An Asymmetric Key Establishment Protocol for Multiphase Self-Organized Sensor Networks

Emmanouil Magkos[1], Panayiotis Kotzanikolaou[2] Dimitris D. Vergados[3], Michalis Stefanidakis[1]

[1]**Ionian University, Department of Computer Science, Platia Tsirigoti, 49100, Corfu, Greece.**
e-mail: {emagos, mistral}@ionio.gr

[2]**University of Piraeus, Department of Informatics, Karaoli & Dimitriou, 18534, Greece.**
e-mail: pkotzani@unipi.gr

[3]**University of the Aegean, Department of Information and Communication Systems Engineering, Karlovassi, Samos, GR-832 00, Greece.**
e-mail: vergados@aegean.gr

***Abstract*: We propose an asymmetric key establishment protocol for decentralized sensor networks (DSN's). The protocol supports multiphase deployment *i.e.,* the sensor nodes are deployed in the network in multiple groups, known as "generations". After their deployment, the nodes of the first generation are engaged in a key establishment protocol, in order to establish secure communication channels with their neighbors. Then, each forthcoming generation initializes a new key establishment phase, which allows the new nodes to communicate with other nodes of their own generation as well as with the older nodes of the network. The protocol extends the hybrid scheme of Kotzanikolaou et al [6]. By making the scheme of [6] fully asymmetric, the proposed protocol corrects a security weakness found in [6] while it does not induce further computation and communication costs.**

1. Introduction

Recent work on key establishment for sensor networks has shown that it is feasible to construct sensors capable of performing (limited) public key cryptographic protocols through Elliptic Curve (EC) cryptography [10] – see for example [4, 5, 8]. Especially in [8] a practical implementation of Elliptic Curve Diffie-Hellman (ECDH) key exchange over F_{2^P} is presented for establishing a shared symmetric key between two sensor nodes, using 163-bit EC public keys in a 8-bit, 7.3828 MHz MICA2 mote [3]. Recently, Kotzanikolaou *et al.* [6] proposed a key establishment protocol for uniform self-organizing sensor networks, where sensor nodes establish pairwise keys, by employing limited public key cryptography. The protocol of [6] is a hybrid protocol which combines standard ECDH key agreement with symmetric techniques. In this way, key establishment maintains many of the advantages of public key cryptography, while the computation, communication and storage costs are realistic for sensor nodes.

Our contribution. In this paper, we propose a fully asymmetric key establishment protocol for sensor nodes, which corrects a security weakness in the Kotzanikolaou *et al.* [6] protocol. While in [6] an attacker who compromises a node and gets all its keying material is able to get access to the victim's past communication, the proposed protocol provides *forward secrecy* to the attacked node. The protocol has the same computation and communication costs with the hybrid protocol of [6], while the extra protection comes at the cost of increasing the storage requirements for each sensor node in the network.

2. Related Work

The hybrid key establishment protocol proposed in Kotzanikolaou et al [6] is comprised of two parts. The "asymmetric" part of the protocol is based on:

- Elliptic Curve Diffie-Hellmann (ECDH) [10] for key agreement between sensor nodes. Each sensor is pre-deployed with a static key pair which the sensor will use, throughout its life, during the bootstrapping phases with its neighbors.

- Implicit certificates [12] for entity authentication. In order to reduce the communication and computation overhead induced by standard certificates, an efficient construction for implicit certificates, based on EC-schnorr signatures, is used in [6]. Such certificates are issued by an off-line authority.

To mitigate the effect of known-key attacks, and in an effort to reduce the scalar multiplications required by protocols such as MQV / MTI [7], the nodes in [6] use symmetric techniques to securely exchange random nonces; These are then used in the key derivation process for key freshness. The "symmetric" part of the protocol is a variation of the AKEP2 protocol [1] for authenticated key agreement with explicit key confirmation using initial trust between nodes. Initial trust is comprised of generation-wide symmetric keys, which the sensors are pre-deployed with, during an off-line protocol with a key authority. Generation-wide keys are only used during bootstrapping and then they are permanently deleted.

