Design an efficient three-party authenticated key exchange protocol in the cloud environment

Chung-Yi Lin [a,]*, Yuh-Min Chen [a], Shu-Yi Liaw[b], Chen-Hua Fu[c]

[a] Institute of Manufacturing Information and Systems, National Cheng Kung University, Tainan, ROC
[b] Department of Business Administration, National Pingtung University Science and Technology, Pingtung, ROC
[c] Department of Information Management, National Defense University, Taipei, ROC
* Corresponding author E-mail address: p98991109@gmail.com

ABSTRACT

With the emergence of cloud computing, conventional three-party authenticated key exchange (3PAKE) protocols face two common problems; one is that the server and users are not in the same domain, and therefore, the shared authenticated keys may be unknowingly compromised. The other problem is these protocols require higher on-line computation cost and on-line communication cost during session key agreement, which can create excessive overhead for cloud clients using devices with low computational capacity. This paper proposes a novel protocol based on the self-verified token pair scheme employed by Chen et al. The proposed protocol does not require that the authenticated keys be stored in clouds to achieve user authentication. Furthermore, it overcomes the security threats existing in the protocol developed by Chen et al. and improves the performance in session key agreement to make it suitable for 3PAKE in cloud environments.

Keywords: 3PAKE, Authenticated key exchange, Cloud environment

I. INTRODUCTION

Since the first key exchange protocol was proposed by Diffie and Hellman [1], a number of researchers have proposed schemes with similar authentication functions such as [2][3][4][5]. Steiner et al. proposed that users share their authenticated keys with a trusted server for mutual authentications in subsequent session key agreements [6]. This became known as the three-party authenticated key exchange (3PAKE) protocol. However, the protocol developed by Steiner et al. is unable to detect on-line guessing or off-line password-guessing attacks. To eliminate the threat of such attacks, some schemes using server public keys such as [7][8]. Nevertheless, these schemes rely on public-key infrastructure create excessive computational overhead, making them unsuitable for environments with limited computational resources.

Although recent 3PAKE protocols such as [9][10][11][12] do not use server public keys, the emergence of cloud computing presents new challenges. First of all, the server and the users are not necessarily in the same trusted domain. Even if a cloud could replace the server and assist two communication parties in mutual authentication, storing the shared authenticated keys in the cloud would pose the risk of keys being compromised. The equipment used by cloud clients, such cell phones and e-readers, are extremely limited with regard to computational capabilities, memory, communication bandwidth, and battery power [13][14]. The means of adapting conventional 3PAKE protocols to cloud environments is thus an issue worthy of further investigation. For cloud clients using 3PAKE with such devices, we concur that some of the systems and applications may be required to expend on-line resources for computation and communication simultaneously, which could lead to excessive overhead.

Chen et al. proposed a protocol with a self-verified token pair scheme that does not require the additional storage of sensitive tables in the server to achieve user authentication [15]. In this scheme, the server hides the secret key derived in the self-verified token pair given to a specific user. Subsequently, the server requires only its private key and the self-verified token pair sent by the user to obtain the shared authenticated key. This provides an ideal method to secure authenticated keys stored in clouds; however, Yang and Chang pointed out that this protocol is unable to withstand stolen-verifier attacks [16]. Nose further noted that once an adversary obtains an authenticated key, a key-compromise impersonation attack can be initiated [17]. Yang and Chang also pointed out that the protocol proposed by Chen et al. requires higher on-line computational costs and communication loads in reaching session key agreement, and therefore, the protocol is inappropriate for the limited resources of mobile devices.

To develop a 3PAKE protocol suitable for cloud environments, this paper proposes a novel protocol based on the scheme established by Chen et al. with the objective of obtaining better security and performance in session key agreement.

