International Journal of Innovative Computing, Information and Control
Volume **9**, Number **11**, November **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 4247–4261

ROBUST AND PRIVACY PROTECTION AUTHENTICATION IN CLOUD COMPUTING

Jheng-Jia Huang[1], Wen-Shenq Juang[2,*], Chun-I Fan[1] and Horng-Twu Liaw[3]

[1]Department of Computer Science and Engineering
National Sun Yat-sen University
No. 70, Lienhai Rd., Kaohsiung 80424, Taiwan
jhengjia.huang@gmail.com; cifan@faculty.nsysu.edu.tw

[2]Department of Information Management
National Kaohsiung First University of Science and Technology
No. 2, Jhuoyue Rd., Nanzih, Kaohsiung 811, Taiwan
*Corresponding author: wsjuang@nkfust.edu.tw

[3]Department of Information Management
Shih Hsin University
No. 1, Lane 17, Sec. 1, Mu-Cha Rd., Taipei 116, Taiwan

Received September 2012; revised March 2013

Abstract. *In the cloud environments, the cloud user can use low computing devices to connect to the clouds. For checking the identities of the user and the cloud server, they must do mutual user authentication before using the cloud services. Also, the transmitted messages between them must be protected. These environments are different with traditional client-server environments since a cloud user may use many various devices to connect to the cloud server to do authentication. It will face the following problems. The first problem is that some devices only have low computing and memory ability. The second problem is that the cloud user may have many various devices for storing secret sensitive information. If some devices are lost, it is hard to recover the sensitive information stored in these devices. In order to solve all the above problems, we propose a lightweight robust and privacy protection authentication scheme for cloud computing. Our proposed authentication scheme can prevent the insider attack even if the secret information stored in a cloud database is compromised and the offline dictionary attack even if the secret information stored in a user's device is compromised. Also, the cloud user can freely choose his password to use our proposed authentication scheme.*
Keywords: Cloud computing, Information security, Privacy protection, Identity authentication, Low-resource device

1. Introduction.

The cloud computing [1,2,8,10,15,22] is growing rapidly in recent years. In a cloud computing environment, the user can transfer some computing loading to the cloud server since the server has high computing ability. Therefore, the user does not need the device that has strong computing ability to connect to the cloud server. In this environment, the user can use many various devices to login to various clouds.

In this environment, the user authentication between the user and the cloud may face the following four problems. The first problem is that the attacker may try to forge the cloud server to get the cloud user's data and information. The second problem is that the attacker may try to get the cloud user's identity information from the communication messages between the cloud server and the cloud user. The third problem is that the attacker may try to generate many fake request messages to cause the DDOS attack [11].

The fourth problem is that the secret information stored in the cloud user's device may be compromised when this device is lost [5,28].

In a cloud environment, the user can use her/his various devices to use the cloud services via authentication [4,13,24,25]. It is very easy to cause the device lost problem since the cloud user may have many portable devices. The cloud devices may store the same secret information. It is very dangerous since some cloud devices may be lost. Therefore, to provide a secure and robust authentication scheme that the user can use her/his various devices to do authentication with the cloud server is very important.

In order to solve all the above problems, we propose a lightweight robust and privacy protection authentication scheme for cloud computing. Our proposed authentication scheme can satisfy the following properties: 1. low communication and computation cost; 2. no password table; 3. free choosing and changing passwords by users; 4. no time-synchronization problem; 5. mutual authentication between the user and the cloud server; 6. revoking the lost card without changing the user's identity; 7. identity protection; 8. providing session key agreement; 9. preventing the offline dictionary attack with the smart device; 10. no quota limit of the number of a user's smart devices; 11. without the real user's identity information in the user's smart device; 12. increasing the security level of any smart device in the cloud environment.

2. **Preliminary.**

2.1. **Elliptic curve cryptosystem.**

The elliptic curve cryptosystem (ECC) was introduced by N. Koblitz [18] and V. S. Miller [21] in 1985. The level of security of ECC is based on the elliptic curve discrete logarithm problem (ECDLP). ECC is one of the asymmetric cryptosystems. One characteristic of the ECC is that when the user inputs the same data, the ECC can transform it to different ciphertexts. We introduce DLP in the following. We assume that p is a large prime number, g is p's primitive root, and g is less than p. If the attacker can know $[y, g, p]$, it is still difficult to compute x. The key owner can use the formula $y = g^x \mod p$ to compute the public key y, and send the public key to any person. The sender can use this public key to encrypt the data d. The step is in the following.

1. The sender chooses a random number r.
2. The sender computes $b = g^r \mod P$ and $c = d * y^r \mod P$ and sends b and c to the key owner.
3. After the key owner receiving b and c, the key owner computes $c * (bx)^{-1} \mod P$ to obtain the data d.

The elliptic curve cryptosystem (ECC) has two advantages.

1. Low computing cost: ECC is a very strong cryptosystem. Even though the ECC computing cost of 160 bits size is lower than RSA computing cost of 1024 bits, it also has the same security level. Therefore, the ECC can be used in the low computing resources devices like a PDA, a smart card and a cellphone.
2. Randomness: One characteristic of ECC is that when the user inputs the same data, the ECC can transform its different ciphertexts.

2.2. **Juang et al.'s scheme.**

2.2.1. *The parameter generation phase.*

1. The server chooses a large prime P and finds a generator point G.
2. The server selects a random number x and computes the public key $PS = (x \times G)$.
3. The server publishes the parameters (PS, P, EP, G, n).

2.2.2. *The registration phase.*

1. The user selects a password PW_i and a random number b, and computes $h(PW_i \| b)$. Then the user sends the parameters $ID_i, h(PW_i \| b)$ to the server.
2. After the server getting the message, if ID_i is a new identifier, the server sets the card identifier $CI_i = 1$ and stores the record ID_i, CI_i in its registration table. If ID_i is not a new identifier, the server sets the card identifier $CI_i = CI_i + 1$ and stores the record $ID_i, CI_i = CI_i + 1$ in its registration table. Therefore, the server generates $b_i = Es(h(PW_i \| b) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i \| b)))$ and $V_i = h(ID_i, s, CI_i)$. The authentication tag is $h(ID_i \| CI_i \| h(PW_i \| b))$. The server issues a smart card to the user. The contained parameters are b_i, V_i, ID_i, CI_i.
3. The user stores the parameters b_i, V_i, ID_i, CI_i, b into the smart card.

