Towards a Secure Mutual Authentication and Key Exchange Protocol for Mobile Communications

Yi-Jun He
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, NT, Hong Kong
yjhe@cse.cuhk.edu.hk

Moon-Chuen Lee
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, NT, Hong Kong
mclee@cse.cuhk.edu.hk

***Abstract*—We analyze several mutual authentication and key exchange protocols (MAKEPs), and present a number of essential properties of the protocols for secure mobile communications. To address the weaknesses of existing protocols, we propose an improved version of MAKEP known as EC-MAKEP. Besides supporting the essential features present in the existing protocols, the proposed protocol also provides the user anonymity and forward secrecy properties that many of the existing protocols do not support. Further, the proposed protocol compares favorably with ES-MAKEP, an improved version of the early MAKEPs, in terms of computation cost and communication bandwidth. In addition, EC-MAKEP supports an implicit authentication of server and a dual authentication of client.**

Keywords-MAKEP; Elliptic Curve; forward secrecy; user anonymity.

I. INTRODUCTION

The mutual authentication and key exchange protocol (MAKEP) presented in [1] aimed to provide secure authentication between a user and a server, and to enable them to determine jointly a session key. This session key can then be used to establish a secure communication channel between the user and the server. So far, different types of mutual authentication and key exchange protocols have been proposed. In general, such protocols could be grouped into two categories: public key based protocols, and symmetric key based protocols.

In the public key based protocols [2], each party holds a pair of private and public keys. The private key is kept by the owner, and used either for decryption (confidentiality), or encryption (signature) of messages. The public key is published to be used for the reverse operation. They provide arbitrarily high levels of security and do not require an initial private key exchange. However, when implemented on low-power wireless devices, these operations can be very inefficient. Further, these protocols require the support of the Public-Key Infrastructure (PKI) for authentication purpose; and the high complexities of the underlying crypto operations could prevent the public key cryptosystems from being widely deployed in most of the applications running on low-power wireless devices.

In symmetric key based protocols [3], a common key is used by both communication partners for encryption and decryption. The symmetric key crypto algorithms are much faster than the public key crypto algorithms when implemented in wireless devices. However, a symmetric key based protocol requires the two communication entities to share a long-lived key before starting their communication. So, how to securely distribute the long-lived key to each communication entity is an important issue. If it is not securely distributed, the attackers could make use of the long-lived key to break the protocol. Moreover, in order to communicate with different entities, each entity needs to possess a set of distinct long-lived keys for communicating with different partners. Hence key management is another problem when deploying symmetric key based protocols.

In general, a good mutual authentication and key exchange protocol should possess the following properties [4,5,6]:

User Anonymity: In mobile communications, most users require their identity and private information to be kept confidential. This property assures the anonymity of a mobile user and prevents an attacker of a malicious entity from getting hold of confidential information of an individual user.

Forward secrecy: This property ensures that if the long-term private keys of one or more of the entities are compromised, the secrecy of previously established session keys should not be affected.

Data Integrity: A system with this property implies that it can verify if any data received from the sender has been modified during transmission.

Known-key security: If one session key has been obtained by an adversary, the protocol should ensure that neither the private keys nor other session keys (past or future) would be compromised as a result.

Key control: The secret session key between any two entities should be jointly determined; neither entity can predetermine the session key.

Key-compromise impersonation resilience: If the long-term private key of an entity A is compromised, the protocol would allow the adversary to impersonate A; but it should not allow the adversary to impersonate other entities to A.

Unknown key-share resilience: An entity A cannot be coerced into sharing a key with any entity C when in fact A

RGC Direct Grant with Project ID 2050347

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
 ISBN 978-963-9799-18-9
DOI 10.4108/wiopt.2008.3402

thinks that it is sharing the key with another entity *B*.

In addition, based on the considerations for a mobile communication environment, the protocol should allow for the limitations of mobile devices, including low wireless bandwidth, and limited computation power. The list below could be seen as performance measurement criteria [7].

1. *Minimum number of passes*: To reduce latency time, the number of message exchanges required between entities should be kept minimal.
2. *Efficient usage of bandwidth*: Due to the low bandwidth in the mobile communications, the total number of bits transmitted should be kept as small as possible.
3. *Limited computational capability*: Since the mobile device computation capability is generally limited, the protocol should reduce the number of cryptographic operations, and employ more offline computations than online computations as much as possible.

This paper proposes a relatively secure and efficient MAKEP for mobile communications between individual lower power wireless devices and a powerful server. The proposed protocol aims at being equipped with the above mentioned essential security features.

