Wireless Netw (2011) 17:1217–1234
DOI 10.1007/s11276-011-0343-x

Public-key based access control in sensornet

Haodong Wang · Bo Sheng ·
Chiu C. Tan · Qun Li

Published online: 24 April 2011

Abstract Symmetric cryptography has been mostly used in security schemes in sensor networks due to the concern that public key cryptography (PKC) is too expensive for sensor devices. While these schemes are efficient in processing time, they generally require complicated key management, which may introduce high memory and communication overhead. On the contrary, PKC-based schemes have simple and clean key management, but cost more computational time. The recent progress in PKC implementation, specially elliptic curve cryptography (ECC), on sensors motivates us to design a PKC-based security scheme and compare its performance with the symmetric-key counterparts. This paper proposes a practical PKC-based access control for sensor networks, which consists of pairwise key establishment, local access control, and remote access control. We have implemented both cryptographic primitives on commercial off-the-shelf sensor devices. Building the user access control as a case study, we show that PKC-based protocol is more advantageous than those built on symmetric cryptography in terms of the memory usage, message complexity, and security resilience. Meanwhile, our work also provides insights in integrating and designing PKC-based security protocols for sensor networks.

H. Wang (✉)
Department of Computer and Information Science,
Cleveland State University, Cleveland, OH 44115, USA
e-mail: hwang@cis.csuohio.edu

B. Sheng
Department of Computer Science, University of Massachusetts
Boston, Boston, MA 02125, USA
e-mail: shengbo@cs.umb.edu

C. C. Tan
Department of Computer and Information Science,
Temple University, Philadelphia, PA 19122, USA
e-mail: cctan@temple.edu

Q. Li
Department of Computer Science, College of William and Mary,
Williamsburg, VA 23187, USA
e-mail: liqun@cs.wm.edu

1 Introduction

A main challenge of large scale sensor networks is the deployment of a practical and robust security mechanism to mitigate the security risks exposed to the unattended and resource constrained sensor devices. Motivated by the fact of insufficient hardware resources, a great deal of research has focused on the symmetric-key-based solutions [4, 5, 7, 13, 32] for light-weight computation. These symmetric-key schemes, however, require complicated key management that may cause high memory and communication overhead. This drawback has not yet been investigated by experimental work so it is not clear how these schemes perform in a realistic system.

Recent progress in implementation of elliptic curve cryptography (ECC) on sensors [9, 12, 14] proves public key cryptography (PKC) is now feasible for resource constrained sensors. Given the efficient low-layer primitive in place, the high-layer PKC-based security scheme design in sensor networks, however, is not straightforward due to the special hardware characteristics and requirements of sensor networks. Therefore the performance of PKC-based security schemes is still not well investigated. This paper compares the symmetric cryptography and PKC-based schemes through an experimental study on an important sensor network security problem: user access control. Our results suggest the PKC-based user access control scheme is more advantageous in terms of the memory usage, message complexity, and security resilience.

Sensor data access control becomes an important security component as the in-network data storage applications [15, 31] have been proposed for the sensor platforms with cheap and large storage capacity. To protect the data, sensors have to authenticate the user, and control the access to their data. Existing access control schemes on the Internet [8, 16] are not feasible for sensor networks due to the limited power, memory and processing capabilities of the sensor hardware. Access control in sensor networks also differs from the conventional schemes in that it is not enough to simply deny unauthorized users' accesses to the data. An unauthorized user should not be allowed to use the network since the network bandwidth is very limited and, more importantly, the battery power of each node may be depleted after malicious users flood messages to the network.

The aforementioned special sensor hardware and network requirements motivate us to design the user access control scheme in a very different fashion. Our basic idea in this paper is to authenticate the user locally by the sensors in the user's vicinity and transfer the endorsements of the local sensors to the remote sensor for data query. In this way, unauthorized data access request will be rejected locally so that DoS attacks trying to deplete the battery power of the network will be blocked locally. The access control proposed in this paper is composed of several components. First, the sensors in proximity need to exchange pairwise keys for secure communications. Second, the user needs to get authenticated by the local sensors either for local sensor data access or for remote sensor data access. Third, the local sensors also need to help the user and the remote sensor build a pairwise key to achieve end-to-end security.

While existing symmetric key schemes [4, 5, 7, 13, 32] can achieve some of the security goals, several significant drawbacks such as high memory and communication overhead in key management, and security vulnerabilities, as we will show in our experimental study, make the symmetric cryptography bases solutions not desirable. We propose an ECC-based, practical and security resilient PKC-based user access control suite. Our approach not only embraces the cryptographic primitive tweaking to achieve the computation and communication efficiency, but composes of a carefully designed and novel threshold-endorsement protocol to address the issue of denial-of-service (DoS) in remote access control. We have implemented all protocols on widely used MICAz and TelosB motes. Our performance evaluation compares the proposed access control suite with prior work which are based on symmetric-key and the prevalent RSA on Internet through comprehensive experiments and rigorous analysis.

In summary, we make the following contributions in this paper. (1) We provide a detailed comparison of symmetric cryptography and PKC-based user access control protocols. Our evaluations are based on actual implementation on commercial off-the-shelf sensor hardware. To the best of our knowledge, this is the first experimental study on real world implementation for two fundamentally different cryptographic primitives. (2) We have designed a suite of ECC-based access control protocols including pairwise key sharing between neighboring sensors, local access control, and remote access control. We believe the integral security application sheds new insights into the practicality of the PKC-based scheme in sensor networks, helps to build a deeper understanding of the security protocol design in a resource constrained system. (3) We have implemented the public key primitive WM-ECC on MICAz and TelosB motes and HP iPAQ. Compared with the widely used and comprehensive TinyECC [12], our implementation gives more efforts in improving the performance on a specific curve.

2 Related work

The user authentication and communication encryption have received extensive attentions [8, 16, 18] for security in large network system. Kerberos [16] has been widely used in distributed client-server authentication and session key establishment. Fox et al. proposed a lightweight version of Kerberos, Charon [8], for mobile devices. Both schemes are centralized; a central server has to be on-line to assist user's request. In sensor networks, SPINS protocol [18] shares the same security architecture. While the centralized schemes have many attractive security features, the communication overhead becomes a major issue when the network size scales, specially for the extremely energy constrained sensor nodes in a large network. For the same reason, the security schemes [21, 35] relying on a central server are not desired in the security mechanism design in large-scale sensor networks.

A number of key establishment schemes based on predistribution have been explored recently [4, 5, 6, 7, 13, 25]. While these symmetric-key-based schemes are computationally efficient, the trade-off has to be paid for complicated key pre-distribution and key management, which incur large communication overhead during the key discovery. In their recent work [19], Di Pietro et al. proposed an interesting key discovery scheme based on pseudorandom key pre-deployment. The scheme leverages the hash computation to find the matching keys so that the communications for key index exchanges in [7] can be avoided. Unfortunately, this mechanism exposes the security vulnerability in node compromising if the adversary launches a "smart attack". The PKC-based pairwise key schemes proposed by Zhang et al. [33, 34] achieve some nice security features by using ID-based cryptography. Although the schemes are very novel, the ID-based cryptography is still not feasible for resource constrained sensors. Wang et al. [28] also discusses user access control scheme by the public key scheme. However, this scheme is just a simple PKC-based one hop authentication and does not provide a comprehensive user access control solution.

The most related research to the user access control is [26, 27, 32]. Zhang et al. [32] proposes several schemes to restrict and revoke the access privilege of a mobile sink. The limitation of the scheme is that the mobile sink's moving track has to be predetermined by the base station. Our scheme, however, addresses a more general user/sensor communication problem. The mobile sink can be regarded as one type of special users in our scheme. Although [26] describes a symmetric-key-based local endorsement scheme which is similar to the threshold endorsement in this paper, the symmetric-key scheme suffers larger communication overhead and requires prohibitive amount of memory storage space. The message authentication scheme in [27] enables the relay nodes to filter the illegal messages (injected by the adversary) and prevent the DoS attacks, but cannot determine how much access privilege a legitimate user has. It is complementary to the remote access control scheme proposed in this paper.

Precursors in ECC implementation on Berkeley Motes include Sun Research Labs [9], EccM [14] and TinyECC [12]. TinyECC is the first public available and widely used efficient ECC implementation on primary field on Berkeley Motes. It is a full package implementing 128, 160, and 192-bit in total six SECG/NIST recommended curves on three generic sensor platforms (MICAz, TelosB, and Imote). Recently, TinyECC further implements flexible user library reconfigurability that allows users to have the choice to pick one or more desired optimization techniques for their own applications. TinyECC is very comprehensive, and its performance is well tuned. Our WM-ECC focuses solely on curve secp160r1 with many available optimization techniques. Due to this special effort, our WM-ECC on this curve is more efficient than TinyECC.

3 System model and assumptions

We consider a large scale wireless sensor network deployed in a variety of environments. Data access to the stored data on each node is protected according to the attributes of the data. The examples include data type (temperature, light, or noise), data location, and data collection time. A user equipped with a portable computing device, such as a PDA, interacts with the sensor network for data query and retrieval. This device is more powerful than the sensor nodes, so it is capable of more computationally intensive tasks. User can query either "local" sensors through direct communication links, or "remote" sensors (that are outside of direct communication range) through multihop routing by intermediate sensor nodes.

We assume a certification authority (CA) is responsible for generating all security credentials for sensors. During the deployment, each sensor is pre-loaded its private key, public key and the corresponding certificate. The user acquires his certificate from the CA through an out-of-band security channel. The certificate includes an access control list which defines his access privilege. To query the sensor network, the user needs to attach the certificate with the query message. The contacted sensor checks the access list and verifies the user's privilege. The verification is performed in a distributed fashion without involvement of the CA. The contacted sensor grants the user the answers that are compliant with the access privilege. If the users cannot be verified, the query will be denied.

The adversary may launch either passive attacks or active attacks, or both. The passive attack includes message eavesdropping, traffic monitoring and analysis. For active attack, we mainly focus on following three types. The first is node compromise. The compromised sensor may capture the legitimate user information while being accessed and reveal it to the malicious third party. Second, user collusion can help malicious users to subvert the system and gain more access privilege. Third, the adversary may inundate user queries in the network to deplete the battery power of sensor nodes. We assume that at most $t - 1$ sensors (where t is a security parameter) can be compromised and an unbounded number of users can collude since it is not hard for mischievous users to share information and orchestrate an aggregated analysis to the collected information.

