A Node Revocation Scheme Using Public-Key Cryptography in Wireless Sensor Networks

PO-JEN CHUANG, SHAO-HSUAN CHANG AND CHIH-SHIN LIN
Department of Electrical Engineering
TamKang University
Tamsui, Taipei County, 251 Taiwan
E-mail: pjchuang@ee.tku.edu.tw

Generally deployed in an unattended environment, a sensor network can be easily assaulted or compromised by adversaries. Network security becomes a major problem. A distributed node revocation scheme is effective in reducing the damages a compromised node may cause to a sensor network, but its operation tends to consume large-scale memory space of the hardware-constrained sensor nodes. To reduce such complexity, this paper presents a new distributed voting revocation scheme based on the one-way hash chain, the certificate revocation list and the public-key cryptography. Performance evaluation shows that our scheme outperforms the other target schemes in enhancing network security at reasonable calculation cost which is acceptable to the sensor nodes.

Keywords: wireless sensor networks, network security, distributed node revocation schemes, one-way hash chains, certificate revocation lists, public-key cryptography, performance evaluation

1. INTRODUCTION

Energy efficiency is a basic concern for a wireless sensor network (WSN), whose sensor nodes are battery-powered and are difficult to get recharged after distribution. Network security is also a major problem – because usually deployed in unattended environments to collect the needed information, a WSN can be easily assaulted by adversaries. A number of security enhancing devices, including key management [e.g., 1-5], have been established for WSNs. Key management involves key distribution and revocation [6]. Key distribution disseminates secret information to the principals to initialize secure communications. Key revocation, by contrast, aims to remove secrets that attackers may have snapped from a compromised node and will result in revocation of such a compromised node. Node revocation is critical in reducing the damage (e.g., information leakage) a compromised node may cause to a sensor network. A node revocation scheme, designed mainly to cut off all links between compromised nodes and normal nodes, can work in a centralized way (such as the EG scheme [6] in which the base station (BS) is responsible

Received September 15, 2008; revised January 20 & May 6 & July 27, 2009; accepted September 3, 2009.
Communicated by Chin-Laung Lei.
* This work was supported in part by the National Science Council, Taiwan, R.O.C., under Grant No. NSC 97-2221-E-032-018 and was presented at the International Conference on Future Generation Communication and Networking, Jeju Island, Korea, Dec. 2007.
for conducting revocation decisions) or in a distributed way (such as the CPS, CGPM and CC schemes [7-9]). A distributed scheme can react more quickly – thus reducing delay time and communication cost – as the compromised node’s neighbor nodes will initiate and broadcast revocation decisions without the assistance of the BS.

To enhance the communication security of a sensor network at reduced hardware cost, this paper introduces a new distributed node revocation scheme utilizing the features of the one-way hash chain, the certificate revocation list (CRL), the public-key cryptography (PKC) [10-12], and some new designs. Our scheme adopts the concept of CRL but transforms it from centralized revocation to distributed voting. Later discussions will show that our new distributed approach can revoke compromised nodes more instantly than traditional CRL. The public-key cryptography seems unsuitable for a hardware-constrained wireless environment. Advanced technology nevertheless has made such application possible and even promising. In our scheme, the BS will sign a certificate to each node before deployment – each certificate carrying a unique ID that authorizes the node’s access to such services as data authentication or node revocation. When a node detects another node in the network, say node A, has been compromised, it will broadcast a Compromise Revocation Vote (CRV) with node A’s certificate ID to all neighbor nodes. Receiving such a packet, a neighbor node will authenticate its validity by the hash value attached in the CRV. If the validity is authenticated, the neighbor node will add one more revocation vote against A. When the received revocation votes against A exceed a threshold value t, the neighbor node will disconnect its link to A. That is, A becomes a revoked node and is deprived of all connections to the normal nodes to save the network from the adversary’s further attacks.

As performance analysis shows, the PKC distributed voting of our node revocation scheme substantially tightens up the network security of a WSN but incurs high calculation cost. To reduce such cost, we adopt the one-way hash function and XOR operation because the two together manage to keep the required energy consumption at a level sensor nodes can tolerate. Node revocation is indeed a rare case that happens only when the system is under attack. But when it happens, how to maintain the network security becomes an extremely important issue. When node revocation must be performed, the key consideration for us will be to attain a desirable balance between the security gain and the incurred cost. In view of this, we consider our new scheme feasible and practical for hardware-constrained WSNs – because it can effectively resist adversaries’ advanced attacks at reasonable and acceptable calculation cost.

