

Attacks and Solutions on Aydos-Savas-Koç's Wireless Authentication Protocol

Kumar Mangipudi¹, Nagaraja Malneedi¹, Rajendra Katti¹, Huirong Fu^{2*}

1. Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58105, USA
2. Department of Computer Science, North Dakota State University, Fargo, ND 58105, USA
{kumar.mangipudi, nagaraja.malneedi, rajendra.katti, huirong.fu}@ndsu.nodak.edu

Abstract- Aydos, Savas and Koç proposed a wireless authentication and key agreement protocol (ASK-WAP) based on Elliptic Curve Cryptography (ECC). We find that this protocol is vulnerable to a man-in-the-middle attack, a denial-of-service attack and an impersonation attack. In this paper, we present the above mentioned attacks on the ASK-WAP. We also propose a variant of ASK-WAP, the User Authentication Protocol (UAP) so that it resists these attacks. Furthermore, we analyze the security and performance of the proposed UAP. The results show that our proposed UAP is much more secured and is also efficient with few message exchanges and less computations.

I. INTRODUCTION

Authenticated key exchange protocols provide the communicating parties a random shared key which can subsequently be used to communicate confidentially. These protocols provide an efficient means of establishing keys and therefore solving the problems associated with key management.

The objective of authenticated key exchange protocols is that the communicating parties execute a scheme and when it is terminated, each of the party should have certain assurance that they know other's true identity and share a new and random session key derived from contribution 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 needs to authenticate to the server often demand for few message exchanges and less computational loads.

There are numerous authentication protocols ever since 1976 Diffie and Hellman proposed the Diffie-Hellman (DH) key exchange based on the discrete logarithm problem [1]. Since the original Diffie-Hellman protocol is vulnerable to the man-in-the-middle attack, modifications were proposed to resist such an attack [2]. Later, Bellare and Merritt presented a password based key exchange protocol for two party communications known as Encrypted Key Exchange (EKE) [3]. Further an efficient and elegant scheme for EKE considered for standardization by the IEEE P1363 Standard working group is AuthA. However, in [12] the authors provided some new security results and also enhanced AuthA,

* Corresponding Author

This material is based upon work supported by the National Science Foundation under Grant No. 0313842. 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.

from denial-of-service attacks. In [13] Zhang showed that Strong Password only Authenticated Key Exchange (SPEKE), a password authenticated key exchange protocol defined in [14] was susceptible to password guessing attack. Wong and Chan [4] proposed a mutually authenticated key exchange for low power computing devices which was later proven insecure against unknown key-share attacks by Shim [15]. Zhu et al presented a password based authenticated key exchange protocol based on RSA for imbalanced wireless networks in [5]. Further, protocols proposed by Beller et al [6], Aziz and Diffie [7] address mutual authentication and key agreement issues for low end devices.

Recently, Elliptic Curve Cryptography (ECC) has gained a lot of attention as ECC implemented devices require less storage, less power, less memory, and less bandwidth as compared to the traditional RSA based cryptosystems. Hence, it is more promising to implement ECC in constrained platforms, such as wireless devices, handheld computers, smart cards, and thin-clients.

Aydos, Savas and Koç proposed a Wireless Authentication and key agreement Protocol (ASK-WAP) based on the ECC for user-server authentication [8]. It has many advantages in terms of bandwidth, storage requirements and computational burden as compared to [4, 5, 6, 7] and was implemented in [9]. In this paper, we show that the ASK-WAP is vulnerable to a man-in-the-middle attack, a denial-of-service attack and an impersonation attack. We propose a variant of ASK-WAP, the User Authentication Protocol (UAP) such that it resists these attacks. We also analyze the security and performance of UAP. The rest of the paper is organized as follows. Section II includes details of the ASK-WAP proposed in [8, 9]. The above mentioned attacks and the proposed UAP are detailed in Section III and Section IV, respectively. The security and performance analysis of the UAP are described in Section V followed by conclusions in Section VI.

