A PKC-based Node Revocation Scheme in Wireless Sensor Networks

Po-Jen Chuang, Shao-Hsuan Chang, and Chih-Shin Lin
Department of Electrical Engineering
Tamkang University
Tamsui, Taipei County
Taiwan 25137, R. O. C.
E-mail: pjchuang@ee.tku.edu.tw

Abstract

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 concept of threshold secret sharing, the certificate revocation list and the public-key cryptography.

1. Introduction

Generally deployed in unattended environments to collect the needed information, a sensor network can be easily assaulted by adversaries. Network security becomes a major problem. A number of security devices, including **key management** [e.g., 1-5], are established to enhance the security. Key management involves key distribution and revocation [6]. Key distribution disseminates secret information to the principals to initialize secure communications. Key revocation, on the other hand, aims to remove secrets that may have been snapped by attackers from a compromised node and thus results in revocation of the 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 is designed mainly to cut off all links between compromised nodes and normal nodes. A node revocation scheme can work in a **centralized** way, such as the EG scheme [6], in which the base station (BS) is responsible for conducting and broadcasting revocation decisions. It can also work in a **distributed** way, i.e., the compromised node's neighbor nodes will initiate and broadcast revocation decisions without the assistance of the BS. The CPS [7], CGPM [8] and CC [9] are all distributed node revocation schemes. A distributed node revocation scheme can react more quickly than a centralized scheme and as a result reduce both delay time and communication cost.

This paper introduces **a new distributed node revocation scheme** based on the one-way hash chain [10], the certificate revocation list (CRL) [11] and the public key cryptography (PKC) [2]. 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 the CRL 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 attached hash value. If the validity is verified, the neighbor node will increase the revocation vote count. When the revocation votes against a node exceed the threshold value t, each neighbor node will cut off their links to the revoked node to protect the network from an adversary's further attack. To reduce the calculation cost, we bring in the one-way hash function and the XOR operation. 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.

2. The PKC and CRL in Our Scheme

2.1. The PKC for Sensor Networks

Pursuing secure performance of a sensor network is quite a technical challenge as security mechanisms

must work under the very limited processing and bandwidth constraints of the sensor nodes. PKC is thus considered an inappropriate security mechanism for the sensor network as it needs a lot of calculation and energy resources. Instead, the less complex symmetric encryption is widely taken to maintain the security of a sensor network. Key management for **symmetric-key based protocols** is nevertheless complicated and prone to adversaries' attacks as it can not pre-load all master keys (due to the high mobility and a limited memory space) 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 authentication, one important PKC operation, is to verify the authenticity of another node's public key, i.e., to make certain a public key truly belongs to its claimed owner. In original PKC, such a public-key-authentication operation involves expensive signature verification on a certificate. To authenticate a public key in a more efficient way, [12] manage 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, to save energy.

A public-key based node revocation scheme, such as the elliptic curve cryptography (ECC) [2] or RSA [13], works better in many ways in a sensor environment than the symmetric cryptography.

2.2. Employing the CRL in Our Scheme

The dynamic multicast group management protocol in [11] is proposed to solve such problems as mobility, unreliable links and multi-hop communication cost which are specific to the ad hoc networks. The main idea is to let group members actively involve in the security of the multicast group, thus reducing the communication and computation load on the source. As group members are in charge of group security, **a service right certificate** is used to verify that a node is authorized to join the group and the group members (nodes) can use it to exchange keys without contacting a public-key center (such as the BS). A service right certificate 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, i.e., the CA) as well as its currency. The CA will be the only party to create and update the certificates.

The idea of the service right certificate and its corresponding node revocation scheme helps initiate and shape 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, whose format is [MinSN ｜ CurrentSN ｜ ListOfRevokedCert ｜ Timestamp ｜ Signature]) 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. Cutting links to a compromised node 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 can not be copied by other nodes and is not open, the signature signed by a private key can not be copied either. The node receiving the cryptographic digital signature thus can use the attached public key to verify the contents of the signature.

3. The Proposed Node Revocation Scheme

Verifying the digital signature in a CRL is indeed quite a burden for the energy-constrained sensor networks. To conserve the limited resources of a sensor network, 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 as 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 clearly shorter. The computation and communication cost for verifying a CRL can be thus significantly reduced.

3.1. The Operation of the New Scheme

First define the one-way hash chain. 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, …). With r being the trust root, the one-way hash chain includes a sequence of hashed values, denoted by $h_1 = H(r)$, $h_2 = H(h_1)$, …, $h_i = H(h_{i-1})$ (i = 1, 2,…). Our node revocation

scheme includes the generation of PKC, verification of the status and revocation vote.

