TinyIBE: Identity-Based Encryption for Heterogeneous Sensor Networks

Piotr Szczechowiak and Martin Collier

School of Electronic Engineering, Dublin City University
Glasnevin, Dublin 9, Ireland
{piotr,collierm}@eeng.dcu.ie

***Abstract*—The area of security for Heterogeneous Sensor Networks (HSNs) is still an open research field that requires new cryptographic solutions. Recent results have demonstrated that Elliptic Curve Cryptography (ECC) and Pairing-Based Cryptography (PBC) are computationally feasible on sensor devices. This allows a wide range of novel security mechanisms, like Identity-Based Encryption (IBE), to be considered for Wireless Sensor Networks (WSNs).**

In this paper we present an efficient security bootstrapping mechanism for HSNs that uses IBE and exploits the enhanced capabilities of high-end cluster heads. Our asymmetric security scheme provides authenticated key distribution without using expensive certificates. It also achieves significant savings in communication overhead and in the number of keys stored on sensor devices. We also present TinyIBE, which is to our knowledge, the first implementation of a complete identity-based encryption scheme for sensor networks. Our evaluation results and comparison with the state of the art show that TinyIBE is a superior security scheme for HSNs which provides affordable public key cryptography without requiring hardware acceleration. With this work we prove that ID-based encryption is not only possible on sensor nodes but is an attractive security solution in this application space.

I. Introduction

Wireless sensor networks have gained much attention over the last few years. They allow us to gather precise information about the surrounding environment and sense data from previously inaccessible areas. One of the biggest advantages of sensor networks is their versatility - many configuration modes and various types of sensors that make them suitable for a wide range of applications. This versatility presents challenges when it comes to network protocol design for WSNs. A solution designed specially for a flat network topology is unlikely to be the right choice in a clustered network scenario. Optimal solutions should be tailored to a specific network organisation and this holds true also in case of security protocols.

The security of wireless sensor networks has been the subject of extensive studies in recent years. Most of the work in this area was focused on flat and homogeneous networks where all sensor nodes have the same capabilities in terms of memory storage, CPU power, transceiver speed and energy supply. In such networks all nodes perform similar tasks such as sensing, routing and data processing. However, theoretical calculations and experimental results [1] have shown that homogeneous networks have lower performance and scalability than heterogeneous networks. They are also less energy efficient and suffer from high communication and storage overheads [1]. Recently deployed sensor networks are starting to follow heterogeneous design [2].

A heterogeneous sensor network is a hierarchical WSN that is typically organised into clusters that consists of two types of nodes. More powerful devices (H-sensors) usually take the role of Cluster Heads (CHs) and low-end nodes (L-sensors) become ordinary cluster members. L-sensors are mainly responsible for sensing whereas H-sensors perform additional tasks such as data collection, information processing and reporting to the Base Station (BS). The role of CH is fixed in this kind of a network. The dominating traffic pattern in HSNs is many-to-one, where L-sensors send data to their CHs.

Many existing key distribution schemes for WSNs are designed to set up pairwise keys among nodes without considering the actual communication pattern (e.g. [3]). The L-sensors communicate with only a small number of their neighbours and they do not need to maintain shared keys with all of them. Based on this observation we propose a new security bootstrapping mechanism that takes advantage of the processing capabilities of H-sensors. It uses ID-based encryption and requires small number of keys in the network. Our implementation results show that by using efficient public key techniques we can simplify the key distribution process, significantly reduce the communication overhead and lower the energy consumption to prolong the network lifetime.

II. Security considerations

The main security goals for HSNs include data privacy, integrity and freshness, availability, access control, information and entity authentication. The first goal can be fulfilled by incorporating a link layer security mechanism (e.g. TinySec [4]). Access control should allow only legitimate nodes to join clusters and become CHs. Guaranteeing availability involves minimising the impact of DoS attacks. Authentication ensures the receiver that the message did in fact originate from the claimed sender. In many cases the confidentiality of simple sensor readings is not as important as the origin of the data.