II. ASK-WAP

In this section, we briefly describe the two phases of ASK-WAP. First, the initialization phase is an offline process in which the server and the users obtain certificates (r , s), identities (I), and expiration dates (t) for their public keys from a trusted third party known as Certification Authority (CA) through a secure channel. The signed certificate pair (r , s) is computed by using an Elliptic Curve Digital Signature Algorithm (ECDSA). More details on the ECDSA can be

found in [11]. The obtained certificate expires at the end of the expiration date. As such obtaining certificate (digital signatures) from the CA is a one time process during the validity of the certificate. Second, the mutual authentication phase is an online process in which the user and server authenticate each other whenever they want to communicate securely. For the rest of our discussion we use the subscripts CA, U, S and A to indicate the CA, a user, the server and an adversary, respectively.

A. Initialization Phase

First, an elliptic curve is defined over Galois Field $GF(p)$ (where p is the characteristic of the base field) with suitable coefficients. Then a base point $P = (P.x, P.y)$ (where $P.x$ and $P.y$ are the x and y co-ordinates of the point P , respectively) of large order n , belonging to this elliptic curve group is selected and made public to all the 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}) .

User	CA
1. Choose $d_U \in \{2, n-2\}$	Choose $k_U \in \{2, n-2\}$
2. $Q_U = d_U \times P$	$R_U = k_U \times P$
3. Send Q_U	→ Receive Q_U
4.	Choose unique I_U and t_U
5.	$r_U = R_U \cdot x$
6.	$s_U = k_U^{-1}(H(Q_U \cdot x, I_U, t_U) + d_{CA} r_U)$
7. Receive $Q_{CA}, I_U, (r_U, s_U), t_U$	← Send $Q_{CA}, I_U, (r_U, s_U), t_U$
8. $e_U = H(Q_U \cdot x, I_U, t_U)$	
9. Store $Q_U, Q_{CA}, I_U, (r_U, s_U), t_U, e_U$	

Figure 1: User initialization protocol

1) User initialization protocol: To obtain a certificate and the public key of CA, a user U executes the user initialization protocol as described in Fig. 1. First, the user randomly chooses a private and public key pair (d_U, Q_U) and sends its public key Q_U to CA. Then CA also chooses a random temporary key pair (k_U, R_U) , a unique identity I_U and an expiration date t_U for the user and applies a one-way hash function H on the concatenated value of $Q_U \cdot x$ (x co-ordinate of the point Q_U , the user's public key), I_U , and t_U , i.e., $H(Q_U \cdot x, I_U, t_U)$. Finally, CA computes $s_U = k_U^{-1}(H(Q_U \cdot x, I_U, t_U) + d_{CA} r_U)$ (where k_U^{-1} is the multiplicative inverse of k_U such that $k_U k_U^{-1} \equiv 1 \pmod{n}$ and r_U is the x co-ordinate of the point R_U) and sends Q_{CA}, I_U, t_U and the certificate (r_U, s_U) to the user. The user receives $Q_{CA}, I_U, (r_U, s_U), t_U$ from CA, computes the hash value $e_U = H(Q_U \cdot x, I_U, t_U)$ and stores $Q_U, Q_{CA}, I_U, (r_U, s_U), t_U$ and e_U .

2) Server initialization protocol: Fig. 2 shows the server initialization protocol. The server S obtains its certificate (r_S, s_S) , the public key of CA Q_{CA} , the assigned identity I_S , and the expiration date t_S from CA; computes $e_S = H(Q_S \cdot x, I_S, t_S)$; and stores $Q_S, Q_{CA}, I_S, (r_S, s_S), t_S$ and e_S .

Server	CA
1. Choose $d_S \in \{2, n-2\}$	Choose $k_S \in \{2, n-2\}$
2. $Q_S = d_S \times P$	$R_S = k_S \times P$
3. Send Q_S	→ Receive Q_S
4.	Choose unique I_S and t_S
5.	$R_S = R_S \cdot x$
6.	$s_S = k_S^{-1}(H(Q_S \cdot x, I_S, t_S) + d_{CA} r_S)$
7. Receive $Q_{CA}, I_S, (r_S, s_S), t_S$	← Send $Q_{CA}, I_S, (r_S, s_S), t_S$
8. $e_S = H(Q_S \cdot x, I_S, t_S)$	
9. Store $Q_S, Q_{CA}, I_S, (r_S, s_S), t_S, e_S$	

