Certificate-Based Pairwise Key Establishment Protocol for Wireless Sensor Networks

Pawani Porambage*, Pardeep Kumar*, Corinna Schmitt†, Andrei Gurtov‡ and Mika Ylianttila*

*Centre for Wireless Communications, University of Oulu, P.o.Box 4500, FI-90014 Oulu, Finland
{pporamba, pkumar, mika.ylianttila}@ee.oulu.fi
†Institute of Informatics, University of Zurich, Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
schmitt@ifi.uzh.ch
‡Department of Computer Science and Engineering, Aalto University, FI-00076 Aalto, Finland
gurtov@cs.helsinki.fi

***Abstract*—In order to guarantee the privacy and safety of data transactions in Wireless Sensor Networks (WSNs), secure key transportation and unique node identification have become major concerns. WSNs are deployed in a wide range of applications with a high demand for secure communications. When designing a secure key management protocol for WSNs, special attention should be given to the resource constraints of the devices and the scalability of the network. In this paper, we exploit public-key nature protocols to define a hybrid key establishment algorithm for symmetric key cryptography. We propose an Elliptic Curve Cryptography based implicit certificate scheme and show how to utilize the certificates for deriving pair-wise link keys in a WSN. By a performance and security analysis, we justify that the proposed scheme is well fitting with the functional and architectural features of WSNs. Both experimental results and theoretical analysis show that the proposed key establishment protocol is viable to deploy in a real-time WSN application.**

***Index Terms*—Wireless Sensor Networks, Implicit certificate, Security, Link key establishment, Elliptic Curve Cryptography**

I. Introduction

Wireless Sensor Networks (WSNs) are deployed in wide ranges of applications such as environmental monitoring, health care, industrial automation and battlefields where information security and confidentiality are prime requirements [1]. Depending on the application scenario of the WSN, sensor nodes can be statically located or mobile. The network is constructed with one or few powerful base stations and thousands to millions of sensor nodes, which are inherently resources restricted in terms of memory, battery capacity and computational power [2]. In order to defend malicious attacks and ensure hop-by-hop security, symmetric key encryption is used for secure communication between WSN nodes as motivated in references [3] and [4]. However, it is impractical to incorporate conventional network layer key management protocols in their original formats since they are too expensive for low-power low-performing sensor nodes [3]. Therefore, it is quite challenging to implement an appropriate key management algorithm, which should be compatible with the resource scarcity of sensor nodes and other WSN characteristics.

In this paper, we propose an Elliptic Curve Cryptography (ECC)-based public key cryptography (PKC) solution for secure key management in WSNs. Our main contribution is the design of an implicit certificate-based key establishment protocol for low performing WSNs and the justification of its appropriateness in terms of performance and security strength. We present the real-time implementation results on a simple TelosB senor network in terms of memory and execution time. Moreover, we explain how the new nodes can join the network and change the locations dynamically, obtain certificates, and establish the pairwise keys with the neighboring nodes. The resource consumption and overhead of the protocol are analyzed, in such a way to support the WSN characteristics. The protocol consists of two phases: Authenticated certificate generation for legitimate nodes (*Phase I*) and secure link key establishment (*Phase II*). In *Phase I*, sensor nodes are granted implicit certificates by the resource rich cluster head, which is considered as the Certificate Authority (CA). The sensor nodes send certificate requests to the CA for new certificates or renovating expired certificates. On receiving the certificates, the nodes can calculate their own public and private keys. In *Phase II*, nodes exploit the obtained certificates to establish pair-wise ephemeral keys (i.e., the link keys) with their neighbor nodes. Since we use ECC to design the protocol, it brings an equal security level as RSA, which is a conventional and standardized PKC scheme. However, ECC induces less processing overhead, resource, and time consumptions than RSA [5].

The rest of the paper is organized as follows. Section II presents the related work which motivated our proposed protocol. Section III provides the assumptions, the system model, and the list of notations that we consider relevant. Section IV describes the theoretical design of our proposed certificate-based key establishment protocol. Two phases, certificate generation and link key establishment, are discussed in subsections IV-A and IV-B respectively. Section V gives a performance and security analysis for the proposed protocol. Finally, Section VI concludes the paper and gives future directions for further enhancements of the proposed scheme.

II. Related Work

This section introduces related work that inspired the proposed system model and our developed protocol solution.

WSNs should be designed to maximize the network lifetime, sensing coverage, network connectivity, data delivery ratio and optimum energy consumption [6]. Network structure is modeled according to the sensing model, transmission range, time synchronization, failure model, and location information [7]. In reference [1] Zhou et. al have presented the main security considerations in WSNs as key management, authentication, integrity, availability, secure routing, and intrusion detection. It was pointed out that, the new trends of keying mechanisms for WSNs are the hybrid versions of symmetric and asymmetric key techniques. Three key distribution approaches were addressed as random, deterministic, and location-based. Nevertheless, in all approaches, the nodes have to store the key information of neighboring nodes. Due to the dynamic characteristics of the network and memory restrictions in the devices, these approaches do not fit well for a WSN with millions of sensor nodes. As stated in reference [8] security goals for a particular WSN are depending on its network life time and application scenario. Generally, cryptographic keys play the most important role in initializing any kind of security in WSN. Secure initialization is intrinsic with many other security protocols such as secure routing, authenticated data processing or secure distributed data storage. Therefore, it is significant to have well secured cryptographic keys as link keys for communication channels in WSN. Due to the high resource demand, PKC algorithms such as RSA are not recommended for WSN applications. However, ECC (i.e., a light weight PKC alternative) based security solutions are no more new to WSNs. The utilization of implicit certificates for generating pair-wise ephemeral keys is yet a improving realm. There are several implicit certificate generation schemes for WSNs presented in references [5] and [9]. Elliptic Curve Qu-Vanstone (ECQV) is one of such schemes embedded in ZigBee Smart Energy applications [10].