The organization of this paper is as follows. In section 2, we first introduce the original MAKEP proposed by D. S. Wong and A. H. Chan [8]; then we present several improved MAKEPs proposed in the literature [9,10,11]. Section 3 provides a brief review and the cryptanalysis of ES-MAKEP [11]; it is the latest improved version of MAKEP and is relatively more secure and efficient than other improved MAKEPs. In section 4, we present our proposed improved MAKEP known as EC-MAKEP which has addressed the weaknesses of the contemporary MAKEPs [8,9,10,11]. In section 5, security and performance analysis of EC-MAKEP are presented. Finally, Section 6 concludes the paper.

II. RELATED WORK

In this section, we analyzed several existing MAKEPs which are relatively more efficient than those public-key based protocols, and more secure than the symmetric key protocols.

2.1 Server-specific MAKEP

In 2001, Duncan S. Wong *et al*. proposed a Server-specific MAKEP [8] (say, Ss-MAKEP) for secure wireless communications between a low-power wireless device (client) and a powerful base station (server). To reduce the high computation cost incurred in public key based cryptographic operations, this protocol avoids using any of such operations on the client side. Instead, it uses efficient symmetric key based operations. Furthermore, it does not need to maintain a secure database of the long-lived keys of its clients though such a database is usually required by a conventional symmetric key based scheme. So, it lets each client keep a certified long-lived key and send it securely to the server whenever the protocol is executed; thus the cost of maintaining and searching through such a database could be eliminated. Further, client can change its key anytime by obtaining a new certificate from CA without involving the server. Therefore there is no key synchronization problem between client and server. However, this protocol requires the client to possess a certificate specific to the server before communicating with the server. In other words, each certificate is server-specific. So, a client has to keep a lot of distinct certificates in order to communicate with different servers. As a mobile device has limited storage, it may not be practical to implement the protocol. Moreover, as the protocol requires the client to send its long-lived symmetric key to server when executing the protocol, the server could get to know the long-lived symmetric key of the client; this makes it possible for a malicious server to impersonate the client.

2.2 Linear MAKEP

Duncan S. Wong *et al*. also presented another protocol called Linear MAKEP [7] (say, L-MAKEP) which is an improved version of Ss-MAKEP. In Ss-MAKEP, client and server share a symmetric key K_A, and client authentication is no more than a conventional symmetric key based authentication scheme. So a malicious server is able to impersonate its own clients. However, in L-MAKEP, the server does not share any key with the client; thus, it prevents any server from impersonating its own clients. In addition, instead of keeping numerous distinct certificates, each client generates a number of key pairs which are then sent to the Trusted Authority (TA) to obtain a signature for each key pair. The client then keeps the signatures for the key pairs. Before communicating with the server, the client uses one key pair and its TA signature to compute a certificate to be sent to the server. The total number of times that the client can run the protocol would be limited by the total number of key pairs it possesses unless the client would generate more key pairs and obtain the signatures from the TA for these keys. Further, both the L-MAKEP and the Ss-MAKEP schemes cannot resist unknown key-share attack since the messages transmitted in the protocol do not include at the time both the sender identity and the recipient identity. Therefore, attackers can pretend to be the client or the server to launch the attack.

2.3 Improved Ss-MAKEP and Improved L-MAKEP

Usually, the unknown key-share attack can be prevented by requiring each entity to show the CA that both the private key and the public key before issuing the certificate, since the attacker cannot show the corresponding private key[20]. However, in certain practical situations, the CA may not require the private key. Kyungah Shim [8] proposed an improved Ss-MAKEP (say, ISs-MAKEP) and an improved L-MAKEP (say, IL-MAKEP) to address the problem by including the identities of the sender and recipient in the encrypted messages. Thus, after decrypting a message, the recipient can detect the presence of the attack by checking if the identity in the certificate matches the identity in the encrypted message. Further, when compared with the original protocols, the improved versions did not involve additional computation cost. However, it is reported in [8] that IL-MAKEP still cannot resist the man-in-the-middle attack.

2.4 I-MAKEP

Jim-Ke Jan *et al*. [9] proposed an improved MAKEP (I-MAKEP) based on Girault's method [21] to resist the malicious attacks such as unknown key-share and man-in-the-middle attacks . In Girault's method, the computation of the client public key involves the participation of both the CA and the user; and the client certificate is "embedded" in the public key itself. So client certificates are no longer needed; the client secret key would be computed by the client, and remains unknown to the CA. I-MAKEP adopted this method; it requires the client to keep only one secret key instead of a certificate and many pairs of private keys in the client memory. So, it is different from IL-MAKEP and ISs-MAKEP, which use the certificate for authentication. Therefore, I-MAKEP can resist the man-in-the-middle attack and requires less memory space. In addition, similar to IL-MAKEP, I-MAKEP employs the pre-computation technique to reduce the client computation overhead. However, its online computation cost and the bandwidth requirement are still high. The computation cost, bandwidth requirement, and the number of message exchanges are important considerations when designing key exchange protocols, especially for wireless communications with low power mobile devices. So there is a need for improving the performance of I-MAKEP.

