Certificateless Authenticated Group Key Agreement Protocol for Unbalanced Wireless Mobile Networks

Chung-Fu Lu[*+], Tzong-Chen Wu[*], and Chien-Lung Hsu[‡]

[*]Department of Information Management,
National Taiwan University of Science and Technology,
43, Section 4, Keelung Road, Taipei 106
Taiwan, Republic of China, R.O.C.
D9009102@mail.ntust.edu.tw; tcwu@cs.ntust.edu.tw

[+]Department of Computer and Communication Engineering
Taipei College of Maritime Technology
No.212, Sec.9, Yan-Pin N. Rd., Taipei 111
Taiwan, Republic of China, R.O.C.
peter@mail.tcmt.edu.tw

[‡]Department of Information Management,
Chang-Gung University
259, Wen-Hwa 1st Road, Kwei-Shan, Tao-Yuan 333
Taiwan, Republic of China, R.O.C.
clhsu@mail.cgu.edu.tw

Abstract: In 2004, Bresson *et al*. proposed a mutual authentication and group key agreement protocol for unbalanced wireless networks. Tseng recently proposed a novel secure protocol to improve Bresson *et al*.'s protocol. However, both protocols are based on certificate-based public key systems and insecure against the so-called impersonation attacks. They might be unsuitable for unbalanced wireless mobile networks from the viewpoints of security, computational complexities, and communication overheads. This paper proposes a certificateless authenticated group key agreement (cAGKA) protocol based on elliptic curve discrete logarithms. The proposed cAGKA protocol is more secure and efficient than previously proposed protocols for unbalanced wireless mobile networks due to the following facts: (i) The entity authentication and the authenticity of the intended public keys can be simultaneously verified in a logically single step without requiring any public key certificates. (ii) Bit sizes of the keys and the related messages are relatively smaller than those of the previously proposed protocols for the same security level. (iii) It saves the required communication overheads, and computational complexities. (iv) It achieves mutual authentication, impersonation attack resistance, explicit key confirmation, forward secrecy, contributory key agreement, and group key updating.

Key-Words: Group key agreement, Certificateless, Elliptic curve, Unbalanced wireless networks, implicitly-certified

1 Introduction

Wireless mobile networks which consist of low-power and powerful nodes have attracted significant attentions recently in variety of applications [1-3]. In general, a low-power node is a device with very-restricted computing power and some required memory capacity such as cell phones, personal assistant devices (PDA), and etc. A powerful node is a device with high computing power capabilities and large memory capacity, such as the base stations of cellular mobile networks, the access points of wireless local area networks or the cluster-heads of mobile ad hoc networks. Hence, such wireless mobile networks are also called "unbalanced" wireless mobile network. Due to the wireless and unbalance properties, the wireless mobile networks may suffer from more potential attacks than wired networks. It is a nontrivial challenge to secure the

wireless mobile network under the considerations of the limited computing capabilities and electronic power of low-power nodes, less network bandwidth, greater transmission delay time, more unstable network connection, etc. Most security protocols and mechanisms currently deployed in wired networks might be unsuitable to such unbalanced wireless mobile networks.

Recently, Bresson *et al.* [4] proposed an authenticated group key agreement protocol for unbalanced wireless networks based on public key technology. It allows a cluster of low-power nodes and one powerful node (e.g. wireless gateway) to dynamically agree on a group secret key shared among them for securing communications. In 2005, Nam *et al.* pointed out the critical security flaws inherent in Bresson *et al.*'s protocol to show their protocol cannot achieve forward secrecy, implicit key authentication, and known key security [5]. They also proposed a patch to fix the security flaws. Later, Nam *et al.* [6] further proposed a group key agreement protocol with constant-round for the unbalanced wireless networks under decisional Diffie-Hellman assumption. The Nam *et al.*'s protocol however is non-authenticated one which implies it cannot provide user and message authentication.

In 2006, Tseng [7] showed that Bresson *et al.*'s and Nam *et al.*'s protocols are not contributory key agreement ones in which the group secret keys are derived from the contributions of all participant nodes. Furthermore, Tseng proposed a real contributory key agreement protocol [8] to allow every group node to contribute their shares to the group key generation. It is more efficient than Bresson *et al.*'s and Nam *et al.*'s protocols in terms of the computational complexities but is less efficient than those in terms of the communication overheads.

In Tseng's and Bresson *et al.*'s protocols, each low-power node must generate and transmit digital signature to the powerful node for entity authentication. They both suggest that low-power nodes with limited computing capabilities can prepare these digital signatures beforehand by off-line pre-computing them in advance. However, this paper will demonstrate that these two protocols are both vulnerable to the so-called impersonation attacks since no timestamp or nonce is bound in signing messages. The adversary can intercept the signatures and masquerade as the intended legal nodes by replaying the intercepted digital signatures to cheat the powerful node and other participating low-power node(s). Although the adversary cannot derive the correct group secret key, both above protocols failed in entity authentication. It is easy to see that the adversary might collude with any participating low-power node and obtain the established group secret key from the node. Under such situation, the adversary can masquerade as the legal low-power node to communicate with other participating nodes (including powerful node) without being detected.

In addition, fundamental cryptographic primitives used in all above mentioned protocols are based on discrete logarithm problem with large primes. That means specifications of related cryptographic parameters, private keys, and public keys must be limited to the required larger bit length (e.g., 1024 bits) for the intended security level. Costs of key management, computational performance, communication overheads, and the size of the required storage are heavy to mobile devices. Victor Miller [9] and Neil Koblitz [10] discover elliptic curve cryptography (ECC) as an alternative solution to public key system in the middle of 1980s. Unlike other popular public key systems such as RSA, ECC is based on elliptic curve discrete logarithms (ECDL) that more difficult to challenge at equivalent key sizes. In another words, key size required in ECC is smaller than that of other public key systems at the same security level. Table 1 illustrates the comparison of the required key size used in symmetric cryptosystems, asymmetric cryptosystems/public key systems, and ECC systems at equivalent level of security. For example, the key size used in traditional public key systems requires 1024 bits but 163 bits in ECC. Smaller key size will gain better computational performance, lower key management costs, and bandwidth savings, which is especially suitable for unbalanced wireless mobile networks.

Table 1. Comparison of the required key size at the same security level [11]

Symmetric cryptosystems (Bits)	Asymmetric Cryptosystems (Bits)	ECC (Bits)
80	1024	163
112	2048	233
128	3072	283
192	7680	409
256	15360	571

Moreover, all of the above proposed protocols are based on the certificate-based public key systems. This means the authenticity of all public key will be verified by checking the validity of extra public key certificates issued by a certification authority (CA). All nodes (including powerful and low-power nodes) must get the intended nodes' public key certificates and then check their validity before performing Bresson *et al.*'s, Nam *et al.*'s, and Tseng's protocols. Otherwise, these protocols might suffer from some potential man-in-the-middle attacks. The certificate-based public key systems might be unsuitable for unbalanced wireless mobile networks from viewpoints of computational complexities and communication overheads.

This paper will propose a *certificateless* authenticated group key agreement (*c*AGKA) protocol based on ECDL for unbalanced wireless mobile networks. It achieves the same security requirements of contributory group key agreement, key confirmation, forward secrecy, and mutual entity authentication as mentioned in Tseng's protocol. The new property "certificateless" means that the authenticity of all public keys are implicitly verified in key agreement procedure without requiring and verifying extra public key certificates. Such public keys used in the proposed protocol are also called implicitly-certified public keys [12] or certificateless ones [13]. As compared with Bresson *et al.*'s, Nam *et al.*'s, and Tseng's protocols, the proposed *c*AGKA protocol has the following advantages:

(i) The proposed protocol is secure against the impersonation attacks.

(ii) Its bit sizes of the key and the related messages are relatively smaller than those of above mentioned protocols at the same security level. The costs of key management and message transmission are thus relatively lower.

(iii) It reduces space requirements due to no required storage for storing public key certificates.