II. PRELIMINARIES

A. *System model*

Figure.1 illustrates the proposed system model. The participating parties include Initiator A, Responder B, and the cloud. To achieve mutual authentication, A and B must register with the cloud to obtain the respective authenticated keys that the cloud shares with them. Beforehand, A and B generate the random exponents and components required to produce the session key as well as the verification values of their components. During session key agreement, A and B exchange their components, and through the cloud, they exchange the verification values of their components. When A and B receive the verification value, they use it to verify the integrity of the component sent to them earlier. Subsequently, when A and B wish to establish secure communication at a specific time, they can use their own random exponent and the component from the other party to generate a common session key.

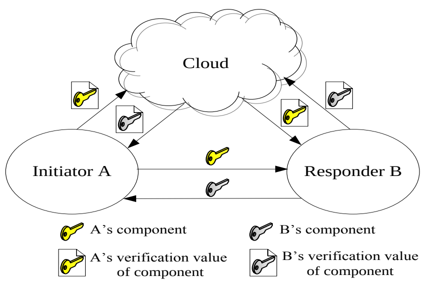

Cloud
Initiator A
Responder B
A's component
B's component
A's verification value of component
B's verification value of component

Figure 1. Conceptual model

B. *Security requirements*

We assume that two types of adversaries exist in the environment: illegal (unregistered) malicious users, referred to as external adversaries, and legal (registered) malicious users, referred to as internal adversaries. Both types of adversary may attempt to steal the shared authenticated keys and private random exponents through stolen-verifier attacks to enable impersonation attacks and crack session keys. Internal adversaries may also try to impersonate Initiator A or Responder B through cloud authentication during session key agreement, thereby tricking the other party into opening a session with them.

In view of the above, we conclude that the security requirements include (1) ensuring the confidentiality of the shared authenticated keys and the private random exponents to oppose stolen-verifier attacks and (2) ensuring that internal adversaries are unable to perform impersonation-of-initiator attacks or impersonation-of-responder attacks.

III. PROPOSED PROTOCOLS

The notation used in the proposed protocol is presented in Table Ⅰ. Sections A and B describe the phase, procedures and computation steps.

A. *Initialization phase*

In this phase, A and B perform the same procedures. Take A for instance.

1) *Registration procedure:*

Prior to this procedure, A must first select psw_A and generate a random number rn_A $(1 < rn_A < q)$ before computing $c\_rn_A = rn_A \oplus h(psw_A)$. Then, A applies for registration in the cloud:

Step 1. The cloud generates a random number σ_A $(1 < \sigma_A < q)$ and computes $sk_A = H(ID_A, \sigma_A)$.

Step 2. Cloud computes $\mu_A = g^{\sigma A} \bmod p$ and generates self-verified token pair $(tok1_A, tok2_A)$ through computing $tok1_A = h(\mu_A, ID_A)$ and $tok2_A = (\sigma_A - x \cdot tok1_A) \bmod q$. Then, cloud delivers $\{sk_A, (tok1_A, tok2_A)\}$ to A.

Step 3. A retrieves rn_A by computing $rn_A = c\_rn_A \oplus h(psw_A)$.

Step 4. A computes $k1_A = h(psw_A, rn_A)$ and $c\_sk_A = sk_A \oplus k1_A$.

Step 5. A saves $\{c\_sk_A, (tok1_A, tok2_A)\}$.

B undergoes the same procedure and saves $\{c\_sk_B, (tok1_B, tok2_B)\}$.

TABLE I. NOTATIONS

Notations	Definition
$h(\ )$	a collision-free one way hash function
$\oplus$	the exclusive operator
$\|\|$	the concatenation operator
p, q	two large primes, whereby $q \mid p-1$
g	a generator, $g^q = 1(mod\ p)$ in Z_p^*
x	the private key of the cloud
sk_A, sk_B	the authenticated keys generated by the cloud shared with A,B
$(tok1_A, tok2_A)$ $(tok1_B, tok2_B)$	the self-verified token pairs generated by the cloud for A,B
ra_A, ra_B	the random exponents generated by A,B.
R_A, R_B	the components of session key generated by A,B
$k1_A, k1_B$	the encryption keys used by A,B to encrypt their authenticated key
$k2_A, k2_B$	the encryption keys used by A ,B to encrypt their random component
v_A, v_B	identity verification values generated by A and B
vc_A, vc_B	identity verification values generated by the cloud to give to A and B
$v\_RA_A,\ v\_RA_B$	the verification values generated by A and B for their component

2) *Pre-process procedure:*

Step 1. A generates one time random number $ra_A \in Z_p^*$ and computes $RA_A = g^{raA} \bmod p$.