Performance of the Kotzanikolaou et al protocol. The protocol in [6] is scalable, with sensors being pre-deployed with a constant number and size of keys, independently of the size of the network. Using the metrics of [5] regarding the costs of each cryptographic action[1], the protocol of Kotzanikolaou et al [6] has a total computation cost of about 645ms per node, 186 bytes

[1]Their computations were performed on Mitsubishi's 16-bit single-chip microprocessors M16C with 10MHz clock.

of communicated messages and 193 bytes of key storage requirements, for a network of five node generations.

A security weakness in the Kotzanikolaou et al [6] protocol. We assume an attacker who eavesdrops all communication lines. We also assume that the attacker learns all generation-wide keys, *e.g.* by compromising at least one newly arriving node in each bootstrapping phase. Then, for any node which is compromised, during the life of the network, the attacker will be able to learn all node's past and future communications with any other node of the network.

Our protocol corrects the above issue and pertains *forward secrecy* for any node whose keying material has been compromised. This is achieved by requiring that each node possesses multiple independent EC public key pairs, one for each generation of nodes. These keys are used in the traditional Diffie-Hellmann model [10] to establish session keys and they are always deleted at the end of each bootstrapping period. Obviously the storage cost for each node is increased in comparison with the Kotzanikolaou et al protocol [6] (a performance analysis will be given in Section 5).

3. An Asymmetric Key Establishment Protocol for Self-Organized DSNs

We propose a key establishment protocol that can be applied to uniform and self-organized sensor networks, *i.e.* we assume that all the nodes are *Restricted Functional Devices* [5] with limited communication, computation and storage capabilities, while the communication between sensor nodes is not supported by any static infrastructure.

The proposed protocol is fully *asymmetric*: the nodes are not pre-deployed with any shared secret information. However, before initialization of the network a trusted authority CA pre-deploys each sensor with the appropriate asymmetric keying information to support authenticated key exchange between any two nodes of the network.

After their deployment all nodes participate in a bootstrapping phase in order to exchange keys with their "neighboring" nodes *i.e.*, other nodes that lie within range. We assume that the nodes are randomly deployed (*e.g.*, via aerial scattering) and that are not aware of their neighbors until their deployment. In *multiphase* deployment it is also possible for sensor nodes to join the network in future time periods, and establish secure channels with nodes of the same and/or of a past generation.

3.1. Notation

Let q denote the order of the underlying finite field F_q and let E be a suitably chosen elliptic curve defined over F_q. Let P denote a base point in E, the generator point, and n be the order of P, where n is prime. Thus $nP = O$ and $P \neq O$ where O is the point at infinity. Let $q_{CA} \in [2, n-2]$ be a random integer selected by the Certification Authority CA and $Q_{CA} = q_{CA} \times P$. The pair of the static secret/public key pair of the CA is q_{CA}, Q_{CA}.

We assume that the network will be eventually constructed from a total of m generations. Let $X^{(i)}$ denote a node X that belongs to the i_{th} node generation and let ID_X denote the identifier of that node. For simplicity and when no further clarification is required, we will denote the node $X^{(i)}$ as X.

3.2. The key pre-deployment phase

During the key pre-deployment phase, the CA predeploys each node $X^{(i)}$ with $(m - i + 1)$ asymmetric key pairs and the corresponding certificates, one for each node generation $i, i+1, i+2, ...m$, in order to enable $X^{(i)}$ to perform bootstrapping with any node $Y^{(k)}$, where $k \geq i$.

