D. Galindo, R. Roman, and J. Lopez, "On the Energy Cost of Authenticated Key Agreement in Wireless Sensor Networks", *Wireless Communications and Mobile Computing*, vol. 12, pp. 133-143, 2012.
http://doi.org/10.1002/wcm.894
NICS Lab. Publications: https://www.nics.uma.es/publications

On the energy cost of authenticated key agreement in wireless sensor networks

David Galindo
University of Luxembourg, 6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
Rodrigo Roman, Javier Lopez
Department of Computer Science, University of Malaga, 29071, Spain

Abstract

Wireless sensors are battery-powered devices which are highly constrained in terms of computational capabilities, memory, and communication bandwidth. While battery life is their main limitation, they require considerable energy to communicate data. Due to this, it turns out that the energy saving of computationally inexpensive primitives (like symmetric key cryptography) can be nullified by the bigger amount of data they require to be sent. In this work we study the energy cost of key agreement protocols between peers in a network using asymmetric key cryptography. Our main concern is to reduce the amount of data to be exchanged, which can be done by using special cryptographic paradigms like identity-based and self-certified cryptography. The main news is that an intensive computational primitive for resource-constrained devices, such as non-interactive identity-based authenticated key exchange (AKE), performs comparably or even better than traditional AKE in a variety of scenarios. Moreover, protocols based in this primitive can provide better security properties in real deployments than other simple protocols based on symmetric cryptography. Our findings illustrate to what extent the latest implementation advancements push the efficiency boundaries of public key cryptography in wireless sensor networks.

1 Introduction

It is well-known that from an efficiency point of view, symmetric key cryptography outperforms public (or asymmetric) key cryptography. Indeed, public key primitives are of the order of hundred of times more computationally intensive that their symmetric key counterparts. Along this line, one would use asymmetric cryptography not for efficiency issues but for achieving specific functionalities, like easier key management or non-repudiation.

The better performance of symmetric key primitives can be even more acute in resource-constrained devices, for which frequently battery life is the main limitation, so the less computationally expensive (and hence less energy consuming) operations the better. This is the reason why in areas like wireless sensor network security, using public key crypto has been considered prohibitive from the very beginning.

Somewhat surprisingly, is precisely in the area of wireless resource-constrained devices where this common wisdom is being challenged. The main reason behind this is the fact that communicating data in these devices requires considerable power, in contrast to wired devices. Therefore, it can be the case that the energy saving of a computationally inexpensive primitive is nullified by the bigger amount of data it requires to be sent. This has already been shown by Großschädl, Szekely and Tillich in [1], where the energy cost of two standardized symmetric and asymmetric key exchange protocols has been evaluated. Specifically, the symmetric key protocol used in that

study is a light-weight variant of authenticated Kerberos [2], while the asymmetric key protocol is an elliptic curve version of Menezes-Qu-Vanstone [3, 4] (ECMQV). The striking result is that in wireless sensor networks (WSN), ECMQV consumes less power than Kerberos[1], due to the fact that it requires 50% less bits to be exchanged.

We go one step further by considering an extreme case of wireless communication, namely, communication between underwater sensor nodes. Classical electromagnetic waves communication is not satisfactory in underwater environments due to the conducting nature of the medium, especially in the case of sea water. Instead, acoustic communication is the most widely used technique, due to the low signal reduction of sound in water [5]. Acoustic communication presents severe limitations in bandwidth and requires a huge amount of energy. According to Morgansen [6], current state of the art in practical scenarios is transmission of 640 bits (80 bytes) per second.

Our contributions. Our *first contribution*, to be found in Section 3, consists on estimating how much energy can be saved by using self-certified key establishment protocols and identity-based key establishment protocols instead of traditional authenticated key exchange (AKE). Regarding self-certified key establishment, no certificates must be sent nor verified to authenticate public key material, thus the savings in communications are clear. Moreover, the computational requirements are also smaller. Optimal communications are reached when using a non-interactive identity-based authenticated key agreement (NIKE) protocol such as Sakai-Ohgishi-Kasahara [7], since it achieves the lowest bandwidth possible (only identities must be exchanged). But despite the latter, the use of NIKE in WSN can look surprising, due to the computationally intensive nature of the bilinear pairing (cf. Chapter 5 in [8]) primitive. However, the implementation of pairings has recently undergone dramatic improvements [9, 10], to such an extent that our estimations based on existing implementation results indicate that, for an equivalent RSA-1024 security level, the Sakai-Ohgishi-Kasahara (SOK) protocol is cheaper than ECMQV energy-wise ! In our *second contribution*, to be found in Section 4, we compare the performance of SOK protocol to well-known symmetric key-based key management systems in UWSN. Taking as a concrete example a lightweight version of the Kerberos protocol [2, 1], we come to the conclusion that SOK outperforms symmetric key-based key management systems in UWSN, both energy and security-wise.

[1] Specifically, this is the case when the communicating nodes are at least two hopes away from the base station.

2 Wireless Sensor Networks

Wireless sensor networks are a very useful tool for solving problems in scenarios that require the acquisition and processing of physical measurements. The principal elements of a sensor network are the sensor nodes and the base station. Sensor nodes (nodes) are wireless-enabled, battery-powered, highly constrained devices that collect the physical information from their environment using an array of sensors such as thermistors, photodiodes, and so on. The base station is a more powerful device that serves as an interface between the nodes and the user. It collects the information coming from sensor nodes, and also send control information issued by the user. There can be from dozens to thousands of sensor nodes on a deployment field, although there is usually only one or more base stations on the same field.

Security is one of the principal concerns while designing protocols and mechanisms for WSN. In fact, sensor networks are inherently insecure due to the features of their nodes and the communication channel. As a result, it is easy for an adversary to manipulate the sensor nodes and the communication channel of an unprotected network on its own benefit. There must be some protocols and security mechanisms that guarantee the resiliency of the network against any kind of external or internal threat. The foundation of these mechanisms and protocols are the security primitives, such as Symmetric Key Cryptography (SKC), Public Key Cryptography (PKC) and Hash functions. Using these primitives, it is possible to assure the confidentiality and integrity of the communication channel, while authenticating the peers involved in the information exchange.

