*Applied Mechanics and Materials* *Online: 2013-09-03*
*ISSN: 1662-7482, Vols. 411-414, pp 135-140*
*doi:10.4028/www.scientific.net/AMM.411-414.135*


# An Impersonating-Resilient Key Agreement Scheme for Wireless Sensor Networks

Lijun Yang[1, a], Chao Ding[1, b] and Meng Wu[1, c]

[1]Nanjing University of Posts & Telecommunications

66 Xin-mofan Road, Nanjing, Jiangsu Province, China

[a]yanglijun1119@gmail.com, [b]dingchao1984@gmail.com, [c]wum@njupt.edu.cn

**Keywords:** sensor network; security; key agreement; identity-based cryptography; efficient implementation

**Abstract.** In this paper, we proposed an authenticated key agreement scheme, TinyIBAK, based on the identity-based cryptography, for wireless sensor networks. The TinyIBAK scheme provides implicit identity authentication and key confirmation, which effectively prevents unauthenticated access to the sensor network and resists impersonation attacks. We implemented our proposal for TinyOS-2.1 based on the MICAz motes, analyzed the memory occupation, and evaluated the time and energy performance with the Avrora toolkit. Experimental results indicate that our proposal consumes an acceptable amount of resources, and is feasible for infrequent key distribution and rekeying in wireless sensor networks.

## Introduction

Wireless sensor networks (WSNs) are vulnerable to various malicious attacks such as eavesdropping, message replay and node compromise, due to its nature of open wireless medium, restricted node resources, and unattended operation manners. Thus, security is very significant for wireless sensor networks deployed in hostile environments.

Key management is a fundamental cryptographic mechanism upon which other security protocols are built. Key pre-distribution is a promising scheme for key management in sensor networks. In this kind of schemes, keying materials are preloaded into sensor nodes before deployment. Eschenauer and Gligor [1] propose the first probabilistic key pre-distribution scheme for pairwise key establishment. The main idea behind the scheme is that each node randomly picks a set of keys from a key pool before deployment so that any two sensor nodes have a certain probability to share at least one common key. Later, other key pre-distribution schemes [2-4] have been proposed to improve the efficiency. However, the common shortcomings of key pre-distribution schemes cannot be neglected. To ensure the key connectivity, most pre-distribution schemes require a large number of keys to be preloaded. With the increase of the nodes number, the key storage occupation and communication costs increase significantly, which makes the pre-distribution schemes inapplicable to large scale sensor networks. On the other hand, it is not flexibility for rekeying and new sensor node addition.

To address the problems aforementioned, researchers have been investigating the possibility of making use of public-key cryptographic (PKC). Several works [5-7] have demonstrated the feasibility of PKC on the resource constrained sensor nodes. Malan et al. [5] implement elliptic curve Diffie-Hellman (ECDH) key exchange on the MICA2 mote to derive a mutual key between pairs of nodes. But it is not authenticated, and hence is subject to the man-in-the-middle attack. Watro et al. [6] propose a scheme named TinyPK based on RSA cryptosystem and Diffie-Hellman algorithms, allowing authentication and key agreement between a sensor network and a third party as well as between two sensor networks. However, TinyPK requires a public key infrastructure (PKI) for public key authentication, which is not affordable in WSNs. Yang et al. [8] propose an approach based on identity-based encryption and Diffie-Hellman algorithms, providing authenticated key agreement between pairs of sensor nodes. But its computation and memory overhead are too high to apply practically. Oliveira et al. [7] implement SOK protocol for sensor networks, in which no interaction is required. However, the shared secret derived through SOK is static, which means that SOK does not support rekeying.

The main idea of identity-based cryptography (IBC) is any information that uniquely identifies entities (e.g. name or email address) can be used as a public key, thus the PKI is unnecessary. In this work, we argue that IBC is ideal for WSNs. In the context of WSNs, the network deployer, can be considered as a trusted entity, thus can play the role of Private Key Generator (PKG) perfectly which takes charge of generating and escrowing nodes' private keys. On the other hand, the private keys can be preloaded into nodes offline in a secure way.