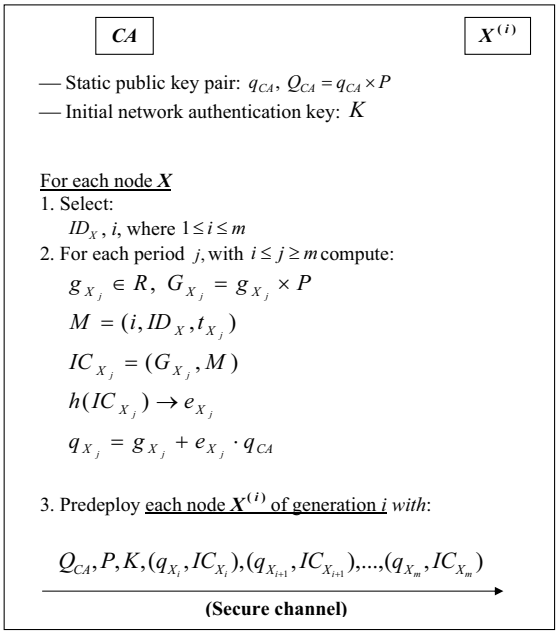

Figure 1: The key pre-deployment phase

For each node generation j, where $j = i, i+1, ..., m$, the CA selects a random number $g_{X_j} \in [2, n-2]$ and computes $G_{X_j} = g_{X_j} \times P$ (see Figure 1). For each random number g_{X_j}, the CA also generates an Implicit Certificate $IC_{X_j} = (G_{X_j}, M_{X_j} = \{i, ID_X, t_{X_j}\})$. Here M_{X_j} includes a fixed generation identifier i of the node, a unique identifier ID_X for the node X and the expiration time t_{X_j} of each Implicit Certificate IC_{X_j}. For each Identity Certificate IC_{X_j}, the CA applies a cryptographic hash function h and from each octet $h(IC_{X_j})$, it obtains an integer e_{X_j}, by using the conversion routine[2] described in [10]. Then, the CA computes the static generation-wide secret keys of the node X as $q_{X_j} = g_{X_j} + e_{X_j} \cdot q_{CA}$, $j = i, i+1, ..., m$. The values g_{X_j} are not given to the node X and are deleted immediately after the key generation process; Otherwise, a compromised node would be able to extract the static secret key q_{CA} of the CA from any couple of the values q_{X_j} and g_{X_j}, for any $j = i, i+1, ..., m$. Observe that each pair (e_{X_j}, q_{X_j}) is an EC-Schnorr signature [9], created by the CA, over the message M_{X_j} of each Implicit Certificate IC_{X_j}. The corresponding public keys Q_{X_j}'s are

[2]Informally, the idea is simply to view the octet string as the base 256 representation of the integer (Section 2.3.8 of [10]).

not stored at node X's memory. Any other node, will be able to recover the appropriate generation-wide public key Q_{X_j} of the node X from the Implicit Certificate IC_{X_j} and the public key Q_{CA} of the CA (see Step 5 in Section 3.3).

Finally, the CA pre-deploys the node $X^{(i)}$ with the necessary network-specific material, such as the CA's public key Q_{CA}, the point P, and an initial symmetric authentication key K (see Figure 1). This 64-bit key K will be used by all nodes as an initial authenticator, in order to avoid processing of fake "hello" messages and prevent trivial DoS attacks. After the initialization of the network the CA will have a passive role, and will not further participate in key establishment. The CA will only be allowed to generate and pre-deploy the keys for the nodes of forthcoming generations.

3.3. The key bootstrapping phase

In this phase, any pair of sensor nodes lying in each other's range, will use their pre-deployed keys in order to perform an authenticated pairwise key establishment. Let $A^{(j)}$, $B^{(i)}$ be two nodes belonging to generations j, i respectively, such that $1 \leq j \leq i \leq m$. Thus, the nodes may belong to the same $(j = i)$ or different $(j < i)$ generation. We describe the bootstrapping phase for the i_{th} period.

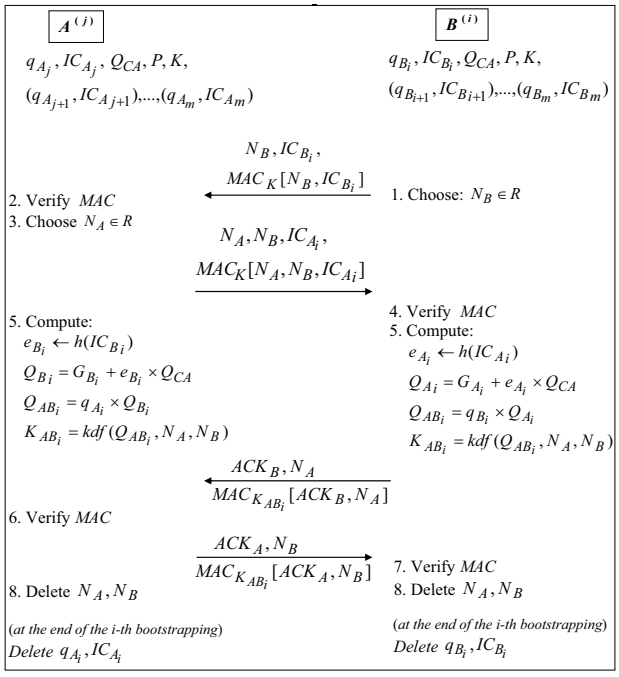

Figure 2: The key bootstrapping phase

If $j = i$ then both nodes will possess, among other things, their i_{th} EC key and the corresponding Implicit Certificate, (q_{A_i}, IC_{X_i}), and (q_{B_i}, IC_{B_i}) respectively, generated during key pre-deployment.