In this paper, we do not address disruption attacks. Disruptions occur when the adversary, by compromising a sensor node, drops legitimate messages or contributes a bogus endorsement share in remote access control (as we will describe later) to invalidate user remote queries. Even though disruption attacks in general are difficult to defend against in sensor networks, a smart adversary is not willing to launch such attacks because incidents of message dropping and user remote access failure may easily trigger system attentions and thus expose the compromised sensors.

4 Pairwise key and local access control

We start the discussion with the secure pairwise key establishment. A lot of sensing tasks (e.g., event detection and user remote access endorsement described in Sect. 5) are achieved through collaborations of multiple neighboring sensors, which require secure peer-to-peer communication to prevent the adversary from eavesdropping. For the same reason, the secure communication channel is also desired when a user queries sensors.

In this section, we design an ECC-based pairwise key establishment scheme for neighboring sensors. A common way to share a secret between two parties is to use Diffie-Hellman (DH) scheme. However, DH cannot be directly used in sensor networks due to the potential Man-In-The-Middle (MITM) attack. We thus develop our key establishment scheme based on ephemeral DH protocol over elliptic curve. We tweak the original DH protocol to defend against MITM attacks. As we will explain later, our scheme is to achieve the best communication and computation efficiency. This PKC-based pairwise key scheme can also be applied for local user access control with a slight modification. We give the brief security and cost analysis in the end.

4.1 Pairwise key establishment between two sensor nodes

We assume the system certification authority (CA) selects an elliptic curve E over the finite field $GF(p)$, where p is a large prime number. We denote P as the base point of E, where P has the order of q (q is a prime number too). CA keeps a system secret x, and publishes the system public key $Q = xP$. We will continue to use this cryptosystem setup throughout this paper.

Similar to the conventional PKI, sensors' public keys need to be certified. Since it is not realistic to have an online CA that can verify the public key in real time, each sensor has to pre-load its certificate that is pre-computed by CA. We first discuss how to generate the private key, the public key and the certificate for each sensor. We first define two one-way cryptographic hash functions, $h_1 : \{0,1\}^* \mapsto [0, q-1]$, $h_2 : \{0,1\}^* \mapsto \{0,1\}^l$, where l is the pairwise key length. Let us consider a sensor node u (we denote u as the sensor ID). CA first selects a random number c_u, generates its certificate $C_u = c_u P$, and calculates $e_u = h_1(u \| C_u)$. The private key of u is derived as $q_u = e_u c_u + x$, and the corresponding public key is $Q_u = q_u P$. Note C_u, q_u and Q_u satisfy the following property:

$$Q_u = e_u C_u + Q. \tag{1}$$

The function of e_u is to bind sensor ID, u, with its certificate, C_u, so that the sensor cannot claim itself as another ID v. As we will explain later, e_u can be utilized to bind a user's access control list with her certificate.

Before the deployment, sensor u is pre-loaded with q_u, Q_u and C_u. Considering a typical 160-bit elliptic curve, these credentials require 100 bytes of memory space.

For two sensors u and v with (q_u, Q_u, C_u) and (q_v, Q_v, C_v) respectively, the ECC-based pairwise key establishment protocol is illustrated in Fig. 1. We denote $(.)^+{}_k$ as a symmetric key encyrption operation by using key k, and denote $(.)^-{}_k$ as a symmetric key decryption operation by using key k. The symmetric key scheme can be any existing scheme, such as AES. Sensor u sends v the key establishment request, which includes u's public key Q_u, certificate C_u and nonce n_0. Sensor v calculates $e_u = h_1(u \| C_u)$, and verifies u's public key by plugging e_u and C_u in Eq. 1. The request will be immediately dropped if the derived public key does not match Q_u. If the verification is successful, v picks a random r_v and generates the challenge in the following steps:

1. v multiplies r_v with Q_u to get a secret ECC point R_v.
2. The hash value of R_v, denoted as $h_2(R_v)$, is used to encrypt the randomly picked secret key, k_v. The hash of n_0, denoted as n_1, forms a nonce chain to defeat the potential security attacks.
3. v computes Y_v by multiplying r_v with the base point P.

Upon receiving ξ_v and Y_v, u can recover k_v because $q_u Y_v = q_u \cdot r_v P = r_v Q_u = R_v$, which is used to encrypt k_v and n_1 by v. After the decryption, u verifies n_1 and continues the execution of the protocol if n_1 is correct. Otherwise, u exits the protocol immediately.

In addition to the challenge generation, v also sends its public key Q_v and the certificate C_v to u. The same verification and challenge are performed by u. Finally, u and v agree on their pairwise key $k_{uv} = k_u \oplus k_v$.

The protocol presented in Fig. 1 is a general purpose scheme which provides security resilience even in an extremely adverse environment. Considering the fact that most pairwise key establishment happens in the network initialization period and, many times, this period of time can be considered active security attack free (e.g., there is no compromise and Man-In-The-Middle attack), the following two optimizations can be applied to achieve better

(1) $u \to v : Q_u||C_u||n_0$

(2) $v :$ verifies Q_u, picks random number $r_v \in [1, q-1], k_v \in \{0,1\}^l$

(3) $v : R_v = r_v Q_u,\ \ Y_v = r_v P,\ \ n_1 = h_2(n_0),\ \ \xi_v = (k_v||n_1)^+_{h_2(R_v)}$

(4) $v \to u : \xi_v||Y_v||Q_v||C_v$

(5) $u : k_v||n_1 = (\xi_v)^-_{h_2(q_u Y_v)}$, verify $n_1 = h_2(n_0)$

(6) $u :$ verifies Q_v, picks random number $r_u \in [1, q-1], k_u \in \{0,1\}^l$

(7) $u : R_u = r_u Q_v,\ Y_u = r_u P,\ n_2 = h_2(n_1),\ \xi_u = (k_u||n_2)^+_{h_2(R_u)}$

(8) $u \to v : \xi_u||Y_u$

(9) $v : k_u||n_2 = (\xi_u)^-_{h_2(q_v Y_u)}$, verify $n_2 = h_2(n_1)$

(10) $u, v :$ agree on $k_{uv} = k_u \oplus k_v$

Fig. 1 ECC-Cert: ECC-based pairwise key establishment scheme between two neighboring sensors: u and v

efficiency. First, since all sensor nodes are honest at that time, the verification of public key is not required. Second, the challenge is only required on one direction when two neighboring sensors try to establish the key. As the result, step (6), (7), (8) and (9) in Fig. 1 are not required in the optimized scheme, and there is not necessary either to verify the public key in step (2).

The optimized scheme requires only two ECC point multiplications compared to three in the general scheme. The further optimization is possible if the sensors have additional storage space. The idea is to select a set of random number $\{r_v\}$, pre-compute the corresponding points, $\{Y_v = r_v P\}$, and store them in the flash memory. When v receives the request, it randomly pick one entry (r_v, Y_v) and immediately sends Y_v to u, and then computes the challenge. In this way, the two ECC point multiplications, $R_v = r_v Q_u$ at v and $R_v = q_u Y_v$ at u can be computed simultaneously. As the result, the processing overhead of pairwise key establishment reduces to only one ECC point multiplication. After the pairwise key is established, v erases the selected (r_v, Y_v) from the storage so that the same random number/point will not be used again. Note the optimized protocol is resilient to passive security attacks. The traffic analysis (if the adversary monitors all network activities) does not reveal any pairwise key secret.

From now on, we denote "ECC-Cert", "ECC-NoCert", "ECC-PreComp" as the general purpose scheme, the optimized scheme and the optimized scheme with pre-computation, respectively, throughout the rest of paper. "ECC-Cert" also can be directly applied for one-hop user access authentication. In that case, the user, say Alice, has to give her access list al_A and certificate C_A, where al_A composes of the user id and the corresponding privilege mask. The contacted sensor builds Alice's public key based on al_A and C_A, and then perform the rest of authentication protocol.

4.2 Cost analysis

The cost of the pairwise key establishment and local access control is determined by the communication and the computation overhead. The communication overhead can be measured by the message complexity. "ECC-Cert" shows u has to send three elliptic curve points (Q_u, C_u and Y_u) and one scalar value (ξ_u). Given a 160-bit ECC cryptosystem, each point has the size of 40 bytes, and each scalar value has the size of 20 bytes. Therefore, u and v have to transmit 140 byte data, and the message complexity for u and v is 280 bytes. Comparatively, in "ECC-NoCert" and "ECC-PreComp", neither sensor needs to send the certificate, then the message complexity reduces to 100 bytes.

In ECC, the point multiplication is much more expensive than other operations, we approximately estimate the computational cost by counting the number of point multiplications. As shown in Fig. 1, "ECC-Cert" requires three point multiplications. Comparatively, "ECC-NoCert" and "ECC-PreComp" only require two and one point multiplication, respectively. In the local access control, with the optimization of pre-computation, the sensor has the similar message overhead as in "ECC-Cert", but has less computation overhead because the pre-computation saves one point multiplication.

4.3 Security analysis

In the security analysis, we consider the following potential threats that an adversary may employ to defeat the proposed challenge-response pairwise key establishment and the local access control protocols.

- **Impersonation**. Suppose an adversary forges an identity w and the corresponding public key Q_w and certificate C_w. Note any one can generate his public key and the certificate by using system public key Q, but no one can derive his private key q_w without the system secret x. It is computationally infeasible to compute his private key q_w without the system secret x. To get q_w from Q_w is equivalent to solve the discrete logarithm problem. Without q_w, the adversary cannot correctly respond the challenge so that the pairwise key request or local query will be immediately rejected by a legitimate sensor. For the same reason, the adversary cannot impersonate the legitimate sensors and users even if he can capture Q_u, C_u, Q_v, C_v in step (1) and (4) in the pairwise key establishment protocol shown in Fig. 1.
- **Replay attack**. The since the chained nonces are used in the protocol in Fig. 1, any replayed message except in step (1) will be dropped immediately. The adversary cannot gain any advantage by replaying the message in step (1) because there is no way to respond the challenge without the corresponding private key.
- **Interleaving**. In the interleaving attack, the adversary selectively combines the messages information from previous or parallel sessions. Due to the challenge-response nature of the protocol, the adversary cannot impersonate or deceive the sensor in the interleaving attack. The reason is that the sensor ID or the user access list is bind with the certificate, so the private key is required to correctly respond the two-way challenge. In addition, the chained nonces allow the legitimate sensors immediately drop the messages from other sessions.