TinyECC is a stable ECC implementation for WSNs. In reference [11] the authors provide implementation details and measurement results of ECC security schemes in WSNs. Basically, they have produced performance results for Elliptic Curve Digital Signaturing Algorithm (ECDSA) and Diffie-Hellman key establishment (ECDH). Several ECC based security schemes have been proposed for WSNs as published in references [12], [5], [13] and [14]. Specifically in reference [14] the authors suggest a hybrid key establishment algorithm using ECDH and implicit certificates. However, they claim the communication and computation overhead only by a theoretical analysis. The scalability and mobility of the protocol are low since the sensor nodes cannot renovate their certificates after deployment. The communication overhead is also high in their protocol due to four message transactions for the key establishment.

III. System Model, Assumptions and Notations

This section introduces the assumed system model, the assumptions and the notations, which are to be used for the key establishment protocol.

A. System Model

According to reference [2] we consider the standard WSN architecture with cluster tree topology as shown in Figure 1. However, based on WSN applications, different topologies

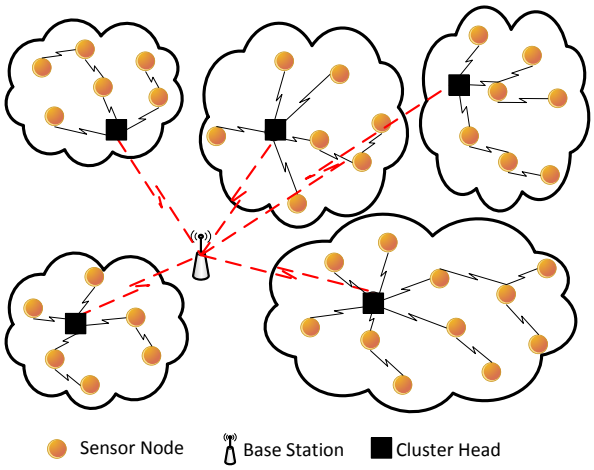

Fig. 1. WSN topology

can be established for both heterogeneous and homogeneous networks. Irrespective of the topology, in every type of data collecting and monitoring WSNs, there is a cluster head (CH) or a network coordinator node. This coordinator node is resource richer than the ordinary senor nodes. This special node is illustrated as a black box in Figure 1 compared with common sensor nodes as circles. The sensor nodes are deployed in predefined clusters under the control of particular CH. Generally, the CH acts as the intermediate coordinator between sensors and the base station. Sensor nodes gather data from their corresponding area and send data to CH or the base station via single hop or multi-hops. There can be scenarios where both sensor nodes and CH can be mobile. In such a scenario, the communication links are very short-term and dynamic changing. For such cases, ephemeral pair-wise link keys are required for secure communication since symmetric key encryption is more cost effective than PKC encryption. The new node addition should be also considered for the network expansion purposes.

B. Assumptions

In our approach, we have five assumptions briefly described in the following. First, comparing with ordinary sensor nodes, CH is a more resource rich entity in terms of memory, battery capacity (or main power) and transmission power. CH performs the role of CA in the particular cluster. Second, Elliptic Curve (EC) parameters, authentication key K, CA's public key (Q_{CA}) and a valid unique identity (ID) are pre-deployed in sensor nodes at the initialization phase (in off-line mode). Third, CA can verify the validity of sensor node identities and recognize whether the sensor node belongs to its cluster or not. Fourth, CH can directly send messages to sensor nodes within its cluster. Sensor nodes can reach the corresponding

CH by single or multiple hops. Finally, physical node capturing attacks are to be identified by beacon messages as explained in references [9] and [4]. When one node is compromised the CH will identify it and broadcast its ID to the cluster. The neighboring nodes of the compromised node will demolish the pre-established pairwise keys with the compromised node.

C. Notations

The notations used in this paper are defined in Table I. EC parameters are denoted by *q, a, b, G, n*. *q* is a prime which indicates finite field F_q. *a* and *b* are coefficients of EC $y^2 = x^3 + ax + b$ where $4a^3 + 27b^2 \neq 0$. *G* is the base point generator with order of *n*, which is also a prime.