Due to its energy efficiency and fast speed, Symmetric Cryptography becomes an interesting choice for securing the foundations of a sensor network. It can provide confidentiality to the information flow, and is also able to provide integrity. There are many optimal SKC algorithms implemented on sensor networks (such as Skipjack), that have small requirements in terms of memory usage and encryption speed (2600 bytes and 25μs/byte for Skipjack, respectively [11]). Moreover, some sensor nodes have transceivers that implement the IEEE 802.15.4 standard, which include a hardware implementation of the AES-128 algorithm.

However, as aforementioned, it is necessary to have certain security credentials in order to open a secure channel between two peers. As a result, if a sensor network relies only on SKC, it is necessary to implement certain key management systems (KMS) that distribute the pairwise keys over the nodes of the network before or after its deployment. The underlying problem here is the typical key management shortcomings of symmetric-key algorithms. To have a glance at these shortcomings, let us introduce some metrics to evaluate key distribution solutions, in particular, those proposed in [12, 13]:

- **Scalability**: Ability to support large networks.
- **Efficiency**: Storage, processing and communication limitations on sensor nodes must be considered:
  - **Storage**: Amount of memory required to store security credentials.
  - **Processing**: Amount of processor cycles required to establish a key.
  - **Communication**: Number of messages exchanged during a key generation process.
  - **Key connectivity**: Probability that two (or more) sensor nodes store the same key or keying material.
- **Resilience**: Resistance against node capture.
- **Extensibility**: Key distribution mechanisms must be also flexible against substantial increase in the size of the network after deployment.

Typical shortcomings of SKC-based key distribution solutions are associated to either scalability, key connectivity, resilience and extensibility properties, being the main advantage of these solutions a low processing time. Public Key Cryptography (PKC) is useful in this context. By using authenticated key exchange protocols, the process of negotiating pairwise keys between previously unknown peers can be greatly simplified, as it enjoys benefits in every single property in the above-mentioned metrics, except for processing time. However, as we shall see, in UWSN the processing time gets its relevance lowered, as bandwidth is by far the most relevant parameter. Thanks to this, a specialized PKC-based key establishment mechanism, namely, non-interactive identity-based key agreement, outperforms previous SKC-based key distribution solutions.

2.1 Underwater Wireless Sensor Networks

There are many potential applications where sensor nodes must be deployed in a lake or in the sea, either for long-term aquatic monitoring (Marine biology, deep-sea archaeology, seismic predictions, pollution detection, oil/gas field monitoring) or short-term aquatic exploration (Underwater natural resource discovery, anti-submarine mission, loss treasure discovery) [14]. These networks have received the generic name of Underwater Sensor Networks (UWSN) [15].

It would seem that the only difference between underwater and terrestrial sensor networks is the water that surrounds the nodes. However, this particular situation imposes additional constraints to the network (cf. Figure 1). For example, underwater sensor nodes must use an acoustic communication channel in order to exchange information wirelessly. This channel has a limited capacity, and the energy consumption and the propagation delay of the channel are very high. Sensor nodes are also especially prone to failure, due to specific underwater threats such as fishing trawlers, underwater life, failure of waterproofing, or mere corrosion. Moreover, underwater nodes are equipped with a limited battery that cannot be recharged due to their location.

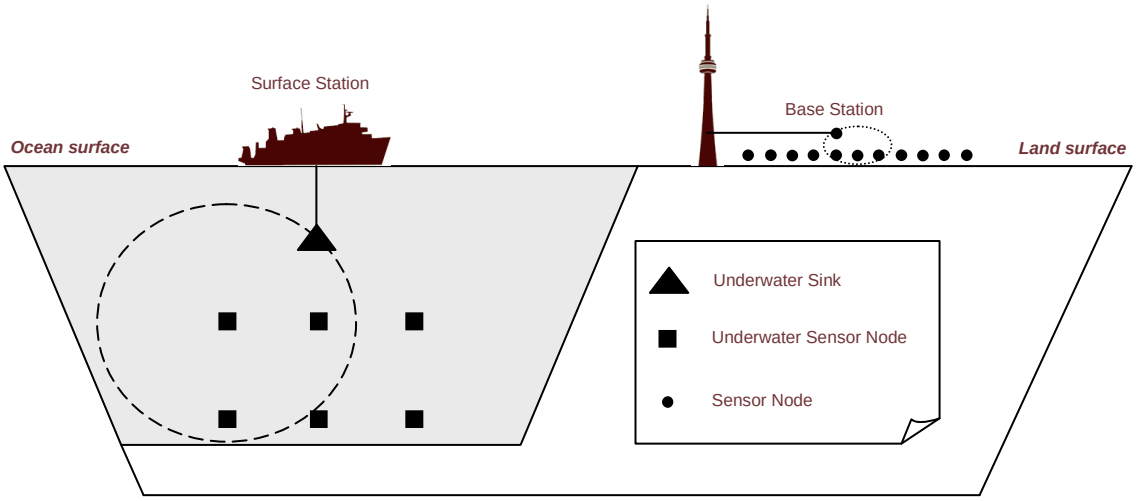

Figure 1: KMS frameworks for WSN

There are other special features of UWSN that must be taken into account: cost, coverage, and hardware capabilities. Unlike terrestrial sensor networks, the deployment of underwater sensor nodes is more costly, due to the complex components of the nodes (e.g. water-proof housing, acoustic modems) and the equipment that must be used to deploy the nodes (e.g. ships). As the deployment of underwater sensor nodes is usually sparser, and because sensor nodes can move due to anchor drift or other external effects, the coverage of the network is reduced. However, the capabilities of underwater sensor nodes are usually greater: they have more memory to perform data caching in case of intermittent connections, and they also have a battery with a higher capacity due to the cost of using the communication channel and the complexity of replacing a drained battery.

Regarding the security requirements of these kind of networks, all the inherent features of UWSN not only have influence on the importance of the security requirements and the characteristics of the security mechanisms, but also limits the number of security mechanisms that could be used to protect the network. For example, auditing is an important property in UWSN, as it is very difficult to physically reach the nodes after their deployment. On the other hand, it is necessary to avoid those protocols whose communication overhead is very high. For example, some key management systems (KMS) must open a communication channel between many nodes in order to create a single pairwise key. As these KMS create an unnecessary amount of messages, it is better to use other scalable KMS where only two nodes interact to create a single pairwise key. It is precisely on the features of the communication channel (acoustic channel) of UWSN where we will focus the remainder of this section.

In these UWSN, it is unpractical to use radio frequency transceivers, because of the severe attenuation factor presented by water. In order to open a communication channel between sensors, it is necessary to use specific underwater acoustic modems. These modems have different features than RF transceivers: they are highly unreliable, their bandwidth is much more limited, and sending or receiving one bit of information carries a high energy penalty. In fact, the cost of using an acoustic communication channel largely impacts the energy required to run any interactive protocol between sensor nodes.