- **Reflection**. A reflection attack is that an adversary sends the identical message back to the message originator for the impersonation purpose. As we explained above, the adversary cannot correctly respond the challenge generated by a legitimate sensor, the attack attempt will fail. Further, a sensor can easily drop a reflected message once the wrong nonces are detected.
- **Forced delay**. The adversary may also block the message between two legitimate parties and resend it in a later time, which is so called the forced delay attack. Obviously, our challenge-response protocol is immured to this attack. The only effect of this attack is to force the two parties to drop the protocol session, assuming the time-out mechanism has been employed in both parties.
- **Chosen-text attack**. In the chosen-text attack, an adversary tries the strategically arranged challenges and tries to extract the other parties private key. As indicated in our protocol, the sensor always uses a emphemeral random number, r_u and r_v, it is impractical for the adversary to compute the private key of a legitimate sensor.

5 Remote access control

Theoretically, a simple extension of the certificate-based local authentication scheme can be used in the remote query. In that case, the challenge-response messages between the user and the remote sensor are routed by a number of intermediate sensors on the routing path. This multi-hop communication pattern, however, poses new security and efficiency issues: (1) potential DoS attacks; (2) high communication overhead for the user authentication and end-to-end security. The two issues are not found in the local query and can not be addressed by the certificate-based scheme due to the following two reasons.

First, because the certificate-based access control achieves end-to-end security, any intermediate sensor has no knowledge about the challenge-response message and cannot detect the DoS attack had the adversary injected a large number of fake queries.

Second, the message overhead becomes critical in the multi-hop communication to reduce the energy consumption of intermediate sensors. The certificate-based scheme requires public key exchanges between two parities. In practice, the public key size (40 bytes) is larger than the typical message size in sensor networks (29 bytes). This overhead may force the sensor to use multiple data packets to transmit the query that otherwise would be done by just one packet. While the certificate-based scheme achieves the user authentication and end-to-end security, it requires two rounds of communications that carry the public keys and incurs the large overhead.

Therefore, we develop a threshold endorsement scheme (inspired by Shamir's secret sharing [22]) to perform the remote access control. The basic idea is that any user has to be authenticated and endorsed by t local sensors before she can send the remote query. Not only do the t local sensors block any DoS attack attempt and transfer the trust (of the authenticated user) to the remote sensor, given the assumption that the adversary can not compromise t sensors, their endorsements also naturally serve as the pairwise key between the user and the remote sensor without any public key transmission. The three components: DoS prevention, user authentication, and message security are integrated organically in the remote access control scheme.

Our scheme is presented as follows. Again, we have an elliptic curve E over finite field $GF(p)$ and a base point P with the order of a prime q. CA maintains a secret polynomial:

$$f(y) = 1 + a_1 y + \cdots + a_t y^t, \tag{2}$$

where $a_i \in GF(q)$ for $1 \le i \le t$. Note that this secret polynomial is slightly different from the one in Shamir's secret sharing scheme in that the term a_0 is equal to 1, which implies one share of this polynomial, namely the share (0, 1), is already known. As the result, only t shares of this polynomial, instead of $t+1$ shares, are enough to reconstruct the polynomial because the known share can serve as the $(t+1)$th share. Therefore, to prevent the secret polynomial from being revealed, the number of malicious sensors must not be more than $t-1$. In this paper, we assume only up to $t-1$ sensors can be compromised.

Before the deployment, each sensor s_i (s_i denotes the sensor ID) is pre-loaded with a secret share z_i, where $z_i = f(s_i)$. Any $t+1$ shares from $t+1$ sensors, without the known share, can reconstruct the secret polynomial by Lagrange interpolation:

$$f(y) = \sum_{i=1}^{t+1} z_i \prod_{j=1, j\neq i}^{t+1} \frac{s_j - y}{s_j - s_i}. \tag{3}$$

When $y = 0$, the $t+1$ secret shares satisfy:

$$\sum_{i=1}^{t+1} z_i l_i = 1, \tag{4}$$

where l_i is the Lagrange coefficient, and determined as $l_i = \prod_{j=1, j\neq i}^{t+1} \frac{s_j}{s_j - s_i}$. It is true that any t shares, $z_1, \ldots, z_t$, plus the known (0,1) share, also satisfy the above equation with different Lagrange coefficients. However, the known share is not the interest of our remote access control scheme, and we focus on the $t+1$ shares from the sensors in the following discussion.

Certification authority also defines a cryptographic hash function $\overline{H}$, mapping a number $\{0,1\}^*$ to a nonzero elliptic curve point on E. The remote access control protocol is given in Fig. 2. We denote $s_1, s_2, \ldots, s_t$ as the local sensors, s_r as the remote sensor. We assume that the ID of the remote sensor for data access is known by some scheme that is beyond the scope of this paper, e.g., resource discovery protocols.

The user, Alice, first performs local access control protocol with t local sensors, $s_1, \ldots, s_t$. After the successful authentications, each local sensor s_i endorses Alice in the following way. First, s_i calculates $R_A = \overline{H}(al_A)$. Note R_A is a point on the elliptic curve E. Then s_i generates its endorsement: $z_i l_i R_A$, where the Lagrange coefficient $l_i = \prod_{j=1, j\neq i}^{t} \frac{s_j}{s_j - s_i} \cdot \frac{s_r}{s_r - s_i}$ (here we use s_r instead of s_{t+1}). In the next step, s_i sends the endorsement to Alice through the secure communication channel established in the local access control as described in Sect. 4 With the t endorsements collected, Alice calculates the elliptic point V_A, which is the summation of the t endorsements. Note only Alice knows the value of V_A. None of t local sensor knows V_A because each sensor only knows its own share of V_A. Now, V_A becomes the shared secret between Alice and the remote sensor s_r. Alice encrypts her access list and query by using $h_2(V_A)$, and then sends the encrypted query along with her access list and l_r ($l_r = \prod_{j=1}^{t} \frac{s_j}{s_j - s_r}$, also calculated by Alice) to the remote sensor s_r. Upon the receipt of the remote access request from Alice, s_r first calculates $R_A = \overline{H}(al_A)$ and computes $V'_A = R_A - z_r l_r R_A$. According to (4), V'_A should be equivalent to V_A because:

$$\sum_{i=1}^{t} z_i l_i R_A + z_r l_r R_A = \left(\sum_{i=1}^{t} z_i l_i + z_r l_r\right) \cdot R_A = R_A. \tag{5}$$

Therefore, s_r can successfully decrypt al_A and *query*. Finally, s_r replies *Alice* with the query result, again encrypted by $h_2(V_A)$.

```
$Alice \rightarrow s_1, \cdots, s_t : al_A || C_A$
for (each sensor $s_i, i = 1, 2, \cdots, t$)
    $s_i$ : perform user authentication
    $s_i$ : compute $R_A = \overline{H}(al_A)$
    $s_i \rightarrow Alice : z_i l_i R_A$
$Alice$ : gets $V_A = \sum_{i=1}^{t} z_i l_i R_A$
$Alice \rightarrow s_r : al_A || l_r || (al_A || query)^+_{h_2(V_A)}$
$s_r$ : compute $R_A = \overline{H}(al_A), V'_A = R_A - z_r l_r R_A$
$s_r : al_A = ((al_A || query)^+_{h_2(V_A)})^-_{h_2(V'_A)}$
$s_r \rightarrow Alice : (reply)^+_{h_2(V_A)}$
```

Fig. 2 ECC-based local threshold endorsement scheme to establish remote pairwise key between the user and the remote sensor

In summary, the main idea of remote access control is to design a mechanism that allows a set of local sensors (because we do not trust a single sensor) to transfer the trust (if the user is authenticated) to the remote sensor, so that the remote sensor does not need to perform the interactive user authentication employed in local authentication, which requires several rounds of communications. This endorsement scheme can be combined with existing en-route filtering schemes, such as SEF [30] and IHA [36], to further prevent the adversary from injecting the data queries through a compromised sensor.

Our scheme can also be extended to work in a sparse network, where t local sensors are difficult to find at one time. In that case, the user moves around and finds t sensors at different locations. All these t sensors perform the same location authentication as described. To produce the endorsement shares, t sensors need to communicate with each other and exchange their ID list and agree on the remote sensor s_r. Note the communications cannot be initiated by sensor themselves since multi-hop communications have to be endorsed as we described previously. For this reason, the user moves back and force, as a carrier, to distribute the node IDs to each of t sensors. Once t sensors share their IDs and agree on s_r, the rest of scheme is the same as described previously.

5.1 Cost analysis

To endorse the user, each local sensor only needs to perform one ECC point multiplication and one hash function $\overline{H}.\overline{H}$ is a special hash function that maps $\{0, 1\}^*$ onto the elliptic curve E. According the study by Boneh et al. [2], this special hash function can be efficiently achieved by two steps: first we hash onto a certain subset $F \subseteq \{0,1\}^*$; then we use a deterministic encoding function to map F onto E. The message complexity for the threshold local endorsement is small. Each sensor only needs to send an elliptic curve point to the user, which has the message size of 40-bytes (for the 160-bit ECC).

5.2 Security analysis

The proposed remote access control scheme is resilient to any sensor compromising attack with no more than $t - 1$ compromised sensors due to the property of the threshold cryptography. Each sensor s_i has its own unique secret z_i. Any $t - 1$ or less shares of secrets are not enough to recover the secret polynomial [22], and cannot be utilized to deduce the value of z_r hold by the remote sensor.