TABLE I
NOTATIONS USED IN CRYPTOGRAPHIC ALGORITHMS

Notation	Description
K	Network-wide symmetric key for initial authentication
r_U	Secret random integer value generated by U
R_U	EC point for certificate request sent by node U
$Cert_U$	Implicit certificate of i^{th} node
e	An integer value used to keep hash value of $Cert_U$
s	An integer value used to compute private key of the requestor node
d_U	Node U's private key
Q_U	Node U's public key
N_U	A random cryptographic nonce generated by node U
K_{UV}	Link key between nodes U and V

IV. OUR SOLUTION

We present the key establishment protocol based on the assumptions and the system model described in Section III. The protocol is mainly divided into two phases: (1) Implicit Certificate Generation and (2) Pairwise Link Key Establishment. Every sensor node has to undergo the first phase at bootstrapping phase and certificate revocation instances. The second phase should be executed whenever two neighboring sensor nodes want to establish a pairwise key.

A. Implicit Certificate Generation (Phase I)

Firstly, EC parameters, authentication key *K*, CA's public key (Q_{CA}) and a valid ID are pre-deployed in each sensor node at the off-line mode. *K* is common to all sensor nodes and the CH (i.e., CA) of the cluster. The flow of the certificate generation scheme is illustrated in Figure 2, where gray boxes indicate value ranges and required equations, and white boxes indicate the operations performed by the entities.

The certificate generation process is inspired by the design principles of ECQV implicit certificate scheme as explained in [10]. When a sensor node wants to obtain a new certificate or renovate an existing one, it broadcasts a certificate request message. The request may reach the CA by single or multiple hops. While creating a certificate request, first, the node generates a random number $r_U \in [1, \ldots, n-1]$ and computes $R_U = r_U \times G$. Secondly, the node produces a cryptographic random nonce N_U and computes $MAC_K[R_U, N_U, U]$, where U is the node identity. Then, the node broadcasts U, R_U and N_U along with MAC. When the CA receives the message, it checks the validity of the node identifier (U) and verifies the MAC. If the identifier is legitimate and the verification is successful, the CA generates a random number $r_{CA} \in [1, \ldots, n-1]$ and computes the certificate $Cert_U = R_U + r_{CA} \times G$. Then the CA calculates s using $Cert_U$, r_{CA} and its own private key (d_{CA}); $e = H(Cert_U)$ and $s = er_{CA} + d_{CA} \pmod n$. Value e should be computed using a one-way cryptographic hash function such as SHA. Later, the CA will send a random nonce N_{CA}, Certificate $Cert_U$ and s along with the MAC on $[Cert_U, N_{CA}, s, U]$.

Since the CA has a higher transmission power, it can directly send the message to the requestor node. When receiving this message, the requestor node U first verifies the received MAC and if it is correct U calculates $e = H(Cert_U)$. The identical hash function should be used as in CA. Then the node can compute its own private key $d_U = er_U + s \pmod n$ and public key $Q_U = d_U \times G$. The CA (i.e., CH) only has to participate in this phase of the protocol in two scenarios, node bootstrapping state and certificate renovation state.

B. Pairwise Link Key Establishment (Phase II)

In this phase, two sensor nodes can use the certificates and pre-computed keys to perform the authenticated pairwise key establishment as shown in Figure 3. Assume that U and V sensor nodes are in the same cluster. Node U initiates the key establishment by choosing a random nonce N_U and broadcasting it along with $Cert_U$, identity U and $MAC_K[Cert_U, N_U, U]$. Likewise in *Phase I* MAC is appended for the initial authentication. Once the neighboring node (V) receives the message, it verifies the MAC. If the verification succeeds, the receiver can ensure that U is an authenticated node. Furthermore V can have an implicit assurance that U is a legitimate node of the given cluster by computing sender's public key Q_U using Q_{CA}; $e = H(Cert_U)$ and $Q_U = eCert_U + Q_{CA}$. According to the following derivation, this calculation also gives exactly the same Q_U as computed by the node U.

$$\begin{aligned} Q_U &= d_U G \\ &= (er_U + s(mod\ n))\ G \\ &= (er_U + er_{CA} + d_{CA}(mod\ n))\ G \\ &= e(r_U + r_{CA}(mod\ n))\ G + d_{CA}G \\ &= e(r_U G + r_{CA} G) + Q_{CA} \\ &= e(R_U + r_{CA} G) + Q_{CA} \\ &= eCert_U + Q_{CA} \end{aligned} \quad (1)$$

Then the node V generates a random nonce N_V and sends it along with $Cert_V$, identity V and $MAC_K[Cert_V, N_V, V]$. In the meantime V computes the pairwise key K_{UV} from its private key d_V and U's public key Q_U; $K_{UV} = d_V Q_U$. Similar to V, upon receiving the message, node U verifies the MAC and if the verification is successful, then it computes Q_V and $K_{UV} = d_U Q_V$. Therefore, at the end of two way message transferring, both parties can derive a common pairwise key for actual secure communication.