The differences between radio transceivers and acoustic modems in terms of the energy consumed by transmitting and receiving one single bit of data

are highlighted in Table 1. It can be seen that the difference in consumption (J per bit) between acoustic modems and RF transceivers is not negligible. For the radio transceivers, we have considered the most popular sensor nodes platforms as of today, which are the MICA2 and the MICAz [16]. The MICA2 transceivers use the 868/916 MHz ISM bands, while the MICAz transceivers use the IEEE 802.15.4 standard. For the acoustic modems, we have considered the UWM2000 and UWM4000 modems [17], which are commonly used in research literature.

These results have been obtained using the information contained in the modem and mote datasheets, under the following assumptions: i) For the UWM2000 modem, we have used the mean of the transmission power indicated in its datasheet (2-8W). ii) For the transceivers used in the MICA2 and MICAz motes, we have considered the most expensive transmission mode, which is theoretically able to send a bit of data to the maximum working range.

3 Authenticated Key Exchange Schemes using Public Key Cryptography

Certificates are needed to establish a trusted link between a public key and the identity of its owner (in our case a sensor node) in order to prevent man-in-the-middle attacks. In a WSN, nodes are supposed to establish pair-wise keys with nodes that belong to the same network, and forbidden to do so with nodes or devices outside the network. Therefore, in key establishment protocols like ECMQV, the nodes must at the beginning exchange their public keys and certificates. It is natural to assume these certificates take the form of a signature by the base station on the identity and public key of the node. In general, nodes public and secret keys are set up by the base station. Such a setting can be viewed as a key-escrowed system, that is, there exists a trusted party who computes the secret keys of the users. As a consequence one is tempted to use different forms of key-escrowed public key paradigms, like identity-based cryptography (even if it does not provide certain properties such as forward secrecy) or self-certified cryptography. Precisely, the main purpose of this section is to analyze their behaviour and energy consumption, in order to discover which protocol is more optimal for an underwater environment.

The concept of identity-based cryptography was proposed by Shamir in [18], aimed at simplifying certificate management inherent to the deployment of public key cryptography. The idea is that an arbitrary string id uniquely identifying a user (such as an e-mail address or a telephone number) can serve as a public key for a cryptographic scheme. The user can not compute the corresponding secret key anymore, but instead it must authenticate itself to a Key Generation Center from which it obtains the corresponding private key $sk[id]$ via a secret channel.

The interest of IBC for WSN is that in IBC systems only the identity of the sensors must be exchanged, and thus public keys and certificates need not be sent. This results in an energy saving for the point of view of the communication between sensors, which can be very considerable depending on the sensor's transmitter. Additionally, in WSN the base station can naturally play the role of the Key Generation Center in an IBC system. The base station embeds the secret key $sk[id]$ prior to its use in the field, and no authentic nor secret channel is needed for key setup.

On the other hand, the self-certified cryptography paradigm [19, 20], advocates for using an implicit certificate, as opposed to the traditional approach, where public keys are authenticated by verifying an explicit certificate. The practical consequence of using an implicit certificate is that the certificate and the public key are merged into a single object, named as self-certified key. This again results in communication savings, since now the sensors exchange identities and self-certified keys, but no certificates. The conceptual consequence is that the link between the identity and the self-certified key is only known to be authentic in the the course of a cryptographic operation (i.e. only the legitimate user will be able to decrypt a ciphertext encrypted with a self-certified encryption scheme).

In the following we study three AKE schemes in the context of underwater sensor networks. These schemes are based on traditional, identity-based and

	MICA2	MICAz	UWM2000	UWM4000
Working range	150 m	100 m	1500 m	4000 m
Throughput	19.2 kbit/s	250 kbit/s	9600 bit/s	4800 bit/s
Tx. consumption	81mW	52.2mW	4000 mW	7000 mW
Rx. consumption	30mW	59.1mW	800 mW	800 mW
μJ per bit (Tx)	4.12 μJ	0.204 μJ	416.66 μJ	1458.33 μJ
μJ per bit (Rx)	16.8 μJ	16.8 μJ	83.33 μJ	166.66 μJ

Table 1: Analysis of the energy consumption of radio-frequency and acoustic modems.

self-certified public key cryptography respectively.

3.1 Some facts on elliptic curves and pairings

Before describing our targeted concrete AKE schemes we need to fix some notation and recall basic concepts about elliptic curves and pairings.

Let GF(p) be a finite field with p elements. An elliptic curve over the field GF(p) can be defined by an equation of the form $y^2 + a_1x + a_3y = x^3 + a_2x^2 + a_4x + a_6$ where $a_1, a_2.a_3, a_4, a_6 \in$ GF(p). A point of the curve is specified by a pair (x, y) satisfying the above equation. It is possible to define an addition law on the points of the elliptic curve together with a special point called the "point at infinity", obtaining an abelian group G of order a certain integer n. We use multiplicative notation for the group G and assume it is finitely generated by an element g, i.e. every element $h \in G$ can be uniquely written as $h = g^\alpha$ for some $\alpha \in \mathbb{Z}_n$. An exponentiation refers to the operation g^r for a randomly taken $r \in \mathbb{Z}_n$. A multi-exponentiation mexp(l) refers to computing $g_1^{r_1} \cdots g_l^{r_l}$, where $g_1, \ldots, g_l \in G$ and $r_1, \ldots, r_l \in \mathbb{Z}_n$, an operation that can be computed more efficiently than just computing l single exponentiations due to an algorithm by Strauss [21]. Additionally, thanks to the technique of point compression, a point $h = (x, y) \in G$ can be represented by just sending the x-coordinate. With the x-coordinate in hand, one can find y by computing one square root in GF(p).