2.5 ES-MAKEP

In 2004, Fuw-Yi Yang *et al*.[10] proposed a protocol known as ES-MAKEP for mobile communications, which involved only 0.1 online modular multiplication on the client side. Its online computation is around ten times faster than the computation of conventional protocols. The number of message exchanges and the message size are smaller than those of the previous protocols Ss-MAKEP, L-MAKEP, ISs-MAKEP, IL-MAKEP and the I-MAKEP. In addition, the proposed protocol can resist not only the unknown key-share attack mentioned above but also the man-in-the-middle attack which Ss-MAKEP and ISs-MAKEP could not resist. More details of the protocol are presented in section 3.

III. CRYPTANALYSIS OF ES-MAKEP

As mentioned above, different approaches have been proposed for improving MAKEPs, such as ISs-MAKEP, IL-MAKEP and ES-MAKEP etc.. However, all of them could not support the security properties forward secrecy, and user anonymity. There is thus a need to develop a new mutual authentication and key exchange protocol to support the foregoing securities features.

In this section, we describe ES-MAKEP in detail, which outperforms the previous protocols in terms of security, message round (no. of messages), message size, server computation cost and client online computation cost. We also analyze the weaknesses of ES-MAKEP, including lacking forward secrecy and user anonymity.

3.1 Brief Review of ES-MAKEP

Before presenting the details of ES-MAKEP, we first introduce the notations and symbols used in [10]:

SK_S: server private key

PK_S: server public key

K: a secret key of symmetric encryption/decryption function

ε_{PK_S} (): an asymmetric encryption function

δ_{SK_S} (): an asymmetric decryption function

E_K(): a symmetric encryption function

D_K(): a symmetric decryption function

h(): a hash function

ID_U: the identification of a client

ID_S: the identification of a server

σ: session key

p, q: a client private key pair

g, n: a client public key pair

$x \| y$: string x concatenates string y

$|n|$: bit length of n

r_{UK}, r_{UF}, r_{UR}: three random numbers selected by client

r_{SK}: a random number selected by server

l: length of session key

Figure 1 depicts the message flows of ES-MAKEP.

Figure 1 depicts the message flows of ES-MAKEP.

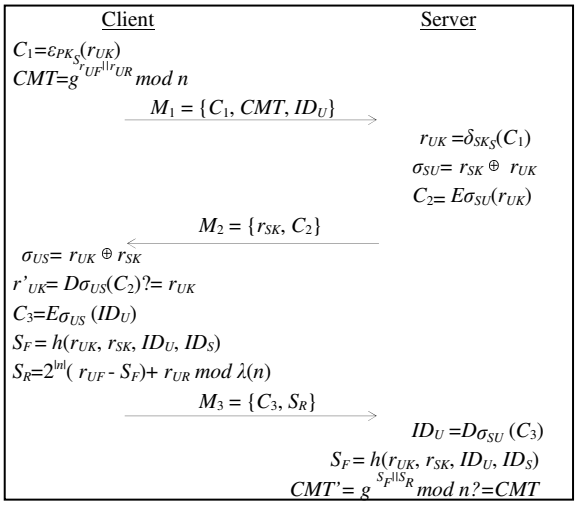

Figure 1. Original ES-MAKEP

The steps of the protocol are as highlighted below.

Step 1: As an offline initialization step, two large prime numbers p and $q \in \{0,1\}^{k/2}$ are randomly chosen such that $p = 2p' + 1$ and $q = 2q' + 1$. Then, the client selects a random value g of order $\lambda(n)$ from the multiplicative group $g \in Z_n^*$, where $n=pq$ and $\lambda(n)=lcm(p-1, q-1)=2p'q'$. Then client announces the public key pair (n,g) to the public, and keeps its private key pair (p,q) as a secret.

Step 2: Client encrypts the random number r_{UK} using the server public key, and computes $CMT=g^{r_{UF}||r_{UR}} \bmod n$. Then, client sends server the message M_1 including C_1, CMT, and client identity information ID_U to ask for initiating a new session.

Step 3: On receiving M_1, server decrypts the ciphertext C_1 to obtain r_{UK}, and calculates the session key σ_{SU} using r_{UK} and the random number r_{SK} it selects. Server also encrypts the random number r_{UK} using the session key. Then, server sends M_2 which includes r_{SK} and C_2 to client.