Step 1. The node $B^{(i)}$ initiates the key establishment, by choosing a nonce N_B and broadcasting it along with its Implicit Certificate IC_{B_i} for the i_{th} generation. The node B also broadcasts a Message Authentication Code (MAC) of the above values, generated with the network-wide key K.

Step 2-3. The neighboring node A verifies the received MAC and retrieves B's generation identifier i from IC_{B_i}. Then, the node A chooses from its storage the corresponding certificate IC_{A_i} and completes the initial handshaking, by choosing a nonce N_A and sending it to B along with the suitable Implicit Certificate IC_{A_i} and a MAC on (IC_{A_i}, N_A, N_B) generated with K.

Step 4-5. On receiving this message, the node B checks the received MAC and if it verifies correctly, the node B uses the received Implicit Certificate IC_{A_i} and the public key Q_{CA} of the CA, in order to compute the public key of node A for the i_{th} generation as $Q_{A_i} = G_{A_i} + e_{A_i} \times Q_{CA}$. Observe that at this point B cannot yet verify the authenticity of the key Q_{A_i}: as soon as A proves knowledge of q_{A_i}, the node B will have *implicit* [12] assurance that it is talking to A and that all information included in the certificate is genuine (*i.e.* signed by the CA).

The node B computes the static pair key $Q_{AB_i} = q_{B_i} \times Q_{A_i}$. The final pairwise key K_{AB} is computed by applying a key derivation function kdf over Q_{AB_i} and N_A, N_B [10]. The function kdf is implemented through an one-way cryptographic hash function, such as SHA-1 [11]. Then, the node B chooses a random acknowledgement (ACK_B) number, computes a MAC on (ACK_B, N_A) with the pairwise key K_{AB_i} and sends $MAC_{K_{AB_i}}[ACK_B, N_A]$ to the node A. The MAC will provide key confirmation to node A, since it will prove that the corresponding secret key q_{B_i} was used.

Step 5-6. At parallel, the node A will use the Implicit Certificate IC_{B_i} and the public key Q_{CA}, in order to compute the public key of node B as $Q_{B_i} = G_{B_i} + e_{B_i} \times Q_{CA}$. The node A similarly computes the static pair key $Q_{AB_i} = q_{A_i} \times Q_{B_i}$. The pairwise key is again computed as $K_{AB_i} = kdf(Q_{AB_i}, N_A, N_B)$. At this time node A will verify the MAC it had previously received by B, in order to confirm that the appropriate secret key of node B was used in the computation of K_{AB_i}.

Step 7-8. In order to provide key confirmation regarding its own secret key q_{A_i}, the node A will similarly compute a MAC with the key K_{AB_i} on a random acknowledgement number (ACK_A) and send it to the node B. After this verification, both nodes will delete the random values N_A, N_B. Note that from the key K_{AB_i} the two nodes can derive two different keys, one for encryption and one for authentication [10].

At the end of the i_{th} bootstrapping phase and after the nodes have performed a key establishment with each of their neighbors, the nodes A and B will delete their i_{th} EC keys and certificates (q_{A_i}, IC_{X_i}), and (q_{B_i}, IC_{B_i}) respectively. For key freshness in subsequent periods, the nodes may periodically update the pairwise key K_{AB_i} using an one-way hash function. The time

interval between subsequent renewals may depend on the data traffic volume, as well as on the strength of the underlying cryptographic primitives.

Remark. Consider two nodes which are not able to communicate during the i'th bootstrapping phase (e.g. due to a weak signal, or a node being busy). In the proposed protocol they will be able to establish session keys in a future bootstrapping phase by using the corresponding public key pair. This was not possible in [6] where all nodes deleted their generation-wide keys at the end of each bootstrapping phase.

4. Security Considerations

Our threat model includes both *passive* and *active* attacks. The proposed protocol extends the Kotzanikolaou et al [6] protocol by combining standard ECDH key agreement [10] and implicit certificates [12] for mutual authentication without any prior secret information being shared between nodes. The protocol is a four-pass challenge-response protocol for authenticated ECDH key agreement with explicit key confirmation. We make use of random nonces for message and key freshness, and MACs for data integrity. We assume that the underlying primitives are secure.