2.2.3. *The precomputation phase.*

1. The smart card generates the random number r, and computes and stores the parameters $e = (r \times G)$ and $c = (r \times Ps) = (r \times x \times G)$ as a point over EP before the start of the log-in phase.

2.2.4. *The log-in phase.*

1. The user inserts his smart card into a card reader and inputs his password PW_i. Then, the smart card sends the parameters b_i and $E_{V_i}(e)$ to the server.
2. After receiving the message, the server decrypts b_i by the secret key s. The server computes $V_i = h(ID_i, s, CI_i)$ and decrypts $E_{V_i}(e)$ to obtain e. The server checks if
   (a) the authentication tag is equal to $h(ID_i \| CI_i \| h(PW_i \| b))$;
   (b) ID_i is registered;
   (c) CI_i is stored in the table.

   If any of the verifications is false, the server will stop the log-in request. If all of the verifications are true, the server selects a random number u and computes $c = (e \times x) = (r \times x \times G)$ and $Ms = h(c \| u \| V_i)$. Therefore, the server sends the parameters u, Ms to the smart card.
3. After the smart card getting the message, the smart card computes and checks if Ms is equal to $h(c \| u \| V_i)$. If it is not, the smart card will stop the log-in phase. Otherwise, the smart card computes $MU = h(h(PW_i \| b) \| V_i \| c \| u)$ and $Sk = h(V_i, c, u)$. Then, the smart card sends the parameter MU to the server.
4. After the server getting the message, the server computes and checks that the MU is equal to $h(h(PW_i \| b) \| V_i \| c \| u)$. If it is not, the server sends a wrong password message to the user. The user can input the password PW_i. Then, the smart card can compute MU. Then, the smart card sends the parameter MU to the server again. If the number of the password verifications exceeds the allowed time, the server will stop the log-in request. Otherwise, the server accepts the log-in request and computes $Sk = h(Vi, c, u)$.

2.2.5. *The password-changing phase.*

1. The user inputs a new password PW_i^* and a new random number b^*. Then, the smart card computes $E_{Sk}(ID_i, h(PW_i^* \| b^*))$. The smart card sends the parameter E_{Sk} to the server.
2. After the server getting the message, it decrypts E_{Sk} to obtain ID_i and $h(PW_i^* \| b^*)$ by Sk. The server computes a new $b_i^* = E_s(h(PW_i^* \| b^*) \| ID_i \| CI_i \| h(ID_i \| CI_i \| h(PW_i^* \| b^*)))$ and sends the parameter $E_{Sk}(b_i^*)$ to the smart card.
3. After the smart card getting the message, the smart card decrypts it by Sk. Then, the smart card stores the new parameters b_i^* and b^* in its memory.

2.3. **Sun et al.'s scheme.**

2.3.1. *The parameter generation phase.*

1. The server chooses an elliptic curve E over a finite field F_p. The set of all the points on E is denoted by $E(F_p)$.
2. The server chooses a point $G \in E(F_p)$ that the subgroup generated by G has a large order n.
3. The server publishes the parameters p, E, G, n.

2.3.2. *The registration phase.*

1. The user selects the sub-identifier ID_U. Then, the user sends the sub-identifier ID_U to the server.
2. After the server getting the message, the server checks if this ID_U is valid. If it is true, the server chooses the sub-identifier ID_S and creates the identifier $ID = ID_U \| ID_S$. Then, the server sends an identifier ID to the user. The server generates $V = h(ID\| K_S) \oplus h(PW)$ and $IM = EK_S(ID\|r)$, where the password PW and the identity protection random number r are selected by the server.
3. The server sends the password PW and the smart card parameters to the user, where the smart card parameters are the public parameter IM and the private parameter V.

2.3.3. *The authentication phase.*

1. The user inserts his smart card into a card reader and inputs his password PW.
2. The smart card chooses a random integer rc from $[1, n-1]$ and computes $Gc = rc \times G$. Then, the smart card sends the parameter IM, Gc to the server.
3. After the server receiving the message, the server decrypts the parameter IM by the master secret key K_S to obtain $ID\|r$. The server checks if this ID is valid. If this verification is false, the server stops this phase. If this verification is true, the server creates $Gs = rs \times G$, where the random number rs is selected from the interval $[1, n-1]$ by the server. The server computes $K_{SU} = h_1(h(ID\|K_S)\|(rs \times G_c))$ and $M_S = h_2(K_{SU}\|G_c\|G_s)$ and sends the parameters M_S, Gs to the smart card.
4. After the smart card receiving the message, the smart card computes $V' = V \oplus h(PW)$ and $K_{SU} = h_1(V'\|(rc * G_s))$. Then, the smart card checks if the value M_S is equal to $h_2(K_{SU}\|G_c\|G_s)$. If it is not, the smart card stops this phase. If it is true, the smart card computes $M_U = h_2(K_{SU}\|G_s)$ and sends the parameter M_U to the server.
5. After the server receiving the message, the server checks if the parameter M_U is equal to $h_2(K_{SU}\|G_s)$. If it is not, the server stops this phase. Otherwise, it is successfully authenticated between the server and the user. Then, the server establishes the session key K_{SU}.

2.3.4. *The password-changing phase.*

1. The user inserts his smart card into a card reader and inputs his old password PW. The user demands to change the password. The user inputs his new password PW^*.
2. The user's smart card computes $V^* = V \oplus h(PW) \oplus h(PW^*)$ and $h(ID\|K_S) \oplus h(P W^*)$. Then, the smart card updates the parameter V with the parameter V^*.

2.4. **Singh et al.'s scheme.**

2.4.1. *The parameter generation phase.*

1. The server selects an elliptic curve E over a finite F_P. The discrete logarithm problem is hard in $E(F_P)$ and a point $G \in E(F_P)$ that the subgroup is generated by a large order n.
2. The server publishes the parameters p, E, G, n.

2.4.2. *The registration phase.*

1. The server creates $V = h(ID\|K_S) \oplus h(PW)$ and $IM = EK_S(ID\|r)$, where the password PW and the identity protection random number r are selected by the server and the parameter K_S is the private key of the server.