Inspired by the synergy between IBC and WSNs, we proposed an identity-based authenticated key agreement scheme, TinyIBAK, based on bilinear paring, for wireless sensor networks. The TinyIBAK scheme provides implicit identity authentication and key confirmation, which effectively prevents the unauthenticated node from accessing the sensor network and resists impersonation attacks. Compared with other ID-based approaches, our proposal is more efficient, and flexible for rekeying. When compare with traditional key pre-distribution schemes, TinyIBAK achieves many advantages in terms of security strength, key connectivity, scalability, communication and storage overhead, and enables efficient secure rekeying. To evaluate the feasibility and performance of our proposal, we implemented TinyIBAK on the MICAz mote based on RELIC cryptographic toolkit [9]. Experimental results indicate that our proposal consumes an acceptable amount of resources.

## Proposed TinyIBAK Scheme

**TinyIBAK.** In this section, we describe our proposal, TinyIBAK. TinyIBAK is involved with three kinds of entities, i.e. any two nodes intent to agree a session key, represented as node A and node B, and a management system owned by the network deployer (plays the role of PKG). TinyIBAK is comprised of three steps, including Setup, Extract, and Key Agreement. Prior deployment, the management system performs Setup and Extract operations offline in a secure environment, and preloads corresponding information including private key, public key, and public system parameters into each node. After deployment, each node performs neighbor discovery, broadcasts its ID within its neighborhood, and performs the Key Agreement step of TinyIBAK as shown in Fig. 1.

*Setup*: takes a security parameter $k$ as input, and returns the master key and system parameters. For a given security parameter $k$, the management system does the followings.

(1) Chooses a pairing-friendly supersingular elliptic curve $E/\mathbb{F}_q$, and runs the IBC parameter generator to generate the cyclic groups $(G_1,+)$ and $(G_T,\bullet)$ of the prime order $n$, an arbitrary generator $P$ of $G_1$, and a bilinear pairing $\hat{e}: G_1 \times G_1 \to G_T$;

(2) Chooses the master key $s \in_R Z_n^*$, and computes the system public key $P_{pub} = sP$;

(3) Chooses a cryptographic secure hash function $H_1:\{0,1\}^* \to G_1$ and $H_2: G_T \to \{0,1\}^k$;

(4) Publishes the tuple $\Pi = (q, G_1, G_T, \hat{e}, P, P_{pub}, H_1, H_2)$ as system parameters and keeps the master key $s$ secret.

*Extract*: takes the system parameters, master key, and a node identifier as input and returns the node's ID-based long-term key. For each node $i$ with identifier $ID_i$, the management system

(1) Computes $Q_i = H_1(ID_i)$;

(2) Computes the private key $d_i = sQ_i$;

(3) Preloads $\langle ID_i, d_i, Q_i, \Pi\rangle$ into every node of the network.

*Key Agreement*: For two nodes A and B to establish an authenticated session key, they should do as follows:

(1) To start a key agreement session with the intended responder B, the initiator A chooses a random $a \in_R Z_n^*$, computes the key token $W_A = aQ_A$, and sends $ID_A$ and $W_A$ to B.

(2) On receiving the initiation message from node A, node B does the following.
   a) Chooses a random $b \in_R Z_n^*$, and computes the key token $W_B = bQ_B$;
   b) Computes $K_{BA} = \hat{e}(W_A + bQ_A, d_B)$, and derives $k_1, k_2$ using KDF, $(k_1, k_2) \leftarrow KDF(K_{BA})$;
   c) Computes the message authentication code (MAC) with the secret $k_1$, $t_B = MAC_{k_1}(ID_B, ID_A, W_B, W_A)$;
   d) Sends $ID_B$, $W_B$ and $t_B$ to node A. $B \rightarrow A: ID_B \| W_B \| t_B$.
(3) On receiving the responding message from node B, node A
   a) Computes $K_{AB} = \hat{e}(d_A, W_B + aQ_B)$, and derives $k_1, k_2$ using KDF, $(k_1, k_2) \leftarrow KDF(K_{AB})$;
   b) Check the received MAC tag $t_B = MAC_{k_1}(ID_B, ID_A, W_B, W_A)$;
   c) Computes the MAC $t_A = MAC_{k_1}(ID_A, ID_B, W_A, W_B)$, and sends it to node B, $A \rightarrow B: t_A$.
(4) On receiving the confirmation message from node A, node B checks the received $t_A = MAC_{k_1}(ID_A, ID_B, W_A, W_B)$.
(5) The shared session key between node A and node B is $k_2$, i.e. $k_{AB} = k_2$.