As described in the protocol, the user knows each share of endorsement: $z_i l_i R_A$, and even $z_r l_r R_A$. Combining all these shares only allows the user to establish shared secret with the remote sensor. These shares can not be used to generate the endorsement for any other access list. Suppose the user has a forged access list al'_A, and the corresponding $R'_A = \overline{H}(al'_A)$. To generate the endorsement shares $z_i R'_A$ $(1 \leq i \leq t)$, the user has to know z_i. However, it is computationally infeasible to retrieve z_i from $z_i l_i R_A$. Meanwhile, the knowledge of $z_i l_i R_A$ cannot be used to derive $z_i l_i R'_A$. The reason is that R_A, R'_A are random elliptic curve points, it is computational infeasible to derive $r_A, r'_A \in GF(q)$, so that $R_A = r_A P$ and $R'_A = r'_A P$. As the result, it is impractical to derive $z_i l_i R'_A$ from $z_i l_i R_A$. For the same reason, the user cannot reuse the acquired secret endorsement to access a different remote sensor.

Since each endorsing sensor establishes a secure communication channel with the user during the local authentication, the adversary cannot capture any share of the endorsement by eavesdropping. Therefore, only the user and the remote sensor share the secret, which is to build the secure communication channel for the remote access.

Finally, we specifically discuss the following potential attacks for the impersonation attempt. Since the first part of the protocol is the user local access control that is executed between the user and the local endorsing sensors, our analysis mainly focuses on the second part, which is between the user and remote sensor.

- **Impersonation**. Since the user has to be authenticated by a group of local sensors before he can access the remote sensor, the impersonation attack is easily defended by the local screening. When the user himself is malicious, the impersonation can be in a different form that the user forges his access list after he is authenticated by the local sensors. However, as we have explained above, the malicious user cannot decrypt the reply from the remote sensor because he does not possess the private key associated to his forged access list.
- **Replay attack**. The remote query answer replied by the remote sensor is encrypted by the secret key, the adversary cannot capture any information through the replay attack. The remote access control protocol, shown in Fig. 2, can be easily modified by including a nonce to allow the remote sensor to detect the reply attack.
- **Interleaving**. There is only one round communication between the user and the remote sensor. The remote sensor receives the query, and then encrypts the reply by using the constructed pairwise key. Without the pairwise key, which is jointly constructed by the local endorsing sensors, the adversary cannot decrypt the reply.
- **Reflection**. The reflection attack cannot be a threat because the protocol between the user and the remote sensor is not a challenge-response authentication. The user cannot understand the remote access request sent by himself, and neither can the remote sensor understand the reply message.
- **Forced delay**. The adversary cannot gain anything from the forced delay attack. As we explained, the reply message from the remote sensor is encrypted.
- **Chosen-text attack**. Our protocol can be easily improved to defeat the chosen-text attack by including a random number, e.g., nonce, in the remote query access request from the user and the reply message from the remote sensor.

6 System implementation

We implement our ECC-based user access control on MICAz motes [10], the most recent MICA family motes from UC Berkeley. MICAz is powered by a ATmega128 microcontroller, which features an 8MHz, 8-bit RISC CPU, 128K bytes flash memory (ROM) and 4K RAM. The MICAz runs TinyOS [24] version 1.1.15.

6.1 WM-ECC implementation

Because of the resource stringency on tiny sensor motes, the implementation of the computationally expensive public key primitive is a major challenge. We first study the feasibility of two public key primitives: RSA and ECC. We have implemented the 1,024-bit (key size) RSA and the 160-bit (key size) ECC security primitive (WM-ECC) on MICAz motes. These two cryptosystems are comparable because they offer the same security level [17]. Note that ECC offers more security per bit than RSA, which is a very attractive feature for wireless communications. We summarize the performance comparison in Table 1. We find RSA is much more expensive than ECC in terms of computation time and memory overhead. RSA signature generation takes 21.5 s, more than 16 times slower than 1.35 s of ECC signature. Even though, in a special case when the public key size is only 17 bits (NIST suggested smallest public exponent), the RSA public key operation can be done in 0.79 s, the slower private key operation is

Table 1 The performance comparison between 1,024-bit RSA and 160-bit ECC on MICAz motes

	Key size (bytes)	Sign (s)	Verify (s)
RSA	128	21.5	21.5 (0.79s)
WM-ECC	20	1.35	1.96

In a special case, when RSA public key is only 17 bits, the verification can be finished in 0.79 s prohibitive in many applicaitons, including the remote access control scheme we have proposed. Furthermore, RSA key size is much larger than ECC. The RSA private key size is 128 bytes, more than 6 times larger than ECC private key size. Combining the above two factors, we believe ECC is a better public key system for sensor networks.

We choose a SECG recommended elliptic curve, secp160r1, in our WM-ECC implementation. Due to the resource constraint, the standard ECC implementation can not be directly ported to MICAz motes. The extensive optimizations have to be performed to allow the ECC primitive to be fitted in the limited memory space, and to achieve practical processing time for doing ECC exponentiations. Therefore, we have implemented the customized ECC primitive exclusively for MICAz motes. The implementation has been done in TinyOS environment with a combination of NesC and assembly languages. We specifically target to improve on the performance of the most frequently called modules of long integer multiplication, long integer division, and modular reduction. Our implementation comprises of a number of techniques for optimization, including hybrid multiplication [9] for faster multi-precision multiplication, Great Division [23] for efficient modular inversion, and modular reduction optimizing for Pseudo-Mersenne numbers. To precisely manage the limited CPU resources (registers), we have implemented these key modules in AVR assembly language. Due to the space limit, we omit the detail description of the above optimization and refer the interested reader to [29] for the details. Our experiments verify that the optimizations are very effective.

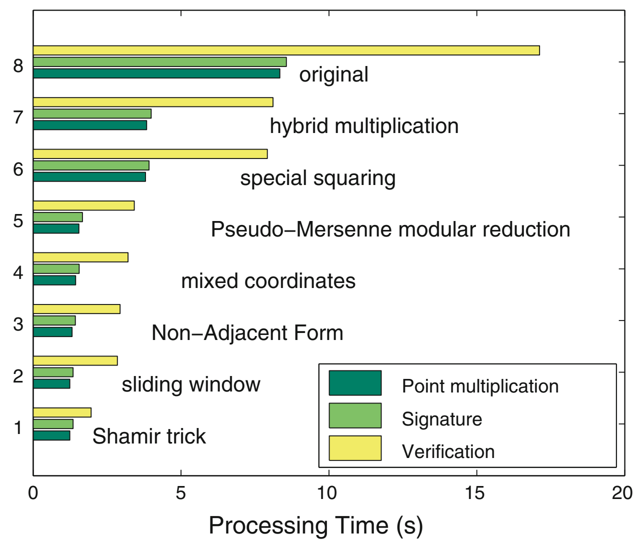


Fig. 3 Running time for WM-ECC operations (point multiplication, ECC signature, and ECC verification) by applying various optimization techniques

Figure 3 shows the performance of our ECC implementation on MICAz by applying the optimizations consecutively. After applying the hybrid multiplication, the multi-precision multiplication, including squaring, speeds up for 7 times. As the result, the overall running time reduces to more than half. The adoption of Pseudo-Mersenne modular reduction achieves more than 10 time speed-up for modular operations, and accordingly improves the overall performance by further 60%. The mixed coordinates, which are used to reduce the number of slower inversion operations, contributes 6% improvement. We further convert the bit-string of the multiplicand to a Non-Adjacent Form to reduce the number of ECC point addition. This effort also contributes 5% performance enhancement. In the sliding-window technique, we pre-compute and store several repeatedly used intermediate values to gain 10% performance improvement. Finally, by using the Shamir's trick that uses the similar pre-computation technique as the sliding-window scheme, we further reduce the running time by 30%.

6.2 User module and other components

Our user module is composed of two parts. We choose an HP iPAQ pocket PC as the user computing module to perform all backend computations. The HP iPAQ features a 522MHz ARM920T PXA270 processor, 64MB RAM and 128MB flash memory. The HP iPAQ is powered by Microsoft Windows Mobile 5.0. Since the iPAQ wireless communication module is not compatible with IEEE 802.15.4/ZigBee on MICAz, we use a MICAz sensor mote to bridge the communication between the user and the sensor motes. The MICAz mote is responsible for communications with the sensor motes in the network. All the data processing is performed at the iPAQ. The two parts communicate through a USB cable.

We implement the same ECC primitive on the iPAQ. Given the powerful processor and plenty of memory, the ECC performance on iPAQ was expected to be much faster. To our surprise, the initial test showed the ECC point multiplication still costs 200 ms, only 6 times faster than MICAz with a more than 60 times faster CPU. The further investigation reveals that C compiler for the mobile devices has poor optimization capability, so that the multi-precision integer operation is not optimized. Therefore, we again re-write the critical components in ARM assembly language. The judicious decision improves the performance from 200 ms down to 40 ms. We summarize the ECC performance results on both platforms in Table 2.

For the hash function, we adopt SHA-1 160-bit implementation from a standard crypto library. For MAC (Message Authentication Code) module, we adopt the RC5 block cipher from TinySec [11]. Both hash and MAC modules are computationally efficient. It takes several

Table 2 The comparison of ECC execution time on various platforms, including MICAz, HP iPAQ and TelosB, for ECC point multiplication (PM), signature generation (Sign), signature verification (Verify) time

	PrivKey (bytes)	PubKey (bytes)	PM	Sign	Verify
MICAz	20	40	1.24 s	1.35 s	1.96 s
iPAQ	20	40	40 ms	48 ms	68 ms
TelosB	20	40	1.44 s	1.60 s	2.26 s

mili-seconds to do a hash operation. The RC5 encryption and decryption take less than 1 ms.

6.3 Other sensor platforms

Our ECC-based access control schemes are not only practical for MICAz motes, but can be deployed on other sensor platforms. We have successfully ported our whole software suite to TelosB motes, the latest research oriented motes developed by UC Berkeley. TelosB is powered by MSP430 microcontroller. MSP430 incorporates an 8 MHz, 16-bit RISC CPU, 48 KB flash memory (ROM) and 10K RAM. The RF transceiver on TelosB is IEEE 802.15.4/ZigBee compliant, the same as on MICAz. Therefore, TelosB and MICAz motes can be mixed together to form a heterogeneous sensor network.

Since TelosB mote has a different hardware architecture, all hardware dependent security primitives have to be re-written for TelosB. We adopt the same optimization techniques explained previously, and find ECC is also practical for TelosB motes. The ECC performance on TelosB is shown in Table 2. Overall, ECC operation on TelosB is only SLIghtly slower than that on MICAz.