Figure 2: Server initialization protocol

B. Mutual Authentication Phase

The mutual authentication phase is executed in real time, i.e., whenever a service is requested by the user or server. For clarity we divide the mutual authentication phase described in [8] into three sub-phases namely the mutual key agreement, the certificate verification, and the session key generation. Fig. 3a shows the mutual key agreement sub-phase between a user

User	Server
1. Send Q_U	→ Receive Q_U
2.	Generate a random number $g_S \in \{2, n-2\}$
3. Receive Q_S, g_S	← Send Q_S, g_S
4. $Q_K = d_U \times Q_S = (d_U d_S) \times P$	$Q_K = d_S \times Q_U = (d_S d_U) \times P$
5. $Q_K \cdot x$: Mutually agreed key	$Q_K \cdot x$: Mutually agreed key
a : Sub-phase 1 (Mutual key agreement)	
User	Server
6. Generate a random number $g_U \in \{2, n-2\}$	
7. $C_0 = E(Q_K \cdot x, (r_U, s_U), t_U, e_U, g_U, g_S)$	
8. Send C_0	→ Receive C_0
9.	$D(Q_K \cdot x, C_0)$: Valid g_S, t_U ?
10.	$C_1 = E(Q_K \cdot x, (r_S, s_S), t_S, e_S, g_U)$
11. Receive C_1	← Send C_1
12. $D(Q_K \cdot x, C_1)$: Valid g_U, t_S ?	
13. $c = s_S^{-1}$	$c = s_U^{-1}$
14. $u_1 = ce_S$	$u_1 = ce_U$
15. $u_2 = cr_S$	$u_2 = cr_U$
16. $R = u_1 \times P + u_2 \times Q_{CA}$	$R = u_1 \times P + u_2 \times Q_{CA}$
17. $v = R \cdot x$	$v = R \cdot x$
18. if $v \neq r_S$, then abort	if $v \neq r_U$, then abort
b: Sub-phase 2 (Certificate verification)	
User	Server
19. $k_M = H(Q_K \cdot x, g_U, g_S)$	$k_M = H(Q_K \cdot x, g_U, g_S)$
20. k_M is the unique session key	k_M is the unique session key
c: Sub-phase 3 (Session key generation)	

Figure 3: Mutual authentication phase

U and the server S . Initially the initiating party, (user) sends its public key Q_U to the initiated party (server). Then the server

generates a random number g_s and sends its public key Q_s , and g_s to the user. Finally, the user and server perform $d_u \times Q_s$ and $d_s \times Q_u$, respectively, to agree upon a mutual key $Q_{k,x}$ (x coordinate of the point Q_k). The following two sub-phases, the certificate verification (Fig. 3b) and session key generation (Fig. 3c), rely on this mutually agreed key $Q_{k,x}$.

In the second sub-phase as shown in Fig. 3b, the user generates a random number g_u , uses a symmetric key encryption algorithm E to encrypt its certificate (r_u, s_u) , the expiration time t_u , the hashed value e_u , and the random numbers g_u, g_s with the mutually agreed key $Q_{k,x}$ to obtain C_0 and sends C_0 to the server. The server decrypts C_0 using a decryption algorithm D with the mutually agreed key $Q_{k,x}$ and checks for the presence of g_s and the validity of t_u . If both tests are valid then the server encrypts its certificate (r_s, s_s) , t_s, e_s and the user's random number g_u to obtain C_1 and sends C_1 to the user. The user checks for the presence of g_u and the validity of t_s . Both parties verify each other's certificate in steps 13 through 18 of Fig. 3b. If invalid, they abort the protocol, otherwise they derive a unique session key k_M by computing the hash on $Q_{k,x}, g_u$ and g_s in the third sub-phase, i.e., the session key generation phase (Fig. 3c).

C. Security of ASK-WAP

Airing the public keys in Fig. 3a does not impose any threat to the system's security because this involves in solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDLP is defined as calculating d given the points Q and P where $Q = d \times P$ [10]. Encrypting the certificates protects from eavesdropping and spoofing attacks. The user and the server achieve anonymity by not exchanging their identities. The random numbers in the protocol guarantee freshness and prevent replay attacks. The reader is referred to [8] for further details.

III. ATTACKS

There is no explicit binding of the user's or the server's identity with their public keys in the ASK-WAP. This is because the user/server computes the hash value $e = H(Q_x, I, t)$ and further they do not exchange their identities. The user and the server without any previous knowledge of each other's public key agree upon a mutual key $Q_{k,x}$ (Fig. 3a) for encryption and decryption of their individual certificates, random numbers, hashed values and expiration dates.