   It is easy to see that due to the bilinear property $K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b)}$, therefore the shared secret establishes between node A and B.

The shared secret is dynamic, because it used the number *a* and *b*, which are randomly chosen for every round of key agreement. In TinyIBAK, two symmetric keys are derived through the key derive function, in which $k_1$ is used to compute the message authentication code (MAC) and $k_2$ is the shared session key used for the secrecy communications between node A and B.

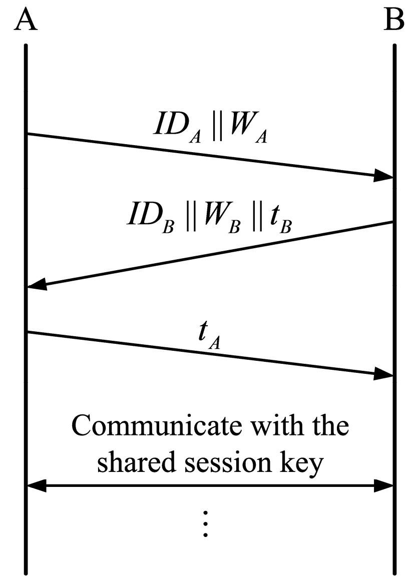


Figure 1 The Key Agreement of TinyIBAK

**Node Addition and Rekeying.** Whenever the new nodes enter the existing network, or failure nodes need replacement, it does not require any new keying material for existing nodes. The management system runs the Setup and Extract steps of TinyIBAK, and preloads $\langle ID_i, d_i, Q_i, \Pi \rangle$ into the new nodes. After deployment, the new nodes run the Key Agreement step to establish shared secret keys with its neighbor nodes.

Whenever rekeying is needed, node A sends rekeying request $Enc_{k_{AB}}(rekeying, nonce)$ to its neighbor node B, and then they both run the Key Agreement step to agree new shared secret key.

Node addition/replacement and rekeying only incur interactions within the neighborhood, and do not affect other nodes of the WSN.

## Security Analysis

The inherent security of the TinyIBAK scheme relies on the difficulty of Bilinear Diffie-Hellman Problem (BDHP). We discuss the security properties of our proposal in the following.

*Forward secrecy:* Assume that the adversary named Eve achieved the private keys $d_A$ and $d_B$, but she still cannot calculate the shared secret between the node A and B in this case. So TinyIBAK provides perfect forward secrecy as previously established session keys remain secure after the secret keys of both communicating parties are compromised.

*Known key security:* Each run of the TinyIBAK between the node A and B produces a unique session key that depends on the random selection of *a* and *b*. Eve, who learned some other session keys, cannot predict new or subsequent session keys, and cannot predict any earlier session keys either. Therefore, the TinyIBAK scheme provides known key security.

*Implicit Authentication:* Only the node A and B can calculate the shared secret between them. Therefore, TinyIBAK provides implicit authentication of node identity. It can effectively prevent the unauthenticated node from accessing the sensor network, and resist impersonation attacks launched by the malicious nodes.

*Key confirmation:* If the tag $t_A$ and $t_B$ are validated, both parties can be sure that the other party does calculate the shared secret $K_{AB} = K_{BA} = \hat{e}\left(Q_A, Q_B\right)^{s(a+b)}$, the other party knows the identity of the entity that it is communicating with, and the communication process of both sides has not been tampered with. Therefore, TinyIBAK achieves mutual authentication of the communication, and confirmation of the shared key.