Step 2. A retrieves rn_A by computing $rn_A = c\_rn_A \oplus h(psw_A)$.

Step 3. A computes $k2_A = h(rn_A, psw_A)$.

Step 4. A divides ra_A into n blocks with the same bit-lengths as $k2_A$, such as $(r1_A, r2_A, \cdots, rn_A)$, and computes $(c1_A, \cdots, cn_A) = (r1_A \oplus k2_A, \cdots rn_A \oplus k2_A)$.

Step 5. A saves $\{RA_{A,} (c1_A, \cdots, cn_A)\}$.

B undergoes the same procedure and saves $(RA_{B,} c\_ra_B)$.

B. *AKE phase:*

1) *Round 1:*

Step1. A retrieves $k1_A$ by computing $rn_A = c\_rn_A \oplus h(psw_A)$ and $k1_A = h(psw_A, rn_A)$.

Step2. A retrieves sk_A by computing $sk_A = c\_sk_A \oplus k1_A$.

Step3. A generates v_RA_A by computing $v\_RA_A = h(ID_A, RA_A)$.

Step4. A generates T_A and identity verification value $v_A = h(ID_B, T_A, v\_RA_A, sk_A)$.

Step5. A sends (ID_A, RA_A) and $\{ID_A, ID_B, T_A, v\_RA_A, v_A, (tok1_A, tok2_A)\}$ to B and the cloud, respectively.

2) *Round 2:*

Upon receiving the message delivered by A, B does the following:

Step 1. B retrieves $k1_B$ by computing $rn_B = c\_rn_B \oplus h(psw_B)$ and $k1_B = h(psw_B, rn_B)$.

Step 2. B retrieves sk_B by computing $sk_B = c\_sk_B \oplus k1_B$.

Step 3. B generates v_RA_B by computing $v\_RA_B = h(ID_B, RA_B)$.

Step 4. B generates T_B and identity verification value $v_B = h(ID_A, T_B, v\_RA_B, sk_B)$.

Step 5. B sends (ID_B, RA_B) and $\{ID_B, ID_A, T_B, v\_RA_B, v_B, (tok1_B, tok2_B)\}$ to A and the cloud, respectively, and saves RA_A temporarily. Similarly, upon receiving the message from B, A saves RA_B temporarily.

3) *Round 3:*

After receiving the messages from A and B, the cloud follows the steps below:

Step1. The cloud authenticates A and B.

Step1.1. The cloud verifies whether the timestamps respectively created by A and B are fresher than those of the last request.

Step1.2. The cloud retrieves sk'_A by computing $\sigma'_A = tok2_A + x \cdot tok1_A \bmod q$ and $sk'_A = H(ID_A, \alpha'_A)$. Similarly, the cloud also retrieves sk'_B by computing $\sigma'_B = tok2_B + x \cdot tok1_B \bmod q$ and $sk'_B = H(ID_B, \sigma'_B)$.

Step1.3. The cloud verifies v_A and v_B, such as $v_A = ?H(ID_B, T_A, v\_RA_A, sk'_A)$ and $v_B = ?H(ID_A, T_B, v\_RA_B, sk'_B)$.

Step1.4. The cloud generates vc_A and vc_B, such as $vc_A = H(ID_B, v\_RA_B, sk'_A)$ and $vc_B = H(ID_A, v\_RA_A, sk'_B)$ for A and B, respectively.

Step1.5. The cloud sends (ID_A, ID_B, vc_A) and (ID_B, ID_A, vc_B) to A and B, respectively.