2. BACKGROUND STUDY

The Distributed CGPM Scheme [8] Involving the idea of threshold secret sharing [13], CGPM defines the cryptographic hash of a random polynomial q(x) as \( H(q(x)) = H(a_0 \| a_1 \| a_2 \| \ldots \| a_{t-1}) \), H being a hash function and \( a_0, a_1, \ldots, a_{t-1} \) being the coefficients of q(x). It also adopts the authenticated-encryption (AE) modes to save energy and fit the sensor networks. The stages of a revocation session include the following four states:

- The pending state: For node initialization, CGPM first computes – offline – \( s_{total} \) random polynomials of degree \( t \) for each of the \( n \) nodes in the network. \( s_{total} \) is the number
of revocation sessions against any target node. For each node $B$ of node $A$’s $m$ participants and each revocation session $s$ against target $B$, load the revocation vote from $A$ against $B$ based on the proposed random polynomial $q_{Bs}$. The vote consists of the secret share $\{q_{Bs}(x_{AB}), x_{AB}\}$ and AE encrypted with the activation mask $Mask_{AB}$ that $B$ gives to $A$. The points at which the secret-sharing polynomial is evaluated (e.g., $x_{AB}$) are generated such that no two participants are with the same revocation secret share.

- **The active state**: When $A$ detects its neighbor $B$ has been compromised, it will vote against $B$ by an unencrypted hop-limited broadcast in the current revocation session $s$, $\{q_{Bs}(x_{AB}), x_{AB}\}$ along with the log $m$ Merkle authentication values that can verify this vote. This local neighborhood broadcast allows only nodes which can verify the authenticity of the vote to disseminate the broadcast. If the vote is correct, these nodes will proceed to the next revocation voting step.

- **The completed state**: When $A$’s session state transits to “complete”, it will count the number of votes received in the active state. If $A$ gets at least $t$ revocation votes (including its own, if any), it will compute the revocation polynomial of $B$, $q_{Bs}$, for this session and also the hash of the polynomial, $H(q_{Bs})$. This hashed value is then broadcast through the entire network and all participants of $B$ will verify this preimage against the value stored in their memory, $H^{r}(q_{Bs})$. If the verification succeeds, all shared keys with $B$ are deleted and $B$ is marked as revoked. The broadcast is then disseminated to the other participants of $B$ until the entire network is covered.

- **The non-current state**: If $A$ gets fewer than $t$ verified revocation votes, this revocation session fails. Each local participant notifies the BS of the failed revocation session to ensure that new nodes can be deployed with the most current revocation session in the correct state. Local participants of $B$ then proceed to request the masks for the new session $s + 1$, i.e., $Mask_{AB}(S + 1)$.

### The Distributed CC Scheme [9]

To further reduce the memory complexity of the CPS scheme, the CC scheme adopts the same distributed voting but utilizes the concept of threshold secret sharing [13] to decrease the storage demand. Threshold secret sharing is a group security mechanism whose basic idea is let each group member possess only a part of a key. To obtain the key requires the cooperation of a “number” of members and the “number” should reach the threshold value $t$. Assume $r$ is the number of members in a group, $r$ is the threshold, and $D$ is the key. We have $q(x) = D + a_1x + \ldots + a_{r-1}x^{r-1}$, which means $a_1 \sim a_{r-1}$ being constants and successively generate $r$ sub-keys $q(1), q(2), \ldots, q(r)$ which are then distributed to all members in the group, one for each. To obtain $D$, it takes at least $t$ members to combine their sub-keys together (because $t$ equations can calculate $t$ variables $(D, a_1 \sim a_{r-1})$ and get $D$).

To employ the concept of threshold secret sharing, the scheme installs $q(x) = D + a_1x + \ldots + a_{r-1}x^{r-1}$ to each node in the network before deployment. $(D$ and $a_1 \sim a_{r-1}$ are constants; $r$ is the threshold of votes needed to revoke a node.) When a node $A$ sets up links with the other $r$ nodes (its voting members), it will produce $q(1), q(2), \ldots$, and $q(r)$ and respectively distribute them to the nodes, each with an attached value Hash $(D \mid a_1 \mid a_2 \mid \ldots \mid a_r)$. If a voting member detects something suspicious with $A$, it will broadcast one vote against $A$ (with value $q$) to the other nodes. After receiving the vote, the other voting members will store it in their memory. Any voting member realizing the number of stored votes against $A$ reaches the threshold value $t$ will move to calculate $D$ and $a_1 \sim a_r$, and get
the value hash \( (D || a_1 || a_2 || \ldots || a_t) \). If the obtained hash value equals the originally stored hash value, the voting member determines \( A \) has been compromised and disconnects its link to \( A \). Space complexity is reduced to as low as \( O(r \times t) \).