4.1. Secure key generation

By using a private offline interface between each sensor node and the CA, during pre-deployment phase, both active and passive attacks against the key generation process (such as unknown key share attacks and small subgroup attacks [7]) are thwarted, provided that the CA is honest and takes all reasonable measures in the key generation process.

4.2. Known-key security

Clearly, if a node's keying material is revealed at any time, all its present and future communication is revealed, given that the attacker is also an eavesdropper. In the following we will examine whether the forward secrecy property is maintained throughout the execution of the protocol.

Forward secrecy The derivation of a session key K_{AB_i} is based on the static EC keys q_{A_i} and q_{B_i}, and the random values N_A, N_B. Since N_A, N_B are transported in the clear, they do not add security to the scheme. However, the nodes A, B delete their EC keys q_{A_i}, q_{B_i} respectively, at the end of the i'th bootstrapping phase and after they have performed a key establishment with each of their neighbors. Recall that in subsequent periods, the nodes will update the pairwise key K_{AB_i} using an one-way hash function. Thus, compromising a node at a given time does not reveal past communications of the attacked node. This would require knowledge of the particular EC keys used for any past session key.

4.3. Node authentication

The proposed protocol uses implicit certificates [12] to support mutual entity authentication. Our specific construction is based on EC-Schnorr [9] signatures, which are provably secure under the *random oracle* model given that the discrete logarithm problem over a subgroup $< G >$ is untractable [2].

Security against impersonation attacks. During the bootstrapping phase, the node A uses the implicit certificate of B and the public key of the CA to reconstruct the public key Q_B of node B. In step 5, the node B uses its private key q_B for the construction of the ECDH key K_{AB} and returns a MAC created with K_{AB}, the node A will have implicit assurance that it is talking to B and that all information included in the certificate is genuine (*i.e.* signed by the CA).

Security against fake generation attacks. Similarly to [6], fake generation attacks are precluded: the generation number of the node is included in its implicit certificate. A compromised node cannot present itself as a node of an earlier or future generation, or else the verification of the certificate will fail.

5. Performance Evaluation

Computational complexity. In order to produce comparable results with related work, we use the metrics of [5] regarding the costs of each cryptographic action. Their computations were performed on Mitsubishi's 16-bit single-chip microprocessors M16C with 10MHz clock. The same metrics were used in the hybrid protocol of [6]. The costs per action are shown in Figure 3. The cost of fixed-point scalar multiplication is reduced, by having a pre-computed look-up table stored in the ROM area of each sensor. A block cipher, such as AES is assumed for the construction of the keyed-hash function. The keyed-hash function is used for the computation/verification of MACs. The SHA-1 algorithm is used for the evaluation of hash values, for random number generation and as the key derivation function kdf. The computation evaluation of our protocol shows a total cost per node of about 633 msec. This cost is about 20% lower than the cost of the hybrid protocol of [5] (760 msec) and slightly better than the protocol of [6] (645 msec), computed with the same metrics.

Cryptographic Action	**Cost/action (msec)**	**Number of actions per node**	
		Node A	**Node B**
Scalar multiplication (random point)	480	1	1
Scalar multiplication (fixed point)	130	1	1
EC addition	3	1	1
Keyed Hash Function evaluation	3	4	4
Hash Function evaluation	2	2	2
Random number generation	2	2	2
Total Computation Cost per node		633	633

Figure 3: Computational costs per node

Communication complexity. The proposed protocol requires a total of 4 message exchanges for key establishment, including the protocol initiation, exchange of MACs and key confirmation. Assuming a node ID is 64 bits, a generation ID and the expiration time are 8 bits,

the Elliptic Curve modulus is 160 bits, the cipher-blocks and MACs are 128 bits, and the random nonces are 64 bits, then the communication cost of the protocol is 1440 bits or 180 bytes, equivalent to the 180 bytes required in [5] and slightly better than the 186 bytes required in [6].

Key storage requirements. Assuming that the nodes are pre-deployed with keys that allow communication with nodes of k generations (including their own generation), the total storage requirements during key pre-deployment are $384 + k \times 400$ bits. We compare the storage costs with the costs of [6].

