user's random number g_{US} for this session. Similarly, if S is able to decrypt C_1 using $Q_K.x$ and check for the presence of g_S in Step 12 of Figures 2 and 5, then S can be sure that only U can produce the correct response g_S. □

2) Key Agreement: Once the certificates are verified, a unique session-key $k_M = H(g_{US},\ g_{S_,},\ e_S,\ e_U)$ is derived from the contributions of both parties in Step 20 of Figures 2 and 5. Thus, no single party has complete control on the selection of the session-key, which is the main goal of a key agreement protocol [28].

3) Key Freshness:

   Lemma 2. *The included random numbers guarantee key freshness.*

   Proof. Let g'_U, g'_S, and g_U, g_S, be the random numbers generated in two different sessions. Since $g'_U \neq g_U$, therefore $Q'_R(g'_U \times Q_S) \neq Q_R(g_U \times Q_S)$ and hence the calculated temporary-keys $Q'_{K.x}$ and $Q_{K.x}$ and g'_{US} and g_{US} are different and so are the obtained session-keys $k'_M(H(g'_{US}, g'_{S_,}, e_S, e_U))$ and

Server		**CA**
1. Generates a random number $g_U \in \{2, n-2\}$		
2. $Q_R = g_U \times Q_S = (g_U d_S) \times P$		Generates a random number $g_S \in \{2, n-2\}$
3. Sends Q_R	→	Receives Q_R
4. $Q_K = g_U \times P$		$Q_K = d_S^{-1} \times Q_R = (d_S^{-1} g_U d_S) \times P = g_U \times P$
5. $g_{US} = Q_K.x$: Temporary-key		$g_{US} = Q_K.x$: Temporary-key
6.		$C_0 = E(Q_K.x, g_{US}, g_{S_,}, I_S, t_S)$
7. Receives C_0	←	Sends C_0
8. $D(Q_K.x,\ C_0)$: Valid g_{US}, t_S?		
9. Calculates $e'_S = H(Q_S.x,\ I_S,\ t_S)$, Is $e'_S = e_S$?		
10. $C_1 = E(Q_K.x,\ (r_U,\ s_U),\ I_U,\ Q_U.x,\ t_U,\ g_S)$		
11. Sends C_1	→	Receives C_1
12.		$D(Q_K.x, C_1)$: Valid $g_S,\ t_U$?
13.		$e_U = H(Q_U.x,\ I_U,\ t_U)$
14.		$c = s_U^{-1}$
15.		$u_1 = ce_U$
16.		$u_2 = cr_U$
17.		$R = u_1 \times P + u_2 \times Q_{CA}$
18.		$v = R.x$
19.		if $v \neq r_U$, then abort
20. $k_M = H(g_{US}, g_S, e_S, e_U)$: Unique session-key		$k_M = H(g_{US}, g_S, e_S, e_U)$: Unique session-key
21. Destroys g_{US}, g_S		Destroys g_{US}, g_S

Figure 5: Mutual authentication phase in USAP

$k_M(H(g_{US},\ g_{S_,},\ e_S,\ e_U))$, where H is a collision free hash function [22]. □

4) Key Confirmation: If U does not terminate the protocol in Step 9 and Step 8 of Figure 2 and 5, respectively, then S knows that C_0 is received correctly and it can execute the rest of the protocol. Similarly, if S does not terminate the protocol in Step 12 of Figures 2 and 5, then U knows that C_1 is received correctly and it can proceed to execute the rest of the protocol. Further, if U and S do not return a failure after verifying each other's certificates, then they confirm to each other that the key is generated correctly [13]. Thus, our protocols provide key confirmation.

5) User Anonymity: We note that in both protocols, U airs C_1, which is an encrypted value of his/her certificate, identity, hashed value and other parameters. Let us assume that the encryption is secure, then it is impossible for an adversary to determine any user specific data without the knowledge of the temporary-key $Q_{K.x}$. Thus, our protocols achieve user anonymity.

6) Server Anonymity: In USAP, based on ECDLP, it is impossible for an adversary to determine a server's public key or other sensitive information such as hashed value and identity from random point $Q_R = g_U \times Q_S$ and the aired encrypted value C_0. Thus, USAP achieves server anonymity.

7) Confidentiality: Assuming that it is hard for an adversary to determine the temporary-key $Q_{K.x}$, the encrypted messages C_0 and C_1 achieve confidentiality of the exchanged messages during the mutual authentication phase. Furthermore, the agreed unique session-key k_M secures the rest of the further communication.