The Public-key Cryptography (PKC) for Sensor Networks  Pursuing secure performance is quite a technical challenge for WSNs as security mechanisms must work under the very limited processing and bandwidth constraints of sensor nodes – that makes the PKC an inappropriate security mechanism because it needs a lot of calculation and energy resources. The not-so-complex symmetric encryption is instead widely used. Key management for symmetric-key based protocols is nevertheless complicated and prone to adversaries’ attacks. The major problem with symmetric encryption lies in its failure to pre-load all master keys, which sensor nodes need if to communicate with security managers. By contrast, a public-key based protocol yields more flexibility and scalability, especially in large sensor networks where new devices keep entering the cluster.

Public key (PK) authentication is to verify the authenticity of another node’s public key. In original PKC, such a PK authentication operation involves expensive signature verification on a certificate. To conduct PK authentication more efficiently, [14] manages to save the needed memory space by letting each node carry a one-way hash value of those public keys. When two nodes exchange their public keys, each can simply compute the one-way hash value of the received public key and check if the result matches the value stored in their memory. Public-key based node revocation schemes may be able to reduce their complexity to fit in WSNs but may not achieve as desirable security as they do in traditional networks. They nevertheless work better in many ways in a sensor environment than the symmetric cryptography. For example, the elliptic curve cryptography (ECC) [12] and RSA [15] are two major PKC mechanisms which have been applied to WSNs with encouraging and inspiring results.

Employing CRL in Our Scheme  A dynamic multicast group management protocol is proposed [11] to solve such problems as mobility, unreliable links and the cost of multi-hop communication, which are specific to ad hoc networks. The main idea of this protocol is: Group members should actively involve in the security issue of the multicast group to reduce the source’s communication and computation load. As all members are in charge of group security, a Service Right Certificate (SRC) is used to verify that a node is authorized to join the group and to exchange keys with other group members without contacting a PK center (such as the BS). An SRC has the following properties: Any participant can read a certificate to determine its owner’s name and public key, and to verify the authenticity of the certificate (originating from the Certificate Authority (CA)) as well as its currency. The CA will be the only party to create and update the certificates.

The idea of SRC and its corresponding node revocation scheme helps initiate our new node revocation scheme. In our initial design, we assume a CA pre-distributes certificates to every node before deployment and each certificate has its own sequence number. When a node discovers another node has been compromised, it will broadcast a certificate revocation list (CRL) along with the sequence number of the compromised node to the neighbor nodes. Receiving the list, the neighbor nodes can authenticate it by the attached signature and, if the signature is correct, cut off the links with the compromised node. Severing links from any compromised nodes in this way helps the network avoid
an adversary’s further attacks and maintain its normal performance. The digital signature in a CRL employs the asymmetric cryptographic technology to produce the private keys. As a private key cannot be copied by other nodes and is not open, the signature signed by a private key cannot be copied either. The node receiving the cryptographic digital signature thus can use the attached public key to verify the contents of the signature.

**CRL in the Sensor Networks** As data in a wireless sensor network are disseminated in a distributed way, it is particularly important to protect the service of data access. Our previous discussion shows that the PKC schemes are eligible for sensor networks and that certificates can be distributed to nodes offline to reduce communication cost. Basically the BS will serve as the CA responsible for checking a deployed node’s access control lists and policies before generating a new service certificate to it. Based on [11], a node in our scheme will initially take the following steps to revoke a compromised neighbor node in the sensor networks.

1. Periodically multicast a CRL. The list whose format is \([\text{MinSN} \mid \text{CurrentSN} \mid \text{ListOfRevokedCert} \mid \text{Timestamp} \mid \text{Signature}]\) will be reliably multicast. (All downstream nodes are required to acknowledge its receipt.) \(\text{ListOfRevokedCert}\) contains all the certificates that have been revoked since \(\text{MinSN}\), thus all certificates with an \(\text{SN}\) in the \([\text{MinSN}, \text{CurrentSN}]\) interval but not in the revocation list are valid.
2. \(\text{CurrentSN}\) will be incremented and a new revocation list will be issued when a certificate is revoked by the source. To maintain a fixed-sized revocation message, the source may increase \(\text{MinSN}\) if the revocation list is getting too long, send the new CRL to its downstream nodes and hold data transmission to the downstream nodes before getting their acknowledgement of receiving the new CRL.
3. Nodes which receive the packet containing the CRL will check the validity of the last signature. If it is correct, they will cut off their links with the node whose certificate sequence number is in the \(\text{ListOfRevokedCert}\). By removing the danger of a compromised node in this way, we are able to preserve the transmission safety of a sensor network.