2.4.3. *The authentication phase.*

1. The user inputs his password PW and the smart card chooses the random integer rc from $[1, n-1]$. Then, the smart card computes $Gc = rc \times G$ and sends the parameters IM, Gc to the server.
2. After the server receiving the message, the server decrypts the parameter Gc to obtain $ID\|r$. The server verifies if this ID is valid. If this verification is false, the server stops this phase. If this verification is true, the server creates $Gs = rs \times G$, $K_{SU} = h(h(ID\|K_S)\|(rs \times G_c))$, and $M_S = h(K_{SU}\|G_c\|G_s)$. The server sends the parameters M_S, Gs to the smart card.
3. After the smart card receiving the message, the smart card computes $V' = V \oplus h(PW)$ and $K_{SU} = h(V'\|(rc \times G_s))$. Then, the smart card verifies if the value M_S is equal to $h(K_{SU}\|G_c\|G_s)$. If it is not, the smart card stops this phase. If it is true, the smart card computes $M_U = h(K_{SU}\|G_s)$ and sends the parameter M_U to the server.
4. After the server receiving the message, the server verifies if the parameter M_U is equal to $h(K_{SU}\|G_s)$. If it is not, the server stops this phase. Otherwise, it is successfully authenticated between the server and the user. Then, the server establishes the session key K_{SU} with the user.

2.4.4. *The password-changing phase.*

1. The user inputs his old password PW, and requests the smart card to change the password. Then, the user inputs his new password PW^*.
2. The user's smart card creates $V^* = V \oplus h(PW) \oplus h(PW^*)$ and $h(ID\|K_S) \oplus h(PW^*)$. Therefore, the smart card updates the parameter V with the parameter V^*.

2.5. **Li et al.'s scheme.**

2.5.1. *The parameter generation phase.*

1. The server chooses a large prime P and finds a generator point G.
2. The server selects a random number x and computes the public key $PS = (x \times G)$.
3. The server publishes the parameters (PS, P, EP, G, n).

2.5.2. *The registration phase.*

1. The user selects the password PW and the random string b. Therefore, the user sends the parameters $ID, h(pw\|b), N_0$ to the server.

2. After the server receiving the message, the server creates the card identifier CI and stores the parameters ID, CI, N_0 in the registration table. The server computes the parameter $b_{ID}^{N_0} = E_s((ID\|CI\|N_0)\|(ID\|CI\|N_0) \oplus h(pw\|b)\| \times h((ID\|CI\|N_0)\|((ID\|CI\|N_0) \oplus h(pw\|b))))$ and V_{ID}. Then, the server sends the parameters $b_{ID}^{N_0}$, V_{ID}, ID, CI to the user.
3. After the user receiving the message, the user stores the parameters $b_{ID}^{N_0}$, V_{ID}, ID, CI, b into the smart card.

2.5.3. *The precomputation phase.* The smart card generates the random number r. Then, the smart card computes and stores the parameters $e = (r \times G)$ and $c = (r \times Ps) = (r \times x \times G)$ as a point over EP before the start of the log-in phase.

2.5.4. *The log-in phase.*

1. The user inputs his password PW. The smart card sends the parameters $b_{ID}^{N_0}$, $E_{V_{ID}}(N_1\|e)$ to the server, in which $N_1 \in_R [0,1]^{64}$.
2. After the server receiving the message. The server decrypts $b_{ID}^{N_0}$ by the master key s to obtain $ID, CI, N_0, h(pw\|b)$. Then, the server checks if the parameters ID, CI, N_0 are in the registration table, and the decryption operation can get the hash tag $h((ID\|CI\|N_0)\|((ID\|CI\|N_0) \oplus h(pw\|b)))$. If this checking is false, the server stops this phase. If this checking is true, the server computes $V_{ID} = h(ID\|s\|CI)$ and decrypts $E_{V_{ID}}$ by V_{ID} to obtain N_1, e. Then, the server updates the parameters ID, CI, N_0 in the registration table by ID, CI, N_1. Then, the server computes $c = xe$ and $M_s = h(c\|u\|V_{ID})$ and sends the parameters $Nb1, u \oplus h_{64}(b_{ID}^{N_1}), M_s$ to the smart card, where $b_{ID}^{N_1} = E_s((ID\|CI\|N_1)\|(ID\|CI\|N_1) \oplus h(pw\|b)\| \times h((ID\|CI\|N_1)\|((ID\|CI\|N_1) \oplus h(pw\|b))))$ and $Nb1 = b_{ID}^{N_1} \oplus (h(N_1\|e\|1)\|h(N_1\|e\|2)\|h(N_1\|e\|3))$ and $h_{64}(b_{ID}^{N_1})$ is the first $64bits$ of $h(b_{ID}^{N_1})$.
3. After the smart card receiving the message, the smart card computes the parameter $b_{ID}^{N_1}$ from $Nb1$. Then, the smart card computes the parameter u from $u \oplus h_{64}(b_{ID}^{N_1})$. The smart card checks if M_s is equal to $h(c\|u\|V_{ID})$. If this checking is false, the smart card stops this phase. If this checking is true, the smart card believes that the server is real. Then, the smart card updates the parameter $b_{ID}^{N_0}$ to $b_{ID}^{N_1}$ and computes the parameter $M_u = h(h(pw\|b)\|V_{ID}\|c\|u)$. The smart card sends the parameter M_u to the server.
4. After the server receiving the message, the server checks if M_u is equal to $h(h(pw\|b)\| V_{ID}\|c\|u)$. If this checking is false, the server stops this phase. If this checking is true, the server believes that the smart card is real. Then, the session key is $k = h(V_{ID}\|c\|u)$.