All of the above security requirements can be fully addressed only by building upon a solid key distribution framework. Key management is an essential cryptographic mechanism upon which other security primitives are built. The main focus of this paper is to design a solid and scalable key distribution scheme for HSNs that solves the shortcomings of existing solutions.

978-1-4244-3518-0/09/$25.00 © 2009 IEEE

A. Possible attacks

Like any wireless networks HSNs are vulnerable to different kinds of attacks including jamming, eavesdropping and spoofing. There are also additional attacks that are exclusive to sensor networks. Deployments in public areas introduce a risk of a physical attack and limited battery power opens the door for a whole range of DoS attacks resulting in a node's energy depletion. HSNs that use node identities are also prone to replication and Sybil [5] attacks.

Perhaps the biggest problem in this kind of networks is the node capture attack. L-sensors are typically low-cost devices that do not have secure storage for cryptographic keys or tamper-proof hardware. An active attacker may easily subvert a node, intercept messages and decode them using the derived secret key. Recent results show that standard sensor nodes, such as MICA2 can be compromised in less than one minute [6]. Capture attacks involving CHs are particulary damaging in HSNs, because these nodes are responsible for critical functions such as data processing and routing. H-sensors store more cryptographic keys and they are the primary target for possible attackers. Should an adversary manage to become a CH, it gains access to valuable information and can stage a number of different insider attacks targeting nodes within the cluster and across the network. Insider attacks are a lot harder to prevent and detect because they are performed by nodes that are regarded as legitimate members of the network. Active attacks like sinkhole, selective forwarding and false message injection can be especially dangerous and may even lead to complete network failure.

III. Related work

Key management in sensor networks has been an active research topic in recent years. Many authors proposed random key pre-distribution schemes [3], [7]. These approaches are promising for small size networks but they do not scale well as they require that each node is pre-loaded with a large number of keys. Most of the key pre-distribution schemes were designed for homogeneous sensor networks and once applied in HSNs they suffer from high communication and storage overhead. Du et. al [8] tried to address these issues by proposing an asymmetric key pre-distribution scheme that uses the capabilities of H-sensors. The authors of [9] and [10] also tried to adopt key pre-distribution schemes to fit heterogenous network design. However, all these solutions are based on symmetric key techniques and they do not provide a perfect trade-off between resilience and storage.

Compared with symmetric key cryptography, Public Key Cryptography (PKC) provides a more flexible and simple interface, without the need for key pre-distribution, pair-wise key sharing or complicated one-way key chain schemes. Gura et al. [11] demonstrated that ECC is computationally feasible in WSNs. Furthermore Hu et al. [12] showed that even RSA is possible on sensor nodes under the condition that we use a dedicated hardware accelerator for cryptographic operations.

Despite the fact that public key primitives are computationally feasible on sensor nodes, protocols based on them are not. They require exchange and storage of large keys, which is especially expensive for L-sensors. Most PKC schemes needs also authentication of public keys which is usually performed by expensive certificates and digital signatures.

Recent papers [13], [14] have demonstrated fast and efficient implementations of PBC primitives on a range of WSN platforms. This fact enables practical implementations of IBE schemes and thus opens new ways for achieving security in sensor networks. The authors in [15], [16], [17] envisioned the use of IBE as a security solution for WSNs. All three papers proposed the Boneh and Franklin IBE scheme [18]. However the feasibility of such solution for practical applications was not tested. None of the works have implemented a complete IBE scheme on sensor devices.

IV. Proposed scheme

Our scheme is a PKC based algorithm and is much more computationally expensive than standard symmetric key algorithms. However we only use it as a bootstrapping mechanism to distribute pair-wise keys after network deployment.