#### 3. THE PROPOSED NODE REVOCATION SCHEME

**Our Verifying Mechanism** As mentioned, the last slot of the CRL in the certificate revocation scheme is the digital signature and verification of the digital signature is indeed quite a burden for the energy-constrained sensor networks. To conserve the limited resources, our node revocation scheme adopts a new and efficient verifying approach based on the one-way hash chain in [10] to replace the costly verification of the digital signature. The one-way hash function is adopted because it can map an input of an arbitrary length to an output of a certain fixed length, such as 20 bytes for the SHA-1 algorithm and 32 bytes for the SHA-2 algorithm. Compared with traditional PKC which consists of a signature (at least 256 bytes) and other data including the public key (at least 128 bytes), the hashing value of the one-way hash function is apparently shorter. The computation/communication cost for verifying a CRL is thus significantly reduced.

**The Operation of our New Scheme** Assuming \(H\) is a one-way hash function and \(r\) is a
randomly selected number, a hash chain can be derived by iteratively hashing \( r \): \( H_i(r) = H(H_{i-1}(r)) \) \((i = 1, 2, \ldots)\). With \( r \) being the trust root, the one-way hash chain includes a sequence of hashed values, denoted by \( h_1 = H(r), h_2 = H(h_1), \ldots, h_i = H(h_{i-1}) \) \((i = 1, 2, \ldots)\). The flowchart in Fig. 1 gives the operation steps of our node revocation scheme.

**Generating the PKC**  Note that in an ad hoc network, a node’s PKC will be generated by itself and authenticated by an offline CA. In our design for a sensor network, however, the BS becomes the CA to distribute the PKC to each node before deployment and each node will authenticate the PKC via the BS after deployment.

The BS in our scheme will pre-distribute a public key \( PK_U \) and a private key \( SK_U \) to a node \( U \) before it is deployed. Assuming all system clocks are synchronized in the sensor network, the maximal lifetime \( T \) for every PKC will be the same, and so will be the update interval \( L \) \((T/L = j (j > 1))\). Our assumption that all system clocks are synchronized in the sensor network is indeed based on the synchronizing mechanisms in [16-18] which can synchronize nodes within an accuracy of less-than-10-microsecond synchronization errors.) Node \( U \) then defines the starting valid date as \( D \) when receiving both the public and private keys. The update point for the \( i \)th period \([D_{i-1}, D_i]\) is denoted as \( D_i = D + i \times L \) \((i = 1, 2, \ldots, j; D_0 = D)\). \( U \) moves on to select a random number \( r \) and generate the corresponding one-way hash chain by hashing \( r \) \( j \) times. Node \( U \) then sends a PKC request containing \( PK_U \), \( D \) and \( h_j \) to the BS. The BS will authenticate the received PKC request and, if the authentication is built, examine the hash values submitted by the other nodes. If finding another \( h_j' = h_j \), the BS will inform Node \( U \) that the \( r \) it sends over is unqualified. \( U \) needs to select another random number and generate its corresponding one-way hash chain. After re-examining the hash value, the BS then issues node \( U \) a PKC, denoted as \( PKC_U = E_{CA}(CID, U, PK_U, D, h_j) \), where \( CID \) represents the certificate’s ID and \( E_{CA} \) indicates all of the information encrypted by the private key of CA.
Verifying the Status After deployment, nodes build secure links between each other by exchanging their certificates. In our scheme, the certificate of each node includes a one-way hash value which can indicate the certificate change (with time) and help to judge if a certificate is out of time or not. The process of certificate verification is listed below and also depicted in Fig. 2.

1. When a node (the certificate verifier) needs to verify a $PKC_U$, it will send a query including the $CID$ to the corresponding certificate owner (i.e., node $U$) which then sends its $PKC_U$ to the requesting node.
2. Upon receiving the requested certificate, the node obtains the starting valid date $D$ and the hash value $h_j$ from the $PKC_U$.
3. If the current time is $T'$, the node then computes parameter $i = [(T' - D)/L]$ by $T'$.
4. The node moves on to calculate the value $h_{END}$ by hashing $h_j, i$ times.
5. The node then checks to see if $h_{END} = h_j$. If yes, $PKC_U$ is considered valid at present and remains valid until the next updating point. Otherwise, it is invalid (due to invalid date $D$ or invalid hash value $h_j$).