Step 4: Upon receiving the message M_2, client calculates the session key $\sigma_{US}= r_{UK} \oplus r_{SK}$, and decrypts the ciphertext C_2 to obtain r'_{UK}. Client authenticates server by checking if r_{UK} equals r'_{UK} since only server could compute σ_{SU}. Thus, if messages M_1 and M_2 are successfully transmitted, σ_{SU} and σ_{US} should have the same value. Accordingly, r_{UK} and r'_{UK} should be equal. After authenticating server, client computes the quantities $S_F= h(r_{UK}, r_{SK}, ID_U, ID_S)$ and $C_3=E_{\sigma_{US}}(ID_U)$. Then it solves S_R using (1).

$$2^{|n|}r_{UF} + r_{UR} = 2^{|n|}S_F + S_R \bmod \lambda(n)$$
$$S_R=2^{|n|}(r_{UF} - S_F)+ r_{UR} \bmod \lambda(n) \qquad (1)$$

At last, client sends the response message $M_3 = \{C_3, S_R\}$ to server.

Step 5: Server computes the quantities $S_F= h(r_{UK}, r_{SK}, ID_U, ID_S)$ and $CMT'= g^{S_F||S_R} \bmod n$. Then, server checks if CMT is equal to CMT'. Based on Adi Shamir and Yael Tauman's scheme[11], CMT and CMT' should be equal if all the messages are correctly transmitted.

At this point, ES-MAKEP should have completed the mutual authentication and key exchange process. But from the security perspective, it is not safe enough, as it could not support the security features forward secrecy and user anonymity. Further details of these two problems are presented in the next two subsections.

3.2 Lacking Forward Secrecy

Park, *et. al*. provide a definition for *forward secrecy* in [12] as the following: Even if a long-term private key has been disclosed to an adversary, the session keys established via the protocol using the long-term key would not be compromised. However, ES-MAKEP does not support forward secrecy, since the session key could be computed if the server secret key has been disclosed. Further details of the problem are presented below.

Assume an adversary E is listening to the session of the ES-MAKEP; the server secret key SK_S has been disclosed; and the adversary could obtain C_1. Then it can compute $r_{UK} =\delta_{SK_S}(C_1)$ by using the server secret key SK_S. E could also obtain r_{SK} from M_2; thus the session key σ_{SU} could then be computed as $r_{SK} \oplus r_{UK}$. Since the disclosure of the server secret key SK_S would enable an adversary to compute the session key σ_{SU}, ES-MAKEP does not satisfy the requirement for forward secrecy.

When the above mentioned scenario occurs, the previous session key could be exposed to the attacker. With this session key and those previously intercepted transmitted messages, the attacker can easily get useful information from those messages encrypted using the session key.

3.3 Lacking User Anonymity

Any system supporting user anonymity means that it keeps user secrets confidential or avoids disclosing any confidential user information. Especially in e-business applications, user anonymity is an important issue since online business transactions could incur many security problems if user secrets are disclosed during the process. For instance, an attacker could make use of the user identity to impersonate the user to perform online shopping.

ES-MAKEP suffers from lacking the user anonymity property mentioned above. The user identity ID_U is transmitted in M_1 without any encryption. So, an attacker could obtain M_1 to figure out the ID_U. Thus the user identity could be exposed to the attacker. The attacker could make use of this ID_U to pretend to be a legitimate user and initiate a session with server.

IV. THE PROPOSED PROTOCOL EC-MAKEP

In this section, we introduce our mutual authentication and key exchange protocol known as EC-MAKEP. It possesses the important security features forward secrecy, user anonymity, as well as all the good security features of ES-MAKEP. It is assumed that the server ID has been distributed to its clients before the protocol execution, and that the server has maintained a database to legitimate client IDs. The improved protocol EC-MAKEP is as depicted in Figure 2.