A. Preliminaries

In our model of HSN we assume that the network will consist of a large number of L-sensors and only a small number of H-sensors. Nodes are static and they do not move after deployment. L-sensors are cheap devices that do not have tamper-proof hardware. It is possible for an active attacker to compromise any L-sensor and extract all its secrets. The CHs store valuable data and keying material, which makes them especially attractive for attackers. We cannot assume that every H-sensor will be tamper-resistant, as this would be too expensive in most deployments. The best that we can hope for is that H-sensors will incorporate a cost effective memory protection mechanism which deletes all the secrets once it detects on chip debugging (as proposed in [6]). Our model assumes also that the environment is insecure from the beginning and the attacks are possible right after network deployment (as oppose to [9]).

The base station does not have to be connected to the network at all times. It can be mobile and we assume that it is well protected from physical attacks. In our scheme the base station can establish a secure connection with any node in the network using its master secret. It is also not possible for an attacker to masquerade as a BS without the knowledge of the master key (see section IV-C).

B. Identity-Based Encryption

IBE is a public-key cryptosystem that was designed to solve some of the problems of traditional PKC algorithms. The main idea is that known information that uniquely identifies users (e.g. email address) can be used to derive public keys. Instead of obtaining public keys we can simply generate them and use them to encrypt messages for given entities. The system ensures that only the legitimate recipient can decode such a message. As a result, keys are self-authenticated and additional

means of public key authentication (e.g. certificates) are unnecessary. This system is promising especially for WSNs where we cannot afford a conventional public key infrastructure. The advent of PBC enabled the first practical IBE system proposed by Boneh and Franklin in [18]. This scheme was based on the Weil pairing but also other types of pairings like the Tate or η_T pairing can be used. For a comprehensive mathematical background on pairings please refer to [19].

The main advantage of IBE over traditional PKC systems is that we do not have to store so many public keys. We can derive them from nodes identities that are widely known in the network. We generate a public key for a given node only in case when we want to communicate with it for the first time. IBE allows us to send secure messages without any prior interaction with a given sensor node, thus the communication overhead is limited to minimum. IBE provides all the security services of a traditional public key system but in a more elegant way without the assistance from third party.

ID-based schemes assume the existence of a trusted authority often called the Private Key Generator (PKG). The PKG can use its master key to decrypt any user message. It also has the ability to impersonate anybody in the system. For that reason the PKG must be well protected and unconditionally trusted by all network users. In most systems such an entity simply does not exist. Fortunately in sensor networks the original network deployer is a trusted authority that can play the role of the PKG. Actually it might even be regarded as an advantage that the deployer is in a position to monitor all wireless traffic. The PKG can also generate a unique secret key based on the node's identity and pre-load this information to the node's memory before the deployment phase. At this stage a secure channel clearly exists which allows careful configuration of the network.

C. TinyIBE scheme

Previous proposals for using IBE in sensor networks [16], [17] suggested the Boneh and Franklin scheme [18] for key distribution. However this protocol is symmetric in nature and requires expensive operations for both encryption and decryption. In our security solution we use a simple variant of the Sakai and Kasahara IBE scheme [20]. This IBE method is not as well-known as that of Boneh and Franklin but it has its advantages that perfectly fit the heterogenous sensor network setting. No pairing calculation is required for encryption and we do not have to hash identities to elliptic curve points. These potentially expensive operations are avoided, which makes the whole scheme more affordable especially for the low-end sensor nodes.

Our TinyIBE scheme can be divided into two phases each consisting of two stages. In the pre-deployment phase PKG/BS performs Setup and Extract operations. This is done off-line in a secure environment:

Setup: The BS generates global system parameters that include its own master secret $s \in \mathbb{Z}_q^*$, a pairing-friendly supersingular elliptic curve E over $\mathbb{F}_q$ and random points $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_q)$ of order r, where $Q = sP$. He generates also $g = \hat{e}(P, P)$ and defines two hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \mathbb{F}_q \rightarrow \{0,1\}^n$, for some n. All nodes are pre-loaded with system parameters: $< E/\mathbb{F}_q, Q, P, g, n, H_1, H_2 >$.