(iv) It does not require extra communication and computational costs to transmit and verify the authenticity of the intended public keys since no extra certificates are required. Entity authentication and the authenticity of the intended public keys can be simultaneously verified in a logically single step.

(v) Node joining and leaving issues are considered from the practical viewpoints in the proposed protocol.

The rest of this paper is organized as follows: In Section 2, we briefly review Tseng's protocol. In Section 3, we propose the *c*AGKA protocol. We discuss the security analysis and performance evaluation of the proposed *c*AGKA protocol in Section 4 and 5, respectively. Finally, some concluding remarks are presented in Section 6.

2 Brief Review and Discussions of Tseng's Protocol

2.1 Review of Tseng's Protocol

In 2007, Tseng modified the Bresson *et al.*'s protocol [4] to propose a real contributory group key agreement protocol [8] for an unbalanced wireless network consisting of some low-power nodes and a powerful node. In Bresson *et al.*'s and Tseng's protocols, the fundamental cryptographic primitives are based on discrete logarithm problem with large primes and all public keys are certificate-based ones. In this session, we briefly review Tseng's protocol and demonstrate an impersonation attack on it.

Let p' and q' be two large primes (where $p' = 2q' + 1$), g be a generator for the subgroup $G_{q'}$, and $h(\cdot)$ be a secure one-way hash function, respectively. Without loss of generality, let $N = \{N_1, N_2, ..., N_n\}$ be the set of low-power nodes that want to agree on a group secret key shared among them and a powerful node N_A. Private and public keys of a node *i* are denoted as SK_i and PK_i, where $SK_i \in_R Z_{q'}^*$, $PK_i = g^{SK_i} \bmod p'$, and "$\in_R$" denotes "randomly chosen from". Initially, each low-power node $N_i \in N$ computes x_i^{-1}, $y_i = g^{x_i} \bmod p'$, $\alpha_i = PK_A{}^{x_i} \bmod p'$, and the signature $\delta_i = Sign(SK_i, y_i)$ for y_i, where $x_i \in_R Z_{q'}^*$ and $Sign(SK_i, y_i)$ denotes the signing algorithm (e.g., DSA [14] or ElGamal's digital signature scheme [15]) under the private key SK_i for y_i. The tuple $(x_i, x_i^{-1}, \alpha_i, y_i, \delta_i)$ should be stored in the low-power node N_i in advance.

In Tseng's protocol, all nodes in *N* will cooperate with a powerful node N_A to agree on a group secret key among them by performing the following steps.

Step 1. Each low-power node $N_i \in N$ sends the pre-computed $\{y_i, \delta_i\}$ to the powerful node N_A for entity authentication.

Step 2. The powerful node N_A verifies legitimacy of N_i's identity and generates group secret key by performing the following sub-steps.

Step 2-1. On receiving the $\{y_i, \delta_i\}$ from N_i (for $i = 1,2,...,n$), the powerful node N_A verifies the validity of the signature δ_i by $Verify(PK_i, y_i, \delta_i)$, where $Verify($ $y_i, \delta_i)$ denotes the signature verification under the public key PK_i for the signature δ_i . If δ_i pass $Verify(PK_i,$ $y_i, \delta_i)$, the legitimacy of N_i's identity is verified.

Step 2-2. The powerful node N_A computes $X = g^x \bmod p'$, $z_i = y_i^x \bmod p'$, $\alpha_i' = y_i^{SK_A} \bmod p'$, $C = h(X \oplus z_1 \oplus z_2 \oplus ... \oplus z_n)$, and $k_G = X\prod_{i=1}^{n} z_i$ where $x \in_R Z_{q'}^*$, $i = 1,2,...,n$, and the symbol "⊕" denotes an exclusive-or (XOR) operation. Note k_G is regarded as the shared group secret key. Finally, N_A broadcasts $\{C, (\alpha_i', z_i); i = 1,2,...,n\}$ to all low-power nodes in N.

Step 3. Upon receiving $\{C, (\alpha_i', z_i); i = 1,2,...,n\}$, each low-power node $N_i \in N$ checks if the received α_i' equals to the stored α_i . If it holds, the low-power node N_i further uses the stored x_i^{-1} to compute $X' = z_i^{x_i^{-1}} \bmod p'$ and checks if $C = h(X' \oplus z_1 \oplus z_2 \oplus ... \oplus z_n)$. If it holds, N_i can derive the group secret key by $k_G = X'\prod_{i=1}^{n} z_i$.

2.2 Discussions of Tseng's Protocol

In Tseng's protocol, all public keys are certificate-based ones. Extra public key certificates with respect to all public keys must be issued by the certification authority (CA) and verified by the verifier before using the public keys for withstanding well-known man-in-the-middle attacks. Hence, it requires extra communication and computational costs to transmit and verify the intended public key certificates. The extra costs might be heavy for unbalanced wireless mobile networks and mobile devices.

Moreover, we will show that Tseng's protocol is vulnerable to the impersonation attacks. Consider the scenario of an impersonation attack that the adversary attempts to masquerade as an intended low-power node N_i for cheating other participating nodes and the powerful node. As mentioned above, the legitimacy of N_i's identity is verified by the pre-computed and stored authentication message $\{y_i, \delta_i\}$ with respect to the low-power node N_i. Although the Tseng's protocol have adopted On-line/Off-line signature scheme proposed by Shamir-Tauman's scheme [16], no timestamps or nonce are real bound in the signature δ_i . If the adversary intercepted an authentication message $\{y_i, \delta_i\}$ transmitted by N_i, he can masquerade as the node N_i to perform the Step 1 of Tseng's protocol by replaying $\{y_i, \delta_i\}$ in the next group communication session. The replayed message $\{y_i, \delta_i\}$ will pass the signature verification in Step 2, still the adversary cannot derive the group secret key k_G with the messages broadcasted by the powerful node in Step 3. The powerful node and other participating low-power node(s) will believe that the node N_i impersonated by the adversary participates in the protocol. This implies Tseng's protocol failed in entity authentication. Moreover, if such an adversary is able to collude with any corrupted low-power nodes, he can obtain the group secret key revealed from them. For instance, an adversary can masquerade as the node N_i by replaying $\{y_i, \delta_i\}$ and collude with a corrupted low-power node in the next group communication session. Then, he can always get a copy of the group key and masquerade as the legal low-power node N_i to participate in the group communication without being detected.

3 The Proposed *c*AGKA Protocol

There are three roles involved in the proposed certificateless authenticated group key agreement *c*AGKA protocol: system authority (SA), low-power nodes, and a powerful node as mentioned above. The SA is responsible to generate all necessary system parameters and cooperates with each node to generate valid node's private and public key pair. The powerful node will authenticate the legitimacy of the participant low-power nodes and determine a group secret key shared among them.

According to the Tseng's protocol, we also assume that the SA and powerful node are trusted. The proposed *c*AGKA protocol consists of five

phases: the system setup, the mobile node registration, the authenticated group key agreement, the node leaving, and the node joining phases. Detailed descriptions of these phases are given below.

3.1 System Setup Phase

Initially, the SA determines a large prime p and a non-supersingular elliptic curve $E_p(a,b)$ as $y^2 = x^3 + ax + b \pmod p$, where $a,b \in_R Z_p^*$ and $4a^3 + 27b^2 \bmod p \neq 0$. The SA further determines a large prime q and a base point G of order q over $E_p(a,b)$, where q is a divisor of the number of points on the elliptic curve $E_p(a,b)$. Let O be a point at infinity over $E_p(a,b)$, $Q_{i.x}/Q_{i.y}$ be the x-coordinate/y-coordinate of the point Q_i [17], and $h(\cdot)$ be a secure one-way hash function that accepts a variable length input and produces a fixed length output which is over GF(q). The private and public keys for the SA are respectively defined as s_{SA} and P_{SA}, where $s_{SA} \in_R Z_q$ and $P_{SA} = s_{SA}G$. The SA publishes $(p,q,E_p(a,b),O,h,G,P_{SA})$ while keeps s_{SA} secret.