2.5.5. *The password-changing phase.*

1. When the user wants to change his password, the user must connect to the server. The user inputs his new password pw^* and selects a new random parameter b^*. Then, the smart card computes $E_k((ID\|CI\|N^*)\|h\ (pw^*\|b^*))$ and sends E_k to the server, where $N^* \in_R [0,1]^{64}$.
2. After the server receiving the message, the server decrypts the parameter E_k to obtain $(ID\|CI\|N^*)\|h(pw^*\|b^*)$ by the session key k. Then, the server updates the parameters ID, CI, N^* in the registration table and computes $b_{ID}^{N^*}$. Then, the server encrypts the parameters $b_{ID}^{N^*}, ID, CI, N^*$ by the session key k and sends E_k to the smart card, where $b_{ID}^{N^*} = E_s((ID\|CI\|N^*)\|(ID\|CI\|N^*) \oplus h(pw^*\|b^*)\| \times h((ID\|CI\|N^*)\|((ID\|CI\|N^*) \oplus h(pw^*\|b^*))))$.

3. After the smart card receiving the message, the smart card decrypts the parameter E_k to obtain $b_{ID}^{N^*}, ID, CI, N^*$ by the session key k. Then, the smart card stores the new parameters $b_{ID}^{N^*}, V_{ID}, ID, CI, b^*$ in its memory.

3. Our Proposed Scheme.

In our proposed scheme, it has four participants and five phases. The participants are the cloud user, the cloud user's device, the cloud server, and the key delegation center (KDC) [17,20]. The phases are the parameter generation phase, the registration phase, the pre-computation phase, the log-in phase, and the password-changing phase. Our proposed authentication mechanism not only uses elliptic curve cryptosystem (ECC) [6,12,14,18,27] to identify the cloud user and the cloud server in the log-in phase, but also uses public one way hash functions [19,23,26] to authenticate the cloud user's identity information in the registration phase and the log-in phase. Our proposed authentication mechanism uses the related parameters to create the session key in the log-in phase. The cloud user can use the session key [3,7,9,16,24] to encrypt and decrypt the data of communication between the cloud user and the cloud server after the log-in phase. In our proposed authentication mechanism, the cloud user does not scare collution between the cloud server and the key delegation center (KDC) since the cloud server can get the key stored in the key delegation center (KDC); it cannot decrypt it to get the needed information. Also, the cloud user can choose his password freely.

3.1. Notations.

The notations for our proposed scheme are defined as follows:

1. $h()$: Public one-way hash function.
2. $E_{AK}()$: Secure symmetric encryption algorithm with the secret key AK.
3. $D_{AK}()$: Secure symmetric decryption algorithm with the secret key AK.
4. b: Random number chosen by the user.
5. G: Generator point of a large order.
6. bi: User's authentication tag.
7. sk: Session key agreement's key.
8. $\|$: String concatenation operator.
9. x: Server's private key based on elliptic curve cryptosystems.
10. r: User's private key based on elliptic curve cryptosystems.
11. u: Server's private key based on elliptic curve cryptosystems.
12. Ps: Server's public key based on elliptic curve cryptosystems.

3.2. The parameter generation phase.

In this phase the cloud server can generate some parameters as follows:

Step 1. The cloud server can find a generator point G of order n, in which n is a large divisor, and $n \times G = O$.

Step 2. The cloud server selects a random number x as its private key and keeps it secretly.

Step 3. The cloud server computes the public key $Ps = (x \times G)$ and publishes the parameters (Ps, G, n).

3.3. The registration phase.

Any user in this phase will do it only once. The user can use her/his smart device after this phase to verify identity. When the cloud user needs to register a new one in the cloud server, the cloud user should perform the following steps with the cloud server and the KDC (key delegation center).

1. The cloud user i can use any smart device to the cloud server and input its identity information. Then, the smart device creates the AES key AK and encrypts b by AK. Then, the smart device computes $h(pw\|b)$ and gives $ID_i, h(pw\|b), E_{AK}(b)$ to the cloud registration server, where b is a random number chosen by the cloud user

i. This procedure can be done by the manager of the device in the cloud user i's face.

2. After the cloud server receiving the message, the cloud server creates the changing password identifier CI_i, where CI_i is the changing password's counter for the cloud user. If ID_i is not a new user, the cloud server revokes the registration request. If ID_i is a new user, the cloud server will set $CI_i = 1$. If the user will change the password in the future, the cloud server computes $CI_i = CI_i + 1$. The cloud server selects a random number s and generates the parameters $bi = h(ID_i \| CI_i \| h(pw \| b))$ and $Vi = h(ID_i \| s \| CI_i)$. Then, the cloud server stores the parameters $ID_i, CI_i, bi, Vi, E_{AK}(b)$ in the registration table and gives $Message, Vi$ to the smart device, where $Message$ is the successful message by the cloud server. This procedure can be done by the server in the cloud registration's face.
3. After the smart device receiving the message, the smart device encrypts AK, Vi by pw and gives $ID_i, E_{pw}(AK, Vi)$ to the KDC.
4. After the KDC receiving the message, the KDC stores the parameters ID_i, E_{pw} (AK, Vi) in the KDC database and gives $Message$ to the smart device, where $Message$ is a successful notification message.
5. After the smart device receiving the message, the smart device stores the parameters AK, Vi. The memory of the smart device contains AK, Vi.

3.4. **The pre-computation phase.** The smart device in this phase can request the cloud server to get the parameters. Then, the smart device has to check if the secret key is in this smart device. The smart device selects a random number and stores some parameters for use in the log-in phase. The smart device should perform the following steps with the cloud server and the KDC (key delegation center).

1. The smart device sends the parameter ID_i to the cloud server.
2. After the cloud server receiving the message, the cloud server can feedback ID_i's $E_{AK}(b)$ and $Message$, where $Message$ is the password changing message by the cloud password changing phase.
3. After the smart device receiving the message, the smart device checks if the smart device has the parameters AK, Vi and $Message$. If it is true, the smart device decrypts $E_{AK}(b)$ by AK to obtain b. If the smart device does not have the parameters AK, Vi or $Message$, the smart device has to request the KDC to get the parameter $E_{pw}(AK, Vi)$. Then, the smart device selects a random number r and computes $e = r \times G$ and $c = r \times P_s = r \times x \times G$. Then, the smart device stores the parameters (c, e, b) for use in the log-in phase.

3.5. **The log-in phase.** When the cloud user wants to login the cloud server, the user must input his password pw and decrypt $E_{AK}(b)$ by AK to obtain b. Then, the smart device computes the parameter $h(pw \| b)$. In our proposed scheme, the smart device will complete the pre-computation phase before the log-in phase. The cloud user should perform the following steps with the cloud server and the smart device.