8) Non-repudiation: A user or server's public and private key pairs together with certificates can be used to achieve non-repudiation of the communication by way of digital signature [18].

9) Forward Secrecy on the User Side: In both protocols, the session-keys can be recovered from the previous runs of the mutual authentication by compromising the server's private key. However, compromising the user's private key does not help in revealing the session-keys. Therefore, our protocols provide forward secrecy on the user side and thereby achieving half forward secrecy [28]. This is justified because the highly resourceful severs can support much stronger security than the resource constrained user [13].

6.2 Attack Resistance Analysis

Now we will show that USAP and UAP resist known attacks such as replay, known-key share, unknown-key share, man-in-the-middle, denial-of-service, and impersonation attacks.

1) Replay Attack: The random numbers g_{US} and g_S prevent replay attacks. Suppose an adversary A captures any or all of the aired messages such as Q_R, C_0, and C_1 and masquerades these messages either with

a previous message or any arbitrary values, the entities return a failure as the substituted message does not contain the correct response for a given challenge.

2) Known-key Share Attack: Each run of the protocol between the entities should produce a unique session-key. From Lemma 2, it is clear that the temporary agreed key $Q_{K.x}$ and the session-key k_M between the user and server vary every time a mutual authentication phase is initiated.

3) Unknown-key Share Attack: Suppose an adversary A tries to make U believe that the temporary-key is shared with B, while B believes that the temporary-key is shared with A. To launch the unknown-key share attack, A should be able to identify the exchanged parameters. However, this is not possible without the knowledge of the temporary-key $Q_{K.x}$. Hence, he will not succeed.

4) Man-in-the-middle, Denial-of-service, and Impersonation Attacks: Our protocols explicitly bind a user or server's identity with its public key, certificate pair and hashed value. As such the above mentioned attacks are not possible as shown below.

   a. UAP: In UAP (Step 2 of Figure 2), a user receives the server's public key along with its identity whose binding with the received certificate is verified during the later stage of the protocol (Steps 12 through 19 of Figure 2). Since the user can identify with whom he/she is communicating with based on the transmitted identity and whose binding with the public key is later verified, it is impossible for an adversary A to insert himself between the communicating parties. This is not the case with A-WAP and hence it succumbs to the above mentioned attacks [17].

   b. USAP: Exchange of the public keys is eliminated as CA distributes the server's public key to all the users. Note that distributing the server's public key does not impose any additional threat on the security, which in turn depends on solving the ECDLP. Also note that the user is assured that he/she is communicating with the intended party (server) by verifying a valid g_{US} in Step 8 of Figure 5. Let us suppose that an adversary, A, tries to establish himself as was shown in [17], then, since A cannot solve the ECDLP, therefore for some ${d'_S}^{-1} \neq d_S^{-1}$, A ends up with a different temporary-key $Q'_{K.x} \neq Q_K.x(Q'_K = {d'_S}^{-1} \times Q_R = ({d'_S}^{-1} g_U d_S) \times P = g'_U \times P)$. Hence, the user returns a failure in Step 8 of Figure 5 because the response received g'_{US} is not equal to the expected value g_{US}.

   In USAP, the user instead of performing certificate verification as described in UAP (i.e., Steps 12 through 19 of Figure 2) verifies the calculated hash value e'_S with the hash value e_S that was received along with the server's public key Q_S during the user initialization phase.

Lemma 3. *The process of verifying the hash value (Step 9 of Figure 5) instead of verifying the server's certificates does not pose any additional threat on the security of USAP.*

Proof. Recall that a user in USAP receives the server's public key and hash value during the initialization phase for CA. After the mutual authentication phase is initiated, if the user is able to check the correct response for mutual authentication, then he/ she knows that the server possesses the private key corresponding to its public key (Lemma 1). However, the user verifies the computed hash value e'_S with the received hash value e_S. This is necessary to check the binding between Q_S, I_S, and t_S. Let us now suppose that either the received I_S or t_S or both have changed, then the e'_S and e_S are not equal. Hence, a user returns a failure, because $e_S = H(Q_S, I_S, t_S) \neq H(Q_S, I'_S, t_S) \neq H(Q_S, I_S, t'_S) \neq H(Q_S, I'_S, t'_S)$, where I'_S, t'_S are different from I_S, t_S. Thus, verifying the hash value instead of certificate does not pose any additional threat to the security of USAP. Unlike UAP, this check is only possible in USAP because the server's public key was received via a secure channel during the initialization phase from CA. □