Extract: The BS issues a unique identity ID_X for each node. The identity of every H-sensor is hashed to a value $a = H_1(ID_H)$ and a corresponding private key is generated $D = \frac{1}{s+a}P$. Each H-sensor is also pre-loaded with a broadcast key pair $< B_H, E_H >$ and identities of all the L-sensors in the network (ID_L).

In the above notation sP is a point multiplication operation and $\hat{e}(P, P)$ is a pairing calculation. A pairing is a mapping function between two groups of elliptic curve points. It has the special property of bilinearity: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. The second phase of TinyIBE scheme starts after network deployment and consists of two main stages:

Encrypt: Every L-sensor selects its primary CH, finds the value $a = H_1(ID_H)$ for a given H-sensor and generates random $w \in \mathbb{Z}_q^*$ and $t \in \mathbb{Z}_q^*$. To encrypt the session key t L-sensor creates the ciphertext C_1, C_2:

$$C_1 = w(Q + aP) \qquad C_2 = t \oplus H_2(g^w)$$

He also includes his identity in the message.

Decrypt: After receiving the encrypted message every CH checks the identity of the L-sensor. The decryption process can only start in case of a positive ID check. H-sensor recovers the session key t by using his private key D:

$$t = H_2(\hat{e}(D, C_1)) \oplus C_2$$

The session key is used with a symmetric cipher to encrypt the broadcast public key B_H. The decryption process works thanks to the bilinearity property of the pairing:

$$\hat{e}(D, C_1) = \hat{e}\left(\frac{1}{s+a}P, w(Q + aP)\right) =$$
$$= \hat{e}\left(P, sP + aP\right)^{\frac{w}{s+a}} = \hat{e}\left(P, P\right)^w = g^w$$

After the decryption step every L-sensor can use its session key to securely communicate with a given CH. He can also authenticate broadcasts by using the Elliptic Curve Digital Signature Algorithm (ECDSA). Secure broadcasts are important when remote network reprogramming is needed.

Figure 1 presents all the TinyIBE operations that are performed after network deployment. The need for ID exchange between nodes cannot be considered an overhead incurred by the key distribution protocol. The majority of existing WSN routing algorithms already requires that nodes know each other IDs. We also assume that H-sensors are equipped with a lot more storage space than L-sensors. This allows them to store all the L-sensor identities without introducing too much overhead. Additionally, ID sizes are negligible when compared to public key and certificate sizes. It is more convenient to maintain a list of eligible L-sensor identities rather than storing all their public keys - like in [21].

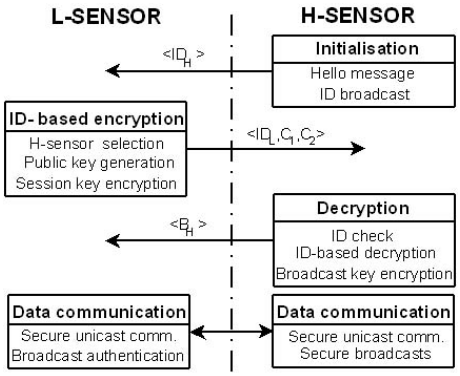

Fig. 1. Key distribution during the on-line phase of TinyIBE.

TinyIBE provides authenticated key distribution without using certificates. The message encrypted for a specific CH can only be decrypted with a correct private key D which is in the possession of a single H-sensor and the BS. Every H-sensor authenticates himself to any L-sensor simply by decrypting the message and sending back data encrypted with the session key. This prevents L-sensors from sending sensor readings to a fake CH. Even if an adversary steals an identity of a node or takes fake identities he still cannot act as a CH or a BS without obtaining the proper private key D or the master secret s. Those values are well protected and our protocol guarantees that the master secret is not revealed even if an attacker subverts all the nodes in the network.