When verifying a certificate, one needs to have the $h_j$ of node $U$ to verify $PKC_U$. There are two ways to obtain the $h_j$ of node $U$. We can pre-distribute the $h_j$ of $r$ neighbor nodes to each node before it is deployed to a sensor environment, to fortify the transmission security. This will consume only a little memory of the sensor nodes (to store the security information) and incur no extra overhead. Such a pre-distribution approach, however, may cause nodes to carry some un-useful information – because all the $r$ neighbor nodes are not necessarily within the transmission range after deployment. For improvement, we choose to let each node get the $h_j$ of its neighbor nodes after deployment. That is, a node will ask its neighbor nodes to send over their $h_j$ through security channels after they are distributed to the topology. This may improve the problem that faces the pre-distribution approach but will increase overhead and meanwhile jeopardize transmission security (because the required verification information is provided by unverified nodes).
We thus decide to adopt the first approach, i.e., the pre-distribution approach to obtain the $h_j$ of node $U$ to verify $PKCU$.

**The Revocation Voting** Node Revocation Using CRV vs. Our Distributed Revocation Voting Scheme.

- **Node Revocation Using CRV** – To revoke a compromised node, a certificate revocation scheme uses the CRV, a structure containing the revocation message. When a node finds out another node is compromised, it takes the following two steps:

  1. The node first broadcasts to the neighbor nodes a CRV whose format is $[U, D, RevocationCID \oplus h_j, Timestamp]$. RevocationCID $\oplus h_j$ contains the ID of the certificate to be revoked. Each certificate has its own ID. The Timestamp can be used to prevent repeated broadcasting of a CRV, thus saving the energy.

  2. When a neighbor node receives the CRV packet and finds that $h_j$ of $RevocationCID \oplus h_j$ is correct, it will calculate the $RevocationCID$ by $RevocationCID \oplus h_j \oplus h_j$. The $h_j$ to be used later can be authenticated by the value received during the connection time and if $RevocationCID \oplus h_j \oplus h_j$ is correct, the value of $RevocationCID$ can be thus obtained to revoke the compromised node.

   In such a scheme, it is very likely that the attacker gathers the CRV, alters the value of $RevocationCID \oplus h_j$ and broadcasts the altered value to the neighbor nodes to confuse them into revoking the wrong (normal) nodes. To give an example, suppose the original $RevocationCID$ is 0001 and the previous $h_j$ stored in a node’s memory is 0010. In the regular situation, the node will figure out $RevocationCID \oplus h_j$ as 0011. However, if an attacker intrudes a neighbor node or eavesdrops in the communication range and finds out the 0011 value, it can alter 0011 to 0100 and broadcast 0100 out. Such an altered false value will mislead the other nodes to obtain a wrong $RevocationCID$ 0110 and proceed to revoke the normal node whose CID = 0110 by mistake.

- **Our Distributed Revocation Voting Scheme** – To prevent similar mistakes from happening, our new scheme adopts the voting way to revoke the compromised nodes. It sets a voting threshold value $t$ to carry out the requested node revocation. When the number of wrong revocation votes is below the threshold $t$, node revocation will not be conducted. In our scheme, each node will have $m$ voting participants, $m \gg t$, and $d$ secure links with the neighbor nodes, $d$ being its dimension degree. Out of security consideration, we define $d \gg t$. The value of $t$ should exceed 1 (revocation vote) at least and is so defined as $d = m \gg t > 1$. The actual value should be set according to the number of deployed nodes in the network. Setting up a proper value of $t$ is important as it can substantially affect the performance of a node revocation scheme. An improperly large $t$ will make the voting scheme difficult to operate, whereas an unfeasibly small $t$ will shake the security of such a scheme and draw more attacks from adversaries.

   The value of $RevocationCID \oplus h_j$ in a CRV indicates a revocation vote. When a node receives a number of votes to revoke a compromised node by its voting members and the number exceeds the threshold $t$, it will revoke the key shared with the compromised node.
in the key ring and cut off its link with the target node. As the voting members of the node may not be its neighbors, how to inform all of them of this revocation event becomes a critical problem. To solve the problem, we consult an approach in [8] which enables all voting members to receive the revocation list through broadcast propagation, even if they are not neighbors with each other. To conserve the limited resources of a sensor network, our scheme also employs the XOR operation, instead of other encryption algorithms. Energy consumption of an XOR operation is so small that it is nearly negligible at performance evaluation. To put our distributed voting node revocation scheme into work, we let each node store the $r$ hash values of the $r$ nodes in its key ring, thus generating $O(r)$ space complexity. Compared with the space complexity $O(S_{total} r \log r)$ of [8], our scheme apparently needs less storing device.