3.2 Mobile Node Registration Phase

When a mobile node N_i associated with a distinguished identifier I_i wants to join the system, he will cooperate with the SA to perform the following steps to generate a valid private and public key pair:

Step 1. The mobile node N_i randomly chooses an integer $v_i \in_R Z_q$, computes $V_i = v_iG$, and then sends $\{I_i, V_i\}$ to the SA.

Step 2. On receiving $\{I_i, V_i\}$ sent from the node N_i, the SA checks whether the identifier I_i is unregistered. If it holds, the SA computes and returns $\{P_i, s_i\}$ to the node N_i, where $P_i = V_i + h(r_i \| I_i)G = (P_{i.x}, P_{i.y})$, $s_i = h(r_i \| I_i) + (P_{i.x} + I_i) \cdot s_{SA} \bmod q$, $r_i \in_R Z_q$, and "‖" is the concatenation symbol. Note that P_i is regarded as N_i's public key issued by the SA.

Step 3. The node N_i computes a private key x_i as $x_i = s_i + v_i \bmod q$. Further, N_i can verify the validity of the private key x_i by checking if $x_iG = P_i + (P_{i.x} + I_i)P_{SA}$. If it holds, (x_i, P_i) is a valid key pair of N_i.

3.3 Authenticated Group Key Agreement Phase

Without loss of generality, let $N = \{N_1, N_2, ..., N_n\}$ be the set of n low-power nodes that want to agree on a group secret key shared among them. All the low-power nodes will cooperative with a powerful node N_A to generate the group secret key. Responsibility of the N_A are to authenticate the identity of all low-power nodes and determines the group secret key k_G for them. The procedure for the authenticated group key agreement phase is stated as follows (as depicted in Fig. 1).

Step 1. Each low-power node N_i computes r_i^{-1}, $R_i = r_iG$, $B_i = x_i(P_A + (P_{A.x} + I_A)P_{SA})$, $C_i = r_i(P_A + (P_{A.x} + I_A)P_{SA})$, and $a_i = h(B_{i.x} \| C_{i.x} \| I_A \| I_i \| t_i)$, where $r_i \in_R Z_q$ and t_i is the current timestamp. Finally, N_i sends $\{I_i, P_i, R_i, t_i, a_i\}$ to N_A.

Step 2. The powerful node N_A verifies the legitimacy of N_i's identity and generates group secret key by performing the following sub-steps.

Step 2-1. On receiving $\{I_i, P_i, R_i, t_i, a_i\}$ from N_i (for $i = 1,2,...,n$), the powerful node N_A checks whether $t_i' - t_i \leq \Delta t$, where t_i' is the timestamp of receiving $\{I_i, P_i, R_i, t_i, a_i\}$ and Δt is the preset acceptable delay threshold. If it holds, the powerful node N_A computes $B_i' = x_A(P_i + (P_{i.x} + I_i)P_{SA})$, $C_i' = x_AR_i$, and verifies the legitimacy of the low-power node N_i by checking if $h(B_{i.x}' \| C_{i.x}' \| I_A \| I_i \| t_i)$ equals to the received a_i. If it does not hold, N_A requests N_i to re-send valid $\{I_i, P_i, R_i, t_i, a_i\}$. Otherwise, both of the identity authentication and the authenticity public key P_i for N_i are verified.

Step 2-2. The powerful node N_A computes $R_A = r_AG$, $z_i = h(R_{A.x} \| B_{i.x}' \| C_{i.x}'$

$\| t_A)$, $k_G = h(R_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$, $m = h(k_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$, and $Y_i = r_A R_i$, where t_A is the current timestamp of powerful node N_A, $r_A \in_R Z_q$ and $i = 1,2,...,n$. Finally, N_A broadcasts $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2,...,n\}$ to all low-power nodes. Note that k_G is the shared group secret key. If the broadcast message is lost in transmission, it may cause the nodes are not able to join the group until those nodes send another join request. The power node should unicast or broadcast the above message to those nodes which did not get the group key.

Step 3. On receiving $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2,...,n\}$ from the powerful node N_A, each low-power node N_i checks whether $t_i'' - t_i \le \Delta t'$, where t_i'' is the timestamp of receiving $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2, ...,n\}$ and $\Delta t'$ is the preset acceptable delay threshold. If it holds, the node N_i can compute $R'_A = r_i^{-1} Y_i$ and verify the legitimacy of N_A's identity and the authenticity of N_A's public key P_A by checking if $h(R'_{A.\mathrm{x}} \| B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| t_A)$ equals to the received z_i. If it holds, the low-power node N_i can further derive the group secret key $k'_G = h(R'_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$ and verify the validity of the group secret key by checking if $h(k'_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$ equals to the received m. If it holds, k'_G is the group secret key shared among the powerful node and all participating low-power nodes.

3.4 Node Leaving Phase

When a low-power node N_i wants to leave the group, the remaining nodes must update the group secret key for ensuring the confidentiality of the future communications. The procedure of the group secret key updating is described below (as depicted in Fig. 2).

Step 1. The powerful node N_A computes $R'_A = r'_A G$, $z'_i = h(R'_{A.\mathrm{x}} \| B'_{i.\mathrm{x}} \| C'_{i.\mathrm{x}} \| t'_A)$, $Y'_i = r'_A R_i$, $k'_G = h(R'_{A.\mathrm{x}} \| z'_1 \| z'_2 \| ... \| z'_{j-1} \| z'_{j+1} \| ... \| z'_n \| t'_A)$, and $m' = h(k'_G \| I_A \| I_1 \| I_2 \| ... \| I_{j-1} \| I_{j+1} \| ... \| I_n)$, where t'_A is the current timestamp of N_A, $r'_A \in_R Z_q$, and $i = 1,2,..., j-1, j+1,...,n$. The node N_A then broadcasts $\{I_A, P_A, t'_A, m', (Z'_i, Y'_i, I_i); i = 1,2,..., j-1, j+1,...,n\}$ to the remaining low-power nodes N_i's, where $N_i \in N \setminus N_j$. Note that k'_G is the shared group secret key.

Step 2. On receiving $\{I_A, P_A, t'_A, m', (z'_i, Y'_i, I_i); i = 1,2,..., j-1, j+1,...,n\}$ from the powerful node N_A, each low-power node $N_i \in N \setminus N_j$ checks whether $t''_i - t'_A \le \Delta t''$, where t''_i is the timestamp of receiving $\{I_A, P_A, t'_A, m', (z'_i, Y'_i, I_i); i = 1,2,..., j-1, j+1,...,n\}$ and $\Delta t''$ is the preset acceptable delay threshold. If it holds, the low-power node $N_i \in N \setminus N_j$ computes $R''_A = r_i^{-1} Y'_i$ and verify the legitimacy of N_A's identity and the authenticity of N_A's public key P_A by checking if $h(R''_{A.\mathrm{x}} \| B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| t'_A)$ equals to the received z'_i. If it holds, the low-power node N_i can further derive the group secret key $k''_G = h(R''_{A.\mathrm{x}} \| z'_1 \| z'_2 \| ... \| z'_{j-1} \| z'_{j+1} \| ... \| z'_n \| t'_A)$ and verify the validity of the group secret key by checking if $h(k''_G \| I_A \| I_1 \| I_2 \| ... \| I_{j-1} \| I_{j+1} \| ... \| I_n)$ equals to the received m'. If it holds, k''_G is the group secret key shared among the powerful node and all participating low-power nodes.

3.5 Node Joining Phase

When a low-power node N_{n+1} wants to join the group in progress, he needs to obtain the group secret key. All participant nodes and the new node N_{n+1} cooperates with each to perform the following steps (see also Fig. 3) to generate a new group secret key shared among them.