It is also important to prevent fake L-sensors from taking part in the network. H-sensor can only setup a shared key after checking L-sensor's identity. If the ID is not on the list CH contacts the BS to clarify if this ID is valid. In case when two L-sensor claims the same ID, the H-sensor informs the base station of a possible adversary in the network. The communication between the BS and H-sensors is secured using the Encrytp/Decrypt mechanism described earlier. Both parties use ID-based encryption, so the communication is authenticated in both directions. The BS can also perform periodic checks on the lists of nodes connected to different H-sensors. In this way the BS can detect intruders that claim the same IDs in various parts of the network.

New node addition is easy using TinyIBE and does not require any new keying material for existing nodes. The BS performs the Setup and Extract operations and deploys a node with a new ID. The H-sensor will contact the BS to validate the new ID during the session key establishment phase. Any external party can deploy its own L-sensors and add them to an existing network. They only have to agree on the identities with the BS using a secure channel. When an L-sensor becomes an orphan he can quickly join another cluster by encrypting a message for his secondary CH. In the case of node compromise the BS can revoke keys by simply removing the ID and the corresponding session key from the list that is held by every CH. This prevents an L-sensor from establishing a secure connection with any H-sensor of his choice.

V. TinyIBE implementation

In order to check the feasibility and performance of TinyIBE we decided to implement it on three commonly used sensor devices. We used the Imote2 platform as an H-sensor and we modeled L-sensors as the MICAz and Tmote Sky nodes. All three platforms use the same CC2420 radio transceiver which allows them to communicate within the same frequency band.

There is a clear correlation between the level of security achieved, and the processing power required. Sensor networks should use a minimum 80-bit security level as it is the current standard in information security. In order to achieve a similar security level for pairing based cryptography we need to work with values that have more than 1024-bits. This fact makes the pairing computation the most expensive operation in TinyIBE. The pairing calculation in our scheme is performed only by the more powerful H-sensors during the decryption process. This allows us to save on code space and execution time on low-end devices.

For the pairing implementation we chose the η_T [22] algorithm. Our choice is motivated by our recent results [14] that demonstrated the efficiency of the η_T implementation on embedded sensor devices. The η_T pairing is evaluated over a binary field $\mathbb{F}_{2^m}$ where m in our case defines the length of the binary polynomial that represents field elements. In accordance with the required security level we chose a supersingular elliptic curve $y^2 + y = x^3 + x$ over the binary field $\mathbb{F}_{2^{271}}$ with the embedding degree $k = 4$. Due to space limitation we cannot present all the details regarding the pairing implementation. For a detailed description and parameters explanation please refer to our recent paper [14].

Our TinyIBE scheme was implemented using the MIRACL [23] library which provides all the necessary tools to perform operations on elliptic curves. MIRACL is a publicly available C library that was primary designed for desktop class computers. Our implementation includes a dedicated version of MIRACL which was specially optimised for constrained WSN platforms. The crucial arithmetic operations used in the binary field arithmetic (e.g. binary polynomial multiplication) were implemented in assembly language for all our target platforms.

A. Performance evaluation

In the first phase of the TinyIBE scheme the PKG preloads all the necessary information to sensor nodes. This is performed off-line before the network deployment. In what follows we focus only on the operations that are carried out by the sensor nodes during the session key establishment phase.

The Encrypt step involves hashing and calculation of two point multiplications. The calculation of C_1 is the most expensive step during the encryption process. An H-sensor public key can be regarded as the value $(Q + aP)$ which is fixed for a given CH. If we assume that the encryption step will be performed multiple times for the same H-sensor, we can cache this value and save one point multiplication during subsequent encryptions. However in most cases the encryption step will be performed only once for a given CH. The calculation of aP is a fixed point multiplication with a 160-bit scalar and we

can use precomputation methods to speed up this operation. To compute C_2 we need to perform the g value exponentiation but this operation is negligible when compared with the C_1 calculation.

The decryption step is pretty straight forward and requires one η_T pairing calculation and one hashing to retrieve the session key. The pairing takes as the first parameter H-sensor's private key which is a constant value. This fixed parameter can be exploited for precomputation that accelerates the pairing computation at the expense of some storage overhead. In our implementation we have put more focus on the efficient memory utilisation and we did not use precomputation. Our performance evaluation excludes also the ID check operation performed by every H-sensor before the decryption step.