*Resilience against Node Capture Attack:* In the TinyIBAK, each sensor is preloaded with one unique private key. After the session key established, each pair of nodes has a different shared key. Thus, compromising *c* sensors does not affect the security of communication among other pairs of nodes. Therefore, TinyIBAK provides strong resilience to the node compromise attack.

## Implementation and Performance Evaluation

**Plaform and implementation.** In order to evaluate the feasibility and performance of TinyIBAK, we implement it on the MICAz platform, a popular choice among the research community. The MICAz is based on the low-power 8-bit microcontroller ATmega128L, which operates at 7.3828MHz and offers 4KB of RAM memory and 128KB of program space. The MICAz runs TinyOS and embeds an IEEE 802.15.4 compliant CC2420 transceiver with a claimed data rate of 250 kbps.

Our implementation involves a management system (a PC with TinyOS installed) and two MICAz sensor nodes. The management system performs the Setup and Extract steps, and preloads the system parameters and node's private key into each sensor node. The sensor nodes perform the Key Agreement step. All the software programs running on the sensor nodes have been developed in the NesC language. We developed and tested our program first on TOSSIM [10] and Avrora [11], popular simulation and analysis toolkits for MICAz motes. These tools together give the strength of code development and debugging. These programs were then installed on MICAz sensor nodes.

As recommended by NIST [12], we adopt an 80-bit security level (RSA-1024 equivalent), which can be achieved by choosing $n \geq 2^{160}$ and $q^k \geq 2^{1024}$ for pairing-based cryptography. In accordance with the required security level we choose the supersingular elliptic curve $y^2 + y = x^3 + x$ with the embedding degree $k = 4$ defined over the binary field $\mathbb{F}_{2^{271}}$ for the pairing imlementation.

The implementation of TinyIBAK scheme is based on the RELIC cryptographic toolkit [9], which provides the necessary tools to perform operations on elliptic curves. RELIC is a publicly available and open source library which is specifically designed for resource-constrained devices.

**Performance Evaluation.** Since the Setup and Extract phases of our proposal are performed offline before the network deployment, here we only focus on the operations carried out by the sensor nodes during the Key Agreement phase.

The TinyIBAK scheme involves the computations of one bilinear pairing, two 271-bit scalar point multiplications, one scalar point addition, two hashing functions and one MAC. The empirical computation overhead of TinyIBAK is shown in Table 1. *Initiation* comprises the parameter configuration of RELIC and the loading of the system parameters. *Parameter Computation* comprises the random chosen of $a$ ($b$) and the computation of key token $W_A$ ($W_B$). *Shared Key Computation* comprises the computation of the shared secret and the derivation of the shared key.

Table 1 Computation overhead of TinyIBAK

| | Initiation | Parameter Computation | Shared Key Computation | MAC Computation | MAC Verification | Total |
|---|---|---|---|---|---|---|
| CPU Cycles | 9,614,652 | 121,582 | 16,115,641 | 156,815 | 157,058 | 26,165,748 |
| Energy [mJ] | 29.60 | 0.37 | 49.91 | 0.48 | 0.49 | 80.85 |
| Time [s] | 1.304 | 0.016 | 2.186 | 0.021 | 0.021 | 3.55 |

As shown in the Table 1, the TinyIBAK scheme takes 3.55s and consumes 80.85mJ in total to execute on ATmega128L micro-controller. These data show that TinyIBAK consumes an acceptable amount of resources, given that these operations are performed very rarely and mainly at the beginning of the network operation. After the key agreement phase, nodes will use much cheaper symmetric key encryption methods.

The memory overhead of TinyIBAK includes the RELIC code, TinyOS code, node's *id*, private key, public key, system parameters and the established shared keys with its neighbors. A private key (public key) requires a point on an elliptic curve, which is represented by coordinates ($x, y$) from a finite field with 271-bit elements. Given $x$ and a single bit of $y$, however, one can easily derive $y$. Thus, a private key (public key) can be compressed to 34 bytes.