To give an example, Fig. 3 depicts a sensor network with four nodes ($A$, $B$, $C$ and $D$) and a voting threshold $t = 2$. Before the network is deployed, BS preloads neighbor nodes’ security information to each node (e.g. preloading $B$’s, $C$’s, and $D$’s security information to node $A$). After deployment, node $A$ establishes secure links with nodes $B$, $C$ and $D$, making $B$, $C$ and $D$ its voting members. The preloaded information in each node will be as Fig. 3 illustrates. Now assume $D$ suspects $A$ has been compromised and thus broadcasts $CIDA \oplus h_D$ to $B$ and $C$. After receiving the packet, $B$ and $C$ use the preloaded information to authenticate $CIDA \oplus h_D$. If the value is correct, they will take it down as the first received revocation vote against $A$. If, some time later, $C$ also broadcasts $CIDA \oplus h_C$ to neighbor nodes $B$ and $D$ and the value of $CIDA \oplus h_C$ is also authenticated, node $B$ now holds two valid revocation votes against $A$ and will thus move on to revoke the key shared with $A$ and cut off its link to $A$.

4. PERFORMANCE ANALYSIS AND COMPARISON

Our New Scheme vs. CGPM [8]  The CGPM holds the following properties: completeness, soundness, bounded time revocation completion, unitary revocation and revocation attack resistance. Due to its distributed voting approach, our scheme attains completeness and soundness. Though without original designs for bounded time revocation completion, our scheme can set a bounded time $\Delta t$ based on the discussion in [8]. After the time period $\Delta t$, if the revocation votes against a specific node stays below the threshold value, the revocation attempt will be aborted. Nodes receiving this revocation vote will report the situation to the BS and remove information regarding this abortive revocation.
from its memory to save the space.

Our scheme certainly follows the property of unitary revocation, an important property for any node revocation schemes – since a revocation decision will be extensively broadcast to the entire network. As to revocation attack resistance, our scheme allows no arbitrary node revocation – partly because it adopts the distributed voting approach to avoid such an attack and partly because it employs the one-way hash function and the XOR operation to encrypt the cast vote. Satisfying all the above properties indicates our node revocation scheme is able to attain reliable security for sensor networks.

**Limitations**  Like other node revocation schemes, our scheme also has limitations. For instance, it is very likely that the attacker gathers the CRV, alters the value of \( \text{RevocationCID} \oplus h_j \) and broadcasts the altered value to the neighbor nodes to confuse them into revoking the wrong (normal) nodes. To prevent such a situation from happening, we choose to revoke compromised nodes in a voting way: Node revocation will not happen when revocation votes (or wrong revocation votes) stay below the set threshold – to keep the network from any easy or handy attacks.

**Defending Attacks**  The feature designs of our scheme enable it to defend most of the known attacks, such as de-synchronization attacks or Sybil attacks. In de-synchronization attacks, an adversary will send nodes forged messages with wrong sequence numbers to influence the operation of nodes. Our scheme can defend this attack type because

1. It does not need any sequence number for certificate verification but adopts the start valid date \( D \) and current time \( T \) to compute \( h_j \) and verifies the certificate by \( h_j \).
2. The certificate in our new scheme is also encrypted by the private key of CA, which the attackers can not forge easily.
3. Even if the attackers are able to forge the certificates, it is hard for them to learn about each hash value on the one way hash chain.

In Sybil attacks, attackers will disguise themselves as legal nodes trying to communicate with the other nodes and then compromise them to snatch away the encrypted security information. This type of attacks can hardly escape our new scheme because when such an impersonated attacker tries to communicate with the other nodes, our scheme will figure it out right away by requesting to verify certificates first.

**Space Complexity**  Before node deployment, the space complexity for our scheme is \( O(r) \) – because each node will preload its own private key and the hash values of its \( r \) neighbors to maintain the normal operation of distributed voting. After deployment, each node needs to keep room for the public keys of \( r \) neighbor nodes – the space complexity for our scheme remains \( O(r) \), smaller than \( O(S_{\text{total}} r \log r) \) of the CGPM scheme [8].

**Energy Consumption**  To evaluate energy consumption, we adopt the following processing time on the sensor side [2]; (1) The random point scalar/fixed point multiplication is 480/130 msec; (2) The SHA-1 algorithm takes 2 msec to digest a 128-bit binary string on the M16C; (3) The Cipher Block Chaining mode takes less than 3 msec to decrypt a
256-bit ciphertext; (4) The sensor needs to do one 160-bit modular multiplication, which takes less than 3 msec, and one 160-bit modular addition, which takes less than 3 msec.