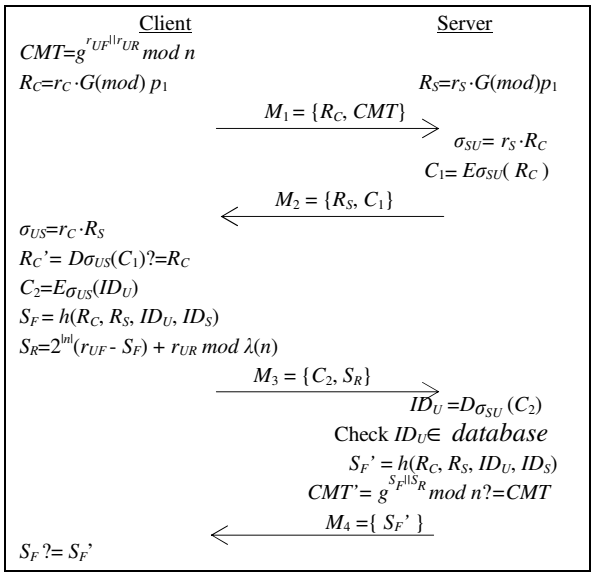

Figure 2. The proposed protocol EC-MAKEP

1. As an offline initialization step, the ECDH algorithm has been preset to use a big prime p_1 and two other parameters a and b satisfying the equation $y^2=(x^3+ax+b) \bmod p_1$, to form an elliptic group $E_{p1}(a,b)$. Then, it chooses the basic point $G=(x,y)$ with order q_1, where

$q_1$is the minimum integer satisfying $q_1 \cdot G = O$, O being a point at infinity. In addition, two large prime numbers p and $q \in \{0,1\}^{k/2}$ are randomly chosen such that $p = 2p' + 1$ and $q = 2q' + 1$. Then, the client selects a random value g of order $\lambda(n)$ from the multiplicative group $g \in Z_n^*$, where $n=pq$ and $\lambda(n)=lcm(p-1, q-1)=2p'q'$. Then client announces the public key (n,g) to the public, and keeps its private key pair (p,q).

2. In order to communicate with the server, the client chooses an integer $r_C < q_1$, $r_{UF} \in_R \{0,1\}^l$ and $r_{UR} \in_R Z_{\lambda(n)}$. Then it computes $R_C = r_C \cdot G (mod) p_1$ through an Elliptic Curve Cryptography (ECC) point multiplication operation. According to the ECC property, R_C is an ECC point. Then client computes $CMT = g^{r_{UF} || r_{UR}} \bmod n$, and sends R_C and CMT to server.

3. Server chooses an integer $r_S < q_1$, and computes $R_S = r_S \cdot B (mod)\ p_1$. According to the ECC property, R_S is an ECC point.

4. After receiving R_C, server computes session key $\sigma_{SU} = r_S \cdot R_C$ and uses the symmetric encryption algorithm $E_K()$ to encrypt R_C with the encryption key σ_{SU}. Then server sends the encrypted value C_1 and R_S to client.

5. Client computes session key $\sigma_{US} = r_C \cdot R_S$, and employs the symmetric decryption algorithm $D_K()$ to decrypt C_1 with the decryption key σ_{US} to obtain $R_{C'}$. If $R_{C'}$ is equal to R_C, σ_{SU} and σ_{US} must be equal; otherwise, the values sent by server to client or the values sent by client to server could have been changed by an attacker.

6. Client encrypts ID_U using session key σ_{US} as the encryption key, and computes the quantities $S_F = h(R_C, R_S, ID_U, ID_S)$, and $S_R = 2^{|n|}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n)$. Then client sends C_2 and S_R to server.

7. Server authenticates the client by checking if the ID_U exists in the client ID database; it then computes the quantities $S_F' = h(R_C, R_S, ID_U, ID_S)$ and $CMT' = g^{S_F' || S_R} \bmod n$. Server compares CMT' with CMT it received from the client, and then sends S_F' to client.

8. Client checks if S_F and S_F' equal.

V. Analysis of EC-MAKEP

The security evaluation of a mutual authentication and key exchange protocol is normally based on the list of security features mentioned in section 1. In addition, computation costs and bandwidth requirements are two other common criteria used to evaluate the performance of a MAKEP implemented in a wireless environment. In the following sub-sections, we analyze our proposed EC-MAKEP based on the above-mentioned features or criteria.

5.1 Implicit Authentication of Server

The implicit authentication of server is done by the client without using directly server ID or certificate. The proposed protocol EC-MAKEP supports an implicit authentication of server as the following. It is assumed that the server ID has been distributed safely to client before protocol execution. If messages M_1 and M_2 are successfully transmitted, client can compute S_F using R_C, R_S, ID_U and ID_S. After obtaining ID_U from client, server computes S_F', and sends it to client. Client compares S_F' received from server with S_F to authenticate the server indirectly because both S_F' and S_F have been computed using ID_S.