V. EVALUATION

This section provides a detailed evaluation of our proposed protocol, which is presented in terms of a performance eval-

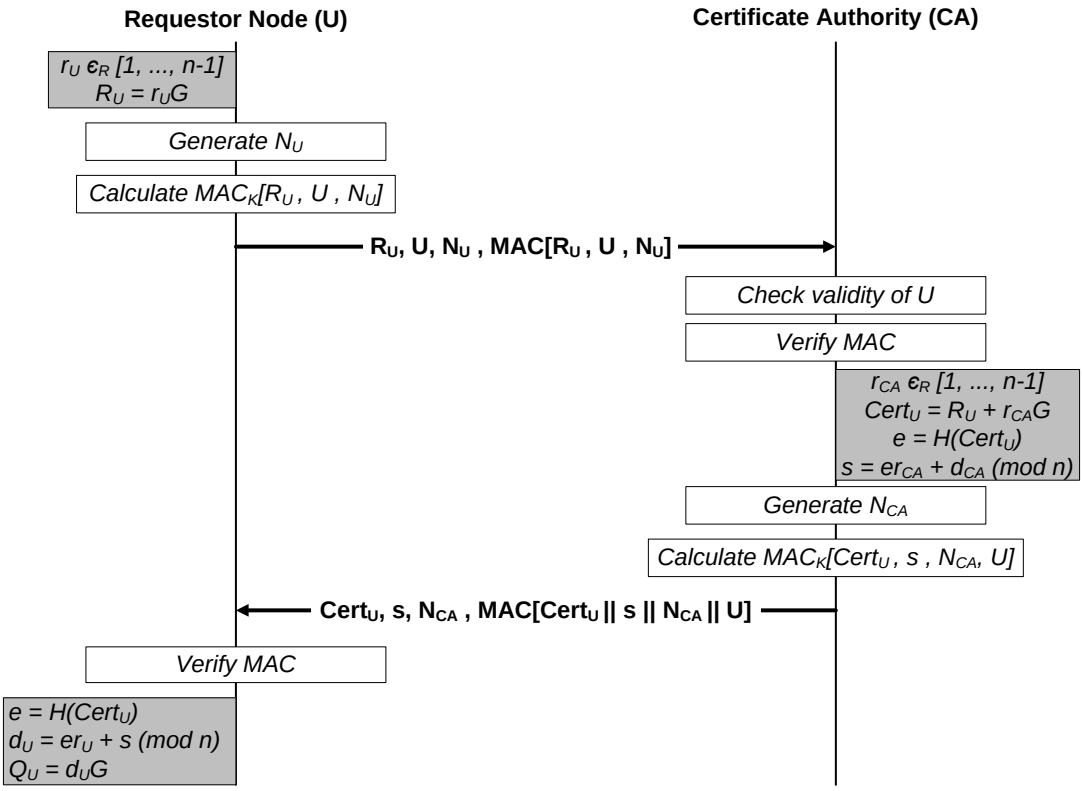

Fig. 2. Implicit certificate generation (Phase I)

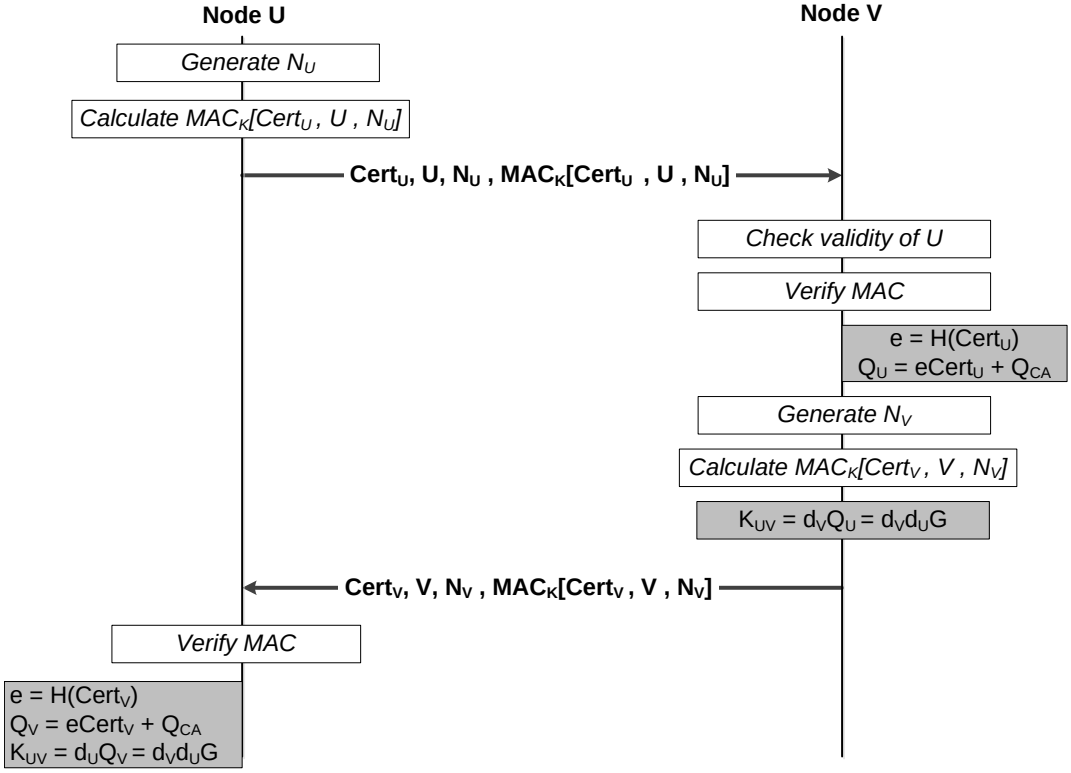

Fig. 3. Pairwise link key establishment (Phase II)

uation, a network scalability analysis, and a security analysis. The subsection V-A presents the numerical measurements we obtained for the memory consumption and timing values of the key establishment protocol. Subsections V-B and V-C contribute a theoretical justification of scalability and security strength of the protocol. Finally, a comparison with existing ECC related approaches is given in Section V-D.