In [2], the scalar multiplication of fixed points $P$ and $PKC_d$ are replaced by a precomputed look-up table in the ROM area and need at least one expensive elliptic-curve scalar multiplication of a random point in the key exchange phase. The time taken in the key exchange phase will be 480/130 msec for a random point scalar/fixed point multiplication. Thus the entire protocol execution time on M16C is about 480 msec + 130 msec * 2 + 20 msec = 760 msec, which is taken as the time complexity for the key distribution phase in our scheme.

After key distribution, the calculation time for a node to accept a certificate issued by the BS or to return the certificate to the BS will be far less than the time consumed in the key distribution phase. At this phase, instead of calculating the time-consuming fixed point multiplication and random point scalar multiplication, we will conduct only the modular multiplication, the modular addition and the hash function $h_j$ to verify the signature issued by the BS. According to [2], in a sensor network implemented on Mitsubishi’s M16C microprocessor with the size of 5.2Kbyte code/data, the time taken for one 160-bit modular multiplication and one 160-bit modular addition are both less than 3 msec.

Our analysis on the ECC algorithm shows that the algorithm uses two modular multiplications to verify the signature, two modular multiplications + one modular addition to sign the signature, and 2 msec to hash $h_j$, a total of $3 \times 2 + 3 \times 2 + 3 + 2 = 17$ msec. Our distributed voting scheme needs to hash the one-way hash function and the needed time is subject to the value of the threshold $t$, which will be $2t$ msec. A larger $t$ will cost more but will ensure higher system security. Thus the total time to process the revocation information at the sensor side will be $17 + 2t$ msec. $17 + 2t$ msec may be longer than the symmetric cryptography takes but is definitely much shorter than the traditional PKC scheme requires, and we believe the hardware of sensor nodes can afford this amount of calculation time.

After node deployment and secure-link establishment, the revocation mechanism will be put into work when a node is compromised. According to [2], the time for setting up the communication key will be 1437 bits or 180 bytes, which is considered close to the communication complexity of our scheme, acceptable to the sensor nodes and thus not to be included in our discussion here.

When a sensor node detects a neighbor node may have been compromised, it will broadcast a CRV to other neighbor nodes to inform of the attack. A CRV will consume 352 bits or 44 bytes bandwidth, in which the node ID, the valid date $D$ of the certificate, the Timestamp together takes 64 bits and the RevocationCID $\oplus h_j$ value alone consumes 160 bits or 20 bytes. 44 bytes, the amount of communication bandwidth our scheme needs for the revocation operation, is indeed feasible for a sensor network.

When we notice that our new scheme generates more complexity than the symmetric cryptography in computing loads, we also notice that sensor nodes – which can complete

<table>
<thead>
<tr>
<th>Table 1. Energy consumption on the sensor side.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing time on sensor</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td>Our scheme</td>
</tr>
<tr>
<td>Reference [2]</td>
</tr>
</tbody>
</table>
a node revocation in 17 + 2t msec – are still capable of conducting node revocation under our scheme. In calculating communication complexity, we know the amount of data transmission and the needed energy consumption are in direct proportion. [19, 20] provide two formulas to calculate the energy consumption of transmitting 1 bit data with distance d:

\[
E(d) = \begin{cases} 
50nJ + 10pJ \times d^2 & d \leq 75m \\
50nJ + 0.0013pJ \times d^4 & d > 75m.
\end{cases}
\]

To perform node revocation in our scheme, a sensor node will broadcast CRV to notify other voting members in the local area only. If we assume the radius of such a local broadcast to be 246.06ft, the energy consumption a broadcasting sensor node requires can be obtained by Eq. (1). As stated above, a revocation vote by a sensor node takes 352 bits, which can be translated by Eq. (1) into energy consumption of 37.4 μJ (352 × (50n + 10p × 75^2) = 37.4 μJ).

According to the assumption that the initial/dying energy of a sensor node is each 2J/0.002J [19, 20], the maximum number of revocation votes for each sensor node will be \((2 - 0.002)/37.4\mu J = 53422\).

Based on the result and also on the fact that a node revocation scheme will be executed only when a compromised node is detected, we are positive that our new scheme can work well in the resource-limited sensor network.

A Sum-up Performance Comparison For easier reference, Table 2 gives a comprehensive sum-up review on the focal content of this investigation. (The listed security properties are the material from [8].)