Table I presents the evaluation results for our TinyECC implementation. All the memory and timing results were obtained using cycle accurate simulators. We used the AVR Studio for the Atmega128L processor (MICAz) and the IAR Embedded Workbench in case of both MSP430 (Tmote) and Xscale (Imote2) platforms. The energy consumption was measured experimentally for the MICAz and Tmote Sky nodes. We used the data sheet figures in case of the Imote2 (data not available for 416MHz clock setting).

The encryption times for L-sensors are acceptable, given that these operations are performed very rarely and mainly at the beginning of the network operation. However we need to apply methods to minimize the vulnerability to DoS attacks. Constant requests for encryption/decryption would quickly deplete node's energy. After the key distribution phase nodes should use much cheaper symmetric key encryption methods.

The 32-bit Imote2 platform handles well the decryption process and achieves best performance at maximum frequency of 416MHz (70 decryption operations per minute). TinyIBE is the fastest IBE scheme at the moment as we are not aware of any other implementations on sensor nodes.

The memory footprint on both L-sensors is considerable (especially on Tmote with its 48KB of ROM) and it is mainly due to the large size of the ECC library. However this includes also all the necessary library code needed to perform the ECDSA during the secure broadcast verification. RAM utilization may seem high for low-end devices but the memory is reserved only for the duration of cryptographic operations. The values showed in Table I present only the peak numbers for stack usage during the program execution.

One of the biggest advantages of TinyIBE when compared with the state-of-the-art is the significant storage saving in terms of cryptographic keys. Assume that the number of H-sensors and L-sensors in a HSN is M and N, respectively. In the key pre-distribution scheme proposed by Eschenauer and Gligor (E-G [3]) the total number of pre-loaded keys in the network is equal to: $m(M+N)$, where each sensor is pre-loaded with m keys. The value of m depends on the key pool size (p) and the probability of sharing at least one key between two nodes (ps). Du et al. proposed asymmetric pre-distribution (A-P [8]) where H-sensors are pre-loaded with more keys (x) than L-sensors (y). The number of pre-loaded keys can be

TABLE I
TINYIBE EVALUATION RESULTS

Platform	Encryption			
	Time	**ROM**	**RAM**	**Energy**
MicaZ (7.38MHz)	3.93s	39.6KB	2.9KB	92.67mJ
Tmote (8.19MHz)	2.62s	30.3KB	3.2KB	27.12mJ
Platform	**Decryption**			
	Time	**ROM**	**RAM**	**Energy**
Imote2 (13MHz)	462ms	32.87KB	4.12KB	12.12mJ
Imote2 (104MHz)	57.7ms	32.87KB	4.12KB	3.76mJ
Imote2 (416MHz)	14.4ms	32.87KB	4.12KB	N/A

calculated as $xM + yN$ where $xy = m^2$. The ECC scheme proposed in [21] decreases the number of necessary keys to $M(N+3)+2N$. Our TinyIBE scheme assumes that each H-sensor is pre-loaded with only 3 keys and the total number of pre-loaded keys in the network ($3M$) is independent of the number of L-sensors. This feature allows our scheme to scale gracefully with the number of nodes in the network.

Figure 2 presents the comparison between the described schemes. The parameters were set as: $M = 30$, $m = 100$, $p = 5000$, $ps = 0.87$, $x = 500$, $y = 20$. In all cases our protocol requires much less storage space for the pre-loaded keys (constant number of 90 keys) than other solutions. The storage savings of TinyIBE are increasing drastically with the number of L-sensors in the network.