5.2 Dual Authentication of Client

Dual authentication here means that besides using client ID to authenticate the client, the client could also be authenticated implicitly. In EC-MAKEP, server first authenticates client by checking if the decrypted client identity ID_U is in its client ID database. After computing $S_F' = h(R_C, R_S, ID_U, ID_S)$, and $CMT' = g^{S_F' || S_R} \bmod n$, the server can authenticate the client for the second time by comparing CMT' and CMT received from the client. As only client knows the secret key pair (p,q), it can compute $\lambda(n)=lcm(p-1, q-1)$ using its secret key pair, and $S_R = 2^{|n|}(r_{UF} - S_F) + r_{UR} \bmod \lambda(n)$. Based on Adi Shamir and Yael Tauman's scheme[11], when $2^{|n|} r_{UF} + r_{UR} = 2^{|n|} S_F + S_R \bmod \lambda(n)$, $CMT = g^{r_{UF} || r_{UR}} \bmod n = g^{S_F || S_R} \bmod n = CMT'$. Thus if CMT and CMT', the server could successfully authenticate the client again since only the client could compute CMT and S_R.

5.3 Security Properties

User Anonymity. In EC-MAKEP, ID_U is not included in M_1 anymore; it is sent in M_3 in an encrypted form $E_{\sigma_{US}}(ID_U)$. As ID_U is encrypted using the session key σ_{US} computed as $r_C \cdot R_S$, only the server could compute the session key and decrypt C_2 to obtain the client identity ID_U. The server can authenticate client by checking if ID_U is within its client ID database. In this way, the client identity information could not be made accessible to an attacker. Thus, the user identity can be kept confidential, and the requirement for user anonymity can be met.

Forward Secrecy. In EC-MAKEP, the client sends a pre-computed ECC point R_C to server. Similarly, the server sends a pre-computed ECC point R_S to client. The session key would then be computed as $r_S \cdot R_C$ by the server, and as $r_C \cdot R_S$ by the client. Now, using $\varepsilon_{PK_S}(r_{UK})$ instead of R_C in message M_1 from client to server, and using r_{SK} instead of R_S in message M_2 from server to client can help both parties to establish a secret session key satisfying the requirement for forward secrecy. Suppose the attacker could somehow get hold of R_C, R_S and the server private key; it still cannot successfully obtain the previous session key because according to the property of the ECC algorithm, the attacker cannot compute r_S or r_C using R_C and R_S. Therefore, the compromise of the long-term private key of the server does not lead to the disclosure of the previous session key. Thus, EC-MAKEP can support forward secrecy.

Data Integrity. A system supporting data integrity implies that it can check if the data received from the client are correct; that is, it can check if the data transmitted to the receiver have been modified. In EC-MAKEP, client can check R_C and R_S by using session key σ_{US} to decrypt C_1 to see if R_C is equal to R_C', since R_S was used to compute σ_{US} and R_C was encrypted using σ_{SU}. If R_C and R_S were successfully transmitted, the session

keys σ_{SU} and σ_{US} should be the same, and R_C should be equal to R_C'. The other messages transmitted in the protocol can also be verified as below. The client first computes $S_F = h(R_C, R_S, ID_U, ID_S)$; it then computes S_R using S_F, which is subsequently sent to the server. Since the parameters R_C, R_S, ID_U, ID_S used in computing S_F have all been transmitted in the network, if any one of them has been modified during the transmission, the S_F calculated by client would not be equal to the one calculated by server. As a result, *CMT* would not be equal to *CMT'*. So, all the data transmitted during the execution of EC-MAKEP can be verified; so the protocol facilitates the validation of data integrity.

Known-key security. With the proposed protocol EC-MAKEP, if the current session key has been compromised, the other session keys (past and future), and the private keys of the client and the server could still be safe. Since session key computation uses a random value of client (server) and a pre-computed value of server (client), and the random value is different in each session, so the past or future session keys have no relation with the current one. Further, as the private keys of the client and server are not involved in the computation of the session key, they would not be compromised even if the session key has been disclosed.

Key control. In the protocol EC-MAKEP, both client and server cannot predetermine the session key being established, because the establishment of the session key involves both a random value and a pre-computed value. Each of the values comes from a different entity, so neither the client nor the server can determine the session key before the communication.

Key-compromise impersonation resilience. With the protocol EC-MAKEP, even if the private key of the client has been exposed, an attacker can impersonate neither the client nor the server because the protocol does not use the private keys of the two entities in the key exchange and authentication process; it uses the pre-computed value R_C and R_S instead.

5.4 Computation Cost and Bandwidth

Computation Cost. To simplify the estimation of the computation cost, we divide the computation cost of the protocol into two parts: offline computation and online computation. R_C, *CMT* and R_S can be computed before the client communicates with server via the proposed protocol; so such computations can be regarded as offline computations. Moreover, the costs of additions, hash operation $h()$, symmetric encryption $E_K()$, and decryption $D_K()$ would not be included since the costs of these operations are much smaller than the cost of the elliptic curve point multiplication operation.