Energy estimations on elliptic curve group element exponentiation are based on the work of Gura *et al.* [22] and Scott and Szczechowiak [23]. The elliptic curve used in [22] is the standardized SECG `secp160r1` curve over GF(p) with $p = 2^{160} - 2^{31} - 1$, which achieves security level equivalent to that offered by standard RSA based solutions with 1024-bit keys. The time needed to perform an exponentiation on that curve by using the ATmega128L microcontroller [24] (one of the most popular microcontrollers for sensor nodes, featuring a 8-bit/7.3828 processor, 128 KB flash memory and 4KB SRAM memory) is reported to be 0.81s, which in terms of energy is 19.1mJ. However, [23] reports to have reduced with respect to [22] the number of clock cycles for a full 160x160 bit multiplication from 3106 to 2651 clock cycles, which would translate to reduce 19.1mJ to 16.30mJ. Computing one multi-exponentiation mexp(2) takes 22% more energy than that of a single exponentiation, while one multi-exponentiation mexp(3) takes about 44% more energy than a single exponentiation according to Brumley [25]. It turns out that $p = 3 \bmod 4$ and computing a square root modulo p, by virtue of Algorithm 3.36 in [26], takes no more energy than 0.10% that of one exponentiation.

With regards to bilinear maps, let $\mathbb{G} = \langle \mathbf{g} \rangle$ be a cyclic group of order q for prime $q > 3$. A map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ to a group $\mathbb{G}_1$ is called a *bilinear* map, if it satisfies the following two properties:

Bilinearity: $e(\mathbf{g}^a, \mathbf{g}^b) = e(\mathbf{g}, \mathbf{g})^{ab}$ for all integers a, b

Distorted: $e(\mathbf{g}, \mathbf{g}) \neq 1$ in $\mathbb{G}_1$.

See [27, 28] for ways of constructing bilinear maps. For efficiency estimations, we use the work of Szczechowiak *et al.* [10]. $\mathbb{G}$ is defined as the group points of the supersingular elliptic curve $y^2 + y =$

$x^3 + x$ over $GF(2^{271})$, while $\mathbb{G}_1 = GF\big((2^{271})^4\big)$ and e is defined as the η_T pairing [29]. This bilinear map also provides equivalent RSA-1024 bit security. [10] reports that computing η_T on this curve requires 62.73 mJ. This constitutes a major advancement in this area, since earlier attempts, for instance [30], reported as much as 712.43mJ.

3.2 ECMQV - Elliptic Curve Menezes-Qu-Vanstone

We recall the elliptic curve version of the Menezes-Qu-Vanstone authenticated key exchange protocol [3, 4], which is one of the most standardized key exchange protocols using public key cryptography. Here public keys are authenticated by using traditional certificates.

Algorithm 3.1 ECMQV key derivation for entity A

Input: Elliptic curve domain parameters G, g, n, secret keys x_A, y_A, public keys $\mathrm{pk}_A, \mathrm{pk}_B$, and ephemeral keys E_A, E_B

Output: A secret key K_{AB} shared with entity with public key pk_B

1: $m \leftarrow \lceil \log_2(n) \rceil / 2$
   $\{m$ is the half bitlength of $n\}$
2: $u_A \leftarrow (u_x \bmod 2^m) + 2^m$
   $\{u_x$ is the x-coordinate of $E_A\}$
3: $s_A \leftarrow (y_A + u_A x_A) \bmod n$
4: $v_A \leftarrow (v_x \bmod 2^m) + 2^m$
   $\{v_x$ is the x-coordinate of $E_B\}$
5: $z_A \leftarrow s_A v_A \bmod n$
6: $K_{AB} \leftarrow KDF(E_B^{s_A} \cdot \mathrm{pk}_B^{z_A} \bmod n)$

In Algorithm 3.1, KDF is a key derivation function, which can be implemented with SHA-160 for example. Node A's public key is $\mathrm{pk}_A = g^{x_A}$, where x_A is A's secret key. Similarly for node B. In the first stage, the nodes exchange and verify certificates vouching for the fact that pk_A and pk_B are public keys from nodes belonging to the network. In a second stage, they exchange their ephemeral keys $E_A = g^{y_A}$ and $E_B = g^{y_B}$, where y_A, y_B are taken at random from the finite field GF(p). We assume certificates are minimalist and take the form of ECDSA [31] signatures (r_A, s_A) and (r_B, s_B) by the owner/manufacturer of the network on the messages $id_A||\mathrm{pk}_A$ and $id_B||\mathrm{pk}_B$ respectively, where $||$ denotes concatenation.

Entity B runs the same algorithm by simply swapping the values $(x_A, y_A, \mathrm{pk}_B, E_A, E_B)$ in Algorithm 3.1 with $(x_B, y_B, \mathrm{pk}_A, E_A, E_B)$ and finally obtains the same key K_{AB} (cf. [4]).

3.3 SOK - Sakai, Ohgishi and Kasahara

In this section we recall a non-interactive authenticated identity-based key establishment scheme. Due to the lack of any standardized identity-based key exchange protocol, we describe a non-interactive scheme due to Sakai, Ohgishi and Kasahara [7, 32], which is the first identity-based authenticated key agreement protocol proposed in the literature.

In the SOK protocol, a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ is included in the domain parameters of the system, together with $\mathbf{g}^z$, where the master secret key z is only known to the base station. Node A's secret key is $sk_A = H(id_A)^z$, while node B's secret key is defined as $sk_B = H(id_B)^z$. Notice that A's identity is id_A and B's identity is id_B.

Algorithm 3.2 SOK non-interactive ID-based key derivation for entity A

Input: Bilinear map domain parameters $\mathbb{G}, \mathbb{G}_1, e, \mathbf{g}^z, n$, the identity id_B and the secret key sk_A

Output: A secret key K_{AB} shared with entity with identity id_B

1: $K_{AB} \leftarrow KDF\big(e(H(id_B), sk_A)\big)$

Entity B runs the same algorithm by simply swapping the values (id_B, sk_A) in Algorithm 3.2 with (id_A, sk_B) and finally obtains the same key K_{AB} thanks to the bilinearity of the pairing,

$$\begin{aligned} e(H(id_B), sk_A) &= e(H(id_B), H(id_A)^z) = \\ e(H(id_B), H(id_A))^z &= e(H(id_A)^z, H(id_B)) \\ &= e(sk_B, H(id_A)) \end{aligned}$$

3.4 SC-ECMQV - Self-certified ECMQV

In this section we explore the energy performance of a variant of ECMQV where implicit public key certificates are used. The base station has a public key $g_0 = g^z$, where $z \in \mathbb{Z}_n$ is secret, and $h : \{0,1\}^* \to \mathbb{Z}_n$ is a public hash function. The self-certified public keys of the sensors are $w_A = g^{r_A}$ and $w_B = g^{r_B}$, with r_A, r_B only known to the base station, while the corresponding secret keys known to the nodes are $x_A = z \cdot h(id_A, w_B) + r_A$ and $x_B = z \cdot h(id_B, w_B) + r_B$ with $x_A, x_B \in \mathbb{Z}_n$. The ECMQV-like public keys $\mathrm{pk}_A, \mathrm{pk}_B$ can be obtained by combining the public information together with self-certified keys, i.e. $\mathrm{pk}_A = g^{x_A} = g_0^{h(id_A, w_A)} \cdot w_A$ and $\mathrm{pk}_B = g^{x_B} = g_0^{h(id_B, w_B)} \cdot w_B$. This allows to build a self-certified version of ECMQV.