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.

To be more specific, 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 to a sensor environment. 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 ($T / L = j$ ($j > 1$)). Node U then defines the starting valid date as D when receiving the public key and the private key. The update point for the ith period $[D_{i-1}, D_i]$ is denoted as D_i : $D_i = D + i * L$ ($i = 1,2,\ldots, j$; $D_0 = D$). 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 anther 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 $PKC_U = SIGN_{CA}(CID, U, PK_U, D, h_j)$, where CID represents the certificate's ID.

Verifying the Status

After deployment, nodes build secure links between each other by exchanging their certificates. As mentioned, nodes in our scheme use the one-way hash chain, instead of the digital signature, to authenticate each other's certificate and thus to maintain the security of the network with reduced energy-consumption. The process of certificate verification is listed below.

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 computes parameter $i = [(T'-D)/L]$ by T'.
4. The node then calculates the value h_{END} by hashing h_{j-i} i times, and checks 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.

3.2. Node Revocation Using Traditional CRL

To revoke a compromised node, a traditional certificate revocation scheme uses the CRL, a signature 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 CRL 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 is used to prevent repeated broadcasting of a list, thus saving the energy.
2. When a neighbor node receives the CRL 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 likely that an attacker gathers the CRL, 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 revocation ID 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 CID 0110 and proceed to revoke the normal node whose CID = 0110 by mistake.

To prevent similar mistakes from happening, our new scheme adopts the voting way to revoke the compromised nodes: When the number of wrong revocation votes is below the threshold t, node revocation will not be conducted.

3.3. Our Distributed Voting Certificate Revocation Scheme

Our new scheme sets a voting threshold value t to carry out the requested node revocation. Each node will have m voting participants, $m>>t$, and d secure links with the neighbor nodes, d being its dimension degree. Out of security consideration, we define $d>>t$. The value of t should exceed 1 (revocation vote) at least and is so defined as $d = m>>t>1$. 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 CRL 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.

Now assume D suspects A has been compromised and thus broadcasts $CID_A \oplus h_{jD}$ to B and C. After receiving the packet, B and C use the preloaded information to authenticate $CID_A \oplus h_{jD}$. 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 $CID_A \oplus h_{jC}$ to neighbor nodes B and D and the value of $CID_A \oplus h_{jC}$ 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 Evaluation

Performance evaluation is conducted to compare certain existing node revocation schemes and our distributed voting revocation scheme. As mentioned earlier, our scheme is expected to generate higher computation and communication cost than the other schemes due to its employment of the PKC mechanism. Such cost is acceptable given the enhanced network security it brings about; it is also affordable as the chance to pay such a cost is rare but critical.

4.1. Complexities for Our New Scheme

Before node deployment, the space complexity for our node revocation scheme will be O(r) because each node is set to preload its own private key and the public keys of its r neighbors. After deployment, each node must reserve rooms for the r hash values of the r nodes in its key ring – to maintain the normal operation of the distributed voting revocation scheme. The space complexity for our scheme remains O(r). Compared with the space complexity O (S_{total}r log r) of [8], our scheme apparently needs less storing space.

4.2. Energy Consumption

Table 1 gives energy consumption on the sensor side for our scheme and the scheme in [2] which also involves the PKC mechanism. To evaluate energy consumption, we adopt the following processing time on the sensor side [2].

1. The random point scalar multiplication is 480 msec and the fixed point multiplication is 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 does a 160-bit modular multiplication, which takes less than 3 msec, and a 160-bit modular addition, which takes less than 3 msec.

Table 1. Energy consumption on the sensor side

	Processing time on sensor	Communication complexity
Our scheme	17+2t msec	352 bits
Reference[2]	760 msec	1437 bits

[2] replaces the scalar multiplication of fixed points P and PK_{CA} by a pre-computed look-up table in the ROM area and needs at least one expensive elliptic-curve scalar multiplication of a random point in the key exchange phase. On the sensor side, the time taken in the key exchange phase will be 480 msec for a random point scalar multiplication, 130 msec for a fixed point multiplication. Thus the entire protocol execution time on M16C is about 480 msec + 130 mesc * 2 + 20 msec = 760 msec, which is also 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

Table 2. Comparison among our scheme and the schemes in [9] and [8]