Though both parties verify each other's certificate in the certificate verification sub-phase (Fig. 3b), an adversary "A" who overhears the mutual authentication can establish himself between the user and server. He establishes a key with the user and another key with the server during the mutual key agreement sub-phase (Fig.3a). However, to proceed further down the protocol, the adversary should have a valid certificate from the CA. Hence, the ASK-WAP is vulnerable to internal attacks where the adversary is from within the system. The rest of this section presents various attacks in the mutual authentication phase of ASK-WAP (Fig. 3).

A. Man-in-the-Middle Attack

Similar to the mutual authentication phase we describe the *man-in-the-middle* attack in three sub-phases. We assume that the user, the adversary and the server have executed the initialization protocol and obtained their respective certificates (r, s) , the public key of CA, assigned identities, and expiration dates from CA. As shown in Fig. 4, a user U initiates the protocol by sending his public key to the server S . An adversary A , who overhears the mutual authentication, bridges himself between the user and the server. It is easy to see the attack being transparent to both the user and server from Fig. 4a through Fig. 4c.

B. Denial-of-Service Attack

An adversary can also launch a *Denial-of-Service* (DoS) attack by simply sending an invalid certificate to the user or by masquerading the user's random number in the certificate verification sub-phase (Fig. 4b). Thus, the user is unable to access the server and hence a DoS attack succeeds.

C. Impersonation Attack

Assume that the adversary has the same functionality as that of the server. Once an adversary bridges himself between the user and the server he may send an invalid certificate to the server or stop responding to the server, but keep communicating with the user. Thus, there is an *impersonation attack* as the user believes that he is communicating with the server, which in the real world is not happening due to the presence of an adversary.

IV. PROPOSED SOLUTION

In this section, we propose a variant of ASK-WAP, the User Authentication Protocol (UAP) where a user needs to authenticate to the server. Similar to the ASK-WAP we describe UAP in two phases: the initialization phase (an offline process through a secure channel) and the user authentication phase (an online process executed whenever a user wants to communicate securely with the server). The rest of this section contains the details of UAP. Note the bold faced steps in Fig.5 through Fig. 7 indicate the differences between the proposed UAP and ASK-WAP.

A. Proposed Initialization Phase

The initialization phase in the proposed UAP has two different initialization protocols: the server initialization protocol defined for the server and the user initialization protocol defined for the user as compared to the ASK-WAP which has the same initialization protocol to both users and server.

1) Server initialization protocol: Fig. 5 shows the server initialization protocol with the CA. The server generates a random public and private key pair (d_s, Q_s) , then sends its public key 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 , sends its public key Q_{CA}, I_s and t_s to the server. The CA stores the server's public key Q_s and expiration date t_s . Finally, the

server receives Q_{CA} , I_S and t_s and stores d_s^{-1} , Q_{CA} , I_S and t_s . Note the server obtains the expiration date t_s instead of the certificate (r_s, s_s) from CA. Once the date t_s expires, the server

is required to perform the server initialization protocol to obtain another valid expiration date.

User	Adversary	Server
1. Send Q_U	→ Receive Q_U	
2.	Send Q_A	→ Receive Q_A
3.		Generate a random number g_s $g_s \in \{2, n-2\}$
4.	Receive Q_s, g_s	← Send Q_s, g_s
5.	$Q_{KAS} = d_A \times Q_S = (d_A d_S) \times P$	$Q_{KAS} = d_s \times Q_A = (d_s d_A) \times P$
6.	$Q_{KAS.x}$: Mutually agreed key	$Q_{KAS.x}$: Mutually agreed key
7.	Generate a random number $g_A \in \{2, n-2\}$	
8. Receive Q_A, g_A	← Send Q_A, g_A	
9. $Q_{KUA} = d_U \times Q_A = (d_U d_A) \times P$	$Q_{KUA} = d_A \times Q_U = (d_A d_U) \times P$	
10. $Q_{KUA.x}$: Mutually agreed key	$Q_{KUA.x}$: Mutually agreed key	