In this version, sensors only exchange their self-certified public keys and the ephemeral keys E_A and E_B, where y_A, y_B are taken at random from the corresponding finite field. It is important to note that they do not exchange nor verify *any certificates*.

Algorithm 3.3 Self-certified ECMQV key derivation for entity A

Input: Elliptic curve domain parameters G, g, n, g_0, h, secret keys x_A, y_A, self-certified keys w_A, w_B and ephemeral keys E_A, E_B

Output: A secret key K_{AB} shared with entity with self-certified public key w_B

1: $m \leftarrow \lceil \log_2(n) \rceil / 2$
   $\{m$ is the half bitlength of $n\}$
2: $u_A \leftarrow (u_x \bmod 2^m) + 2^m$
   $\{u_x$ is the x-coordinate of $E_A\}$
3: $s_A \leftarrow (y_A + u_A x_A) \bmod n$
4: $v_A \leftarrow (v_x \bmod 2^m) + 2^m$
   $\{v_x$ is the x-coordinate of $E_B\}$
5: $z_A \leftarrow s_A v_A \bmod n$
6: $K_{AB} \leftarrow KDF\big(E_B^{s_A} \cdot (g_0^{h(id_B, w_B)} \cdot w_B)^{z_A} \bmod n\big)$

Like ECMQV, entity B runs the same algorithm by simply swapping the values $(x_A, y_A, w_B, E_A, E_B)$ in Algorithm 3.3 with $(x_B, y_B, w_A, E_A, E_B)$ and finally obtains the same key K_{AB}.

3.5 Bandwidth and energy consumption

As we can see, the SOK protocol only requires the identities id_A, id_B of the sensors involved to compute a pairwise authenticated and confidential key. On the other hand, the communication overhead of the ECMQV protocol is dominated on by the exchange of public keys, certificates and ephemeral keys. Note that SC-ECMQV does not a preliminary round to exchange certificates, due to the nature of the self-certified public keys. On the computational side, SOK has to perform one hash operation $\mathrm{hash}_{\mathbb{G}}$', plus 1 pairing computation. ECMQV has to verify an ECDSA signature (one multi-exponentiation 'mexp(2)'), and to run its protocol (one multi-exponentiation 'mexp(2)' plus one exponentiation 'exp' plus two square roots 'sqrt'). Finally, running the self-certified ECMQV protocol has a computational cost dominated by one exponentiation plus one multi-exponentiation mexp(3) plus two square roots 'sqrt'. Consequently, the overall energy cost and transmission cost of ECMQV for one node amounts to:

$$\begin{aligned} &2\mathrm{mexp}(2) + 1\mathrm{exp} + 2\mathrm{sqrt} \\ &(+\mathrm{trans.}\ 1410\ \mathrm{bits} + \mathrm{recep.}\ 1410\ \mathrm{bits}) \end{aligned} \tag{1}$$

whereas the energy cost and transmission cost of SOK for one node amounts to:

$$\begin{aligned} &1\mathrm{hash}_{\mathbb{G}} + 1\mathrm{pairing} \\ &(+\mathrm{trans.}\ 384\ \mathrm{bits} + \mathrm{recep.}\ 384\ \mathrm{bits}) \end{aligned} \tag{2}$$

and the the energy cost and transmission cost of SC-ECMQV for one node amounts to:

$$\begin{aligned} &1\mathrm{mexp}(3) + 1\mathrm{exp} + 2\mathrm{sqrt} \\ &(+\mathrm{trans.}\ 706\ \mathrm{bits} + \mathrm{recep.}\ 706\ \mathrm{bits}) \end{aligned} \tag{3}$$

considering that i) one packet containing nodes identities, protocol ID, message ID, checksum, and low-level headers and footers, amounts to a total of 384 bits, ii) public keys have 161 bits, iii) each ECDSA

MICA2	Comp.	Comm.		**MICAz**	Comp.	Comm.	
ECMQV	59.33	29.5	88.83	ECMQV	59.33	23.97	83.3
SOK	62.73	8.04	70.77	SOK	62.73	6.53	69.26
SC-ECMQV	43.03	14.77	**57.8**	SC-ECMQV	43.03	12	**55.03**
UWM2000	Comp.	Comm.		**UWM4000**	Comp.	Comm.	
ECMQV	59.33	704.98	764.31	ECMQV	59.33	2291.23	2350.56
SOK	62.73	191.99	**254.72**	SOK	62.73	623.99	**686.72**
SC-ECMQV	43.03	352.99	396.02	SC-ECMQV	43.03	1147.24	1190.27

Table 2: Per node energy cost of authenticated key exchange (in mJ)

certificate has 320 bits, and iv) each ephemeral key contributes with 320 bits.

Therefore, the SOK protocol requires the lowest bandwidth to accomplish its task. This is because the SOK protocol only needs to exchange 384 bits, whereas the ECMQV protocol must exchange 1410 bits and the SC-ECMQV protocol has to exchange 706 bits. That is indeed a very important point, due to the unreliable nature of the acoustic channel, it is essential to exchange as few bits as possible. The main limitation of the SOK protocol is the pairing computation, as it is very energy consuming. For example, the most efficient implementation as of 2009 takes about 2.66s processing time and has 62.73 mJ energy cost in the ATmega128L microcontroller [10]. At first sight this seems a rather large figure, but if we compare this amount of energy needed to transmit data in the UWM2000 and UWM4000 underwater modems, we obtain that computing a pairing takes the same amount of energy than transmitting 146 and 42 bits respectively. Therefore it is fair to consider pairing computation in UWSN as cheap when compared to transmitting-receiving information.