5. CONCLUSION

In wireless sensor networks, a node revocation scheme conducted in a distributed way is effective in reducing the damages caused by a compromised node. However, the operation of such a distributed revocation scheme tends to consume large-scale memory space which is quite a burden for the resource-constrained sensor nodes. To reduce such complexity, this paper presents a new and effective distributed voting revocation scheme.

Our new distributed node revocation scheme utilizes the features of the one-way hash chain, the certificate revocation list (CRL), the public-key cryptography (PKC), and some critical new designs of ours. In our scheme, the BS first signs a certificate to each node before deployment. Each certificate has a unique ID which allows a node to prove it is authorized to access such services as data authentication or node revocation. When a node detects another node in the network has been compromised, it can broadcast a compromise revocation vote (CRV) with the compromised node’s certificate ID to all neighbor nodes. Receiving such a broadcast packet, a neighbor node will authenticate its validity by the hash value attached in the CRV. If the vote is authenticated, the neighbor node will increase the amount of revocation votes. When the received revocation votes against the same node exceed a threshold value t, each neighbor node will cut its link to the revoked node. Thus by severing the links between a compromised node and all nor
Table 2. Comparison among our scheme and the schemes in [8, 9].

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic concept</td>
<td>Using threshold secret sharing to vote</td>
<td>Using threshold secret sharing to vote</td>
<td>Using CRL to vote</td>
</tr>
<tr>
<td>Encryption type</td>
<td>Symmetric cryptography</td>
<td>Symmetric cryptography</td>
<td>Asymmetric cryptography</td>
</tr>
<tr>
<td>Preload information in sensor nodes</td>
<td>• ( q(x) = D + a_1 x^1 + \ldots + a_{r-1} x^{r-1} ) ( q(x) ), ( x ) ( H_s(x) = H(D \parallel a_1 \parallel a_2 \parallel \ldots \parallel a_{r-1}) )</td>
<td>• ( mask_{bas} ) and ( H(q_{bas}) ) • a path of log ( r ) hash tree values for each of ( B )’s neighbors and ( R_B ) • ( Emask_{AB}[q_{bs}(x_{bas}), x_{bas}] )</td>
<td>• The hash value ( h_{END} ) of neighbor nodes • The certificate revocation vote from other nodes</td>
</tr>
<tr>
<td>Space complexity</td>
<td>( O(r \times t) )</td>
<td>( O(S_{total} r \log r) )</td>
<td>( O(r) )</td>
</tr>
<tr>
<td>Votes’ authentication</td>
<td>No</td>
<td>Using the Merkle tree to authenticate votes</td>
<td>Using the hash value in a certificate to authenticate votes</td>
</tr>
<tr>
<td>Network security analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asymmetric cryptography</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td>Check validity of vote</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>bounded time revocation completion</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Unitary revocation</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>revocation attack resistance</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Network Security</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

mal nodes, we are able to protect the network from an adversary’s further attack.

Performance evaluation shows that, when compared with related node revocation schemes, our scheme attains stronger network security at higher calculation cost due to its PKC distributed voting mechanism. To reduce the cost, we bring in the one-way hash function and the XOR operation and manage to reduce energy consumption to an acceptable level for sensor nodes. In fact, when taking both network security and hardware complexity into account, we consider our scheme feasible and practical for the wireless environment because (1) the higher cost is acceptable in trade of the earned enhanced network security and (2) the cost is affordable as the chance to pay such a cost is rare (only when the network is under attacks) though critical.

REFERENCES


Po-Jen Chuang (莊博任) received the B.S. degree from National Chiao Tung University, Taiwan, R.O.C., in 1978, the M.S. degree in Computer Science from the University of Missouri at Columbia, U.S.A., in 1988, and the Ph.D. degree in Computer Science from the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, U.S.A. (now the University of Louisiana at Lafayette), in 1992. Since 1992, he has been with the Department of Electrical Engineering, Tamkang University, Taiwan, where he is currently a Professor. He was the department chairman from 1996 to 2000. His main areas of interest include parallel and distributed processing, fault-tolerant computing, computer architecture, mobile computing, and network security.

Shao-Hsuan Chang (張煒軒) received the B.S. and M.S. degrees in Electrical Engineering respectively in 2003 and 2007 from Tamkang University, Taiwan. He is now a hardware engineer of the Department of Motherboard Hardware of the ASRock Incorporation, Taipei, Taiwan. His research interests include network security and mobile computing.

Chih-Shin Lin (林志信) received the B.S. and M.S. degrees in Electrical Engineering respectively in 2005 and 2007 from Tamkang University, Taiwan, where he is currently pursuing the Ph.D. degree. His research interests include parallel and distributed processing, network security, and mobile computing.