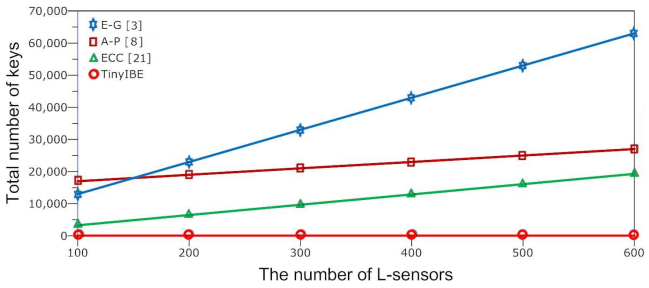

Fig. 2. Pre-loaded key storage space.

TinyIBE introduces also significant savings in communication overhead. In the E-G scheme each sensor broadcast the list of identifiers of keys on their key ring. When the key pool $p = 10000$, m needs to be larger than 150 to achieve a high key sharing probability of 0.9 [3]. Each key identifier requires at least 14 bits and the resulting message will have a size of 263 bytes. In the A-P scheme [8] every L-sensor sends a key list message to its H-sensor which includes L-sensor's ID, the list of key IDs and node's location. This results in a communication overhead of 94 bytes (assuming 2B for ID and 4B for location data). We used the following parameters: $x = 450$, $y = 50$, $p = 10000$. Our TinyIBE scheme requires an exchange of only one message to establish a shared session key. The C_1 value is a point on $E(\mathbb{F}_{2^{271}})$ which can be compressed to 34 bytes and C_2 has the size of the session key (128-bits). The resulting message has 52 bytes. The communication overhead in TinyIBE is independent of the network size and fixed for each node pair. In case of the

key pre-distribution schemes ([3], [8]) the overhead increases with the number of pre-loaded keys and the key pool size. We cannot compare our results with the ECC scheme proposed in [21] as it does not specify the public key encryption method used for session key distribution.

B. Security analysis

It is important to check what is the effect of an L-sensor compromise on the rest of the network. It can be measured by calculating the probability that an attacker can decrypt the communication between any two nodes after capturing c L-sensors (compromising probability). Paper [21] presents the formula to calculate the probability that two sensors have exactly j common keys in the E-G scheme: $p(j) = \binom{p}{j}\binom{p-j}{2(m-j)}\binom{2(m-j)}{m-j}/\binom{p}{m}^2$, where p is the key pool size and m is the number of pre-loaded keys in each node. The compromising probability for the E-G scheme can be calculated as:

$$C(m) = \sum_{j=1}^{m}(1-(1-\frac{m}{p})^c)^j p(j) / \sum_{j=1}^{m} p(j)$$

Similar formulas can be found in [8] to compute the same probability for the A-P scheme.

In Figure 3 we compare the compromising probability of all four schemes. We used the following parameters: $p = 5000$, $x = 125$, $y = 20$ and two different values of m for the E-G scheme. The ECC based protocol and our TinyIBE scheme employ asymmetric cryptography primitives. After the session key distribution phase each pair of nodes has a different shared key. Hence, compromising c L-sensors does not affect the security of communication among other pairs of nodes. The compromising probability in both schemes is always equal to zero for all c values. For the key predistribution schemes the compromising probability increases significantly with the number of pre-loaded keys (m or y). Therefore the resilience to the node compromise attack is much higher in PKC-based protocols.

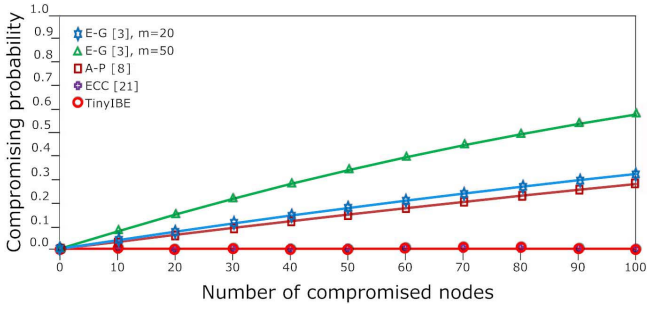

Fig. 3. The compromising probability.