a: Affect of adversary in the mutual key agreement sub-phase

User	Adversary	Server
11. Generate a random number g_U $g_U \in \{2, n-2\}$		
12. $C_0 = E(Q_{KUA.x}, (r_U, s_U), t_U, e_U, g_U, g_A)$		
13. Send C_0	→ Receive C_0	
14.	$D(Q_{KUA.x}, C_0)$: Valid g_A, t_U ?	
15.	$C_1 = E(Q_{KUA.x}, (r_A, s_A), t_A, e_A, g_U)$	
16. Receive C_1	← Send C_1	
17. $D(Q_{KUA.x}, C_1)$: Valid g_A, t_S ?		
18. $c = s_A^{-1}$	$c = s_U^{-1}$	
19. $u_1 = ce_A$	$u_1 = ce_U$	
20. $u_2 = cr_A$	$u_2 = cr_U$	
21. $R = u_1 \times P + u_2 \times Q_{CA}$	$R = u_1 \times P + u_2 \times Q_{CA}$	
22. $v = R.x$	$v = R.x$	
23. if $v \neq r_A$, then abort	if $v \neq r_U$, then abort	
24.	$C_0 = E(Q_{KAS.x}, r_A, s_A, t_A, e_A, g_A, g_s)$	
25.	Send C_0	→ Receive C_0
26.		$D(Q_{KAS.x}, C_0)$: Valid g_s, t_A ?
27.		$C_1 = E(Q_{KAS.x}, (r_s, s_s), t_s, e_s, g_A)$
28.	Receive C_1	← Send C_1
29.	$D(Q_{KAS.x}, C_1)$: Valid g_A, t_S ?	
31.	$c = s_S^{-1}$	$c = s_A^{-1}$
32.	$u_1 = ce_S$	$u_1 = ce_A$
33.	$u_2 = cr_S$	$u_2 = cr_A$
34.	$R = u_1 \times P + u_2 \times Q_{CA}$	$R = u_1 \times P + u_2 \times Q_{CA}$
35.	$v = R.x$	$v = R.x$
36.	if $v \neq r_s$, then abort	if $v \neq r_A$, then abort

b: Affect of adversary in the certificate verification sub-phase

User	Adversary	Server
37. $k_{MUA} = H(Q_{k.x}, g_U, g_A)$	$k_{MUA} = H(Q_{k.x}, g_U, g_A)$	
38. k_{MUA} is the unique session key	k_{MUA} is the unique session key	
39.	$k_{MAS} = H(Q_{KAS.x}, g_A, g_s)$	$k_{MAS} = H(Q_{KAS.x}, g_A, g_s)$
40.	k_{MAS} is the unique session key	k_{MAS} is the unique session key

c: Affect of adversary in the session key generation sub-phase

Figure 4: Affect of adversary in the mutual authentication phase in ASK-WAP

2) User initialization protocol: All users perform the user initialization protocol with the CA as shown in Fig. 6 when they first subscribe. Similar to ASK-WAP, the 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 the public key of the server. Then CA sends the assigned identity I_u , an expiration date t_u , the generated certificate (r_u, s_u) along with the server's public key Q_s and the server's public key expiration date t_s to the user. Note the CA does not send its public key Q_{CA} to the user as in the ASK-WAP. Distributing the server's public key to all the users ensures that every user knows the server's public key prior to the authentication scheme.

Server	CA
1. Choose $d_s \in \{2, n-2\}$	
2. $Q_s = d_s \times P$	
3. Send Q_s	→ Receive Q_s
4. Calculate d_s^{-1}	Choose unique I_s and t_s
5. Receive Q_{CA}, I_s, t_s	← Send Q_{CA}, I_s, t_s
6. Store $d_s^{-1}, Q_{CA}, I_s, t_s$	Store Q_s, t_s

Figure 5: Proposed server initialization protocol

User	CA
1. Choose $d_u \in \{2, n-2\}$	Choose $k_u \in \{2, n-2\}$
2. $Q_u = d_u \times P$	$R_u = k_u \times P$
3. Send Q_u	→ Receive Q_u
4.	Choose unique I_u and t_u
5.	$r_u = R_u \cdot x$
6.	$s_u = k_u^{-1}(H(Q_u \cdot x, I_u, t_u) + d_{CA} r_u)$
7. Receive $Q_s, (I_u, r_u), s_u, t_u, t_s$	← Send $Q_s, (I_u, r_u), s_u, t_u, t_s$
8. $e_u = H(Q_u \cdot x, I_u, t_u)$	
9. Store $Q_u, Q_s, (I_u, r_u), s_u, t_u, t_s$	