Upon receiving the message from the cloud, A and B respectively use RA_B and RA_A to verify vc_A and vc_B, such as $vc_A = ?h(ID_B, h(ID_B, RA_B), sk_A)$ and $vc_B = ?h(ID_A, h(ID_A, RA_A), sk_B)$. If they pass verification, then A and B confirm the integrity of RA_B and RA_A.

Thereafter, when A and B wish the establish secure communication at a given time, A computes $rn_A = c\_rn_A \oplus h(psw_A)$ and $k2_A = h(rn_A, psw_A)$, retrieves ra_A by computing $ra_A = (c1_A \oplus k2_A \| \cdots \| cn_A \oplus k2_A)$, and obtains the common session key by computing $session\ key = (RA_B)^{raA} \bmod p$. In the same manner, B retrieves ra_B and obtains the common session key by computing $session\ key = (RA_A)^{raB} \bmod p$.

IV. Security analysis

In this section, we assess whether the proposed protocol meets the security requirements. Before the security demonstration, we emphasize the following facts to assist in security analysis: (1) discrete logarithm problems (DLP) and D-H problems (DHP) are difficult to solve, (2) one way hash functions are collision-free and irreversible, and (3) clouds can protect their private keys from being compromised. These are widely accepted baselines in security. Furthermore, we assume that one way hash functions and random number algorithms are sufficiently secure and have the same bit-lengths. In addition to the original parties A, B, and the cloud, the parties of the security demonstration also include external adversary E and internal adversary I.

A. *Proposition1: The proposed protocol can withstand stolen-verifier attacks*

We use party A is attacked as an example. The proof of our proposition is as follows. To ensure the confidentiality of authenticated key sk_A and random exponent ra_A, A computes $k1_A = h(psw_A, rn_A)$ and $k2_A = h(rn_A, psw_A)$ to generate encryption keys $k1_A$ and $k2_A$. Then, A uses $k1_A$ to encrypt sk_A as $c\_sk_A = sk_A \oplus k1_A$ and A uses $k2_A$ to encrypt the blocks divided from ra_A, thereby deriving $(r1_A \oplus k1_A, \cdots, rn_A \oplus k1_A)$. As E or I use the same attack approaches, we will explain the attack using an attack by E as an example. Suppose E attacks A and steals c_sk_A, c_ra_A, and c_rn_A. To obtain sk_A and ra_A, E must first gain $k1_A$ and $k2_A$. Therefore, E attempts to gain psw_A or rn_A from c_rn_A. Because $c\_rn_A = rn_A \oplus h(psw_A)$, E cannot know either or. Even if E performs a brute force attack, they cannot differentiate the real psw_A or rn_A from c_rn_A. Consequently, E can only guess. Say psw_A and rn_A are both 128 bits. The chances of correctly guessing psw_A or rn_A is $1/2^{128}$, and thus, it is unlikely that E will be able to derive $k1_A$ or $k2_A$. This demonstrates that the proposed protocol can achieve our proposition.

B. The protocol can withstand impersonation attacks

The proof of Proposition 1 shows that neither E nor I can obtain the authenticated key of A. As a result, they cannot pass the authentication of the cloud with messages forged as though they were from A. However, I possesses a shared authenticated key with which both impersonation-of-initiator and impersonation-of-responder attacks could be attempted. Below is our proof with regard to Proposition 2.

CASE 1 Impersonation-of-initiator attack:

Suppose I impersonates A and attempts to establish a session with B. I sends (ID_A, v_RA_I) and $\{ID_I, ID_B, T_I, c\_RA_I, v_I, (tok1_I, tok2_I)\}$ to B and the cloud, respectively. Upon receiving the message, B sends (ID_B, v_RA_B) and $\{ID_B, ID_A, T_B, c\_RA_B, v_B, (tok1_B, tok2_B)\}$ to A and the cloud, respectively. Subsequently, I must intercept messages from the channel between B and A and tamper with the message from the channel between B and the cloud by changing it to $\{ID_B, ID_I, T_B, c\_RA_B, v_B, (tok1_B, tok2_B)\}$ in an attempt to pass the authentication of the cloud. B will think that they are establishing a session with A when in fact they are doing so with I. However, I cannot impersonate A because the verification value generated by B is $v_B = h(ID_A, T_B, c\_RA_B, sk_B)$; if I changed ID_A to ID_I, the cloud would discover that $v_B \neq h(ID_I, T_B, c\_RA_B, sk_B)$. As a result, I would be unable to pass the authentication of the cloud.