1. After the cloud user inputs the password and the smart device has finished the pre-computation phase, the smart device encrypts the parameters $h(pw \| b)$ and e by Vi and sends $E_{Vi}(h(pw \| b), e)$ to the cloud server, where $Vi = h(ID_i \| s \| CI_i)$.
2. After the cloud server receiving the message, the cloud server decrypts the parameter E_{Vi} by Vi to obtain $h(pw \| b)$ and e. The cloud server computes the parameter bi^*, where $bi^* = h(ID_i \| CI_i \| h(pw \| b))$. Then, the cloud server checks if bi^* is equal to bi. If it is false, the cloud server sends a wrong password message to the user. The user inputs the password pw and b to compute $h(pw \| b)$. Then, the smart device

sends the parameter $h(pw\|b)$ to the cloud server again. If bi^* is equal to bi, the cloud server selects a random number u and computes $c = e \times x = r \times x \times G$ and $Ms = h(Vi\|c\|u)$. Then, the cloud server sends the parameters u and Ms to the smart device.

3. After the smart device receiving the message, the smart device computes Ms and checks if the parameter Ms is equal to $h(Vi\|c\|u)$. If it is not, the smart device revokes the log-in phase. Otherwise, the smart device computes Mu and sk, where $Mu = h(h(pw\|b)\|Vi\|c\|u)$ and $sk = h(Vi\|c\|u)$. Then, the smart device sends the parameter Mu to the cloud server. At this time, the cloud server is authenticated by the smart device.
4. After the cloud server receiving the message, the cloud server computes Mu and checks if the parameter Mu is equal to $h(h(pw\|b)\| \ Vi\|c\|u)$. If it is not, the cloud server revokes the log-in phase. Otherwise, the cloud server accepts the log-in request and computes a session key sk, where $sk = h(Vi\|c\|u)$. At this time, the smart device is authenticated by the cloud server. Both sides can use the session key sk to decrypts and encrypts the messages of communication.

3.6. **The password-changing phase.** The cloud user can change his password in this phase when the user thinks that she/he should do it or the user lost his smart device any time. If the cloud user's password is changed successful, the smart device can update the new parameter Vi^* in the pre-computation phase. When the cloud user needs to change his password, the cloud user needs to agree on a session key with the server through the log-in phase in advance.

1. After the user inputs his new parameters pw^* or b^* and the smart device has finished the log-in phase. The smart device checks if b has changed. If it is true, $E_{AK}(b) = E_{AK}(b^*)$. If b has not been changed, $E_{AK}(b) = E_{AK}(b)$. Then, the smart device encrypts the parameters $h(pw^*\|b^*)$, $E_{AK}(b)$ by sk and sends $E_{sk}(h(pw^*\|b^*), E_{AK}(b))$ to the cloud server, where $sk = h(Vi\|c\|u)$.
2. After the cloud server receiving the message, the cloud server decrypts the parameter E_{sk} by sk to obtain $h(pw^*\|b^*)$ and $E_{AK}(b)$. Then, the cloud server selects a random number s and computes CI_i, bi^*, Vi^*, where $CI_i = CI_i + 1$, $bi^* = h(ID_i\|CI_i\| h(pw^*\|b^*))$ and $Vi^* = h(ID_i\|s\|CI_i)$. Then, the cloud server encrypts the parameter Vi^* by sk and stores the parameters CI_i, bi^*, Vi^*, $E_{AK}(b)$. Then, the cloud server sends the parameters $Message$ and $E_{sk}(Vi^*)$ to the smart device, where $Message$ is the password changing successful message by the cloud server.
3. After the smart device receiving the message, the smart device decrypts E_{sk} by sk to obtain Vi^* to store the parameter Vi^* in the smart device. Then, the smart device encrypts the parameters AK, Vi^* by pw and sends ID_i and $E_{pw}(AK, Vi)$ to the KDC.
4. After the KDC receiving the message, the KDC stores the parameters ID_i, E_{pw} (AK, Vi) in the KDC database and gives $Message$ to the smart device, where $Message$ is successful message by the KDC.

4. Discussion.

4.1. Security analysis.

4.1.1. *Mutual authentication.* In our proposed scheme, the goal of mutual authentication is to create a robust and powerful session key sk between the cloud user and the cloud server. In Step 2 of the log-in phase of our proposed scheme, after the smart device receiving u and Ms from the cloud server, it will compute Ms and verify if $Ms =$

$h(Vi\|c\|u)$. Then, the smart device computes the session key $sk = h(Vi\|c\|u)$. In Step 3, after the cloud server receiving Mu from the smart device, it will compute Mu and verify if $Mu = h(h(pw\|b)\|Vi\|c\|u)$. Then, the cloud server computes the session key $sk = h(Vi\|c\|u)$. The session key sk has the parameters $(Vi\|c\|u)$. The parameter (Vi) contains $h(ID_i\|s\|CI_i)$. It can verify the user. The parameter (c) contains $(e \times x = r \times x \times G)$. It is computed by the smart device in the pre-computation phase. Therefore, the parameter (c) is fresh, which can only been used once by this phase. The parameter (u) is a random number and chosen by the cloud server in the log-in phase. Each time has the different session key in the log-in phase. Therefore, an attacker cannot forge and intercept the session key in each phase.

4.1.2. *Preventing the replay attack.* Our proposed scheme can prevent this attack because the random number r is chosen by the smart device, and the random number u is chosen by the cloud server in the log-in phase. It will change in the log-in phase every time. Therefore, the attacker cannot get the random number r. Therefore, the attacker cannot create the parameters Ms and Mu to pass the identity authentication and login the server. In other way, the attacker cannot get the user's password pw to decrypt AK and Vi. The attacker cannot use the parameter AK to decrypt $E_{AK}(b)$ in the pre-computation phase. Therefore, the attacker cannot decrypt the parameter $E_{AK}(b)$ to create the parameter $E_{Vi}(h(pw\|b), e)$. Therefore, the attacker cannot pass the checking to use the replay attack to login the log-in server.