	The CC Scheme [9]	The CGPM Scheme [8]	Our scheme
Basic concept	Using threshold secret sharing to vote	Using threshold secret sharing to vote	Using CRL to vote
Encryption type	Symmetric cryptography	Symmetric cryptography	Asymmetric cryptography
Preload information in sensor nodes	➢ $q(x) = D+a_1x^1+\ldots+a_{t-1}x^{t-1}$ ➢ q(x), x ➢ $Hq(x)=H(D \parallel a_1 \parallel a_2 \parallel ...\parallel a_{t-1})$	➢ $mask_{Bas}$ and $H^2(q_{Bs})$ ➢ a path of log r hash tree values for each of B's neighbors and R_B ➢ $Emask_{ABs}[q_{Bs}(x_{ABs}), x_{ABs})]$	➢ The hash value of neighbor nodes ➢ The certificate revocation vote from other nodes
Space complexity	O(r×t)	$O(S_{total}rlogr)$	O(r)
Votes' authentication	No	Using the Merkle tree to authenticate votes	Using the hash value in a certificate to authenticate votes
Network Security	low	medium	high

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*2 + 3*2 + 3 + 2 = 17 msec. Then 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. As mentioned, when a sensor node detects a neighbor node may have been compromised, it will broadcast a CRL to other neighbor nodes to inform of the attack. A CRL 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 feasible for a sensor network.

4.3. Performance Comparison

Table 2 lists our comprehensive comparison between our new scheme and two other distributed revocation schemes, i.e., the CGPM scheme in [8] and the CC scheme in [9].

As the table indicates, our new scheme outperforms the other two schemes in enhancing network security because it is the only scheme to employ the PKC mechanism for key-distribution and the asymmetric cryptography for encryption. The CGPM scheme [8] achieves higher network security than the CC scheme mainly because it uses the Merkle tree to authenticate the validity of votes while the CC scheme sets no mechanisms to authenticate the revocation votes.

5. Conclusions

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 of the resource-constrained sensor nodes. To reduce such complexity, this paper presents a new distributed voting revocation scheme based on the one-way hash chain, the concept of threshold secret sharing, the certificate revocation list (CRL) and the public-key cryptography (PKC). Performance evaluation shows that, when compared with related node revocation schemes, our scheme achieves 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 the hardware-constrained sensor nodes.

Node revocation is indeed a rare case which happens only when the system is under attack, but when it happens, how to maintain network security becomes a critically important issue. As security analysis shows our new revocation scheme outperforms the other target schemes in resisting adversaries' advanced attacks, we consider it feasible and practical to earn such enhanced network security at reasonable and acceptable calculation cost.

6. References

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. 7th Annual ACM Int'l Conf. on Mobile Computing and Networks*, July 2001, pp. 189-199.

[2] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang, "Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks," *Proc. 2nd ACM Int'l Conf. on Wireless Sensor Networks and Applications*, Sept. 2003, pp. 141-150.

[3] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *Proc. 10th ACM Conf. on Computer and Communication Security*, Oct. 2003, pp. 62-72.

[4] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. on Computer and Communications Security*, Oct. 2003, pp. 52-61.

[5] A. Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy, "Scalable Cryptographic Key Management in Wireless Sensor Networks," *Proc. 24th Int'l Conf. on Distributed Computing Systems Workshops*, Mar. 2004, pp. 796-802.

[6] L. Eschenauer, and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. 9th ACM Conf. on Computer and Communication Security*, Nov. 2002, pp. 41-47.

[7] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. 2003 IEEE Symp. on Security and Privacy*, May 2003, pp. 197–213.

[8] H. Chan, V. D. Gligor, A. Perrig, and G. Muralidharan, "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks," *IEEE Trans. on Dependable and Secure Computing*, Vol. 2, pp. 233-247, July-Sept. 2005.

[9] P.-J. Chuang and T.-H. Chao, "A Node Revocation Scheme for Sensor Networks," *Proc. IASTED Int'l Conf. on Wireless Sensor Networks*, July 2006.

[10] J. Li, Y. Zhu, H. Pan, and S. Liu, "A Distributed Certificate Revocation Scheme Based on One-Way Hash Chain for Wireless Ad Hoc Networks," *Proc. 2nd IEEE Int'l Conf. on Mobile Technology, Applications and Systems*, Nov. 2005.

[11] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, "Secure Multicast Groups on Ad Hoc Networks," *Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003, pp. 94-102.

[12] W. Du, R. Wang, and P. Ning, "An Efficient Scheme for Authenticating Public Keys in Sensor Networks," *Proc. 6th ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing*, 2005, pp. 58-67.

[13] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK Securing Sensor Networks with Public Key Technology," *Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, Oct. 2004, pp. 59-64.