CASE 2 Impersonation-of-responder attack:

Supposing A wishes to establish a session with B, A sends (ID_A, v_RA_B) and $\{ID_A, ID_B, T_A, c\_RA_A, v_A, (tok1_A, tok2_A)\}$ to B and the cloud, respectively. I tampers with the message from the communication channel between A and the cloud by changing it to $\{ID_A, ID_I, T_A, c\_RA_A, v_A, (tok1_A, tok2_A)\}$. I intercepts the messages in the communication channel between A and B. Moreover, I sends (ID_B, v_RA_I) and $\{ID_I, ID_A, T_I, c\_RA_I, v_I, (tok1_I, tok2_I)\}$ to A and the cloud, respectively, in an attempt to pass the authentication of the cloud and make A think a session is being established with B, when in fact the session is with I. However, I is unable to impersonate B because the verification value generated by A is $v_A = h(ID_B, T_A, c\_RA_A, sk_A)$; if I changed ID_B to ID_I, the cloud would discover that $v_A \neq h(ID_I, T_A, c\_RA_A, sk_A)$. As a result, I would be unable to pass the authentication of the cloud.

V. Efficiency Analysis

We compared the proposed protocol with that proposed by Chen et al. with regard to performance in communication costs and on-line computation costs.

A. Communication cost

We compared the communication costs of the two protocols with regard to communication efficiency and the size of messages being transmitted.

1) Communication efficiency:

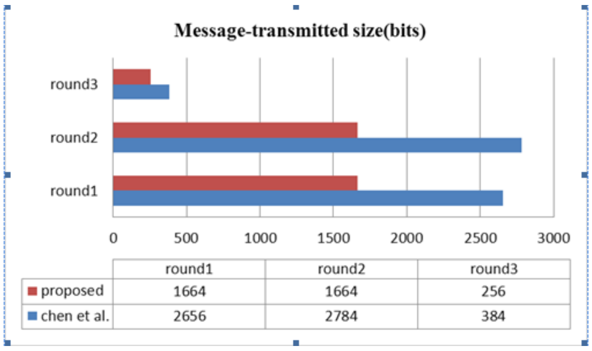
We used the number of rounds and times of communication to examine the communication efficiency of the protocol. The proposed protocol requires 3 rounds and 6 communications. In contrast, the protocol proposed by Chen et al. requires 3 rounds and 7 communications. The protocol proposed in this paper demonstrates superior communication efficiency.

2) Message transmission size:

Chen et al. assumed that the sizes of p and q are 1,024 bits and 160 bits, the output size of one way hash function is 128 bits, the size of the timestamp is 96 bits, and the size of ID is negligible. We thus compared the sizes of messages transmitted from 3 rounds using the proposed protocol and that proposed by Chen et al. based on the aforementioned assumptions. Figure 2 presents the results. In both the first and second rounds, the costs of the proposed protocol were 96+128*3+160+1024 bits, whereas in the third round, the cost was 2*128 bits. Thus, the total size of messages transmitted was 3584 bits. This result is approximately 2240 bits smaller than that using the protocol proposed by Chen et al.