4.1.3. *Preventing the offline dictionary attack with the lost smart device.* The traditional methods would store the data with the high secret information into the device or the smart card. Different with the traditional methods, in our proposed scheme, the smart device does not spend a lot of resources to store parameters, because the parameters $ID_i, E_{pw}(AK, Vi)$ are stored in the key delegation center. Therefore, if the attacker gets the user's smart device and tries to get the information in the device, it is not useful. The user's smart device does not contain the parameter pw. Even if the attacker gets the smart device, it also is unable to forge the parameter $h(pw\|b)$ to login the log-in server.

4.1.4. *Using in the cloud environment with any smart device.* The traditional methods would store the data with the high secret information into the device or the smart card. In our proposed scheme, we delegate the parameters ID_i, CI_i, bi, Vi, $E_{AK}(b)$ in the cloud registration table. When the cloud user lost his smart device or get a new cloud device, it can change his password in the password-changing phase and update two keys AK, Vi with any smart device in the pre-computation phase. Therefore, the user's rights and interests will not be affected.

4.1.5. *Preventing the insider attack.* In this attack, the insider may use his authority to get a user's password. In our proposed scheme, the smart device computes the parameter $h(pw\|b)$ with hash function before sending it to the cloud log-in server. Therefore, the stored password in the server database is encrypted. Our proposed scheme can prevent this attack because the attacker cannot get the complete password parameter pw.

4.2. **Cost and functionality consideration.**

4.2.1. *No password table.* In the registration phase, the cloud server always needs the password table to store the cloud user's password. It not only wastes the database storage memory, but also increases the risk of insider attack. In our proposed scheme, the hashed password with a random number $h(pw\|b)$ is encrypted in the parameter $bi = h(ID_i\|CI_i\|h(pw\|b))$. It can increase the security level of the log-in phase.

4.2.2. *Choosing and changing passwords by users.* In Li's scheme, in order to increase the entropy of the password, the server always gives the random password to the user, but this password is hard to remember for user. Therefore, if the user wants to inquire or change her/his password. It will increase the extra cost for the server. In our proposed scheme, the cloud user can choose his password pw and the random number b to do authentication. It would increase the entropy of the password. The random number b will be encrypted by AK. Then, the ciphertext will be sent to the key delegation center to store it. It not only can increase the entropy of the password and the high security level, but also can provide the funtion to choose and change the password.

4.2.3. *No time-synchronization problem.* In the log-in phase, in order to prevent the replay attack, timestamps can be used to prevent this problem. This approach will cause the time-synchronization problem. In our proposed scheme, we use two nonces r and u to compute Ms and Mu. Challenge and response between the user's smart device and the cloud login server is used in our scheme for preventing the replay attack. Therefore, we do not have the time-synchronization problem in our proposed scheme.

4.2.4. *No quota limit of the number of a user's smart devices.* In the registration phase, the user's smart device encrypts the parameter b by the key AK and stores the ciphertext in the cloud server. In the pre-computation phase, the user's smart device checks if the smart device does not have AK, Vi or $Message$. If yes, the smart device has to request the key delegation center to get the new parameter $E_{pw}(AK, Vi)$. It can prevent the user from logining the server with the new smart device. In the password-changing phase, the parameter CI_i changes every time when the user changes his password. Therefore, the parameter Vi will be different when the parameter CI_i is changed. Then, the smart device sends the parameters ID_i, $E_{pw}(AK, Vi)$ to the key delegation center. After receiving ID_i and $E_{pw}(AK, Vi)$, the key delegation center stores the parameters ID_i and $E_{pw}(AK, Vi)$ in the key delegation center database and sends $Message$ to the smart device.

4.2.5. *Identity protection.* In our proposed scheme, there are two kinds of the identity protection: 1. identity protection of the user's smart device; 2. identity protection of communication between the cloud server and cloud user. In identity protection of the user's smart device, the user's smart device does not store any personal information or the high secret information in the smart device. It only has encryption or decryption operation keys AK and Vi. In identity protection of communication between the cloud server and cloud user, it only sends the parameters $h(pw\|b)$ and e. The parameters are encrypted by Vi. Therefore, the attacker cannot get the cloud user's identity information of communication between the cloud server and the cloud user.

4.2.6. *Providing session key agreement.* After the log-in phase, any communication can use the session key to encrypt or decrypt the messages. In the pre-computation phase of our proposed scheme, the user's smart device will choose the nonce r. Then, the cloud log-in server chooses the nonce u in the log-in phase. The nonces will be different every time. After this phase, the cloud log-in server and the user's smart device can create the session key parameter sk by himself.

4.2.7. *Without any user's identity information in the user's smart device.* We assume that the attacker can get the smart device. She/he will use any tools to get the information in the smart device. In our proposed scheme, we can solve the smart device lost problem because we store the user's identity information and the encryption parameters in the cloud server and the key delegation center. Therefore, the user's smart device only temporarily stores the parameters AK and Vi in the cloud device.

4.2.8. *Low communication and computation cost.* In P1 of Table 1, our scheme, Juang et al.'s scheme [24], Sun et al.'s scheme [4], Singh et al.'s scheme [13], and Li et al.'s scheme [25] use the password. The password length is 128 bits. In P2 of Table 1, our scheme has the parameters $[AK, Vi]$. The memory length is $128 + 128 = 256$ bits. Juang et al.'s scheme [24] has the parameters $[bi, Vi, IDi, CIi, b]$. The memory length is $384 + 128 + 32 + 32 + 64 = 640$ bits. Sun et al.'s scheme [4] has the parameters $[V, IM]$. The memory length is $128 + 128 = 256$ bits. Singh et al.'s scheme [13] has the parameters $[V, IM]$. The memory length is $128 + 128 = 256$ bits. Li et al.'s scheme [25] has the parameters $[b, N0, ID, VID, ID, CI]$. The memory length is $384 + 128 + 32 + 32 + 64 = 640$ bits. In P3 of Table 1, our scheme has the parameters $[E_{Vi}(h(pw\|b), e), u, Ms, Mu]$. The memory length is $384 + 64 + 128 + 128 = 704$ bits. Juang et al.'s scheme [24] has the parameters $[bi, E_{Vi}(e), u, Ms, Mu]$. The memory length is $384 + 384 + 64 + 128 + 128 = 1088$ bits. Sun et al.'s scheme [4] has the parameters $[IM, GC, MS, GS, MU]$. The memory length is $128 + 326 + 128 + 326 + 128 = 1036$ bits. Singh et al.'s scheme [13] has the parameters $[IM, GC, MS, GS, MU]$. The memory length is $128 + 326 + 128 + 326 + 128 = 1036$ bits. Li et al.'s scheme [25] has the parameters $\left[b_{ID}^{No}, E_{V_{ID}}(N_1\|e), Nb1, u \oplus h64(b_{ID}^{N_1}), MS, MU\right]$. The memory length is $896+576+128 = 1600$ bits. The detail efficiency comparison among our scheme and related schemes is shown in Table 1.