7 Performance Analysis

In this section, we study the performance of our protocols in terms of computational burden (i.e., the number of cryptographic and arithmetic operations performed), the total number of bits and messages exchanged during each run in the mutual authentication phase from a user's perspective and compare those results with recently proposed AKA protocols such as A-WAP and MSR-Hybrid [13]. The comparison from a user's perspective is justified because the user is a low-end resource constrained device with limited battery power as compared to the highly resourceful servers. Note that a direct comparison between our protocols and A-WAP and MSR-Hybrid cannot be made because of the added security features that our protocols support. For example, MSR-Hybrid neither provides user anonymity nor encrypts certificates and A-WAP is susceptible to attacks and fails to preserve anonymity. Also note that in the MSR-hybrid and A-WAP protocols, the user and server verify each other's certificates issued by CA in the mutual authentication phase and do not assume the availability of an on-line CA that is similar to the one in PKI infrastructure for verifying the certificates.

In our analysis, we assume a key length of 160-bit and 1024-bit for ECC and Rabin cryptosystem [1], respec-

Table 1: Comparison of UAP and USAP with other recently proposed AKA protocols based on ECC

	USAP	UAP	A-WAP	MSR-Hybrid
Random number generation	1	1	1	1
Fixed point multiplication (FP)	2	4	3	3
Symmetric encryption (E)	1	1	1	1
Symmetric decryption (D)	1	1	1	1
Inverse (s^{-1})	-	1	1	1
Multiplication (ce)	-	2	2	3
Hash (H)	2	2	1	3
Small modular exponentiation	-	-	-	1
Message exchanges	3	4	4	6
Total bits exchanged (number)	1920	2560	2560	4448
Total execution time (msec)	280	440	410	455

tively, and the key deriving function (KDF) and MAC operation referred in [13] are the same as the hash function, H, mentioned in this paper. Table 1 tabulates a comprehensive list of computational and communicational overheads along with execution times for all four AKA protocols that are being compared in the mutual authentication phase from a user's perspective. The timings are based on the implementation of the various operations listed in Table 1 on a Mitsubishis M16C microprocessor, details of which can be found in [13]. In ECC based AKA protocols, the time consuming operation is point multiplication. There are two kinds of point multiplications: the fixed-point multiplication (FP) and random-point (RP) multiplication. The only difference between the two is that the prior knowledge of the point in an FP operation allows for pre-computations, resulting in a lesser execution time than that is required for an RP. Examples of an FP and RP are $g \times P$ and $g \times Q_R$, respectively, where the base point P is a known public parameter and the point Q_R is randomly generated during the run of the protocol. The total execution time for a protocol largely depends upon the number of FP and RP operations. In [13], the authors' report average execution times for FP and RP operations as 130 msec and 480 msec, respectively.

A user in each of the protocols listed in Table 1 is required to perform only FP operations, while the time consuming RP operations are shifted to the server side. The reported execution time for MSR-Hybrid protocol was 455 msec whose breakup is given as follows. The three FP operations and a small modular exponentiation require 390 msec and 45 msec of processing time, thereby leaving room of about 20 msec for the rest of the operations. Considering similar conditions for executing the rest of the protocols, the calculated execution times for USAP, UAP and A-WAP are 280 msec (260 msec for two FP operations + 20 msec for the rest of the operations), 440 msec (420 msec for four FP operations + 20 msec for the rest of the operations), and 410 msec (390 msec for three FP operations + 20 msec for the rest of the operations), respectively.

The number of bits exchanged in USAP is calculated as follows. The user transmits/receives 1920 bits whose breakup is given as: Q_R (320 bits), $C_0 = E(Q_K.x, g_{US}, g_{S_,}, I_S, t_S)$ ($4 \times 160 = 640$ bits), $C_1 = E(Q_K.x, (r_U, s_U), I_U, Q_U.x, t_U, g_S)$ ($6 \times 160 = 960$ bits) in Steps 3, 7 and 11 of Figure 5. We do not transmit the key $Q_K.x$ and thus it is not counted. Similar calculations result in exchange of 2560 bits, 2560 bits and 4448 bits in UAP, A-WAP and MSR-Hybrid, respectively.

In summary, the USAP is efficient in terms of computations and the total number of bits exchanged as compared to UAP, A-WAP, and MSR-Hybrid in addition to being secure and preserving anonymity of both the user and server. The UAP is scalable but only at the cost of server anonymity and increased computational and communicational overhead as compared to USAP.