B. On-line computation cost:

The session key agreement requires on-line computation; therefore, we compared the proposed protocol and that proposed by Chen et al. with regard to the computation costs required in the AKE phase. The symbols used to analyze the computation cost are presented in Table Ⅱ. As introduced in previous studies as [18][19][20], the relationships of the various computation costs can be determined as shown in Table Ⅲ. Table Ⅳ compares the total computation costs required by A, B, and the server (cloud) in the proposed protocol and that proposed by Chen et al. Figure 3 displays the computation costs required in each round. To facilitate the comparisons in Figure 3, we converted the costs of T_{EXP} and T_{MUL} into cost of T_H. The results of the comparisons indicate that the proposed protocol imposes significantly lower computation costs than that proposed by Chen et al. with the exception of the cloud, which induces computation costs higher than those of the server in the protocol developed by Chen et al. Regardless, clouds are likely to have greater computation capabilities, and therefore, such costs will not cause excessive computational overhead.



	round1	round2	round3
🟥 proposed	1664	1664	256
🟦 chen et al.	2656	2784	384

Figure 2. Comparison of size of messages transmitted.

TABLE II. SYMBOLS USED TO ANALYZE COMPUTATION COSTS

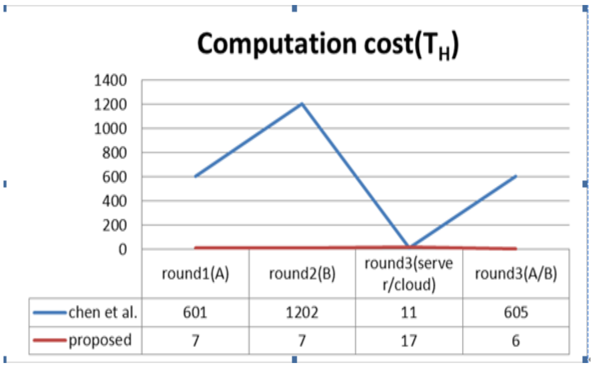
Symbol	Definition
T_{EXP}	Cost of one modular exponential operation
T_{MUL}	Cost of one modular multiplication operation
T_{ADD}	Cost for one modular addition operation
T_H	Cost of one hash function operation
$T_\oplus$	Cost of one exclusive OR operation

TABLE III. RELATIONSHIPS AMONG VARIOUS OPERATIONS

$1\ T_{EXP} \approx 240\ T_{MUL}$
$1\ T_{EXP} \approx 600\ T_H$
$1\ T_H \approx 0.4\ T_{MUL}$
T_{ADD} is negligible
$T_\oplus$ is negligible

TABLE IV. COMPARISON OF COMPUTATIONAL COSTS

3PAKE	Computation cost			
	A	B	Server (Cloud)	Total
Chen et al.	$2T_{EXP}+4T_H$	$2T_{EXP}+4T_H$	$2T_{MUL}+6T_H$	$4T_{EXP}+2T_{MUL}+14T_H$
proposed	$7T_H$	$7T_H$	$2T_{MUL}+12T_H$	$2T_{MUL}+26T_H$



	round1(A)	round2(B)	round3(server/cloud)	round3(A/B)
chen et al.	601	1202	11	605
proposed	7	7	17	6

Figure 3. Comparison of computational costs required in each round.

The above analysis demonstrates that the proposed protocol has better communication efficiency and transmits smaller messages that the protocol developed by Chen et al. Furthermore, in the AKE phase, the proposed protocol significantly reduces computational costs for the two communication parties with regard to session key agreement.

VI. CONCLUSION

The objective of this paper was to improve the protocol proposed by Chen et al. in terms of security and performance during session key agreement to make it applicable to 3PAKE in cloud environments. Security analysis shows that the proposed protocol is able to withstand stolen-verifier attacks as well as impersonation attacks. Efficiency analysis indicates that the proposed protocol has higher communication efficiency and transmits smaller messages than the protocol developed by Chen et al. In addition, the proposed protocol significantly reduces the overhead of on-line computation for the two communication parties during session key agreement. We can thus conclude that the proposed protocol is suitable for 3PAKE in cloud environments. This paper assumes that the cloud is honest and follows the required security service agreement. However, malicious clouds are still possible, and we therefore plan to develop a 3PAKE protocol capable of withstanding malicious attacks even from the clouds themselves.