Step 1. The N_{n+1} chooses $r_{n+1} \in_R Z_q$ to compute r_{n+1}^{-1}, $R_{n+1} = r_{n+1}G$, $B_{n+1} = x_{n+1}(P_A + (P_{A.\mathrm{x}} + I_A)P_{SA})$, $C_{n+1} = r_{n+1}(P_A + (P_{A.\mathrm{x}} + I_A)P_{SA})$, and $a_{n+1} = h(B_{(n+1).\mathrm{x}} \| C_{(n+1).\mathrm{x}} \| I_A \| I_{n+1} \| t_{n+1})$, where t_{n+1} is the current timestamp of low-power node N_{n+1}. Finally, N_{n+1} sends $\{I_{n+1}, P_{n+1}, R_{n+1}, t_{n+1}, a_{n+1}\}$ to N_A.

Step 2. The powerful node N_A verifies legitimacy of N_{n+1}'s identity and generates group secret key by performing the following sub-steps.

Step 2-1. On receiving $\{I_{n+1}, P_{n+1}, R_{n+1}, t_{n+1}, a_{n+1}\}$ from N_{n+1}, the powerful node N_A checks whether $t'_{n+1} - t_{n+1} \le \Delta t$, where t'_{n+1} is the timestamp of receiving $\{I_{n+1}, P_{n+1}, R_{n+1}, t_{n+1}, a_{n+1}\}$ and Δt is the preset acceptable delay threshold. If it holds, the powerful node N_A computes $B'_{n+1} = x_A(P_{n+1} + (P_{(n+1).\mathrm{x}} + I_{n+1})P_{SA})$ and $C'_{n+1} = x_A R_{n+1}$, and checks if $h(B'_{(n+1).\mathrm{x}} \| C'_{(n+1).\mathrm{x}} \| I_A \| I_{n+1} \| t_{n+1})$ equals to the received a_{n+1} for verifying the legitimacy of the low-power node N_{n+1} by. If it does not hold, N_A requests N_{n+1} to re-send valid $\{I_{n+1}, P_{n+1}, R_{n+1}, t_{n+1}, a_{n+1}\}$. Otherwise, both of the identity authentication and the authenticity public key P_{n+1} for N_{n+1} are verified.

Step 2-2. The powerful node N_A computes $R'_A = r'_A G$, $z'_i = h(R'_{A.\mathrm{x}} \| B'_{i.\mathrm{x}} \| C'_{i.\mathrm{x}} \| t'_A)$, $k'_G = h(R'_{A.\mathrm{x}} \| z'_1 \| z'_2 \| ... \| z'_{n+1} \| t'_A)$, $m' = h(k'_G \| I_A \| I_1 \| I_2 \| ... \| I_{n+1})$, and $Y'_i = r'_A R_i$, where t'_A is the current timestamp of N_A, $r'_A \in_R Z_q$, and $i = 1,2,...,n,n+1$. Furthermore, N_A broadcasts $\{I_A, P_A, t'_A, m', (z'_i, Y'_i, I_i); i = 1,2,...,n,n+1\}$ to all low-power nodes. Note that k'_G is the shared group secret key.

Step 3. On receiving $\{I_A, P_A, t'_A, m', (z'_i, Y'_i, I_i); i = 1,2,...,n,n+1\}$ from the powerful node N_A, each low-power node $N_i \in N \cup N_{n+1}$ checks whether $t''_i - t'_A \le \Delta t''$, where t''_i is the timestamp of receiving $\{I_A, P_A, t'_A, m', (z'_i, Y'_i, I_i); i = 1,2,...,n,n+1\}$ and $\Delta t''$ is the preset acceptable delay threshold. If it holds, the low-power node $N_i \in N \cup N_{n+1}$ computes $R''_A = r_i^{-1} Y'_i$ and verify the legitimacy of N_A's identity and the authenticity of N_A's public key P_A by checking if $h(R''_{A.\mathrm{x}} \| B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| t'_A)$ equals to the received z'_i. If it holds, the low-power node N_i can further derive the group secret key $k''_G = h(R''_{A.\mathrm{x}} \| z'_1 \| z'_2 \| ... \| z'_n \| z'_{n+1} \| t'_A)$ and verify the validity of the group secret key by checking if $h(k''_G \| I_A \| I_1 \| I_2 \| ... \| I_n \| I_{n+1})$ equals to the received m'. If it holds, k''_G is the group secret key shared among the powerful node and all participating low-power nodes.

4 Security Analysis and Comparisons

In this section, we will discuss some security considerations of the proposed *c*AGKA protocol. The security of the proposed protocol is based on the assumptions of the elliptic curve discrete logarithm (ECDL) problem [17-19] and the one-way hash function (OHF) [12, 20]. Security considerations are analyzed from the following six perspectives: confidentiality of private keys, entity authentication, authenticity of public keys, confidentiality and confirmation of the established group secret key, group key contribution, and forward secrecy.

4.1 Confidentiality of Private Keys

Consider the scenario of a compromising attack that an adversary or a malicious registered node attempts to derive SA's private key s_{SA}. With the knowledge of SA's public key $P_{SA} = s_{SA}G$, the adversary will face the ECDL problem to derive s_{SA}. A malicious registered node can successfully

compromise SA's private key s_{SA} with the SA's sending $s_i = h(r_i \| I_i) + (P_{i.\mathrm{x}} + I_i) \cdot s_{SA} \bmod q$ only if he can derive r_i first. However, it is computationally infeasible to derive r_i from $P_i = V_i + h(r_i \| I_i)G$ under the ECDL and the OHF assumptions.

Similarly, consider the scenario of a compromising attack that a malicious adversary (including any registered node) attempts to derive node's private key x_i. Since the node's private key x_i is computed by $x_i = s_i + v_i \bmod q$, the adversary will face the ECDL assumption to derive v_i from $V_i = v_i G$. The private key satisfies the verification $x_i G = P_i + (P_{i.\mathrm{x}} + I_i)P_{SA}$. With knowledge of $\{P_i, I_i, P_{SA}\}$, it is computationally infeasible to derive x_i under the ECDL problem.

Consider the scenario of an attack that an adversary attempts to derive nodes' private keys x_i by the intercepted messages $\{I_i, P_i, R_i, t_i, a_i\}$'s and $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2,...,n\}$. From the equations $B_i = x_i(P_A + (P_{A.\mathrm{x}} + I_A)P_{SA})$ and $a_i = h(B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| I_A \| I_i \| t_i)$, the adversary will face the ECDL and OHF assumptions to compromise the private key x_i. Similarly, the adversary cannot derive the private key x_A from $z_i = h(R_{A.\mathrm{x}} \| B'_{i.\mathrm{x}} \| C'_{i.\mathrm{x}} \| t_A)$, $B'_i = x_A(P_i + (P_{i.\mathrm{x}} + I_i)P_{SA})$, and $C'_i = x_A R_i$.

4.2 Entity Authentication

The proposed *c*AGKA protocol provides mutual authentication for verifying the legitimacies of the powerful node and the low-power nodes with each other. To authenticate the legitimacy of the participating low-power node N_i, the powerful node can check if $h(B'_{i.\mathrm{x}} \| C'_{i.\mathrm{x}} \| I_A \| I_i \| t_i)$ equals to the received a_i, where $B'_i = x_A(P_i + (P_{i.\mathrm{x}} + I_i)P_{SA})$ and $C'_i = x_A R_i$. Since $B_i = x_i(P_A + (P_{A.\mathrm{x}} + I_A)P_{SA}) = x_A(P_i + (P_{i.\mathrm{x}} + I_i)P_{SA}) = B'_i$ and $C_i = r_i(P_A + (P_{A.\mathrm{x}} + I_A)P_{SA}) = x_A R_i = C'_i$, the adversary can successfully generate a valid a_i for cheating the powerful node only if he knows x_i or x_A. Security of the private keys is based on the ECDL and the OHF assumptions as analyzed above. Since the timestamp t_i is included in a_i, the adversary cannot replay the intercepted messages to masquerade as a valid low-power node. The proposed protocol can adopt the existing time synchronization schemes [21,22] to achieve a synchronization objective. This also implies the proposed *c*AGKA protocol can withstand the impersonation attacks.