8 Conclusions and Future Work

In this paper, we have proposed two ECC based authentication and key agreement protocols, the UAP and USAP. Both are suitable for wireless applications. The proposed protocols not only provide a variety of security features but also are efficient in terms of message exchanges and computations. Moreover, they can easily be implemented in a majority of applications and thus are not limited to ad-hoc networks. As a part of our future work, we will concentrate on achieving both server anonymity and scalability in our AKA protocols.

Acknowledgment

This work was supported by the National Science Foundation under Grant No. 0313842 and Grant No. 0542374. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We thank the anonymous reviewers for their valuable suggestions in improving the quality of

this paper.

References

[1] W. Aiello, S. M. Bellovin, M. Blaze, J. Ioannidis, O. Reingold, R. Canetti, and A. D. Keromytis, "Efficient, DoS resistant, secure key exchange for Internet protocols," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 48-58, Nov. 2002.

[2] M. Aydos, E. Savas, and C. K. Koc, "Implementing network security protocols based on elliptic curve cryptography," in *Proceedings of the Fourth Symposium on Computer Networks*, pp. 130-139, Istanbul, Turkey, May 20-21, 1999.

[3] M. Aydos, T. Yanik, and C. K. Koc, "High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor," in *IEEE Proceedings on Communications*, vol. 148, no. 5, pp. 273-279, Oct. 2001.

[4] A. Aziz and W. Diffie, "A secure communications protocol to prevent unauthorized access, privacy and authentication for wireless local area networks," *IEEE Personal Communications*, vol. 1, no. 1, pp. 25-31, 1994.

[5] F. Bao and R. H. Deng, "Privacy protection for transactions of digital goods," in *Proceedings of international conference on information and communications security*, LNCS 2229, pp. 202-215, Springer-Verlag, 2001.

[6] M. J. Beller, L F. Chang, and J. Yacobi, "Privacy and authentication on a portable communications systems," *IEEE Journal on Selected Areas in Communications*, vol. 11, pp. 821-829, Aug. 1993.

[7] S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," *IEEE proceedings of the Symposium on Security and Privacy*, pp. 72-84, May 1992.

[8] E. Bresson, O. Chevassut, and D. Pointcheval "New security results on encrypted key exchange," *7Th International Workshop on Theory and Practice in Public Key Crtptography - PKC 2004*, LNCS 2947, pp. 145-158, Springer-Verlag.

[9] Certicom Research, *Standard for Efficient Cryptography*, SEC 1: Elliptic curve cryptography, version 1.0, Sep. 20, 2000, Certicom Corporation, URL: http://www.secg.org/collateral/sec1.pdf.

[10] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, Nov. 1976.

[11] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," in *Proceedings of the 8th IEEE Computer Security Foundation Workshop*, pp. 24-29, 1995.

[12] D. Harkins and D. Carrel, *The Internet key exchange (IKE)*, Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force, Nov. 1998.

[13] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast authenticated key establishment protocols for self organizing sensor networks," in *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pp. 141-150, San Diego, California, USA, Sep. 2003.

[14] IEEE P1363, *Standard Specifications for Public-key Cryptography*, Draft 13, Nov. 1999.

[15] D. Jablon, "Strong password-only authenticated key exchange," *Comput. Commun. Rev., ACM SIGCOMM*, vol. 26, no. 5, pp. 5-26, Oct. 1996.

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

[17] K. Mangipudi, N. Malneedi, R. Katti, and H. Fu, "Attacks and solutions on Aydos-Savas-Koc's wireless authentication protocol," *Symposium on security and network management, IEEE Global Telecommunications conference*, pp. 2229-2234, Dallas, Texas, Nov.-Dec. 2004.

[18] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Boston, MA Kluwer Academic Publishers, 1993.

[19] A. Perrig, R. Szewczyk, D. Tygar, V. Men, and D. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521-534, 2002.

[20] P. Rogaway, M.Bellare, and D.Boneh, *Evaluation of Security Level of Cryptography*, ECMQVS, SEC1, Jan. 2001.

[21] M. Sandirigama, A. Shimizu, and M. T. Noda, "Simple and secure password authentication protocol (SAS)," *IEICE Transactions on Communications*, vol. E83-B, no. 6, pp. 1363-1365, 2000.

[22] B. Schneier, *Applied Cryptography: Protocols Algorithms, and Source Code in C*, Publisher, John Wiley & Sons, New York, 1996.

[23] K. Shim, "Cryptanalysis of mutual authentication and key exchange for low power wireless communications," *IEEE Communications Letters*, vol. 7, no. 5, pp. 248-250, May 2003.