A. Performance Evaluation

Our experimental setup is implemented on TelosB mote platform [15], which has IEEE 802.15.4 compliant CC2420 RF transceivers. The hardware includes 8 MHz, 16-bit MCU with 10 Kbyte RAM and 48 Kbyte ROM. CC2420 RF transceiver has a maximum data rate of 250 kbps and frequency band of 2400 MHz. The proposed scheme is developed in NesC on TinyOS 2.1.2 [16]. ECC (i.e., for EC arithmetic operations) and natural number (NN) (i.e., for large natural number operations) interfaces are utilized from TinyECC configurable library [11]. *secp160r1* EC domain parameters are used as defined in [17]. TinyECC provides EC optimization techniques such as Barrett Reduction to speed up modulo operations, Hybrid multiplication and squaring for integer multiplication, Projective Coordinate Systems for the point addition, Sliding Window for scalar multiplication, and Shamir's trick for summing two scalar multiplications.

The experimental setup comprises three TelosB nodes, one as the CA and the rest as the cluster nodes. For the sake of simplicity and comparison, CA functionalities are also implemented on a sensor node itself. The measurements are taken in terms of execution time and memory (i.e., RAM and ROM) consumption. ECC operations are extremely costly than other cryptographic operations (i.e., MAC, SHA-1) [11]. Therefore, we have considered three different techniques of EC operation optimizations (i.e., provided with TinyECC) for taking time and memory readings: *Disable all the optimization techniques, enable all the optimization techniques,* and *disable Shamir's trick*. SHA-1 is used as the one-way cryptographic hash function. The memory consumption values are measured for requestor operations, CA operations and pairwise key calculation with three optimized techniques. The *check_size.pl* script is used to obtain RAM and ROM sizes required by each operation. The execution times are measured directly on the sensor nodes for the collective operations such as protocol initialization, request generation, certificate generation, certificate verification and pairwise key computation.

As depicted in Figures 4 and 5, when the EC optimization techniques are disabled, the memory consumptions of ROM and RAM sizes are very small for all three operations (i.e., Requestor operations, CA operations and Pairwise Key Calculation).

After enabling EC optimization techniques, massive increments can be seen for the memory utilization values of all operations. Nevertheless, by disabling Shamir's trick, we can save 802 bytes of ROM size and 676 bytes of RAM size for every operation. Key calculation consumes lower memory size

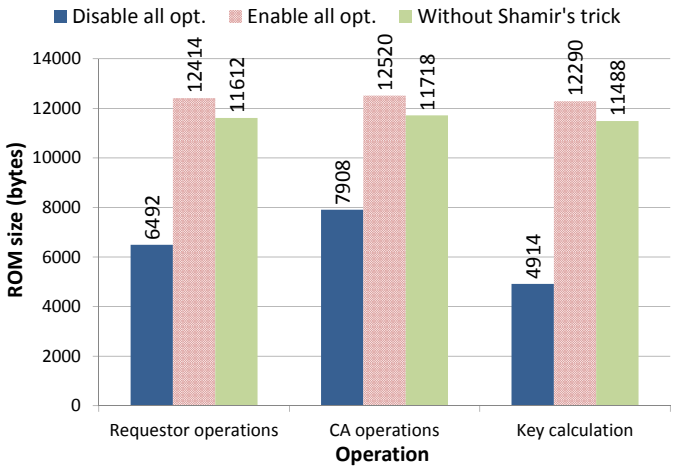

Fig. 4. ROM size of certificate generation and key calculation

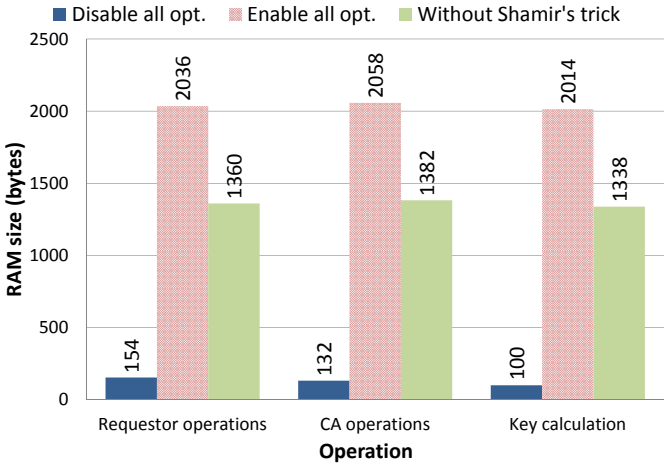

Fig. 5. RAM size of certificate generation and key calculation

than the certificate generation operation. However, for last two optimization techniques (i.e., enabling all EC optimizations and disabling Shamir's trick only), for each case separately, the ROM consumption values are showing slight variations around a common value for all three operations. A similar behavior can be seen for the RAM utilization.