Figure 6: Proposed user initialization protocol

B. Proposed User Authentication Phase

The proposed user authentication phase is executed in real time, i.e., whenever the user wants to set up a secure communication with the server. Fig. 7 shows the user authentication phase. The user U, generates a random number g_u , calculates $Q_R = g_u \times Q_s$, and transmits Q_R to the server. Transmitting Q_R is not of security concern as this is similar to airing the public key. The user also calculates $Q_K = g_u \times P$ to obtain the mutually agreed key $Q_{K \cdot x}$. First, the server performs $d_s^{-1} \times Q_R$ to obtain the mutually agreed key $Q_{K \cdot x}$. Then the server encrypts the random numbers g_s, g_{us} (g_{us} is the same as the key, required for data freshness) and the expiration date t_s with the mutually agreed key and finally sends this encrypted value C_0 to the user. We find it redundant for the user to verify the server's certificate as the server's public key is obtained from CA. However, the user decrypts C_0 , checks for the presence of the random number g_{us} and compares the server's expiration date t_s with that of the server's expiration date supplied by CA during the user initialization protocol. If valid, the user obtains C_1 by encrypting its certificate (r_u, s_u) , the expiration date t_u , the hashed value e_u , and the server's

random number g_s with the mutually agreed key. Then the server decrypts C_1 , checks for the presence of the random number g_s and user's expiration date t_u . If the check fails it aborts the protocol, otherwise it verifies the user's certificate. If valid, both parties generate the unique session key k_M and destroy $Q_{K \cdot x}, g_u$, and g_s from their memory.

User	Server
1. Generate a random number $g_u \in \{2, n-2\}$	
2. $Q_R = g_u \times Q_s = (g_u d_s) \times P$	Generate a random number $g_s \in \{2, n-2\}$
3. Send Q_R	→ Receive 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. $Q_{K \cdot x}$: Mutually agreed key	$Q_{K \cdot x}$: Mutually agreed key
6. $g_{us} = Q_{K \cdot x}$	$g_{us} = Q_{K \cdot x}$
7.	$C_0 = E(Q_{K \cdot x}, g_{us}, g_s, t_s)$
8. Receive C_0	← Send C_0
9. $D(Q_{K \cdot x}, C_0)$: Valid g_{us}, t_s ?	
10. $C_1 = E(Q_{K \cdot x}, r_u, s_u, t_u, e_u, g_s)$	
11. Send C_1	→ Receive C_1
12.	$D(Q_{K \cdot x}, C_1)$: Valid g_s, t_u ?
13.	$c = s_u^{-1}$
14.	$u_1 = c e_u$
15.	$u_2 = c r_u$
16.	$R = u_1 \times P + u_2 \times Q_{CA}$
17.	$v = R \cdot x$
18.	if $v \neq r_u$, then abort
19. $k_M = H(Q_{K \cdot x}, g_s)$	$k_M = H(Q_{K \cdot x}, g_s)$
20. k_M is the unique session key	k_M is the unique session key
21. Destroy $Q_{K \cdot x}, g_s$	Destroy $Q_{K \cdot x}, g_s$

Figure 7: Proposed user authentication phase

V. ANALYSIS

In this section, we analyze the security and compare the performance of our proposed UAP with the ASK-WAP in terms of number of computations, messages exchanged, and required storage.

A. Security Analysis

Our proposed UAP shares the same security features of the ASK-WAP, in addition to having extra security features to resist the attacks (cf. Section IV).

- 1) Shared Features: The following are the features common to both UAP and ASK-WAP.
 - The presence of random numbers guarantees freshness and prevents replay attacks.
 - Encrypting the certificates and expiration dates prevents eavesdropping and spoofing attacks.
 - The user anonymity is preserved.
- 2) Additional Features: Our proposed UAP possesses the following additional features as compared to the ASK-WAP.