REFERENCES

[1] W. Diffie, M.Hellman, "New directions in cryptography, " IEEE Trans Inform Theory, vol. 22, pp.644–54, 1976.

[2] S.M. Bellovin, M. Merrit, "Encrypted key exchanged: Password-based protocols secure against dictionary attacks, " in: Proc. of IEEE Symposium on Security and Privacy, pp.72–84, 1992.

[3] V. Boyko, P. Mackenzie, S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman, " in: Proceedings of EUROCRYPT'2000, Advances in Cryptology. Springer-Verlag, pp.156–171, 2000.

[4] O. Goldreich, Y. Lindell, "Session-key generation using human passwords only, " in: Proceedings of CRYPTO'2001, Advances in Cryptology. Springer-Verlag, vol. 2139, pp.408–432, 2001.

[5] R. Gennaro, Y. Lindell, "A framework for password-based authenticated key exchange, " In: Proceedings of EUROCRYPT'2003, Advances in Cryptology. Springer-Verlag, vol. 2656, pp.524–543, 2003

[6] M. Steiner, G. Tsudik, M. Waidner, "Refinement and extension of encrypted key exchange, ", ACM SIGOPS Operating Systems Review, vol. 29, pp.22-30, 1995.

[7] C.L. Lin, H.M. Sun, T.Hwang, "Three-party encrypted key exchange: attacks and a solution, " ACM SIGOPS Operating System Review, vol. 34, pp.12-20, 2000.

[8] H.-M. Sun, B.-C. Chen, T, Hwang, "Secure key agreement protocols for three-party against guessing attacks, " The Journal of Systems and Software, vol. 75, pp.63-68, 2005.

[9] R. Lu, Z. Cao, "Simple three-party exchange protocol, " Computers & Security, vol. 26, pp.94-97, 2007.

[10] T.F. Lee, T. "Hwang, Simple password-based three-party authenticated key exchange without server public keys, " Information Sciences, vol.180, pp.1702-1714, 2010.

[11] T.Y. Chang, M.S. Hwang, W.P. Yang, "A communication-efficient three-party password authenticated key exchange protocol, " Information Sciences, vol. 181, pp.217-226, 2011.

[12] C. Lv, M. Ma, H. Li, J. Ma, Y. Zhang, "An novel three-party authenticated key exchange protocol using one-time key, " Journal of Network and Computer Application, vol. 36, pp.498-503, 2013.

[13] M. Conti, et al., “Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence, ” Journal of Pervasive and Mobile Computing, vol. 8, pp.2-21, 2012.

[14] Q. Shi, N. Zhang, D.-L. Jones, “Efficient autonomous signature exchange on ubiquitous networks, ” Journal of Network and Computer Application, vol.35, pp.1793-1806, 2012.

[15] T.Z. Chen, W.B. Lee, H.B. Chen, “A round- and computation-efficient three-party authenticated key exchange protocol, ” The Journal of Systems and Software, vol. 81, pp.1581-1590, 2008.

[16] J.H. Yang, C.C. Chang, “An efficient three-party authenticated key exchange protocol using elliptic curve cryptography for mobile-commerce environments, ” The Journal of Systems and Software, vol. 82, pp.1497-1502, 2009.

[17] P. Nose, “Security weaknesses of authenticated key agreement protocols, ” Information Processing Letters, vol. 111, pp.687-696, 2011.

[18] Y.P. Liao, S.S. Wang, “A secure dynamic ID based remote user authentication scheme for multi-server environment, ” Computer Standards & Interfaces, vol.31, pp.24-29, 2009.

[19] Z. Li, J. Higgins, M. Clement, “Performance of finite field arithmetic in an elliptic curve cryptosystem, ” IEEE Computer Society, pp.249 – 258, 2001.

[20] R.C. Wang, W.S. Juang, C.L. Lei, “A Web Metering Scheme for Fair Advertisement Transactions, ” International Journal of Security and Its Applications, vol. 2, pp.49-56, 2008.