TABLE II
EXECUTION TIME

Operation	Disable all opt. (ms)	Enable all opt. (ms)	Without Shamir's trick (ms)
Initialization	1	5227	2709
Certificate request generation	41799	2755	2764
Certificate generation	86377	5724	5728
Certificate verification	42129	2755	2758
Key computation	86891	5767	5768

Table II shows the execution time of distinctive operations

(i.e., protocol initialization, request generation, certificate generation, certificate verification and pairwise key computation) under three aspects of EC optimization techniques. Once we disable all the EC optimization techniques, the algorithm initialization time has become very close to zero. However, initialization takes 5227 ms when all the optimizations are enabled. Without Shamir's trick, the initialization time is 2709 ms, which is approximately two times faster than with all the optimizations. Contrasting to initialization operation, all the other operations have been speed-up 20 times faster by enabling EC optimization techniques.

When we consider both execution time and memory allocation, it is clear that the optimizations accelerate the operations of the protocol though they consume reasonably large memory. However, only by eliminating Shamir's trick, we cannot impose a significant impact on the overall performance of the protocol. Therefore, in cooperation, the best solution for our key establishment protocol is to enable all the other techniques except Shamir's trick (i.e., third technique). According to the last optimization aspect, the sensor node takes 8231 ms for protocol initialization, request generation and certificate verification. On CA's side, the execution time is 8437 ms for protocol initialization and certificate computation. Therefore the computation time for the certificate generation phase at both ends is 16668 ms. The key establishment phase expends 8477 ms for initialization and key calculation. From the given observations, we can claim the feasibility of deploying the proposed scheme is viable in wireless sensor nodes. Memory costs for our protocol are tolerable by the extreme resource constrained sensor nodes such as TelosB mote platform. Moreover, timing values can be reduced with the further improvements of more efficient implementation of ECC basic operations.

In both, certificate generation phase and key establishment phases, the communication cost is restricted to two way message transfers for the purpose of minimizing the communication overhead. Since each message contains an EC point (44 bytes), node ID (2 bytes), random nonce (4 bytes), and MAC value (8 bytes), the average message size is about 58 bytes. Therefore, we have changed the default size (29 bytes) of the data of *message_t* header field in TinyOS 2.1.2, according to the protocol requirement (58 bytes). Similar to memory and time consumption optimizations, we can further reduce the message sizes by using well designed EC curves.

B. Scalability

Our protocol supports the scalability of the network (i.e., expanding the network with the new node addition) and the location changes of the sensor nodes with in the same cluster. When a new node is added to the network, a valid node identity, keying information (i.e., K and Q_{CA}) and EC domain parameters should be stored while the node is at the off-line mode. Figure 6 illustrates how our protocol supports a new node addition to the network, within a particular cluster. In *Stage 1*, at the bootstrapping phase, the newly added node (marked white double circled) can send the certificate request and obtain a certificate from the CA for computing its own keys. Therefore, the size of the network is not necessary to be pre-defined during the initial design phase and the deployment phase.

After receiving a new node request, the CA only needs to verify the validity of the sensor node identities to issue the certificate. In *Stage 2*, the new node receives its certificate. Finally in *Stage 3*, the node can establish the pairwise link keys with its neighbors, using that received certificate.

Similarly, the sensor nodes do not need prior knowledge about their neighbors. Whenever a new node is added to the network or it changes the neighboring set, it can establish the ephemeral pairwise link keys, with the corresponding neighbors using the certificate. The certificates always provide an implicit assurance for the sensor nodes that they are authenticated nodes in the given cluster. Even though the sensor nodes frequently change their locations (i.e., also the neighboring set), they can derive the pairwise keys securely without previous awareness of the new neighboring nodes. According to reference [1], if the pairwise keys between neighbors are pre-installed, then there should be a large number of stored keys per node. This may not be desirable for the large scale networks. However, in our protocol such a large scale key pre-installation is not needed at all since the ephemeral link keys have to be established before starting communication. Furthermore, the keys are derived based on their certificates which are exchanged during the initial handshake.

C. Security Analysis

Our proposed certificate based key establishment protocol is developed using one of the lightest PKC schemes ECC. Though it is comparatively more expensive than symmetric key algorithms, it is inherently secured due to the PKC characteristics. However, we have shown in the above section, that the proposed scheme is feasible to deploy in real-time WSNs. While using EC scalar-point multiplication, the scheme is provably secured under the random oracle model that the discrete logarithm problem over the subgroup is untractable. The proposed pairwise key establishment extends the security strength of the standard ECDH key agreement, by using mutually authenticated keying materials (e.g., $Cert_U$ and s). Since the link keys are derived using two pre-evaluated values (e.g., d_U and Q_V), it is implicitly assured the legitimacy and trust between two parties. In order to overcome illegal message alternations by malicious nodes and denial of service attacks (DoS), every message contains MAC with the common authentication key K for preserving data integrity. The availability of the proposed protocol is ensured by giving permission to two legitimate nodes, which possess the certificates granted from the CA, to establish a secure pairwise key for their mutual communication. The freshness of the messages is guaranteed by appending true nonce. In *Phase I*, the origin of the message (i.e., CA) cannot deny being sent the message (i.e., non-repudiation property) since the CA uses its key pair to generate the certificate and private key reconstruction value (s). Likewise, during the key establishment phase, the sender of the messages cannot deny that the messages are sent by itself since the receiver always uses the certificate of the sender to derive its (i.e., the sender's) public key.