This assertion can be verified by analyzing the results shown in Table 2, that shows the energy consumption of a sensor node engaged in authenticated key exchange protocols in normal and underwater sensor networks, in terms of mJ. The energy figures for SOK are taken from [10], where it is asserted that hashing onto the elliptic curve has negligible cost compared to computing the pairing. For both UWSN platforms we considered, SOK is much better than the other protocols, due to the high transmission cost. Note that SC-ECMQV beats ECMQV even when the motes use radio frequency transceivers. This is not surprising, since both the computation and communication costs of using SC-ECMQV are smaller.

4 NIKE and Symmetric Key-based KMS

In the previous section, we have demonstrated how non-interactive identity-based key agreement (NIKE) protocols like SOK are more energy-wise efficient than traditional asymmetric key establishment protocols (e.g. ECMQV) in underwater environments. However, this analysis would be incomplete if symmetric key-based KMS (henceforth known as "symmetric protocols") are not taken into account. As public key cryptography has been deemed as "unsuitable" for many years, there have been many researchers that have tried to solve the problem of distributing the secret keys of a sensor network using only symmetric key cryptography. The existing number of symmetric protocols is high, and as of 2008 is still growing. Therefore, it is necessary to check which of these symmetric protocols also have a small communication overhead, and whether they provide better properties.

There are three major frameworks of symmetric protocols: "Key-Pool" framework, Mathematical framework and Negotiation framework (see Figure 2). All symmetric protocols in the "Key-Pool" framework are based on the following assumption: every node stores a small subset of keys (known as "key

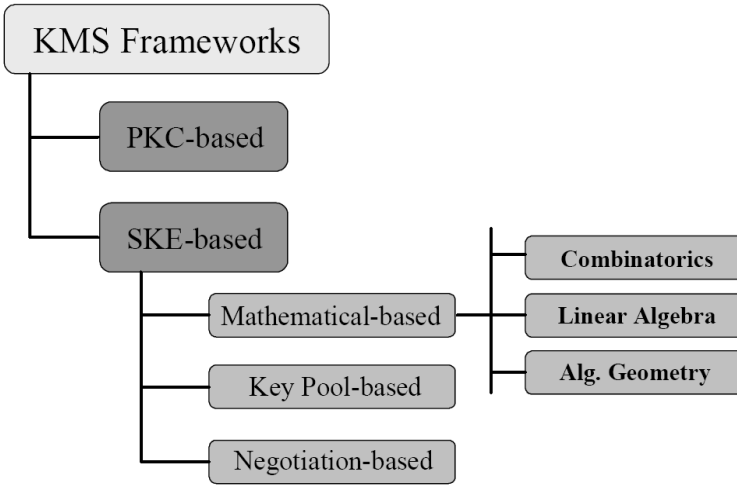

Figure 2: KMS frameworks for WSN

chain") retrieved from a large set of precalculated key (known as "key pool"). Two nodes will share a pairwise key if they have a common key inside their "key chains". If two nodes do not share any pairwise key, they will try to find a "key path" (i.e. protected channel using other nodes as intermediaries) in order to negotiate the key. In the Mathematical framework, two nodes calculate a common pairwise key using mathematical concepts belonging mainly to the fields of Linear Algebra, Combinatorics and Algebraic Geometry. Most of the symmetric protocols belonging to this framework can provide a pairwise key using just the ID of the nodes as an input. Lastly, in the Negotiation framework, sensor nodes exchange information related to their pairwise keys just after the deployment of the network. The content of this negotiation can be either the pairwise key itself or certain fields that allow the authentication of both peers and the derivation of the key.

Since these symmetric protocols do not use asymmetric cryptography, their computational cost in energy terms can be considered negligible. Therefore, the energy consumption of these protocols will be influenced mainly by the cost of sending and receiving information through the wireless channel. In order to effectively compare the symmetric protocols with the SOK scheme, we can establish an upper bound for the number of bits that a symmetric protocol can send and receive before becoming less energy efficient than SOK. If we consider the energy consumption of the SOK scheme, presented in the previous section, such upper bound is 384 bits (i.e. bits exchanged in the SOK scheme) plus the number of bits that could be exchanged using the computational energy of the SOK scheme. These extra bits can be expressed as follows:

$$Tb + Rb = E \tag{4}$$

where T is the energy consumption of sending one bit through the acoustic channel, R is the energy consumption of receiving one bit through the acoustic channel, E is the computational cost of the SOK scheme, and b is the extra number of bits available to the symmetric protocols. Using the values calculated in the previous section, a symmetric key agreement can transmit and receive at most 126 extra bits for the UWM2000 modem or 38 extra bits for the UWM4000 modem in order to be preferred to the SOK protocol ! Therefore, the upper bound of the number of bits that a symmetric protocol can exchange before becoming less energy efficient than SOK is 510 bits for the UWM2000 modem and 422

bits for the UWM4000 modem. Alas, since the size of a single sent/received packet is at least 384 bits (needed to specify nodes identities, protocol ID, message ID, checksum, and low-level headers and footers), it turns out that the maximum number of messages that can be exchanged is upper bounded by 2.

Most protocols belonging to the "Key-Pool" framework can be regarded inefficient in underwater environments due to their excessive reliance on the exchange of information. They need to exchange a large list of "key chain" key IDs (e.g. 50 IDs) in order to find a common pairwise key. These IDs will increment the number of bits exchanged between nodes. Also, if two nodes do not share a pairwise key, it is necessary to open a secure channel with other nodes in order to find a "key path". These additional negotiation messages will affect the energy consumption of the nodes located in the "key path". Note that there are other symmetric protocols that can reduce the list of key IDs to a single ID [33], but those protocols still have the problem of finding a "key path" due to their low key connectivity.

On the other hand, the majority of the protocols that belong to the Negotiation framework can be also considered inefficient. In this framework, a single node must usually negotiate a pairwise key with every neighbor, thus any node must transmit at least one message per neighbor (cf. [34]). However, in ID-based protocols such as SOK, a node only needs to send one single message with its ID, and all of the nodes in the neighborhood will be able to derive a pairwise key. Besides, many protocols engage in complex negotiations that either exchange many messages [35] or use different messages with different radio transmission ranges [36]. We illustrate the negotiation framework by briefly discussing a lightweight version of the Kerberos protocol [2, 1].

In this modified version [1], minor modifications have been put forward in order to minimize the payload of the messages exchanged. In this protocol (see Figure 3), two entities A and B wishing to establish a shared secret key k_{AB} already share a secret key (k_{AT} and k_{BT} respectively) with a trusted third party T.