7 Analysis and evaluation

We evaluate our access control schemes using a combination of theoretical analysis and actual implementation on a sensor platform. The symmetric key schemes compared are: Random-key [7], PIKE [4], Blom [6], and Blundo [32]. Random-key, PIKE and Blom are used to compare the performance of our public key solution when performing pairwise key establishment. Note that of the different symmetric schemes considered, only Blundo is explicitly designed for access control. Thus, we only compare our local access control solution with Blundo.

The metrics used to compare pairwise key establishment are **memory overhead**, **message complexity** and **security resilience**. Since all symmetric-key-based key establishing schemes require key pre-distribution, the memory overhead measures the amount of memory space required for each sensor to achieve a certain degree of key connectivity with its neighboring nodes. The more keys pre-distributed, the higher key connectivity can be achieved. The message complexity measures the amount of communications required for a certain sensor node to establish pairwise keys with its neighboring sensors. In security resilience against the node compromise, we measure the fraction of the compromised communication links as a result of sensor compromise. The communication links here are the direct communication links between any two neighboring sensors.

We implement Random-key scheme and Blundo user access control scheme as the real world comparison. We use the following four metrics: **key establishing time**, **memory overhead**, **message complexity** and **energy consumption**. The key establishing time measures the time duration for a random sensor to establish secret pairwise key with its neighbors. Similarly, the memory overhead measures the exact amount of data space required (in the real implementation) in the access control. The message complexity then shows the amount of messages transmitted during the key establishing procedure. The energy consumption estimates the average communication energy consumed during the key establishment.

Finally, we implement all components in the proposed remote access control. By focusing on the processing delay, we demonstrate the delay is small, which makes our scheme practical in the real world.

7.1 Analytical results

7.1.1 Pairwise key

Random-key [7] can be considered as a base line pairwise key establishment protocol. Each sensor is randomly pre-distributed with a number of secret keys from a system key pool. Any two neighboring sensors try to find a common key to establish a pairwise key by exchanging the key indices. Note, Random-Key is included in the evaluations for completeness, even though it is not particularly well-suited for the scenario of interest. Blom [6] is a variation of Random-key scheme. Instead of pre-distributing random keys, Blom pre-distributes secret vectors from the key spaces (or matrices) maintained by the system. Any two sensors having the vectors from a common key space can establish the pairwise key. PIKE [4] (we only consider PIKE-2D in this paper) is different from the above two schemes in that each sensor, identified by a two-dimensional ID, is pre-distributed at least one common secret key with determined $2 \times \sqrt{N}$ sensors (where N is the number of sensors in the network), which have either the same row-ID or column-ID. Any two sensors establish the pairwise key through the sensor that has the same row-ID with one of the two sensors and the same column-ID with the other.

We provide three variations of our ECC-based pairwise key schemes: ECC-Cert, ECC-NoCert, and ECC-PreComp, which were discussed in Sect. 4.

Our analysis is based on a randomly, uniformly deployed sensor network with 10,000 nodes. On average, each sensor has 20 neighbors. The above parameters are selected according to [1] so that the sensor network can be connected with a probability greater than 99%. The senor node IDs have the size of 2 bytes. The random keys have the size of 10 bytes. With additional 2 bytes for key indices, each pre-distributed random key requires 12 bytes for memory space. We assume the key pool size is 10,000 for both Random-key and PIKE. We choose 160-bit ECC as our public key primitive. Accordingly, an ECC certificate has 40 bytes, an ECC public key has 40 bytes, and an ECC private key has 20 bytes.

The ability to establish a direct pairwise key (not through the third party) between two neighboring sensors is very important, since direct key sharing not only reduces the communication overhead, more importantly, also improves the security resilience. Figure 4a shows the memory overhead required by the key establishing schemes to achieve a direct key between two sensors with different probability.

To increase the probability of establishing direct pairwise keys, Random-key scheme needs to pre-distribute more keys in each sensor node. We can see from Fig. 4a, the memory overhead is increasing linearly when the required key connectivity increases from 0.1 to 0.9. This trend becomes exponential when the connectivity is larger than 0.9. To achieve 100% connectivity, each sensor has to be pre-loaded with 300 keys, which requires 3.6KB memory space. Considering MICAz only has 4KB data space, the 300 keys almost use up all available memory and leave almost no space for the application programs. Thus, the Random-key scheme obviously is not practical to achieve 100% direct key connectivity.

Compared with Random-key scheme, the memory overhead of PIKE only depends on the network size. Given 10,000 sensor nodes, each sensor has to be pre-loaded with $2 \times (\sqrt{10,000} - 1) = 198$ keys. Therefore, the memory overhead for PIKE is constantly $12 \times 198 = 2{,}376$ bytes. Blom scheme with $\lambda = 29$ and $\omega = 50$ (please refer [6] for the details) also introduces high memory overhead as shown in Fig. 4a, specially when the high key connectivity rate is required.

Compared to symmetric key schemes, our ECC-based schemes overall have less memory overhead, specially when the key connectivity is high. In ECC-NoCert, each sensor only needs to store its private key and public key pair, which have the combined size of 60 bytes. In ECC-Cert, each sensor has to store one more certificate, so the memory overhead becomes 100 bytes, 40 bytes more than that of ECC-NoCert. ECC-PreComp has more memory overhead because each sensor needs to store the pre-computed random numbers (20 bytes each) and corresponding elliptic curve points (40 bytes each). Given average 20 neighbors, each sensor at least stores 20 pre-computed values, which account for 1,200 bytes more overhead. As the result, the memory overhead for ECC-PreComp are 1,260 bytes. Note the memory overhead of the public key base schemes do not change for achieving different key connectivity.

When the sensors are captured and compromised, the relative communication links are also compromised. These compromised links include the direct communication links connected to the compromised nodes and indirect communication links due to the leakage of the system secret, such as the subset of system key pool in Random-key scheme. To simplify the analysis, we assume the the compromised nodes are evenly and randomly scattered in the network. Figure 4b plots the number of compromised indirect communication links due to the node compromise.

Our ECC-based public key scheme is ideal under such situation. There is no indirect link compromised due to the

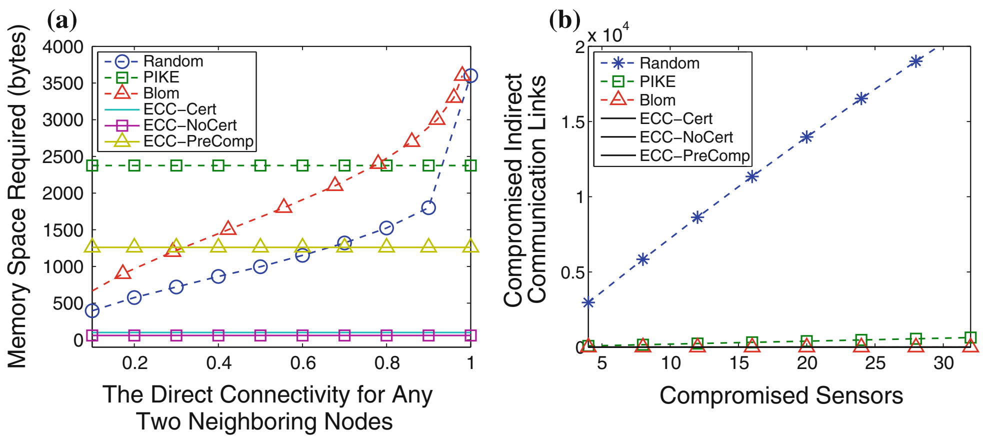


Fig. 4 **a** The memory space required for any two nodes to to establish a direct pair-wise key under different key connectivity rate. **b** The trend of percentage of total communication links compromised with the increasing number of sensors compromised

node compromise. In PIKE scheme, each sensor serves the intermediary for other two sensors. Suppose a sensor with ID (i, j) is compromised. As the result, all potential links between any sensor on the ith row and any sensor on the jth column will be compromised. Given $\sqrt{N}$ sensors on the ith row and $\sqrt{N}$ sensors on the jth column, there are totally N potential links. Considering the network connectivity is $20/N$, on average 20 indirect links can be compromised for each compromised sensor. In Random-key scheme, the communication links, which are not directly connected to the compromised nodes, may also be compromised because the secret keys used in these indirect links might be revealed to the adversary when the nodes are captured. Let the number of captured nodes be x. Given system parameters: total key pool number P and pre-distributed key number k, the expected fraction of the compromised communication links is $1 - (1 - \frac{k}{P})^x$ [6]. Figure 4b indicates that Random-key scheme is more vulnerable to the node compromise attack. When 32 nodes are compromised, more than 20,000 links can be compromised. PIKE scheme performs much better, but the number of compromised indirect links still grows linearly as the number of compromised sensor increases. As we indicated above, our ECC-based schemes are resistant to the node compromise. There is no indirect link can be compromised due to the node capture.

We find Blom scheme is resistant to the node compromise as no indirected link is compromised. As indicated in [6], however, this feature does not hold when the number of compromised nodes keeps growing. The security resilience degrades exponentially when the fraction of the compromised node reaches the certain threshold.

Careful readers may argue that the network parameter selection have an impact on the results of the above memory overhead and security resiliency analysis. For example, the memory overhead of some symmetric-key schemes, such as Random-key and Blom, are low when the key connectivity is low. However, the random graph theory [1] tells that, to have a securely connected sensor networks, the key connectivity has to maintain a certain level. In our example, given 10,000 sensors and 20 neighbors for each, the key connectivity must be greater than 90% in order to have a securely connected network with the probability of 99%. It is true that the requirement for the key connectivity reduces to 50% if the average degree of each node increases from 20 to 36. However, the sensor network in the latter case is almost twice denser than the former one. As the result, the hardware cost is nearly doubled because 16 more sensors are needed in each neighborhood area. We have not found a good way to convert the hardware overhead to the memory overhead and make the comparison against our previous memory overhead analysis, but we believe the hardware cost is an important performance metric and cannot be ignored. In this paper, we use the memory overhead as an example to present the extra cost incurred in the symmetric key schemes.