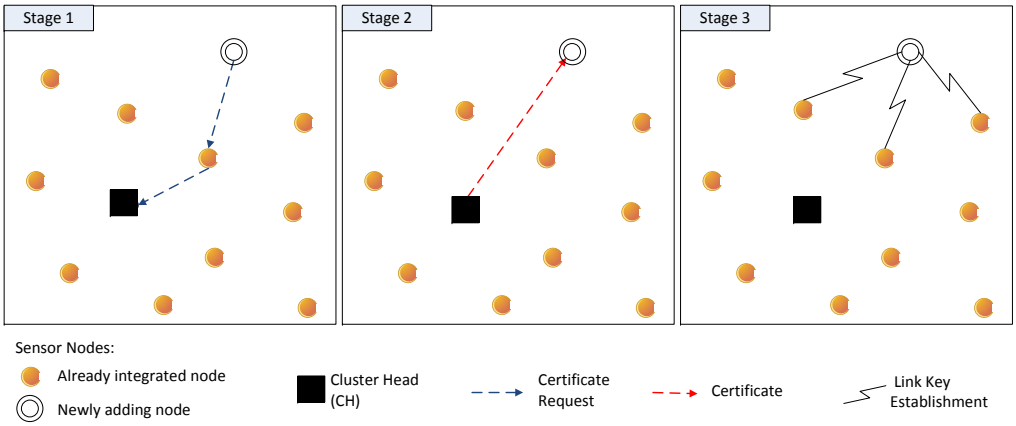

Fig. 6. Behavior of the protocol when a new node enters the cluster.

In the security analysis, we are considering three attacks including, node compromising attacks, masquerade attacks and impersonate attacks. In node compromising attacks, an adversary can physically capture a node and obtain its keys. Similarly *in Phase II*, an attacker can impersonate a legitimate sensor node using its certificate or try to masquerade the key establishment between two legitimate nodes.

Node compromise attacks

Our protocol is resilient to node compromise attacks. If a node U is captured, the adversary can reveal $Cert_U$, Q_U, d_U and Q_{CA}. However, with CA's public key, an adversary can not create a new valid certificate and private key reconstruction data, since they are derived using d_{CA}, which is only known to CA. Though a node pretends to be a forgery CA and issues certificates with Q_{CA}, eventually the fake certificates and public keys are disclosed during the key establishment phase (i.e., calculating $K_{UV} = d_U Q_V$). We assume that the CH can identify compromised nodes using beacon message technique, as explained in references [9] and [4]. Then the CA will broadcast the compromised node ID to the non-compromised nodes. Upon receiving CA's message (i.e., compromised node ID), the other sensor nodes will discard the certificate of the corresponding node and the pre-established pairwise keys. Then the compromised node cannot appear itself as a legitimate node in future, because its certificate and node ID are already abandoned by the legitimate nodes.

Impersonate and Masquerade attacks

In the key establishment phase, nodes are authenticated in order to prevent impersonation attacks and masquerade attacks. The node V computes the node U*'s* public key using its $Cert_U$ and CA's public key Q_{CA}. The pairwise key K_{UV} calculation at both ends will be the same, only if the certificates are issued by the valid CA. Therefore the node V has an implicit assurance that the received certificate is genuine (i.e., issued by the CA). Likewise, when two legitimate nodes initiate a pairwise key, an attacker (without a valid certificate) cannot come in between them and masquerade the key establishment. Since the pairwise key is derived on the basis of the certificates and private keys of both legitimate parties, an attacker cannot proceed it by only using a valid certificate.

D. Comparison With Related Work

The development of our proposed key establishment protocol is inspired by the different ECC based security schemes, which are presented in the Section II. Therefore, in this section we compare the certificate-based pairwise key establishment protocol with the related work.

In reference [11], timing and memory utilization values are presented for three ECC schemes. Among them, ECDSA and ECDH are the most related schemes to our proposed protocol. Table III gives the comparison results between ECDSA scheme and the proposed certificate scheme on behalf of the sensor node and CA. All the empirical results are measured on TelosB sensor nodes. For the sake of comparison, we have considered the enabling of all the ECC optimization techniques except Shamir's trick (i.e., third technique). At both ends our scheme is more efficient than the conventional ECDSA scheme in terms of memory consumptions and timing values. Similarly, Table IV shows the comparison results between ECDH scheme and the proposed key establishment protocol. The values witness the high performing capability of our scheme in the resource constrained sensor nodes.

TABLE III
COMPARISON OF PROPOSED CERTIFICATE SCHEME WITH ECDSA SCHEME

	ECDSA scheme [11]	Proposed solution	
		Requestor operations (sensor node)	CA operation (CH)
ROM (bytes)	12640	11612	11718
RAM (bytes)	1586	1360	1382
Time consumption (ms)	14789	8231	8437

In security aspects, conventional ECDH scheme is vulnerable to impersonate and masquerade attacks since two communicating parties do not have an authentication phase

TABLE IV
COMPARISON OF PROPOSED KEY ESTABLISHMENT SCHEME WITH ECDH SCHEME

	ECDH scheme [11]	Proposed key establishment scheme
ROM (bytes)	12102	11718
RAM (bytes)	1866	1382
Time consumption (ms)	6146	5768

during the key establishment. However, as explained above our key establishment is well secured at both types of attacks. Originally, ECDSA and ECDH schemes do not address the possibility of network scalability. However, in the paper we have analyzed how the proposed scheme supports the scalability of the network. Therefore, the authors of this paper believe that the proposed solution extends the existing pool of security solutions are concerned with ECC and can optimize the key establishment in WSNs.