The messages exchanged are:

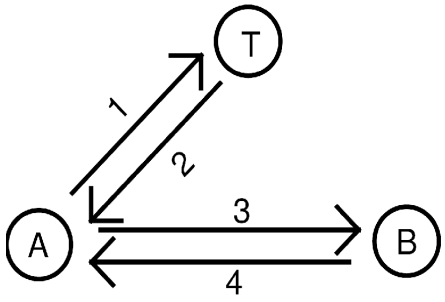

Figure 3: Simplified Kerberos protocol

$$
\begin{aligned}
1 &: A, B, n_A \\
2 &: \{k_{AB}, A, t_S, t_E, n_A\}_{k_{AT}},\ \{k_{AB}, A, t_S, t_E\}_{k_{BT}} \\
3 &: \{k_{AB}, A, t_S, t_E\}_{k_{BT}},\ \{A, t_A\}_{k_{AB}} \\
4 &: \{t_A\}_{k_{AB}}
\end{aligned}
$$

In the first message entity A asks the trusted third party T for a session key that enables A to securely communicate with B. Next, T generates a session key k_{AB} and assembles the reply message 2 consisting of a ticket and other data. The ticket contains the freshly generated session key k_{AB} and is encrypted in the long-term key k_{BT} shared between B and T. Besides the ticket, message 2 also includes the session key k_{AB} in a form readable by A, i.e. encrypted in the long-term key k_{AT} shared between A and T. Having received message 2, entity A decrypts the non-ticket portion of the message to obtain the session key k_{AB}. Next, A produces an authenticator, encrypts it using the key k_{AB}, and sends it along with the ticket to entity B. Entity B first decrypts the ticket using its long-term key k_{BT} and extracts the session key k_{AB}. The session key enables B to decrypt the authenticator from message 3, which, if successful, proves As identity since only a legitimate entity possessing k_{AB} can generate a valid authenticator. Finally, message 4, sent from B to A, confirms Bs true identity and completes the mutual authentication.

In order to determine the size of the messages shown in Figure 3, we assume that node identifiers and timestamps consist of 64 bits, whereas the session key has a length of 128 bits. Furthermore, we count

64 bits for the nonce n_A that is part of messages 1 and 2. Then, if the if the trusted party T is in the range of communication of nodes A and B, running Kerberos requires A to transmit 1344 bits and receive 1564 bits, while B needs to transmit 926 bits and receive 512 bits. Remember that the upper bound of the number of bits that a symmetric protocol can exchange before becoming less energy efficient than SOK is 510 bits for the UWM2000 modem and 422 bits for the UWM4000 modem. Clearly, Kerberos is more inefficient than SOK for the UWM4000 modem, while for the UWM2000 modem they can be considered to perform similarly, if we sum the energy spent by the two nodes. Notice that this comparison is made in the best scenario for Kerberos, that is, nodes A and B can reach the trusted party directly. However, unless the underwater wireless sensor network is small, it is almost always the case in large sensor networks that the sensor nodes are located far away from the base station. The communication energy cost of Kerberos depends not only on the transmit power level, but also on the number of intermediary nodes between A and T. Multi-hop communication between A and T increases the total energy consumption since any intermediary node has to forward (i.e. receive and retransmit) the message to its neighbor located on the route to the final destination. If this is the case, running Kerberos becomes extremely expensive from a global point of view, and SOK is definitely to be preferred.

Still, there are other symmetric protocols that are more energy-efficient than the SOK scheme: several protocols that belong to the Mathematical framework. The communication requirements of these protocols are similar to SOK (e.g. broadcast the ID of the node), but in computational terms the protocols use simpler primitives that do not have a high energy cost. For example, the Polynomial-based key pre-distribution schemes calculate $f(ID_i, ID_j) = f(ID_j, ID_i)$ (being f a bivariate polynomial) [37], whereas Blom-based key pre-distribution schemes calculate $A(ID_i) \cdot G(ID_j) = A(ID_j) \cdot G(ID_i)$ (being A and G specially crafted matrices) [38]. Observe that the Blom schemes may need to include an extra s value in their messages in order to construct the matrix G, although the size of the resulting message is still smaller than the aforementioned upper bound, and also s can be made equal to the node ID.

All these protocols (BROSK, Blom scheme, Polynomial scheme) are more energy efficient than SOK, but they lack certain security properties that may become essential in a real deployments. For example, the resilience of the Mathematical framework protocols is not very good. If an adversary captures enough nodes of the network, it may obtain information of the pairwise keys shared by other nodes. This is not the only disadvantage of these protocols: the scalability and the extensibility of the Blom scheme is unsatisfactory, and the security of both mathematical foundations (Blom schemes and bivariate polynomials) has not been formally demonstrated. About negotiation-based protocols like BROSK, the security of the exchange of pairwise keys can usually be assured only just after the deployment of the network. Therefore, an adversary can eavesdrop the negotiation process of either new nodes that want to establish communication with old nodes or nodes that move from their original position and want to open a secure channel with their new neighbourhood.

In comparison with any symmetric protocol in the Mathematical framework, non-interactive identity-based key agreement protocols like SOK offers better scalability, key connectivity, extensibility, and network resilience. The amount of information that has to be stored inside the nodes is independent of the size of the network, thus there are no size restrictions. Also, all nodes can exchange their IDs at any given time, thus it is possible to open a secure connection between any pair of nodes and to add new nodes to the network. Moreover, if an adversary captures a sensor node, it will only obtain the information related to the node, thus he/she will be unable to eavesdrop any ongoing communication between other nodes. Besides, due to special requirements such as node mobility [6], the batteries of underwater sensor nodes should have a higher capacity. As a result, the execution of a few pairings during the lifetime of the network will have a negligible influence in the node. Nevertheless, for specific deployments that consider energy as a primary factor, it is possible to use optimal symmetric protocols if their disadvantages (low

resilience, extensibility, scalability) are not important for the application.

5 Conclusions

In this work we have focused on the fact that wireless sensor networks consume considerable energy in sending and receiving data. We have studied how two alternative public key cryptography paradigms named as self-certified and identity-based cryptography respectively, can help to improve the energy cost of cryptographic key agreement between peers in a network. Our results bring the news that computationally intensive primitives like identity-based key agreement outperform symmetric and traditional key exchange protocol in specialized environments like underwater wireless sensor networks. Moreover, we have pointed out which symmetric key-based KMS could be better than the alternative public key cryptography paradigms in these UWSN environments, and we have shown how these symmetric protocols do not provide better security properties than the alternative public key paradigms.