On the other hand, each low-power node N_i can authenticate the legitimacy of the powerful node by checking $h(R'_{A.\mathrm{x}} \| B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| t_A)$ equals to the received z_i, where $R'_A = r_i^{-1} Y_i$. The adversary can successfully masquerade as the powerful-node for cheating any low-power node N_i if he can correctly derive B_i, C_i, and R'_A. Security of B_i and C_i is protected under the ECDL and the OHF assumptions as discussed above. It can be seen that the security of $R'_A = r_i^{-1} Y_i$ is also protected based on the ECDL assumption since the adversary will face the ECDL assumption to derive r_i from R_i and then use r_i to compute R'_A.

4.3 Authenticity of Public Keys

Seeing that a valid public key P_i with respect to x_i and I_i has to satisfy the check of the verification equality, $x_i G = P_i + (P_{i.\mathrm{x}} + I_i)P_{SA}$, a malicious adversary N_{adv} (including any registered node) may attempt to forge a valid pair $(I_{adv}, x_{adv}, P_{adv})$ to satisfy this verification equality. Consider the scenario that an adversary N_{adv} attempts to choose an identity information I_{adv} and try to generate a valid certificateless private and public key pair (x_{adv}, P_{adv}) without the assistant of SA. The adversary can first arbitrarily choose his identifier I_{adv} and private key x_{adv}, and then tries to compute the corresponding public key P_{adv} such that $P_{adv} + P_{adv.\mathrm{x}} P_{SA} = x_{adv} G - I_{adv} P_{SA}$. It can be seen that the adversary will face the intractability of the ECDL problem to derive $P_{adv.\mathrm{x}}$ and P_{adv} from this equation. Similarly, the adversary might first determine (I_{adv}, P_{adv}), and then try to derive x_{adv} to satisfy above verification equality. It is obvious to see that x_{adv} is protected under the ECDL assumption. What is more, to generate a valid I_{adv} with the arbitrarily chosen x_{adv} and P_{adv}, the

adversary will be confronted with the difficulty of the ECDL assumption.

4.4 Confidentiality and Confirmation of the Established Group Secret Key

In the proposed *c*AGKA protocol, the group secret key is generated by $k_G = h(R_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$. Only one *secret* variable $R_{A.\mathrm{x}}$ is contributed to key generation. The adversary can successfully compromise R_A for deriving k_G only if he knows r_i or r_A due to $R_A = r_i^{-1} Y_i = r_i^{-1}(r_A R_i) = r_i^{-1}(r_A r_i G) = r_A G$. Compromising r_i from R_i or r_A from Y_i is an ECDL problem. On the other hand, if the adversary attempts to derive k_G from the intercepted message $m = h(k_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$, he will face the intractability of reversing the one-way hash function (i.e. OHF problem). Hence, the confidentiality of the group secret key is protected under the ECDL or OHF assumptions.

In addition, the proposed *c*AGKA provides explicit key authentication (also called key confirmation) in such a way that all participating low-power nodes can explicitly verify the authenticity of the established group secret key. It can see that the message $m = h(k_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$ can be regarded as an authenticator for this purpose. If the group secret key is not correctly computed by $k_G = h(R_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$, it will fail to the verification of *m*.

4.5 Group Key Contribution

We will show that the proposed *c*AGKA protocol is a contributory key agreement one which allows every participating low-power nodes to contribute their shares to the group key generation. It can be seen that the group secret key is computed by $k_G = h(R_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$, where $z_i = h(R_{A.\mathrm{x}} \| B'_{i.\mathrm{x}} \| C'_{i.\mathrm{x}} \| t_A)$ and $R_A = r_i^{-1} Y_i$. The secret random number r_i is secretly determined by a low-power node N_i, and hence contributed to the group key generation. This means that each low-power node equally contributes to the group secret key and guarantees its freshness in each group secret key construction, that is to say, no participant node can predetermine the group secret key. Hence, the proposed protocol is a contributory group key agreement one.

4.6 Forward Secrecy

The forward secrecy guarantees that an adversary who compromises a private key(s) or one group secret key must not reveal previously established group secret keys. For example, when a low-power node wants to join a group, forward secrecy must be achieved to prevent the new member from accessing the previous group communications. As mentioned of the proposed *c*AGKA protocol, the group secret key is $k_G = h(R_{A.\mathrm{x}} \| z_1 \| z_2 \| ... \| z_n \| t_A)$, where $z_i = h(R_{A.\mathrm{x}} \| B_{i.\mathrm{x}} \| C_{i.\mathrm{x}} \| t_A)$ and $R_A = r_i^{-1} Y_i$. Compromising the private key x_i (or x_A) only help to derive B_i and C_i. The group secret key is still protected by the secret R_A. It is easy to see that compromising r_i from R_i or r_A from Y_i is an ECDL problem. Hence, the adversary cannot derive any one group secret key with the compromised private keys.

Consider the scenario that the adversary compromised one group secret key attempts to derive any one previously established group secret key. Since the proposed protocol is a contributory one as mentioned above, the group secret key for distinct session will be refreshed by the random secret values. The group secret keys can be regarded as a random number generated by all participating nodes. Hence, the adversary knowing one group secret key cannot derive previously established one, which implies the forward secrecy is achieved.

4.7 Comparisons of Security Properties

We compare the necessary security properties of the proposed scheme and those of Bresson *et al.*'s [4] and Tseng's protocols [8] in Table 2. From Table 2, we can see that the proposed *c*AGKA protocol is a certificateless contributory key agreement one, while the other two protocols are certificate-based ones. The proposed protocol will have the merits of the certificateless public keys. Bresson *et al.*'s and Tseng's protocols [4, 8] are all insecure against the impersonation attack, since their transmitted messages can be replayed by the adversary. Hence, they cannot achieve the mutual authentication. As we analyzed above, the proposed protocol are secure against the impersonation attack and achieves the mutual authentication. Considering the security of the established group secret key, the proposed protocol and Tseng's protocol can achieve contributory group key agreement, forward secrecy, and key confirmation, while Bresson *et al.*'s

protocol cannot. Key confirmation of Tseng's protocol can be implicitly achieved by checking the correctness of all variables contributed to group key generation, while that of the proposed protocol are explicitly achieved by a key authenticator. We also considered and proposed the group key updating mechanisms for node leaving or joining in our proposed protocol. Moreover, the underlying cryptographic assumption of the proposed protocol is elliptic curve discrete logarithm problem, while that of Bresson *et al.*'s [4] and Tseng's protocols is discrete logarithm (DL) problem. The proposed protocol is hence more secure than the other two protocols under the same key size.

5 Performance Evaluations and Comparisons

In this section, we will evaluate the performance of our proposed *c*AGKA protocol in terms of the computational complexities and the communication overheads. For convenience, we first define the following notations:

- $T_{EM/EA}$: the time for computing a point multiplication/addition operation over an elliptic curve;
- $T_{MM/EXP/INV}$: the time for computing a modular multiplication/exponentiation/inversion;
- T_H: the time for computing a secure one-way hash function h;
- $T_{SIG/VER}$: the time for generating/verifying a signature;
- n: the number of low-power nodes that want to agree on a group secret key shared among them;
- $|a|$: the bit-length of a variable a.

Note that the time for computing a modular addition and that for XOR function are ignored here for that they are negligible as compared to the other complexities measures. From [23-26], the time complexities can be respectively regarded as $T_{EM} \approx 29T_{MM}$, $T_{EA} \approx 0.12T_{MM}$, $T_{EXP} \approx 240T_{MM}$, $T_{INV} \approx 10T_{MM}$, and $T_H \approx 4T_{MM}$.