Table 2 Memory overhead of TinyIBAK

| Scheme | RAM [Byte] | | ROM [Byte] |
|---|---|---|---|
| | ***.data*** | ***.bss*** | ***.text*** |
| TinyIBAK | 1,096 | 1,168 | 57,146 |

Table 2 depicts the memory occupation of TinyIBAK, where the *.bss* and *.data* segments consume RAM while the *.text* segment consumes ROM. The RAM utilization may seem slightly high for MICAz sensor mote but most memory is reserved only for the duration of key agreement. The values showed in Table 2 present only the peak numbers for stack usage during the program execution.

In our TinyIBAK scheme, three messages are required to be exchanged during the key agreement, which can be expressed in the form of Eq. 2

$$\begin{array}{l} \mathrm{A} \rightarrow \mathrm{B} : ID_A \,\|\, W_A \\ \mathrm{B} \rightarrow \mathrm{A} : ID_B \,\|\, W_B \,\|\, t_B \\ \mathrm{A} \rightarrow \mathrm{B} : t_A \end{array} \quad . \tag{2}$$

The key token $W_A$ ($W_B$) is a point on $E\left(\mathbb{F}_{2^{271}}\right)$, thus can be compressed to 34 bytes. Each node identifier occupies 2 bytes. The MAC function was implemented with SHA1, thus the MAC tag has 16 bytes. Therefore, the initiator and the responder are required to transmit 52 bytes, respectively. Here we do not take the extra overhead arose from TinyOS packet encapsulation into consideration.

The communication overhead of TinyIBAK is independent of the network size and fixed for each node pair. This feature allows our scheme to scale gracefully with the number of nodes in the network.

## Conclusions

In this paper, we proposed an identity-based authenticated key agreement scheme, TinyIBAK, based on the bilinear paring, for wireless sensor networks. The TinyIBAK scheme provides several important security properties, such as implicit authentication, key confirmation, and resilience against impersonation attacks and node compromise attacks. To evaluate the feasibility and performance of our proposal, we implemented it for TinyOS-2.1 based on the MICAz motes. The experimental results indicate that our proposal consumes an acceptable amount of resources, and is feasible for infrequent key distribution and rekeying in large scale sensor networks.

## Acknowledgements

This work is financially supported by National Basic Research Program of China (973 Program) under Grants 2011CB302903, the National Natural Science Foundation of China (Grant No. 61100213), the Key Program of Natural Science for Universities of Jiangsu Province (Grant No.10KJA510035), the Specialized Research Fund for the Doctoral Program of Higher Education (20113223120007), the Science and Technology Program of Nanjing (201103003) and the Postgraduate Innovation Project Foundation of Jiangsu Province (Grant No. CXLX11_0411).

## References

[1] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," presented at the Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, 2002.

[2] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," presented at the Proceedings of Symposium on Security and Privacy, 2003.

[3] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure Wireless Sensor Networks," presented at the Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, Fairfax, Virginia, 2003.

[4] D. Liu, P. Ning, and R. Li: ACM Transaction on Information System Security vol. 8(2005), p. 41-77, 2005.

[5] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," presented at the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks(SECON), Santa Clara, California, USA, 2004.

[6] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," presented at the Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, Washington DC, USA, 2004.

[7] L. B. Oliveira, D. F. Aranha, C. P. L. Gouvêa, M. Scott, D. F. Câmara, J. López*, et al.*: Computer Communications vol. 34 (2011), p. 485-493.

[8] G. Yang and H. Cheng: Chinese Journal of Electronics vol. 36(2008), p.1389-13958. In Chinese.

[9] RELIC Cryptographic Toolkit on http://code.google.com/p /relic-toolkit

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," presented at the Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, 2003.

[11] Avrora: The AVR Simulation and Analysis Framework on http://compilers.cs.ucla.edu/ avrora/

[12] Special Publication 800-57: Recommendation for Key Management on http://csrc.nist.gov/ CryptoToolkit/ kms/guideline-1-Jan03.pdf