In addition, it is shown in [20] that the selection of 10,000 keys for the key pool in Random-key for a sensor network with 10,000 yields a weaker security resilience in node compromise attacks than the scheme with an optimal parameter. However, the optimal parameter selection presented in [20] does not fundamentally change the fact that the scheme relying on the pseudo-random key distribution has much less security resilience than that of ECC-based schemes. As indicated in [20], given a 10,000-node WSN with the optimal selection of the parameters, approximately 65 node compromises will lead as many as 25% of links compromised. In ECC-based schemes, with 10,000 nodes and average degree d, 65 compromised nodes only affect approximately $\frac{65 \times d}{10{,}000 \times d/2} = 1.3\%$ of the links (which are the direct links connected to those compromised nodes). In comparison, the Random-key with optimal parameters is still much more vulnerable in node compromise attacks. Based on the above discussion, we believe our security resilience analysis successfully reveals the security issue of the Random-key scheme, even though we use an example with a suboptimal parameter.

7.1.2 *Local user access control*

User access control requires the sensor nodes to authenticate the user and verify the user's access privilege. A symmetric-key user access control based on Blundo's scheme is proposed by Zhang et al. [32]. The Blundo's scheme is very similar to Blom's scheme as we explained previously. The system maintains a symmetric bi-variate polynomial. Each sensor or user is pre-loaded with a secret share of the polynomial. Any sensor and the user can establish a pairwise key by plugging other's public information, such as sensor ID or user access list, into the secret polynomial share. The access control can be achieved by integrating the user access list to the polynomial share, so that the user has to show the genuine access list, otherwise the user can not establish the pairwise key with the sensor and can not pass the authentication. The drawback of this scheme is that it has very limited security resilience against the user collusion attack. The reason is that the system secret, polynomial shares, has to be given to the user. Multiple malicious users may easily gather the information, reconstruct the secret polynomial, and finally compromise the system security. Here we want to emphasize that only public key scheme can fundamentally solve the security hole of the user collusion attack.

We do not compare our ECC-based local user access control to Blundo access control [32]. We instead perform comparison experiment to study other advantages of our

ECC-based local access control. We reserve this part to the next subsection.

7.2 Experimental results

Here, we demonstrate the advantages of our proposed public key schemes through real world experiments. For the comparison purpose, we also implement Random-key scheme and Blundo's scheme based access control scheme on real sensor motes.

7.2.1 Experiment test-bed and parameter setting

We implement the baseline symmetric key scheme, Random-key, on the same test-bed for the comparison of pairwise key establishment. We use 10 MICAz motes to form a sensor neighborhood. Each sensor can directly communicate with any of other nine neighbors. We select the key pool size of 10,000. Each key, with the size of 10 bytes, is identified by a two-byte key index. We first generate 10,000 random keys at a laptop computer. Each mote is randomly pre-distributed with 150 keys. In the experiment, we have adopted the simple scheduling method to avoid message collision, which emulates the optimal communication environment for key establishing. We randomly pick one out of ten motes to initiate the pairwise key establishment with all its neighbors. Even with 150 keys pre-loaded, they are not enough for any mote to establish direct pairwise key with all the neighbors. Therefore, multiple rounds of key establishment have to be performed. After the first round direct key establishment, the initiating mote notifies the neighbors that have already established direct pairwise keys with it and starts the second round of key establishment. The key establishing protocol is exactly the same in the second round except the initiator has changed. Each of the neighbors that have established the direct key is required to perform the indirect key establishing in the second round. Two rounds of key establishing still may not achieve 100% key connectivity for the original initiator. More rounds of such operation could be necessary. In our experiment, however, we limit it to three rounds. That means any two neighboring motes at most have two helpers for establishing indirect pairwise key. This arrangement is supported by the fact indicated in Random-key [7] that the number of pairwise key established through more than 3 hops is negligible.

Finally, we implement the Blundo access control on our test-bed. We first generate a random symmetric polynomial. The coefficients have the size of 10 bytes. The polynomial degree is adjustable for the target security resilience against the node compromise. Each mote is pre-distributed with a secret polynomial share, which is generated by simply plugging in the mote ID. The amount of memory space for storing the polynomial share is determined by the polynomial degree. Similar as the access control test-bed implemented by our ECC public scheme, we use the HP iPAQ as the user module. Again, the iPAQ is attached to a MICAz mote.

For all schemes conducted on our test-bed, we repeat the tests for 20 times, and record the average values.

7.2.2 Pairwise key establishment

Figure 5a illustrates the processing time delay in pairwise key establishing for achieving different degree of key connection. We select two ECC-based schemes: ECC-PreComp and ECC-NoCert for this experiment. It clearly shows that ECC-PreComp is much faster than ECC-NoCert since the former scheme only requires one ECC multiplication for both neighboring sensors, while the latter one requires two. In reality, ECC-PreComp is very practical because the precomputation only introduces a very limited memory overhead compared to that in symmetric key schemes.

Compared to the PKC-based schemes, Random-key scheme has lower processing overhead when the requirement of key connectivity is low. However, this advantage does not hold if more than 80% key connection is required. The reason is that the number of pre-distributed keys is not enough for establishing pairwise keys with all its neighbors. The key establishing time thus increases to infinity. As shown in Fig. 5a, the time jumps to infinite large at key connectivity of 0.8. We restrict the pre-distributed key number due to the limited 4KB memory space in MICAz motes. In our experiment, with 150 key pre-distributed, a mote can only establish direct pairwise key with two out of its nine neighbors. The other pairwise keys are established through the second and the third rounds of key establishing procedure.

Figure 5b further reveals that ECC-based pairwise key scheme has much less message complexity than Random-key scheme. To establish a pairwise key, two neighboring motes only need to transmit 120 bytes for both ECC-NoCert and ECC-PreComp schemes. In Random-key scheme, the broadcasting node has to send all key IDs in its key ring. Given 150 keys and 2 bytes each for key index, the broadcasting mote transmits 300 byte message. All listening neighbors also need to respond the key establishing broadcast, by either replying the challenging message (if there is shared key), or notifying there is no shared key. This message overhead has to be paid in all three key establishing rounds. In wireless sensor networks, high message complexity increases the chance of message collision and thus causes network congestion. The low message complexity is a significant advantage for ECC-based pairwise key establishing schemes.

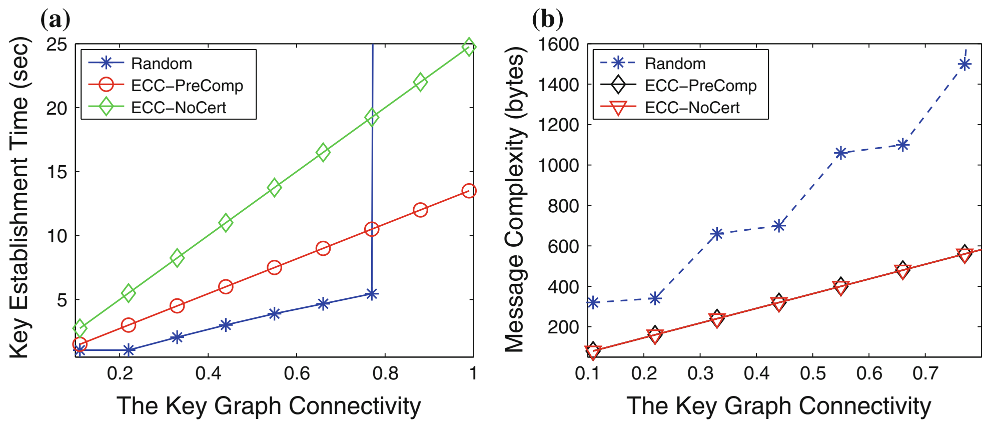


Fig. 5 **a** Key establishing delay for different key graph connectivities. **b** The message complexity for achieving the target key connectivity

Finally, we compare the energy consumption, including communication energy and computation energy, during the pairwise key establishment. We estimate the communication energy consumption by multiplying the total amount of communications by an average communication energy consumption of 18μ J/bit [3]. Since the symmetric-key encryption and decryption are very efficient, we ignore the computation overhead of Random-key scheme. Comparatively, it takes several seconds in the public-key-based schemes, so the computation energy consumption cannot be ignored. The ECC computation energy consumption E can be calculated by $E = U \cdot I \cdot t$, where U is the voltage, I is the current and t is the time duration. According to the MICAz data-sheet, U is 3.0V (two AA batteries), and I is 8mA (the current draw in the active mode). We plot the results in Fig. 6a. The dash-line is the communication energy cost of Random-key scheme. The two solid lines are the combined communication energy and computation energy consummation of two ECC-based schemes. The figure clearly identifies the key drawback of Random-key scheme. The symmetric-key-based scheme consumes more than twice amounts of communication energy than the ECC-based scheme even though the public key scheme consumes more energy in computation. The reason is that message broadcast is required in Random-key scheme. As the result, all neighboring sensors need to listen the broadcasts all the time and consume the energy for receiving the messages.

We argue that processing time is not a significant metric for sensor network application as opposed to the memory usage, message complexity, robustness to sensor compromising. Pairwise key sharing is usually conducted in the initialization phase of sensor network deployment as papers on symmetric key pairwise key sharing argue. Public key protocol for key sharing takes a few more seconds to finish than symmetric key protocols, which is tolerable for network initialization and also for online pairwise key sharing, since it is done only once between two sensors. From the experimental data, we clearly see that our protocol performs better than symmetric key

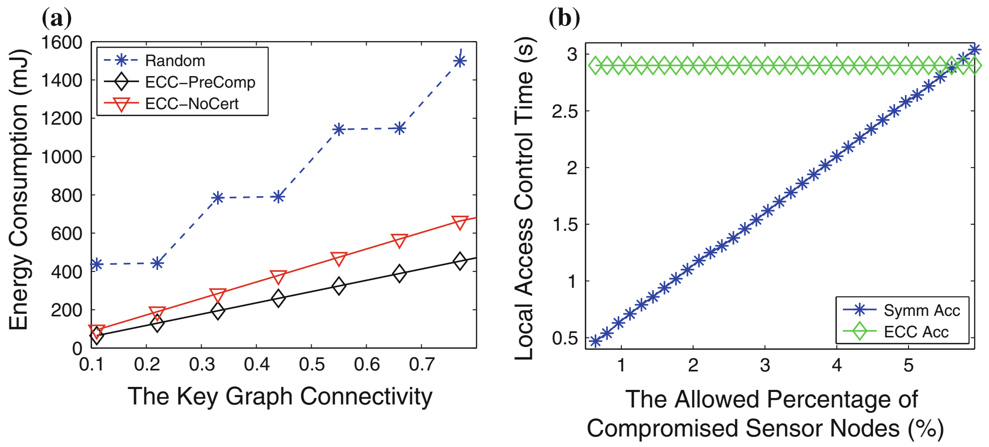


Fig. 6 **a** The energy consumption (including communication and computation) for achieving the target key connectivity. **b** Local authentication time versus security resilience (% of compromised sensors)

protocols in terms of memory usage, message complexity (and thus equivalently energy consumption and system lifetime), and robustness to network compromising.