	Key pre-deployment storage costs per node for k node generations (bits)					
Compared protocols	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$
Kotzanikolaou *et al* [7] $1032 + (k \times 128)$	1160	1288	1416	1544	1672	2312
The proposed protocol $384 + (k \times 400)$ bits	784	1184	1584	1984	2384	4384

Figure 4: Storage per node for k generations

The key pre-deployment storage costs of the protocol allow for a reasonable number k of node generations. For $k \leq 2$ the protocol has lower or almost equal costs with the protocol of [6], while for $k > 2$ the protocol of [6] is lighter in storage costs. For $k = 10$ node generations, the proposed protocol has almost double key pre-deployment storage costs than the protocol of [6]. However, the storage requirements of our protocol are tolerable for applications requiring a limited number of node generations. For example, for $k = 5$ node generations, the key storage costs are 2384 bits or 298 bytes.

This cost includes,the private EC keys of the sensor, the public key of the CA, the base point P and the network-wide initial key K. Note that after each key establishment phase, the sensor node will delete the EC keying material used in this phase. This helps in maintaining an almost constant key space regardless of the generation of pairwise keys. Our protocol is scalable: for a fixed number of generations, the per-node storage and energy resources do not limit the size of the network.

6. Concluding Remarks

In this paper we show that it is feasible to construct authenticated key agreement protocols for uniform DSN's, based on asymmetric (EC) cryptography only, with the same computation and communication cost as in hybrid protocols, such as the Kotzanikolaou et al scheme [6]. The proposed asymmetric scheme improves the security of [6]. More specifically it establishes forward secrecy for past communications in case a node is compromised by an active attacker who is also an eavesdropper. The additional cost of the proposed protocol is the additional storage requirements for each node of the network.

Our protocol supports multiphase node deployment, where nodes may join the network in several time periods and it is scalable, since the number of pre-deployed keys is fixed for a given number of generations and does not depend on the size of the network.

Future work on the specific research area should be focused on more energy-efficient protocols for secure key establishment between sensor nodes in unattended environments. Our findings show that it is feasible to use Elliptic Curve Cryptography as a supplementary security primitive, for special uses of low-energy computing devices. In a future work we will present more specific implementation results.

ACKNOWLEDGEMENT

This Research work is funded by the Ministry of Education and Religious Affairs and co-funded by E.U. (75%) and National Resources (25%) under the Grant "Pythagoras - Research Group Support of the University of the Aegean".

REFERENCES

[1] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of the Advances in Cryptology – Crypto '92*.

[2] D. Brown, R. Gallant, and S. Vanstone. Provably secure implicit certificate protocols. In *Proceedings of the 5th International Conference on Financial Cryptography*, volume 2339 of *LNCS*, pages 156–165. Springer-Verlag, 2002.

[3] Crossbow. Mica2 wireless measurement system datasheet. Available at: http://www.xbow.com/Products/Product_pdf_files /Wireless_pdf/MICA2 _Datasheet.pdf, 2005.

[4] G. Gaubatz, J. P. Kaps, and B. Sunar. Public key cryptography in sensor networks -revisited. In *Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS '04)*, 2004.

[5] Q. Huang, J. Cukier, H. Kobayashi, B. Liu, and J. Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications*, pages 141–150. ACM Press, 2003.

[6] P. Kotzanikolaou, E. Magkos, C. Douligeris, and V. Chrissikopoulos. Hybrid key establishment for multiphase self-organized sensor networks. In *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks WoWMoM '05*, pages 581–587. IEEE Press, 2005.

[7] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.

[8] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proceedings of the 1st IEEE International Conference on Sensor and Ad hoc Communications and Networks*, Santa Clara, California, Octomber 2004. IEEE Press.

[9] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[10] SECG. Standards for efficient cryptography group. SEC 1: Elliptic curve cryptography. Available at: http://www.secg.org/download/aid-385/sec1_final.pdf, 2005.

[11] SHA. Federal Information Processing Standard 180-2: Secure Hash Standard, August 2002.

[12] R. Struik and G. Rasor. Mandatory ECC security algorithm suite. Available at: http://grouper.ieee.org/groups/802/15/pub/2002/May02/02200r1P802-15_TG3-Mandatory-ECC-Security-Algorithm-Suite.pdf, 2002.