First, we discuss the computational complexities of low-power nodes in our proposed *c*AGKA protocol. In Step 1, each low-power node N_i computes r_i^{-1}, $R_i = r_i G$, $B_i = x_i(P_A + (P_{A.x} + I_A)P_{SA})$, $C_i = r_i(P_A + (P_{A.x} + I_A)P_{SA})$, and $a_i = h(B_{i.x} \| C_{i.x} \| I_A \| I_i \| t_i)$. The computational complexities for the step 1 are $4T_{EM} + T_{EA} + T_H + T_{INV}$. In Step 3, the low-power node N_i computes $R'_A = r_i^{-1} Y_i$ and checks whether $z_i = h(R'_{A.x} \| B_{i.x} \| C_{i.x} \| t_A)$ holds or not. Further, the low-power node N_i computes $k'_G = h(R'_{A.x} \| z_1 \| z_2 \| ... \| z_n \| t_A)$ and checks whether $m = h(k'_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$ holds or not. The Step 3 requires $T_{EM} + 3T_H$. Therefore, the computational complexities for each low-power node are $5T_{EM} + T_{EA} + 4T_H + T_{INV}$.

In the following, we consider the computational complexities of the powerful node in our proposed *c*AGKA protocol. In Step 2, the powerful node computes $B'_i = x_A(P_i + (P_{i.x} + I_i)P_{SA})$, $C'_i = x_A R_i$, and checks whether $a_i = h(B'_{i.x} \| C'_{i.x} \| I_A \| I_i \| t_i)$ holds or not. If it pass verifications, the powerful node N_A computes $R_A = r_A G$, $Y_i = r_A R_i$, $z_i = h(R_{A.x} \| B'_{i.x} \| C'_{i.x} \| t_A)$, $k_G = h(R_{A.x} \| z_1 \| z_2 \| ... \| z_n \| t_A)$, and $m = h(k_G \| I_A \| I_1 \| I_2 \| ... \| I_n)$. Thus, computational complexities for the powerful node are $(4n+1)T_{EM} + nT_{EA} + 2(n+1)T_H$.

Comparisons of the computational complexities among the proposed *c*AGKA, Bresson *et al.*'s, and Tseng's protocols are given in Table 3. In Bresson *et al.*'s and Tseng's protocols, all public keys are certificate-based ones. This means that the authenticity of all public keys will be verified by checking the validity of extra public key certificates issued by a certification authority CA. The proposed protocol uses certificateless public keys and hence gains performance efficiency in computational complexities due to no certificate verification. For simplicity, we assume that the public key certificates are implemented by ElGamal signature scheme [15] in Bresson *et al.*'s and Tseng's protocols. The digital signature used in both protocols is also assumed to be implemented by ElGamal signature [15]. From Table 3, it can see that the computational complexities for each low-power node are independent on the number of the low-power nodes in Bresson *et al.*'s and the proposed protocols, but not in Tseng's protocol. Computational complexities for the powerful node in these three protocols are all dependent on the number of the low-power nodes, but the proposed protocol requires lower computational complexities. In summary, the proposed *c*AGKA protocol is more efficient than Bresson *et al.*'s, and Tseng's protocols in computational complexities.

Considering the communication overheads in the three protocols, we let the adopted one-way hash function h be SHA-1 [27] (the bit length of the

output is 160 bits), $|p'| = 1024$ bits, $|q'| = 160$ bits, $|p| = |q| = 163$ bits respectively. For simplicity, the counter c (used in Bresson *et al.*'s protocol [4]), the timestamp t (used in the proposed protocol), and the identity I are all assumed to be 160 bits.

In our proposed *c*AGKA protocol, each low-power node N_i sends $\{I_i, P_i, R_i, t_i, a_i\}$ to the powerful node N_A via uni-cast communication. Thus, the communication overheads sent by each low-power node are $|I| + 4|p| + |h| + |t|$. In Step 2, the powerful node N_A broadcasts $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2,...,n\}$ to low-power nodes. The communication overheads sent by the powerful node are $(n+1)|I| + 2(n+1)|p| + (n+1)|h| + |t|$, where $|h|$ is the bit-length of the adopted hash function. Table 4 shows comparisons of the communication overheads. Since Bresson *et al.*'s and Tseng's protocols are certificate-based ones, they require 2048 bits (i.e., $2|p'|$ bits) for transmitting an extra public key certificate which is assumed to be implemented by ElGamal signature scheme [15]. As seen from Table 4, the communication overheads sent by each low-power node in Bresson *et al.*'s and Tseng's protocols are larger than those in the proposed *c*AGKA protocol. The communication overheads sent by the powerful node in the proposed protocol are larger than those in Bresson *et al.*'s protocol, since it requires extra communication overheads for achieving contributory group key agreement, key confirmation, and mutual authentication. If the proposed *c*AGKA provides the same security requirements as mentioned in Bresson *et al.*'s protocol [4], the communication overheads of the powerful node are only $(326n + 486)$ bits (i.e., the powerful node broadcasts $\{(P_A, t_A, Y_i); i = 1,2,...,n\}$ instead of $\{I_A, P_A, t_A, m, (z_i, Y_i, I_i); i = 1,2,...,n\}$. All the protocols require the same number of rounds in the group key agreement. The Tseng's protocol does not consider and propose the group key updating mechanisms for node leaving or joining process. For the joining and leaving process of the proposed *c*AGKA and Bresson *et al.*'s protocols, the communication overhead depends on how many mobile nodes join/leave in a given time period.

6 Conclusion

Elaborating on merits of the certificateless public keys and elliptic curve cryptography, the proposed *c*AGKA protocol is more efficient and secure than previously proposed protocols for unbalanced wireless mobile networks. Authenticity of public keys is verified together with entity authentication without any public key certificates. We showed that the proposed *c*AGKA protocol is a real contributory group key agreement and provides mutual authentication between low-power nodes and the powerful node. Based on the intractability of solving the ECDL and the OHF problems, our proposed *c*AGKA protocol is secure against some potential active and passive attacks. Moreover, it can gain much efficiency in saving of computational complexities and communication overheads.

Acknowledgment

We would like to thank anonymous referees for their valuable suggestions. This work was supported in part by the Ministry of Economic Affairs (R.O.C.) under grants 96-EC-17-A-19-S1-055, in part by National Science Council under the grants NSC 96-2219-E-001-001, NSC 96-2219-E-011-008, and NSC 97-2221-E-182-032.

References:

[1] C. Toma, M. Popa and C. Boja, Smart card based solution for non-repudiation in GSM WAP applications. *WSEAS Transactions on Computers*, Vol. 7, No. 5, 2008, pp. 453–462.

[2] M. Jurian, I. Lita and D. A. Visan, Efficient mobile communication solutions for remote data acquisition, supervisory and control systems. *WSEAS Transactions on Communications*, Vol. 7, No. 7, 2008, pp. 739–748.

[3] Tie-Jun Pan, Lei-na Zheng, Hua-jun Zhang and Chen-bin Fang, Research of utility prepayment system based on wireless communication. *WSEAS Transactions on Communications*, Vol. 8, No. 1, 2009, pp. 71–80.

[4] E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval, Multual authentication and group key agreement for low-power mobile devices. *Computer Communications*, Vol.27, 2004, pp. 1730–1737.

[5] J. Nam, S. Kim and D. Won, A weakness in the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme for low-power mobile devices. *IEEE Communications Letters*, Vol.9, 2005, pp. 429-431.

[6] J. Nam, S. Kim and D. Won, DDH-based group key agreement in a mobile environment. *J. Syst. Software*, Vol.78, 2005, pp. 73–83.

[7] Y. M. Tseng, On the security of two group key agreement protocols for mobile devices. *In Int Workshop on Future Mobile and Ubiquitous Information Technologies (FMUIT2006)*, Nara, Japan, May 9-12, 2006, pp.59-62. IEEE Computer Society, Washington, DC, USA.

[8] Y.M. Tseng, A secure authenticated group key agreement protocol for resource-limited mobile devices. *The Computer Journal*, Vol.50, 2007, pp. 41–52.