7.2.3 Local access control

We first measure the authentication delay in the local access control. Since the parameter selection in the Blundo's scheme based local access control depends on the security resilience against the node compromise. We test both schemes under various security resilience requirements. The user authentication delay is shown in Fig. 6b. When the security resilience is low, up to 5.5% of sensor nodes allowed to be compromised, the Blundo access control is more efficient than our ECC-based scheme. The reason is that the polynomial operations are much faster than ECC exponentiation. However, the processing overhead of the Blundo based scheme increases as the requirement of security resilient increases. When the requirement of security resilience is more than 5.5%, the processing overhead of the symmetric-key-based scheme becomes slower than our PKC-based scheme. The reason is that the processing overhead of our ECC-based scheme does not change, it always provides the security equivalent to the discrete logarithm problem.

Note, the security concern of user collusion attack has not been revealed yet by this experiment. This security issue has to be considered in real world deployment. Therefore, either higher degree random polynomial or multiple polynomial have to be selected to improve the security. As a result, the processing overhead of the Blundo based access control will be higher. On the contrary, the ECC-based access control scheme does not suffer from user collusion attack, so our scheme can be directly applied to the real world deployment.

Figure 7a shows the comparison of data size of two local access control schemes in the real implementation. It clearly shows that the memory overhead scales linearly in the Blundo based scheme for satisfying different security resilience. The degree of the random polynomial is larger for higher security requirements. As a result, the sensors need more space to store the corresponding coefficients. The data size of the ECC-based scheme, as can be easily predicted, does not change at all.

When Figs. 6b and 7a show that the Blundo access control has poor security scalability in processing time and memory overhead, Fig. 7b displays that it also has poor network scalability in message complexity. Since the Blundo access control scheme uses "Cell Merging" and "Block Compression"[32] to reduce the number of polynomial possessed by the user. The user has to traverse a Merkle-hash tree. The traversal path length is determined by the tree size, which is in turn determined by the number of location blocks, or the network size. Again, our ECC-based user access control has the advantage of excellent network scalability; the message complexity is independent to the network size, fixed at 100 bytes. The figure clearly shows that the Blundo based scheme has more complexity than our public key scheme when the network size is just over 100 blocks. This fact proves our scheme is more favorable for large network deployment.

7.3 Remote access control

In this subsection, we evaluate our remote access control scheme. We first provide the micro-benchmark for the local authenticate and threshold endorsement generation. Then, we provide the overall estimation of the remote access performance. In the experiments, we mainly focus on the user perceived remote access processing delay. Our first

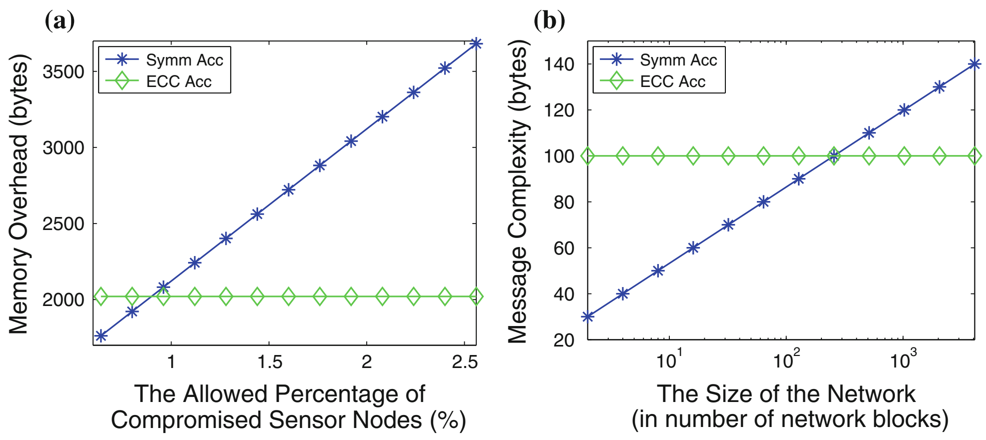


Fig. 7 **a** The memory space required to finish the local user authentication, regarding the different security resilience, in term of the percentage of sensors allowed to be compromised. **b** The message complexity in user authentication, regarding the different network size, in term of the number of network blocks, where each block is the user access area unit

hand experimental results suggest the PKC-based remote access control scheme is very practical.

7.3.1 Local endorsement

The local endorsement procedure can be further divided into user local authentication and endorsement generation. We have already demonstrated the performance of user local authentication in the previous section. To be authenticated by multiple local sensors, a simple and effective optimization can be applied to allow the user to be authenticated in parallel rather than one-by-one. The user first sends its certificate to all the endorsing sensors, so that the endorsing sensor can verify the certificate and generate the challenges simultaneously. Then the user collects all the challenges from each member of endorsing group and responds them one-by-one. This optimization is valid for user authentication because the user device is much more powerful than sensors. As we showed previously, ECC multiplication on iPAQ is more than 30 times faster than MICAz mote. Therefore, the ECC operation overhead on user device is negligible compared to that of sensors. This also explains why such optimization does not work in pairwise key establishment between one sensor and its neighbors.

Figure 8a displays time consumption when the user is authenticated by multiple endorsing sensors. For the comparison purpose, we also show the authentication delay without the optimization. Obviously, the optimized scheme is significantly more efficient. Before optimization, it takes more than 45 s for the user to finish authentication with 16 endorsing sensors. This delay dramatically reduces to 5s after the optimization.

After finishing the user authentication, the endorsing sensors perform threshold endorsement to establish pairwise key between the user and the remote sensor. We continue the above authentication experiment. Each endorsing sensor immediately computes its endorsement share and then sends to the user sequentially. Figure 8b shows the user waiting time to receive all the endorsement shares. With the number of endorsing sensors changing from 4 to 32, the time duration linearly grows from 4.5 to 8.9 s. The measurement includes the user local authentication time (local pairwise key establishing time). The performance is consistent with that displayed in Fig. 8a. The threshold endorsement requires each sensor to perform one more ECC point multiplication at the cost around 1.3s as showed previously.

Upon the receipt of the remote access query, the remote sensor has to verify the authenticity of the remote query by decrypting the message using its own secret share as presented in Sect. 5 The computational complexity of this operation is independent to the number of local endorsing sensors. The only expensive operation at the remote sensor is one ECC point multiplication. It takes only 1.4 s for the remote sensor to calculate its secret share and verify the query.

7.3.2 Complete remote access control

Finally, we are eager to investigate the overall performance of the remote access control, including the threshold signature generation, message propagation, and remote sensor verification. We assume the local endorsing sensors have already established pairwise key with each others. To simplify the experiment, the user directly sends the query to the remote sensor. Then we add the estimated hop-by-hop forwarding delay to estimate the performance for various hop distances. The estimated forwarding delay is the communication delay in sensor RF transceiver. Our estimation fixes the amount of communication delay to 17.5 ms.[1]

Figure 9a shows the estimated overall user remote query response time, given the size of local endorsing group with 4, 8 and 16, respectively. We find the overall remote query delay is short. When the remote sensor is located at 20 hops away, the user query response time is 6.8 s. When the larger size of the local endorsing sensor group is required, the additional overhead increases moderately.

7.3.3 Porting to other sensor platforms

Finally, we demonstrate that our ECC-based user access control suite can also be efficiently deployed on a different sensor platform, TelosB mote. Figure 9b illustrates the performance comparison between the two platforms for remote access control with the setup of 8 local endorsing sensors. The overall access control performance on the two platforms is very close, although the performance on TelosB is slightly worse because ECC on TelosB is slightly slower. In practice, MICAz and TelosB can be deployed together to form a heterogeneous sensor network for user access control purpose because they share the same RF transceiver.

8 Conclusion

This paper proposes a PKC-based access control for sensor networks, which consists of pairwise key establishment, local access control, and remote access control The main idea is to certify the user's access list by a group of local sensors in his vicinity, and pass the local endorsement to the remote sensor if local authentication is passed. ECC and threshold cryptography have been used extensively in the scheme. We have performed a comparison test by implementing both

[1] Based on our experimental result of forwarding a 60 byte payload in MICAz motes.