VI. CONCLUSION

In this paper, we introduced a certificate based pairwise key establishment protocol for WSNs. The proposed key management scheme comprises two phases: For providing certificates for the resource constrained sensor nodes and establishing pairwise link keys for mutual node communication. The security protocol is a PKC based solution used for deriving a common secret key for symmetric key encryption. The novelty is the utilization of implicit certificates for generating pairwise keys. Our experimental results show the feasibility of deploying the proposed scheme in an actual resource constrained WSN. However, the further optimized EC operations may incur less resource consumptions on sensor nodes and accelerate the protocol execution. Moreover, we have discussed and justified the appropriateness of the protocol for the resource utilization and scalability of WSN. Though there is a simple concept behind the proposed scheme, the security analysis has proven the robustness of the protocol for different security pitfalls.

In future, we intend to extend this protocol by changing the content of the certificate in such way to provide higher security for mobile sensor nodes in massive scale IoT networks. We can customize the content of the implicit certificates by adding other information such as the time stamp, location identity or IPv6 over Low power Wireless Personal Area Network (6LoWPAN) identity, depending upon the application requirements. Furthermore, we intend to extend the utilization of implicit certificates for group key management in large scale sensor networks.

ACKNOWLEDGEMENT

This work has been supported by Tekes under Massive Scale Machine-to-Machine Service (MAMMotH) project and Academy of Finland project SEMOHealth.

REFERENCES

[1] Y. Zhou, Y. Fang, and Y. Zhang, "Securing Wireless Sensor Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 10, no. 3, pp. 6–28, 2008.

[2] "IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Std 802.15.4. 2011(Revision of IEEE Std 802.15.4-2006), 2011.

[3] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A Survey of Key Management Schemes in Wireless Sensor Networks," *Comput. Commun.*, vol. 30, no. 11-12, pp. 2314–2341, Sep. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2007.04.009

[4] S. H. Jokhio, I. A. Jokhio, and A. H. Kemp, "Node Capture Attack Detection and Defence in Wireless Sensor Networks," *Wireless Sensor Systems, IET*, vol. 2, no. 3, pp. 161–169, 2012.

[5] K. Malasri and L. Wang, "Design and Implementation of a SecureWireless Mote-Based Medical Sensor Network," *Sensors*, vol. 9, no. 8, pp. 6273–6297, 2009.

[6] X. Li, Y. Mao, and Y. Liang, "A Survey on Topology Control in Wireless Sensor Networks," in *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision*, ser. ICARCV, 2008, pp. 251–255.

[7] Z. Gengzhong and L. Qiumei, "A Survey on Topology Control in Wireless Sensor Networks," in *Proceedings of the 2nd International Conference on Future Networks*, ser. ICFN, 2010, pp. 376–380.

[8] S. Stelle, M. Manulis, and M. Hollick, "Topology-Driven Secure Initialization in Wireless Sensor Networks: A Tool-Assisted Approach," in *Proceedings of the 7th International Conference on Availability, Reliability and Security*, ser. ARES, 2012, pp. 28–37.

[9] R. Lu, X. Li, X. Liang, X. Shen, and X. Lin, "GRS: The Green, Reliability, and Security of Emerging Machine to Machine Communications," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 28–35, 2011.

[10] "SEC4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV), version 0.97," www.secg.org, August 2013.

[11] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proceedings of the 7th international Conference on Information Processing in Sensor Networks*, ser. IPSN. IEEE Computer Society, 2008, pp. 245–256. [Online]. Available: http://dx.doi.org/10.1109/IPSN.2008.47

[12] Y. Liu, J. Li, and M. Guizani, "PKC Based Broadcast Authentication using Signature Amortization for WSNs," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 2106–2115, 2012.

[13] X. H. Le, S. Lee, I. Butun, M. Khalid, R. Sankar, M. (Hyoung-IL) Kim, M. Han, Y.-K. Lee, and H. Lee, "An Energy-Efficient Access Control Scheme for Wireless Sensor Networks based on Elliptic Curve Cryptography," *Journal of Communications and Networks*, vol. 11, no. 6, pp. 599–606, 2009.

[14] P. Kotzanikolaou and E. Magkos, "Hybrid Key Establishment for Multi-phase Self-Organized Sensor Networks," in *Proceedings of the 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, ser. WoWMoM. IEEE, 2005, pp. 581–587.

[15] "TelosB Datasheet," Crossbow Inc., Tech. Rep., 2013. [Online]. Available: http://www.datasheetarchive.com/4--Crossbow*-datasheet.html

[16] "TinyOS Documentation," www.tinyos.net, August 2013.

[17] D. Hankerson, S. Vanstone, and A. J. Menezes, *Guide to Elliptic Curve Cryptography*. Springer, 2004.