Fuw-Yi Yang and Jinn-Ke Jan [10] presented their analysis of computation cost and bandwidth requirement of ES-MAKEP. By following their approach, we have carried out a similar analysis on our proposed protocol EC-MAKEP, and the results are as presented below. Based on the findings of [14,15], an ECC with 160-bit key length could offer roughly the same level of security as RSA with 1024-bit modulus. As to the modulus exponent function $g^x \bmod n$, we set the length of modulus n equal to 1024bits, where x is a 160-bit random integer. The cost of computing such a modulus exponent function is estimated to be about $1.5|x|$ modular multiplications [16], equal to around 240 modular multiplications ($|x|$ indicates the length of x). When the length of p_1 is 160 bits in the elliptic curve point multiplication function, it would be 8 times faster than modulus exponent computation. So it can be deduced that the computation cost of elliptic curve point multiplication is equivalent to around 29 modular multiplications [16]. Tables I and II present the comparisons of computation costs of EC-MAKEP and ES-MAKEP on the client side and the server side respectively.

Note: MMs denotes the computation cost of a modular multiplication $a*b \bmod n$, where a, b, and n are all set to be 1024 bits.

TABLE I. COMPARISON OF COMPUTATION COSTS ON CLIENT SIDE (MMS)

	EC-MAKEP		**ES-MAKEP**	
	Online	***Offline***	***Online***	***Offline***
Message M_1	0	1805 [a]	0	1778
Message M_2	0	0	0	0
Message M_3	0.1 [b]	0	0.1	0
Total	0.1	1805	0.1	1778

TABLE II. COMPARISON OF COMPUTATION COSTS ON SERVER SIDE (MMS)

	EC-MAKEP		**ES-MAKEP**	
	Online	***Offline***	***Online***	***Offline***
Message M_1	0	29 [c]	1536	0
Message M_2	0	0	0	0
Message M_3	1776 [d]	0	1776	0
Total	1776	29	3312	0

a. Computing R_C requires one ECC point multiplication, with a cost of 29MMs. Computing $CMT=g^{r_{UF}||r_{UR}} \bmod n$ needs 1.5*(160+1024)=1776MMs[17].

b. Fuw-Yi Yang *et al.* [10] show that to compute $S_R=2^{|n|}(r_{UF} - S_F)+ r_{UR} \bmod \lambda(n)$ requires 0.1MMs.

c. Computing R_S requires one ECC point multiplication, with a cost of 29MMs.

d. Computing $CMT'= g^{S_F||S_R} \bmod n$ requires 1.5*(160+1024) = 1776MMs.

On the client side, most of the computation cost in EC-MAKEP is incurred offline and it requires a similar online computation cost as ES-MAKEP. Although EC-MAKEP requires a slightly bigger offline computation cost on client side, the small additional cost for computing R_C and R_S are justifiable since the underlying operations could help to provide forward secrecy, and enhance the security of the protocol. Moreover, the offline computation would be performed only once; so the computation cost can be considered insignificant.

On the server side, EC-MAKEP requires a much smaller online computation cost compared with ES-MAKEP, since in EC-MAKEP, server does not need to do asymmetric decryption operation any more. As a result, EC-MAKEP

reduces the computation burden of the server when the protocol runs.

Bandwidth. Table III shows the bandwidth overheads of EC-MAKEP and ES-MAKEP. As suggested in [18], for practical cryptographic operations, we set $|l| = |ID_U| = |r_{UF}| = |S_F| = 160$ bits.

TABLE III. BANDWIDTH OVERHEADS IN EC-MAKEP AND ES-MAKEP (BITS)

	EC-MAKEP	ES-MAKEP
Message M_1	1184[a]	2208
Message M_2	480[b]	320
Message M_3	1184[c]	1184
Message M_4	160	0
Total	3008	3712

a. R_C is computed by using an ECC point multiplication modulo p_1; so the maximum length of R_C would be $|p_1|$=160bits. *CMT* is computed as $g^{r_{UF}||r_{UR}} \bmod n$; thus the length of *CMT* is equal to $|n|$=1024bits.

b. Similar to R_C mentioned above, the length of R_S could be 160bits. $C_1$is estimated to have a length equal to the length of R_C, which is 160bits. And we set the length of ID_S to be 160bits, equal to its length in ES-MAKEP.

c. As in ES-MAKEP, it is assumed the lengths of both C_2 and ID_U are 160bits, and the length of S_R is 1024bits.

As shown in Table III, when compared with ES-MAKEP, EC-MAKEP reduces the bandwidth requirement by 704bits. A smaller bandwidth requirement is certainly an advantage for low bandwidth wireless communication.