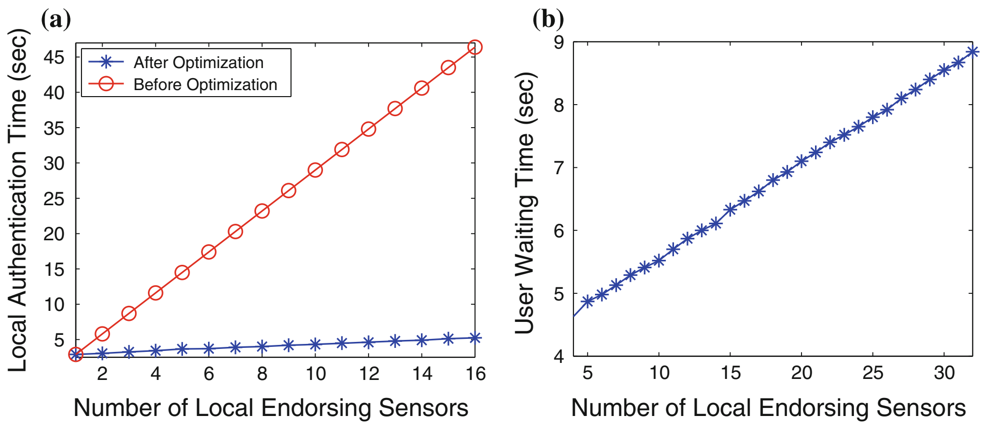


Fig. 8 **a** The user local authentication time, before and after the optimization, by multiple local endorsing sensors. **b** Key establishing time with the remote sensor when the number of endorsing sensor changes from 4 to 32

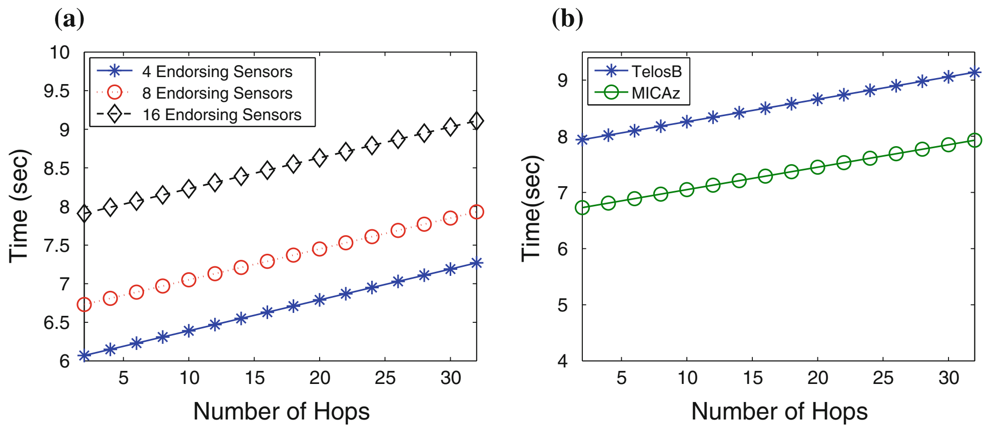


Fig. 9 **a** Remote query time delay. **b** Comparison of remote query delay between MICAz and TelosB

symmetric-key and public-key primitives on popular sensor motes. Our experiment results suggest the PKC-based protocol is more advantageous than the symmetric key in terms of the memory usage, message complexity, and security resilience. To the best of our knowledge, this is the first comprehensive experimental comparison between symmetric-key and public-key scheme in sensor networks.

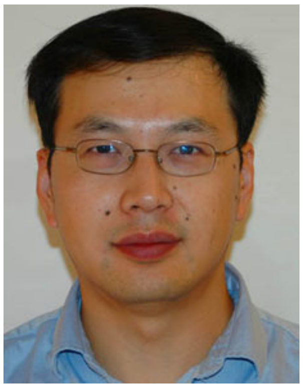
Acknowledgments The authors would like to thank all the reviewers for their insightful comments and kind guidances to improve the paper. This project was supported in part by US National Science Foundation grants CNS-0721443, CNS-0831904, and CAREER Award CNS-0747108.

References

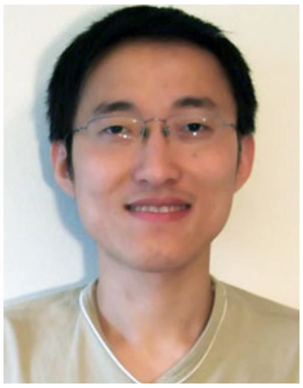
1. Bollobs, B. (1985). *Random graphs*. NY :Acadamic Press Inc.
2. Boneh, D., & Franlin, M. (2001). Identity-based encryption from the weil pairing. In *CRYPTO*, pp. 213–229. Berlin: Springer.
3. Carman, D., Matt, B., Kruus, P., Balenson, D., & Branstad, D. (2000). Key management in ditributed sensor networks. In *DARPA Sensor IT Workshop*.
4. Chan, H., & Perrig, A. (2005). PIKE: Peer intermediaries for key establishment in sensor networks. Miami, FL: INFOCOM.
5. Chan, H., Perrig, A., & Song, D. (2003). Random key predistribution schemes for sensor networks. In *IEEE symposium on security and privacy*, pp. 197–213, Berkeley, California, May.
6. Du, W., & Deng, J. (2003). A pairwise key pre-distribution scheme for wireless sensor networks. In ACM CCS.
7. Eschenauer, L., & Gligor, V. D. (2002). *A key-management scheme for distributed sensor networks*. In ACM CCS, November.
8. Fox, A., & Gribble, S. D. (1996). *Security on the move: Indirect authentication using Kerberos*. In *Mobicom*, New York, November.
9. Gura, N., Patel, A., Wander, A., Eberle, H., & Shantz, S. C. (2004). *Comparing elliptic curve cryptography and RSA on 8-bit CPUs*. In *CHES*, Cambridge, MA, August.
10. Crossbow Technology INC. Wireless sensor networks. http://www.xbow.com/.
11. Karlof, C. , Sastry, N., & Wagner, D. (2004). TinySec: A link layer security architecture for wireless sensor networks. In *SENSYS*, Baltimore, MD, November.
12. Liu, A., & Ning, P. (2005) http://discovery.csc.ncsu.edu/software/TinyECC/.
13. Liu, D., & Ning, P. (2003). Establishing pairwise keys in distributed sensor networks. In *ACM CCS*, Washington, DC, October.
14. Malan, D. J., Welsh, M., & Smith, M. D. (2004) A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *The first IEEE international conference on sensor and Ad Hoc communications and networks*. Santa Clara, CA, October.
15. Mathur, G., Desnoyers, P., Ganesan, D., & Shenoy, P. (2006). Ultra-low power data storage for sensor networks. In *IPSN '06*, New York, NY, USA.

16. Neuman, B. C., & Ts'o, T. (1994). Kerberos: An authentication service for computer networks. *IEEE Communications, 32*(9), 33–38.
17. NIST. (2001). Key management guideline. In *Workshop Document (DRAFT)*, October
18. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, D. (2002). *SPINS: Security protocols for sensor networks*. NY: ACM/Kluwer Wireless Networks Journal (WINET).
19. Di Pietro, R., Mancini, L. V., & Mei, A. (2006). Efficient and resilient key discovery based on pseudo-random key pre-deployment. *Wireless Networks 12*(6).
20. Di Pietro, R., Mancini, L. V., Mei, A., Panconesi, A., & Radhakrishnan, J. (2008) Redoubtable sensor networks. *ACM Transaction on Information and Systems Security, 11*(3), March.
21. Ren, Kui , & Lou, Wenjing (2005). Privacy enhanced access control in pervasive computing environments. In *Proceedings of BroadNet05*, October.
22. Shamir, A. (1979) How to share a secret. C*ommunications of the ACM 22*(11), 612–613
23. Shantz, S. C. (2001). From Euclid's GCD to montgomery multiplication to the great divide. In *Technical report, Sun Microsystems Laboratories* TR-2001-95, June.
24. TinyOS. TinyOS 1.1.15. http://www.tinyos.net, 2006.
25. Traynor, P., Kumar, R., Saad, H. B., Cao, G., & Porta, T. L. (2006). LIGER: Implementing efficient hybrid security mechanisms for heterogeneous sensor networks. In *MOBISYS*, Uppsala, Sweden, June.
26. Wang, H., & Li, Q. (2006). Distributed user access control in sensor networks. In *IEEE international conference on distributed computing in sensor systems*(DCOSS), pp. 305–320, San Francisco, CA, June.
27. Wang, H., & Li, Q. (2010). Achieving robust message authentication in sensor networks: A public-key based approach. *ACM Journal of Wireless Networks (WINET), 16*(4), 999–1009.
28. Wang, H., Sheng, B., & Li, Q. (2006). Elliptic curve cryptography based access control in sensor networks. *International Journal on Sensor Networks, 1*(2).
29. Wang, H., Sheng, B., Tan, C.C., & Li, Q. (2007). *WM-ECC: An elliptic curve cryptography suite on sensor motes*. Technical Report WM-CS-2007-11, College of William and Mary, Computer Science, Williamsburg, VA.
30. Ye, F., Luo, H., Lu, S., & Zhang, L. (2004). *Statistical en-route filtering of injected false data in sensor networks*. In INFOCOM.
31. Zeinalipour-Yazti, D., Lin, S., & Kalogeraki, V. (2005). Dimitrios Gunopulos, Walid A. Najjar. MicroHash: An efficient index structure for flash-based sensor devices. In *FAST*.
32. Zhang, W., Song, H., Zhu, S., & Cao, G. (2005). Least privilege and privilege deprivation: towards tolerating mobile sink compromises in wireless sensor networks. In *MOBIHOC*, Chicago, IL, May.
33. Zhang, Y., Liu, W., Lou, W., & Fang, Y. (2006). Location-based compromise-tolerant security mechanisms for wireless sensor networks. *IEEE Journal on Selected Areas in Communications, 24*(2), 247–260.
34. Zhang, Y., Liu, W., Lou, W., & Fang, Y. (2005). Securing sensor networks with location-based keys. In *WCNC*, New Orleans, Louisiana, March.
35. Zhu, S., Setia, S., & Jajodia, S. (2003). LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *ACM CCS*, Washington D.C., October.
36. Zhu, S., Setia, S., Jajodia, S., & Ning, P. (2004). An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE symposium on security and privacy*, Oakland, CA, May.

Author Biographies



Haodong Wang is an assistant professor in the Department of Computerand Information Science at Cleveland State University. He received his Ph. D in Computer Science at College of William and Mary. His research interests focus information assurance in Cyber-Physical Systems, privacy preserving and user access control in sensor networks,efficient information storage, search and retrieval in pervasivecomputing, and MAC design in IEEE802.11 wireless LAN.



Bo Sheng is an assistant professor in the department of Computer Science at University of Massachusetts Boston. He received his Ph.D. in computer science from the College of William and Mary in 2010. His research interests include wireless networks and embedded systems with an emphasis on efficiency and security issues.

Chiu C. Tan is a research assistant professor at Temple University. He received his Ph.D in Computer Science from the College of William and Mary. His research involves security and privacy problems in cloud computing, social networks, wireless, sensor and vehicular networks.

Qun Li is an assistant professor in the Department of Computer Science at College of William and Mary. He holds a PhD degree in computer science from Dartmouth College. His research interests include wireless networks, sensor networks, RFID, and pervasive computing systems. He received the NSF Career award in 2008.