- The CA distributes the server’s public key to all the users. Thus the users know the server’s public key prior to the authentication phase and hence, is secure from the above mentioned attacks as it does not allow the adversary to establish himself as against to ASK-WAP which allows the adversary to establish himself between the user and the server (refer to steps 1 through 10 of Fig. 4) and thus mounting the attacks.
- Mutually agreed key between the user and the server varies every time the user authentication phase is initiated, in contrast to the ASK-WAP where the same mutually agreed key is established as long as the public keys of both the user and the server are valid.

Table 1: Performance comparison of user’s computational load in the authentication phase of UAP and ASK-WAP

	UAP	ASK-WAP
Random number generation	1	1
Point multiplication ($d \times P$)	2	1
Symmetric Encryption (E)	1	1
Symmetric Decryption (D)	1	1
Inverse (s^{-1})	-	1
Multiplication (ce)	-	2
$u_1 \times P + u_2 \times Q_{CA}$	-	1
Hash (H)	1	1

B. Performance Analysis

We compare UAP and ASK-WAP in terms of computational burden, memory and the number of message exchanges required in the authentication phase. Table 1 shows the comparison of computational loads from the user’s perspective. It is clear from Table 1 that the user is required to perform few computations in UAP as compared to the user in ASK-WAP. From Fig. 7, the server in UAP has similar computational load as that of the server in ASK-WAP. The user stores the server’s public key Q_s , instead of storing the CA’s public key Q_{CA} . Additionally, the user is required to store the server’s expiration date t_s (refer to Fig. 6). Thus, the user requires little extra memory to store t_s (very few bits). Finally, there are only three message exchanges between the user and the server as compared to ASK-WAP which has four.

VI. CONCLUSIONS

In this paper, we have presented three simple attacks on the Aydos-Sava-Koç’s wireless authentication and key agreement protocol. We have also proposed a variant of ASK-WAP, the UAP that resists the man-in-the-middle, DoS and

impersonation attacks. In addition to this, we have analyzed the security and performance of the proposed UAP. The results show that our proposed UAP is much more secured and is also efficient with few message exchanges and less computations.

REFERENCES

- [1] W. Diffie and M. E. Hellman, “New directions in cryptography”, *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, Nov 1976.
- [2] S. B. Wilson and A. Menezes, “Authenticated Diffie-Hellman key agreement protocol”, *Proceedings of Selected Areas of cryptography*, pp. 339-361, 1998.
- [3] 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.
- [4] D. S. Wong and A. H. Chan, “Efficient and mutually authenticated key exchange for low power computing devices”, *Proceedings of 7th International conference on theory and Applications of Information Security*, vol. 2248 pp. 272-289, Gold Coast, Australia, December 9-13, 2001.
- [5] F. Zhu, D. S. Wong, A. H. Chan, and R. Ye, “Password authenticated key exchange based on RSA for imbalanced wireless networks”, *Proceedings of the 5th International Conference on Information Security*, Lecture Notes on Computer Science, vol. 2433, pp. 150-161, 2002.
- [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, August 1993.
- [7] 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.
- [8] M. Aydos, B. Sunar, and C. K. Koc, “An elliptic curve cryptography based authentication and key agreement protocol for wireless communication”, *2nd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Dallas, Texas, October 30, 1998.
- [9] M. Aydos, T. Yanik, and C. K. Koc, “High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor”, *IEEE Proceedings on Communications*, vol. 148 no. 5, pp. 273-279, October 2001.
- [10] A. J. Menezes, “Elliptic curve public key cryptosystems”, Boston, MA Kluwer Academic Publishers, 1993.
- [11] IEEE P1363. Standard specifications for public-key cryptography, Draft 13, November 1999.
- [12] E. Bresson, O. Chevassut, and D. Pointcheval “New security results on encrypted key exchange”, *7th International Workshop on Theory and Practice in Public Key Cryptography – PKC 2004*. F. Bao, R. Deng and J. Zhou Eds, Springer-Verlag, LNCS 2947 pp 145-158.
- [13] M. Zhang, “Analysis of the SPEKE password-authenticated key exchange protocol”, *IEEE Communications Letters*, vol 8, no.1, pp 63-65, January 2004.
- [14] D. Jablon, “Strong password-only authenticated key exchange”, *Comput. Commun. Rev., ACM SIGCOMM*, vol 26, pp. 5-26, October 1996.
- [15] 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.