VI. CONCLUSIONS

It is generally agreed that a good mutual authentication and key exchange protocol should possess the following security properties: *user anonymity*, *forward secrecy*, *data integrity*, *known-key security*, *key control*, and *key-compromise impersonation resilience*. In addition, for a protocol used in a wireless environment, its bandwidth requirement and computation cost should be reduced to a minimal amount. We studied the early mutual authentication and key exchange protocol (MAKEP), and the improved MAKEPs; we also identified their weaknesses. The latest improved protocol known as ES-MAKEP addresses many of the security problems of the previous MAKEPs. This paper proposes the protocol EC-MAKEP which has improved on ES-MAKEP. Our security analysis of the proposed protocol shows that EC-MAKEP satisfies all of the major security requirements for a secured MAKEP. It compares favorably with ES-MAKEP in terms of the above mentioned security requirements. For instance, ES-MAKEP does not support forward secrecy and user anonymity; whereas EC-MAKEP supports both of the two features. In addition, it provides implicit authentication of server, and dual authentication of client, making it more secure than the previous protocols. Moreover, when compared with ES-MAKEP, EC-MAKEP has a smaller online computation cost, and requires a smaller bandwidth. The above advantages make EC-MAKEP more suitable for practical implementation in a wireless environment.

ACKNOWLEDGMENT

The work reported in this article has been supported in part by CUHK under RGC Direct Grant with Project ID 2050347.

REFERENCES

[1] Jakobsson, M., & Pointcheval, D, "Mutual Authentication and Key Exchange Protocol for Low Power Devices," In Financial Cryptography (pp. 178–195). Springer-Verlag. 2001.

[2] Harbitter, A.H.&Menase, D.A., "Performance of public-key-enabled Kerberos authentication in large networks," In Proceedings of 2001 IEEE Symposium on Security and Privacy, pp. 170-183,2001.

[3] R.M. Needham and M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," Commun. of the ACM, 21(12):993-999, 1978.

[4] L. Chen, Z. Cheng, and N. P. Smart; "Identity based authentication key agreement protocols from pairings," Proceedings of the 16th IEEE Computer Society Foundations Workshop (CSFW'03).

[5] Certicom's Bulletin of Security and Cryptography. Code and cipher vol.1, no.2. From www.certicom.com/codeandcipher.

[6] Jaegwan Park, Jaeseung Go, and Kwangjo Kim, "Wireless Authentication Protocol Preserving User Anonymity," SCIS 2001, The 2001 Symposium on Cryptography and Information Security. Oiso, Japan, January 23-26, 2001. The Institute of Electronics, Information and Communication Engineers.

[7] Kook-Heui Lee, Sang-Jae Moon, Won-Young Jeong, and Tae-Geun Kim, "A 2-pass authentication and key agreement protocol for mobile communications," JooSeok Song(Ed.): ICISC'99, LNCS 1787, pp. 156-168, 2000.

[8] D. S. Wong and A. H. Chan, "Mutual authentication and key exchange for low power wireless communications," Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force, IEEE, Vol. 1, 2001, pp. 39-43.

[9] K. Shim, "Cryptanalysis of mutual authentication and key exchange for low-power wireless communications," IEEE Communications Letters, Vol. 7, No. 5, pp.248-250, 2003.

[10] J. K. Jan and Y. H. Chen, "A new efficient MAKEP for wireless communications," In Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04), IEEE, Volume 2, pp. 347-350, 2004.

[11] Fuw-Yi Yang and Jinn-Ke Jan, "A Secure and Efficient Key Exchange Protocol for Mobile Communications" Cryptology ePrint Archive 2004/167, July, 2004, http://eprint.iacr.org.

[12] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," Advances in Cryptology-CRYPTO'01, LNCS 2139, pp. 355-367, 2001.

[13] DongGook Park, Colin Boyd and Sang-Jae Moon. "Forward Secrecy and Its Application to Future Mobile Communications Security" PKC2000, LNCS1751. Spring-Verlag , 2000. 433-445.

[14] Certicom Research, Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, September 20, 2000.

[15] Lin Zhuxing, & Li Zhenglong, "Elliptic-Curve Undeniable Signature Schemes," The 11th information security conference, 331~338, 2001.

[16] A. Jurisic, & A.J. Menezes. ECC Whitepapers: Elliptic Curves and Cryptography. Certicom corp. Available: http://www.certicom.com/research/weccrypt. html.

[17] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, 1987. 48(17) : 203~209.

[18] Niels Fegruson, Bruce Schneier, "Practical Cryptography," John Wiley & Sons, 2003.

[19] A. Lenstra and E. Verheul, "Selecting Cryptographic Key Size," The Third International Workshop on Practice and Theory in Public Key Cryptography, LNCS 1751, 2000. 446~465.