Table 1. The communication and the storage cost comparison among our scheme and related schemes

	P1	P2	P3
Our scheme	128 bits	256 bits	704 bits
Juang et al.'s scheme [24]	128 bits	640 bits	1088 bits
Sun et al.'s scheme [4]	128 bits	256 bits	1036 bits
Singh et al.'s scheme [13]	128 bits	256 bits	1036 bits
Li et al.'s scheme [25]	128 bits	640 bits	1600 bits

P1: Password length
P2: Memory for storing the cryptographic parameters in a smart device
P3: Communication cost of the login for cryptographic parameters

In Table 2, we let $Hash$ be the time of the one hashing operation, Sym be the time of the one symmetric encryption or decryption operation, EC_M be the time for the multiplication operation of a number over an elliptic curve, M be the time for the modular multiplication. We assume that $EC_M \approx 29\ M$ and $EXP \approx 240\ M$, and they are referenced in [12]. We assume that $hash \approx 0.4\ M$ and $sym \approx 0.4\ M$, and they are referenced in [14,27]. We assume that an elliptic curve over a 163-bit field has the same security level as 1024-bit public key cryptosystems such as the RSA or the Diffile-Hellman cryptosystem. In P1 of Table 2, our scheme needs 2 $Hash$ operations. Juang et al.'s scheme [24] needs 3 $Hash$ operations. Sun et al.'s scheme [4] needs 2 $Hash$ operations and 1 Sym operations. Singh et al.'s scheme [13] needs 2 $Hash$ operations and 1 Sym operations. Li et al.'s scheme [25] needs 2 $Hash$ operations and 3 Sym operations. In P2 of Table 2, our scheme, Juang et al.'s scheme [24] and Li et al.'s scheme [25] need 2 EC_M operations. Sun et al.'s scheme [4] and Singh et al.'s scheme [13] do not need the computation cost of the cloud user's device of the pre-computation phase. In P3 of Table 2, our scheme needs 3 $Hash$ operations and 1 Sym operations. Juang et al.'s scheme [24] needs 3 $Hash$ operations and 1 Sym operations. Sun et al.'s scheme [4] needs 4 $Hash$ operations and 2 Sym operations. Singh et al.'s scheme [13] needs 4 $Hash$ operations and 2 Sym operations. Li et al.'s scheme [25] needs 8 $Hash$ operations and 4 Sym operations. In P4 of Table 2, our scheme needs 4 $Hash$ operations and 1 Sym operations and 1 EC_M operations. Juang et al.'s scheme

[24] needs 4 $Hash$ operations and 2 Sym operations and 1 EC_M operations. Sun et al.'s scheme [4] needs 4 $Hash$ operations and 1 Sym operations and 2 EC_M operations. Singh et al.'s scheme [13] needs 4 $Hash$ operations and 1 Sym operations and 2 EC_M operations. Li et al.'s scheme [25] needs 10 $Hash$ operations and 10 Sym operations and 1 EC_M operations. The detail efficiency comparison among our scheme and related schemes is shown in Table 2.

Table 2. The computation cost comparison among our scheme and related schemes

	P1	P2	P3	P4
Our scheme	$2Hash \cong 0.8M$	$2EC_M \cong 58M$	$3Hash + 1Sym \cong 1.6M$	$1EC_M + 1Sym + 4Hash \cong 31M$
Juang et al.'s scheme [24]	$3Hash + 1Sym \cong 1.6M$	$2EC_M \cong 58M$	$3Hash + 1Sym \cong 1.6M$	$1EC_M + 2Sym + 4Hash \cong 31.4M$
Sun et al.'s scheme [4]	$2Hash + 1Sym \cong 1.2M$	NA	$2EC_M + 4Hash \cong 59.6M$	$2EC_M + 1Sym + 4Hash \cong 60M$
Singh et al.'s scheme [13]	$2Hash + 1Sym \cong 1.2M$	NA	$2EC_M + 4Hash \cong 59.6M$	$2EC_M + 1Sym + 4Hash \cong 60M$
Li et al.'s scheme [25]	$2Hash + 3Sym \cong 2M$	$2EC_M \cong 58M$	$8Hash + 4Sym \cong 4.8M$	$1EC_M + 10Sym + 10Hash \cong 37M$

P1: The computation cost of the registration for the server
P2: The computation cost of the pre-computation phase for the cloud user's device
P3: The computation cost of the login for the user's device
P4: The computation cost of the login for the server

The detail functionalities comparison among our scheme and related schemes is shown in Table 3.

Table 3. The functionality comparison

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
Our scheme	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Juang et al.'s scheme [24]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes
Sun et al.'s scheme [4]	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	No	No
Singh et al.'s scheme [13]	Yes	Yes	No	Yes	Yes	No	Yes	No	No	No	No	No
Li et al.'s scheme [25]	No	No	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No

P1: Low communication and computation cost
P2: No password table
P3: Choosing and changing passwords by users
P4: No time-synchronization problem
P5: Mutual authentication
P6: Revoking the lost cards without changing the user's identity
P7: Identity protection
P8: Providing session key agreement
P9: Preventing the offline dictionary attack with the smart device
P10: No quota limit of a user's smart device
P11: Without any user's identity information in the user's smart device
P12: Preventing the insider attack

5. **Conclusions.** In the cloud environment, the cloud server has more computing and storage ability. The user can transfer the major computing loading to the cloud and can store the secret information in the cloud. This will cause that an attacker tries to get this valuable information and do various attacks. Therefore, in this environment, how to provide a secure authentication mechanism that the user can use many various devices to

do authentication and to solve the device lost problem is very important. In this paper, we proposed a lightweight robust and privacy protection authentication scheme for cloud computing. Our proposed scheme can solve the time-synchronization problem, the offline dictionary attack with the smart device, the replay attack, the offline dictionary attack with the lost smart device, and the insider attack. In addition, our proposed scheme can provide many advantages. The advantages are mutual authentication, using in the cloud environment with any smart device, low communication and computation cost, no password table, no quota limit of the number of a user's smart devices, identity protection, providing session key agreement and without any user's real identity information in the user's smart device.