[9] V. Miller, Use of elliptic curves in cryptography. *Advances in Cryptology-CRYPTO'85*, Santa Barbara, California, USA, August 18-22, 1985, pp. 417–426. Springer-Verlag, Berlin.

[10] N. Koblitz, Elliptic curve cryptosystems. *Mathematics of Computation*, Vol.48, 1987, pp. 203–209.

[11] RFC 4992, *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*. TLS Working Group, Internet Engineering Task Force (IETF), 2006.

[12] A. Menezes, P.C. van Oorschot and S. Vanstone, Handbook of Applied Cryptography. *CRC Press*, Boca Raton, 1997.

[13] M. Girault, Self-certified public keys. *Proceedings of Advances in Cryptology - EuroCrypt '91*, Brighton, UK, 8-11 April, pp.490-497, Springer-Verlag, Berlin, 1991.

[14] NIST FIPS 186-2, *Digital Signature Standard (DSS)*. NIST, Gaithersburg, MD, USA, 2000.

[15] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31, 1985, pp. 469–472.

[16] A. Shamir and Y. Tauman, Improved on-line/off-line signature schemes. *Proc. Advances in Cryptology - Crypto'01, LNCS*, 2139, 2001, pp. 355–367.

[17] IEEE P1363, *Standard Specifications for Public Key Cryptography*. The Institute of Electrical and Electronics Engineers, Inc., 2000.

[18] A. Menezes, Elliptic curve public key cryptosystems. *Kluwer Academic Publishers*, Boston, MA., 1993.

[19] I. Blake, G. Seroussi and N. Smart, Elliptic curves in cryptography. *Cambridge University Press*, Cambridge, UK., 1999.

[20] W. Diffie and M. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22, 1976, pp. 44–654.

[21] K. Romer, Time Synchronization in Ad Hoc Networks. *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01)*, Long Beach, CA, USA, 4-5 October, 2001, pp. 173–182.

[22] F. Scuglik, Time Synchronization Possibilities in Wireless networks for Embedded Systems. *WSEAS Transactions on Communications*, 2005, Vol. 4; No. 11, pp. 1215–1218

[23] N. Koblitz, A. Menezes and S. Vanstone, The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, Vol.19, 2000, pp. 173–193.

[24] T.S. Chen, E.T. Hsu and Y.L. Yu, A New Elliptic Curve Undeniable Signature Scheme. *International mathematical forum*, Vol.1, 2006, pp. 1529–1536.

[25] D. Hankerson, J. L. Hernandez and A. Menezes, Software Implementation of Elliptic Curve Cryptography over Binary Fields. *Proceedings of CHES '00*, Worcester, MA, USA, 17-18 August, 2000, pp. 1–24. Springer-Verlag, New York.

[26] S.Contini, A.K. Lenstra and R. Steinfeld, VSH, an Efficient and Provable Collision-Resistant Hash Function. *Advances in Cryptology – EUROCRYPT 2006*, Saint Petersburg, Russia, 28 May - 1 June, 2006, pp. 165–182. Springer-Verlag, Berlin.

[27] NIST FIPS 180-3, *Secure Hash Standard (SHS)*. NIST, Gaithersburg, MD, USA., 2007.