[24] A. Shimizu, T. Horioka, and H. Inagaki, "A password authentication method for contents communication on the Internet," *IEICE Transactions on Communications*, vol. ESI-B, no. 8, pp. 1666-1673, Aug. 1998.

[25] R. Struik and G. Rasor, "Mandatory ECC security algorithm suite," *submissions to IEEE P802.15 Wireless Personal Area Networks*, Mar. 2002.

[26] H. M. Sun, B.T. Hsieh, and S. M. Tseng, "Cryptanalysis of Aydos et al's ECC-based wireless authentication protocol," in *Proceedings of IEEE International Conference on e-Technology, e-Commerce, and e-Service*, pp. 565-568, Taipei, Taiwan, Mar. 2004.

[27] A. Weimerskirch and D. Westhoff, "Identity certified authentication for Ad-hoc networks," in *Proceedings of the 1st ACM Workshop on Security of Ad hoc and Sensor Networks*, pp. 33-40, Fairfax, Virginia, 2003.

[28] S. B. Wilson and A. Menezes, "Authenticated Diffie-Hellman key agreement protocol," in *Proceedings of Selected Areas of cryptography*, pp. 339-361, 1998.

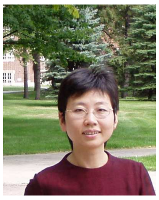
[29] D. S. Wong and A. H. Chan, "Efficient and mutually authenticated key exchange for low power computing devices," in *Proceedings of 7th International Conference on Theory and Applications of Information Security*, LNCS 2248, pp. 272-289, Gold Coast, Australia, Dec. 2001.

[30] M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, vol. 8, no.1, pp. 63-65, Jan. 2004.

[31] F. Zhu, D. S. Wong, A. H. Chan, and R. Ye, "Password authenticated key exchange based on RSA for imbalanced wireless networks," in *Proceedings of the 5th International Conference on Information Security*, LNCS 2433, pp. 150-161, 2002.

Kumar Mangipudi received his Diploma in Electrical and Electronics Engineering in 1994 from State Board of Technical Education, AP, India and his AMIE (B.S) in Electrical Engineering from the Institution of Engineers (India) in 2000. While pursuing his AMIE, he worked on as an Instrumentation Engineer executing the design, testing, and commissioning of instrumentation control systems for process industry at various places in India and Dubai, UAE. In 2002 he obtained his M.S in Electrical Engineering form South Dakota School of Mines & Technology. He is currently a PhD candidate in the department of Electrical and Computer Engineering at North Dakota State University. His interests are in Network Security, Cryptography, VLSI, and Embedded Systems.

Rajendra Katti received his B. Tech degree from the Indian Institute of Technology (Bombay), India in 1983. He received his M.S. in Mechanical Engineering from the University of Idaho in 1985, his M.S. in Electrical Engineering from Washington State University in 1987, where he also earned his PhD in Electrical Engineering in 1991. Dr. Katti teaches courses related to Digital Systems and Computer Architecture. Dr. Katti has received funding from the National Science Foundation in the area of Performance Modeling of Computer Architectures and Cryptography. His interests are in Cryptographic Hardware, Finite Field arithmetic, Fault Tolerant computing and Computer Architecture. He has published over 40 journal and conference papers on these topics. He was a senior design engineer at the Intel Corporation in 2000 and 2001 where he worked in the Design for Testability Group. He has also taught at the Wichita State University in Kansas. He has collaborated with the IBM Almaden Research Center for the development of unidirectional error correcting codes. He is currently an Associate Professor in the Department of Electrical and Computer Engineering at North Dakota State University.



Huirong Fu joined Oakland University as an assistant professor in 2005. Previously, she has been working as an assistant professor at North Dakota State University (NDSU) for three years, and as a post-doctoral research associate at Rice University for more than two years. As a lead professor and the principal investigator in two projects funded by NSF, Dr. Fu has been actively conducting research in the area of information security. Her primary research interests are in information assurance and security, networks, Internet data centers, and multimedia system and services. She has published over twenty papers in top-tier journals and refereed conferences. Aside from research, Dr. Fu has contributed to a substantial growth in new course development at NDSU, especially in the increasingly emerging area of information assurance and security. In addition, due to her great efforts and collaboration with the Designated Center of Academic Excellence in Information Assurance Education at George Washington University, a Portable Educational Network (PEN, a.k.a Mobile Lab) has been successfully built up.