Acknowledgment. This work was supported in part by the National Science Council of Taiwan under the Grant NSC 100-2221-E-327-019-MY2, NSC 101-2219-E-110-003, NSC 101-2219-E-110-005, and "Aim for the Top University Plan" of the National Sun Yat-sen University and Ministry of Education, Taiwan.

REFERENCES

[1] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer and D. H. J. Epema, Performance analysis of cloud computing services for many-tasks scientific computing, *IEEE Transactions on Parallel and Distributed Systems*, vol.22, pp.931-945, 2011.

[2] A. Weiss, Computing in the clouds, *NetWorker – Cloud Computing: PC Functions Move onto the Web*, vol.11, no.4, pp.16-25, 2007.

[3] C. C. Chang and C. Y. Chung, An efficient session key generation protocol, *Communication Technology Proceedings*, vol.1, pp.203-207, 2003.

[4] D. Z. Sun, J. P. Huai, J. Z. Sun, J. X. Li, J. W. Zhang and Z. Y. Feng, Improvements of Juang et al.'s password-authenticated key agreement scheme using smart cards, *IEEE Transactions on Industrial Electronics*, vol.56, pp.2284-2291, 2009.

[5] E.-J. Yoon, S.-H. Kim and K.-Y. Yoo, A security enhanced remote user authentication scheme using smart cards, *International Journal of Innovative Computing, Information and Control*, vol.8, no.5(B), pp.3661-3675, 2012.

[6] E.-J. Yoon, S.-B. Choi and K.-Y. Yoo, A secure and efficiency ID-based authenticated key agreement scheme based on elliptic curve cryptosystem for mobile devices, *International Journal of Innovative Computing, Information and Control*, vol.8, no.4, pp.2637-2653, 2012.

[7] E.-J. Yoon and K.-Y. Yoo, Improving the LEE-LEE'S password based authenticated key agreement protocol, *International Journal of Innovative Computing, Information and Control*, vol.8, no.8, pp.5657-5675, 2012.

[8] G. Boss, P. Malladi, D. Quan, L. Legregni and H. Hall, *Cloud Computing*, Copyright IBM Corporation, 2007.

[9] H. Roh and S. Jung, Session key exchange and mutual authentication scheme between mobile machines in WLAN based ad hoc networks, *Information and Communication Technology Convergence (ICTC)*, pp.482-483, 2010.

[10] J. Baliga, R. W. A. Ayre, K. Hinton and R. S. Tucker, Green cloud computing: Balancing energy in processing, storage, and transport, *Proc. of the IEEE*, vol.99, pp.149-167, 2011.

[11] J. Mirkovic and P. Reiher, *A Taxonomy of DDoS Attack and DDoS Defense Mechanisms*, 2004.

[12] K. Lauter, The advantages of elliptic curve cryptography for wireless security, *Wireless Communications*, vol.11, pp.62-67, 2004.

[13] K. Singh, A. Khalique and S. Sood, A password-authenticated key agreement scheme based on ECC using smart cards, *International Journal of Computer Applications*, vol.2, pp.26-30, 2010.

[14] K. Takashima, Scaling security of elliptic curves with fast pairing using efficient endomorphisms, *Institute of Electronics, Information and Communication Engineers (IEICE)*, vol.E90-A, pp.152-159, 2007.

[15] L. Qian, Z. Luo, Y. Du and L. Guo, Cloud computing: An overview, *CloudCom., LNCS*, vol.5931, pp.626-631, 2009.

[16] M. Bellare and P. Rogaway, Provably secure session key distribution – The three party case, *Proc. of the 27th Annual ACM Symposium on Theory of Computing*, pp.57-66, 1995.

[17] M. K. Franklin and M. K. Reiter, Fair exchange with a semi-trusted third party, *Proc. of the 4th ACM Conference on Computer and Communications Security*, pp.1-5, 1997.

[18] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation*, vol.48, pp.203-209, 1985.

[19] R. C. Merkle, One way hash functions and DES, *Proc. of Advances in Cryptology*, 1998.

[20] R. Ranchal, L. B. Othmane, A. Kim, M. Kang and M. Linderman, Protection of identity information in cloud computing without trusted third party, *IEEE International Symposium on Reliable Distributed Systems*, pp.368-372, 2010.

[21] V. Miller, Use of elliptic curves in cryptography, *Advances in Cryptology Cryptology, Lecture Notes in Computer Science*, vol.218, pp.417-426, 1985.

[22] W. A. Warr, *Cloud Computing*, Wendy Warr and Associates, 2009.

[23] W. S. Juang, Efficient password authenticated key agreement using smart cards, *Computers and Security*, vol.23, pp.167-173, 2004.

[24] W. S. Juang, S. T. Chen and H. T. Liaw, Robust and efficient password-authenticated key agreement using smart cards, *IEEE Transactions on Industrial Electronics*, vol.55, pp.2551-2556, 2008.

[25] X. Li, W. Qiu, D. Zheng, K. Chen and J. Li, Anonymity enhancement on robust and efficient password-authenticated key agreement using smart cards, *IEEE Transactions on Industrial Electronics*, vol.57, pp.793-800, 2010.

[26] X. Wang, Y. L. Yin and H. Yu, Finding collisions in the full SHA-1, *Advances in Cryptology, Lecture Notes in Computer Science*, pp.17-36, 2005.

[27] Z. Li, J. Higgins and M. Clement, Performance of finite field arithmetic in an elliptic curve cryptosystem, *The 9th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pp.249-256, 2001.

[28] Z.-Y. Cheng, Y.-Liu, C.-C. Chang and S.-C. Chang, A smart card based authentication scheme for remote user login and verification, *International Journal of Innovative Computing, Information and Control*, vol.8, no.8, pp.5499-5511, 2012.