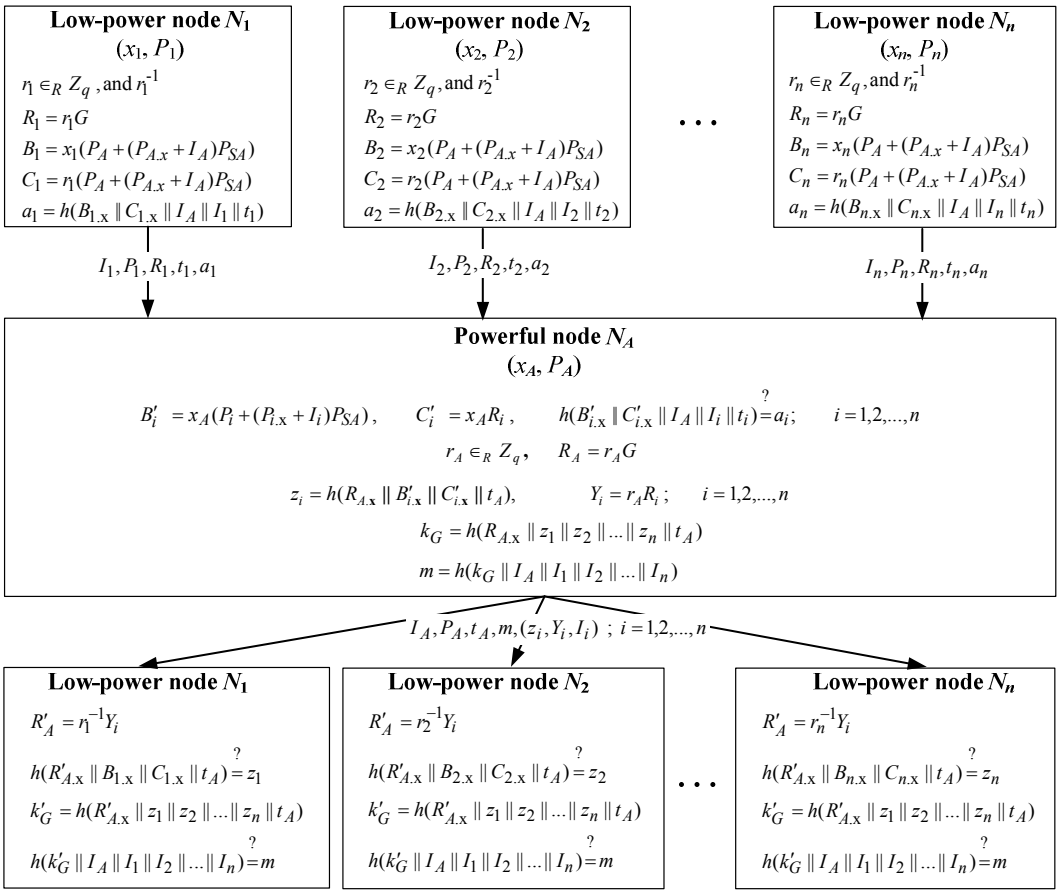

Fig. 1. The proposed authenticated group key agreement protocol

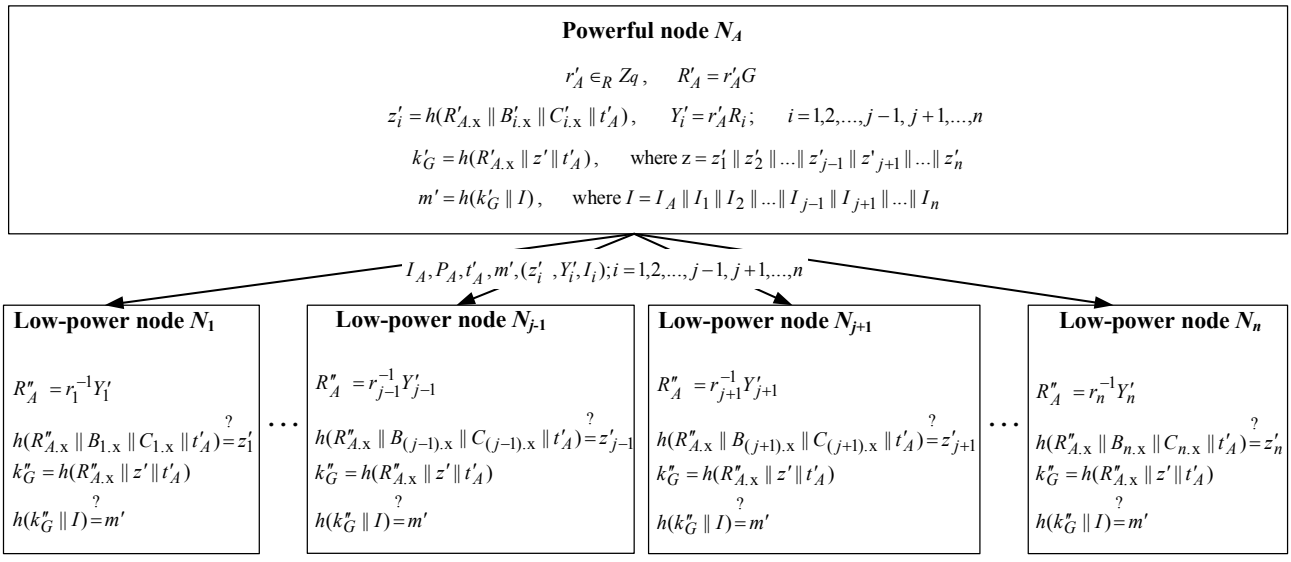

Fig. 2. Node leaving phase

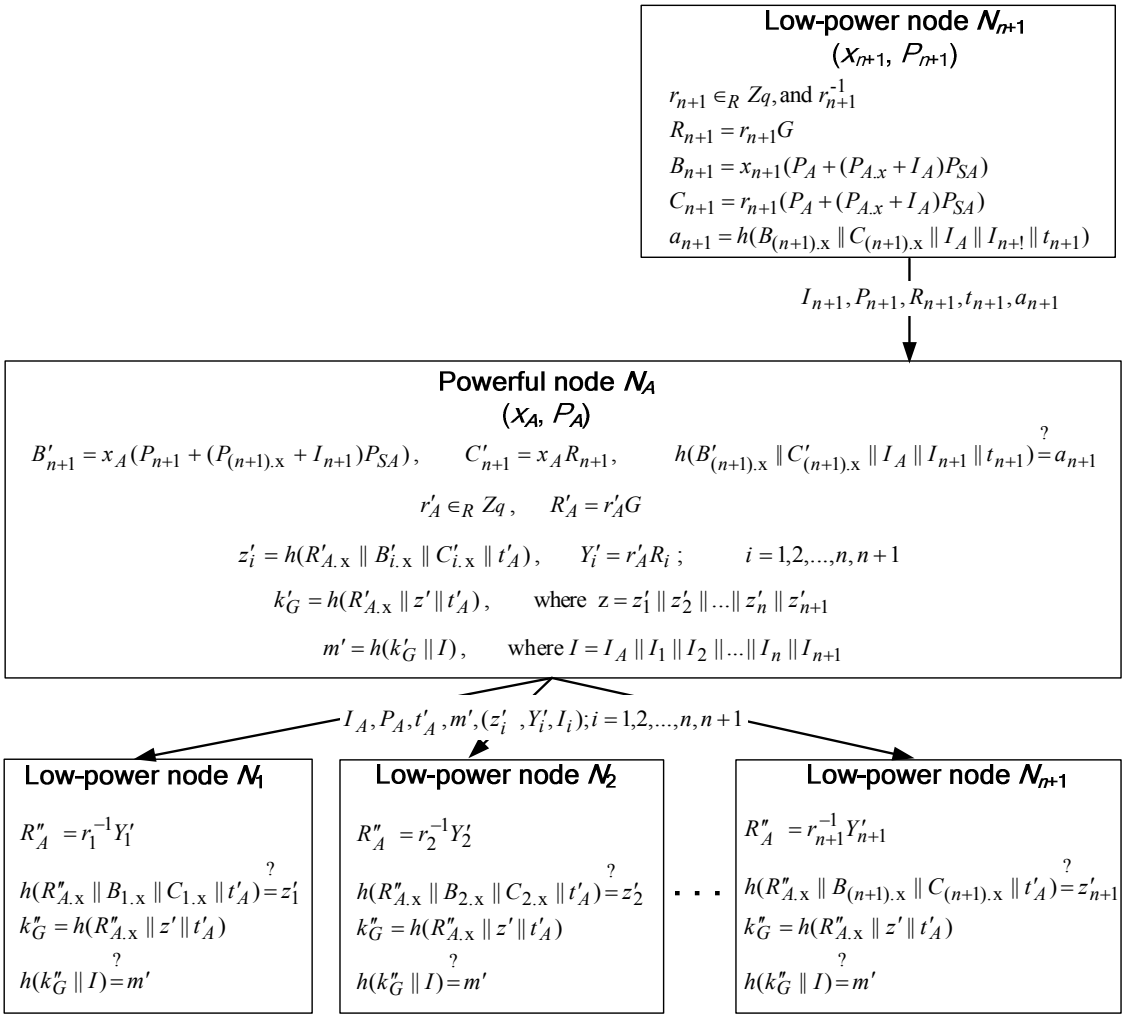

Fig. 3. Node joining phase

Table 2. Comparisons of security properties

	Bresson *et al.*'s protocol [4]	Tseng's protocol [8]	The proposed *c*AGKA protocol
Public keys	Certificate-based	Certificate-based	Certificateless
Mutual authentication	No	No	Yes
Impersonation attack resistance	No	No	Yes
Contributory group key agreement	No	Yes	Yes
Forward secrecy	No	Yes	Yes
Key confirmation	No	Implicit	Explicit
Group key updating (when member joins or leaves)	Yes	No	Yes
Underlying cryptographic assumption	DLP	DLP	ECDLP

Table 3. Comparisons of computational complexities

		Bresson *et al.*'s protocol [4]	Tseng's protocol [8]	The proposed *c*AGKA protocol
Each low-power node	Time complexities	$2T_{EXP} + 2T_H + T_{VER}$ ‡ $+ T_{SIG}$ *	$3T_{EXP} + nT_{MM} + T_H + T_{INV} + T_{VER}$ ‡ $+ T_{SIG}$ *	$5T_{EM} + T_{EA} + 4T_H + T_{INV}$
	Rough Estimation	$\approx 1461T_{MM}$	$\approx (n + 1707)\ T_{MM}$	$\approx 162.12T_{MM}$
powerful node	Time complexities	$nT_{EXP} + (n+1)T_H + 2nT_{VER}$ ‡	$(2n+1)T_{EXP} + nT_{MM} + T_H + 2nT_{VER}$ ‡	$(4n+1)T_{EM} + nT_{EA} + 2(n+1)T_H$
	Rough Estimation	$\approx (1686n+4)T_{MM}$	$\approx (1923n+244)T_{MM}$	$\approx (124.12n+37)T_{MM}$

Remark *: The computational complexity for generating a signature is $T_{SIG} = T_{EXP} + 2T_{MM} + T_{INV} + T_{MA} \approx 252\ T_{MM}$ according to ElGamal signature scheme [15].

‡: The computational complexity for verifying a signature/public key certificate is $T_{VER} = 3T_{EXP} + T_{MM} \approx 721\ T_{MM}$ according to ElGamal signature scheme [15].

Table 4. Comparisons of communication overheads

	Bresson *et al.*'s protocol [4]	Tseng's protocol [8]	The proposed *c*AGKA protocol
Communication overheads sent by each low-power node	$\|I\| + 5\|p'\|$ ‡ **≈ 5280 bits**	$\|I\| + 5\|p'\|$ ‡ **≈ 5280 bits**	$\|I\| + 4\|p\| + \|h\| + \|t\|$ **≈ 1132 bits**
Communication overheads sent by the powerful node	$(n+1)\|I\| + \|c\| + n\|h\| + 2\|p'\|$ ‡ **≈ 320*n* + 2368 bits**	$(n+1)\|I\| + \|h\| + 2(n+1)\|p'\|$ ‡ **≈ 2208*n* + 2368 bits**	$(n+1)\|I\| + 2(n+1)\|p\| + (n+1)\|h\| + \|t\|$ **≈ 646*n* + 806 bits**
Number of rounds	2	2	2

Remark ‡: The costs for transmitting each public key certificate/signature of the Bresson *et al.*'s protocol and the Tseng's protocol are assumed to be implemented with ElGamal signature [15], i.e., $2|p'| = 2048$ bits.