Acknowledgements

The authors wish to thank Prof. Gene Tsudik and Dr. Roberto Di Pietro for their useful input during the development of this paper.

This work has been partially supported by the ARES CONSOLIDER project (CSD2007-00004) and the CRISIS project (TIN2006-09242). The second author was funded by the Ministry of Education and Science of Spain under the "Programa Nacional de Formacion de Profesorado Universitario".

References

[1] Großschädl J, Szekely A, Tillich S. The energy cost of cryptographic key establishment in wireless sensor networks. *ASIACCS*, ACM, 2007; 380–382.

[2] Kohl JT, Neuman BC. The Kerberos network authentication service (V5) 1993.

[3] Menezes A, Qu M, Vanstone S. Some new key agreement protocols providing mutual implicit authentication. SecondWorkshop on Selected Areas in Cryptography (SAC 95) 1995.

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

[5] Liu L, Zhou S, , Cui JH. Prospects and problems of wireless communications for underwater sensor networks. *Wireless Communications and Mobile Computing - Special Issue on Underwater Sensor Networks,* 2008; **8**(8):977–994.

[6] Hickey H. Underwater communication: Robofish are the ultimate in ocean robots, keeping in touch without scientists' help June 2008.

[7] Sakai R, Ohgishi K, Kasahara M. Cryptosystems based on pairing over elliptic curve (in japanese). *The 2001 Symposium on Cryptography and Information Security*, 2001. Oiso, Japan.

[8] Blake I, Seroussi G, Smart N. *Advances in Elliptic Curve Cryptography*, *London Mathematical Society Lecture Note Series*, vol. 317. Cambridge University Press, 2005.

[9] Oliveira LB, Scott M, Lopez J, Dahab R. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *5th International Conference on Networked Sensing Systems*, 2008; 173–180. Available at `http://eprint.iacr.org/2007/482`.

[10] Szczechowiak P, Kargl A, Scott M, Collier M. On the application of pairing based cryptography to wireless sensor networks. *WISEC*, ACM, 2009; 1–12.

[11] Choi KJ, Song JI. Investigation of feasible cryptographic algorithms for wireless sensor network. *Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006)*, 2006.

[12] Camtepe SA, Yener B. Key distribution mechanisms for wireless sensor networks: a survey. *Technical Report TR-05-07*, College of William & Mary March 2005.

[13] Alcaraz C, Roman R. Applying key infrastructures for sensor networks in cip/ciip scenarios. *1st International Workshop on Critical Information Infrastructures Security (CRITIS 2006)*, 2006; 166–178.

[14] Cui JH. Underwatersensor network lab — overview, achievements, plans. `http://uwsn.engr.uconn.edu` 2007.

[15] Akyildiz I, Pompili D, Melodia T. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks Jounal (Elsevier)* 2005; **3**(3):257–279.

[16] Crossbow. Wireless sensor nodes. `http://www.xbow.com/` 2008.

[17] LinkQuest. Underwater acoustic modems. `http://www.link-quest.com/` 2007.

[18] Shamir A. Identity-based cryptosystems and signature schemes. *CRYPTO 1984*, *LNCS*, vol. 196, 1985; 47–53.

[19] Girault M. Self-certified public keys. *EUROCRYPT*, vol. 574, 1991; 490–497.

[20] Petersen H, Horster P. Self-certified keysconcepts and applications. *Communications and Multimedia Security97*, 1997; 102–116.

[21] Strauss. Addition chains of vectors. *American Mathematical Monthly* Aug-Sept 1964; **71**(7):806–808.

[22] Gura N, Patel A, Wander A, Eberle H, Shantz SC. Comparing elliptic curve cryptography and rsa on 8-bit cpus. *CHES*, *Lecture Notes in Computer Science*, vol. 3156, Springer, 2004; 119–132.

[23] Scott M, Szczechowiak P. Optimizing multiprecision multiplication for public key cryptography. Cryptology ePrint Archive, Report 2007/299 2007. `http://eprint.iacr.org/`.

[24] Atmel. Atmega128 product description. `http://www.atmel.com/dyn/products/product_card.asp?part_id=2018` 2007.

[25] Brumley BB. Efficient three-term simultaneous elliptic scalar multiplication with applications. *Proceedings of the 11th Nordic Workshop on Secure IT Systems—NordSec '06*, Fåk V (ed.), Linköping, Sweden, 2006; 105–116.

[26] Menezes A, , van Oorschot P, Vanstone S. *Handbook of Applied Cryptography*. Discrete Mathematics and its Applications, CRC Press, 1997. Available at `http://www.cacr.math.uwaterloo.ca/hac/`.

[27] Boneh D, Franklin M. Identity-Based encryption from the Weil pairing. *SIAM Journal of Computing* 2003; **32**(3):586–615. This is the full version of an extended abstract of the same title presented at *Crypto'01*.

[28] Verheul ER. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology* 2004; **17**(4):277–296.

[29] Barreto PSLM, Galbraith SD, O'Eigeartaigh C, Scott M. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography* 2007; **42**(3):239–271.

[30] Oliveira LB, Aranha DF, Morais E, Daguano F, López J, Dahab R. Tinytate: Computing the tate pairing in resource-constrained sensor nodes. *NCA*, IEEE Computer Society, 2007; 318–323.

[31] X9 ASC. American national standard x9.62-2005, public key cryptography for the financial services industry, the elliptic curve digital signature algorithm (ecdsa) 2005.

[32] Dupont R, Enge A. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics* 2006; **154**(2):270–276.

[33] Mehta M, Huang D, Harn L. Rink-rkp: A scheme for key predistribution and shared-key discovery in sensor networks. *Proceedings*

of the 24th IEEE International Performance Computing and Communications Conference (IPCCC'05), 2005; 193–197.

[34] Zhu S, Setia S, Jajodia S. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks* November 2006; **2**(4):500–528.

[35] Seshadri A, Luk M, Perrig A. Sake: Software attestation for key establishment in sensor networks. *2008 International Conference on Distributed Computing in Sensor Systems (DCOSS'08)*, 2008; 372–385.

[36] Anderson R, Chan H, Perrig A. Key infection: Smart trust for smart dust. *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP 2004)*, 2004; 206–215.

[37] Liu D, Ning P, Li R. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security* 2005; **8**(1):41–77.

[38] Du W, Deng J, Han YS, Varshney P, Katz J, Khalili A. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security* 2005; **8**(2):228–258.