VI. Conclusion

A common misconception is that identity-based encryption is not a suitable security technique for sensor networks. The argument is that IBE suffers from identity theft and replication problems and that it is too heavyweight for sensor devices. The results presented in this paper prove otherwise. Our security scheme is protected against fake cluster heads and it is feasible to implement even on the most constrained sensor nodes. The encryption procedure takes less than four seconds on an 8-bit MICAz platform. Identity theft and replication issues can be addressed by simple ID check procedures. A comparison with the state of the art shows that our TinyIBE protocol introduces significant savings in key storage space and communication overhead compared to existing solutions. Additionally it provides a higher level of security and stronger resilience against node capture attack than any key predistribution technique. TinyIBE was designed especially for HSNs to provide a new and simplified way for key distribution in the network. Our work is the first to prove that identity-based encryption is not only feasible for HSNs but is a superior security bootstrapping method in a heterogenous environment.

References

[1] E. J. Duarte-melo and M. Liu, "The effect of organization on energy consumption in wireless sensor networks," in *In: IEEE Globecom*, 2002.

[2] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," in *SenSys*, 2004.

[3] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02*. ACM, 2002.

[4] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for Wireless Sensor Networks," in *2nd ACM SensSys*, 2004.

[5] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis & defenses," in *IPSN '04*. New York, USA: ACM.

[6] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," *Technical Report*, 2005.

[7] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *SASN '03*. New York, USA: ACM.

[8] X. Du, Y. Xiao, and H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Networks*, vol. 5, 2007.

[9] L. B. Oliveira, H. C. Wong, A. A. F. Loureiro, and R. Dahab, "On the design of secure protocols for hierarchical sensor networks," *Int. J. Secur. Netw.*, vol. 2, no. 3/4, pp. 216–227, 2007.

[10] S. Hussain, F. Kausar, and A. Masood, "An efficient key distribution scheme for heterogeneous sensor networks," in *IWCMC '07*.

[11] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs." in *CHES'04*.

[12] W. Hu, P. Corke, W. Shih, and L. Overs, "secfleck: A public key technology platform for wireless sensor networks," 2009, pp. 296–311.

[13] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the limits of Elliptic Curve Cryptography in Sensor Networks," in *EWSN 2008*, ser. LNCS, vol. 4913. Springer-Verlag.

[14] P. Szczechowiak, A. Kargl, M. Scott, and M. Collier, "On the application of pairing based cryptography to wireless sensor networks," in *WiSec '09, Second ACM conference on Wireless Network Security*, 2009.

[15] B. Doyle, S. Bell, A. F. Smeaton, K. McCusker, and N. O'Connor., "Security considerations and key negotiation techniques for power constrained sensor networks," *The Computer Journal*, vol. 4/49, 2006.

[16] G. Yang, C. Rong, C. Veigner, and H. Cheng, "Id-based key agreement and encryption for wireless sensor networks," *IJCSNS*, vol. 6, 2006.

[17] L. B. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. A. F. Loureiro, "Identity-based encryption for sensor networks," in *PERCOMW '07*.

[18] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[19] I. Blake, G. Seroussi, and N. Smart, "Advances in Elliptic Curve Cryptography." Cambridge University Press, 2005, pp. 183–213.

[20] R. Sakai and M. Kasahara, "Id based cryptosystems with pairing on elliptic curve," Cryptology ePrint Archive, Report 2003/054, 2003.

[21] X. Du, Y. Xiao, S. Ci, M. Guizani, and H. Chen, "A routing-driven key management scheme for heterogeneous sensor networks," in *ICC '07*.

[22] P. S. L. M. Barreto, S. Galbraith, C. O'hEigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Designs, Codes and Cryptography*, vol. 42, pp. 239–271, 2007.

[23] M. Scott, "MIRACL – Multiprecision Integer and Rational Arithmetic C/C++ Library," 2007, *http://ftp.computing.dcu.ie/pub/crypto/miracl.zip*.