Secure-TWS: Authenticating Node to Multi-user Communication in Shared Sensor Networks

Leonardo B. Oliveira
Unicamp
leob@ceset.unicamp.br

Aman Kansal
Microsoft Research
kansal@microsoft.com

Bodhi Priyantha
Microsoft Research
bodhip@microsoft.com

Michel Goraczko
Microsoft Research
michelg@microsoft.com

Feng Zhao
Microsoft Research
zhao@microsoft.com

ABSTRACT

Recent works have shown the usefulness of network and application layer protocols that connect low power sensor nodes directly to multiple applications and users on the Internet. We propose a security solution for this scenario. While previous works have provided security support for various communication patterns in sensor networks, such as among nodes, from nodes to a base station, and from users to nodes, the security of communication from sensor nodes to multiple users has not been sufficiently addressed. Specifically, we explore this design space and develop a security solution, named Secure-TWS, for efficient authentication of data sent by a resource constrained sensor node to multiple users, using digital signatures. We investigate the resource overheads in communication and computation for two most suitable signature schemes – the Elliptic Curve Digital Algorithm (ECDSA) and the Boneh-Lynn-Shacham (BLS) short signature schemes. We implement these schemes on two popular sensor node architectures (based on MSP430 and ARM processors with 802.15.4 radios) and experimentally characterize relevant trade-offs.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and Protection*; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless Comunication*

General Terms

Security

Keywords

Security; Cryptography; Digital Signatures

1. INTRODUCTION

This paper describes the implementation of a security solution for sensor nodes that are shared by multiple users. Shared sensor nodes are useful in many scenarios such as when a common sensing substrate is used by multiple applications. While the sharing of sensors over the Internet is not new [1], recent works have demonstrated the usefulness of methods that connect low power sensor nodes directly to applications, locally and over the Internet, without intermediary gateways, such as to improve interoperability, deployment re-use, and reduced costs [2]. Network protocols at the IP layer [3, 4] and the application layer [2] have shown energy efficient methods to enable this mode of operation. Figure 1 shows these communication scenarios.

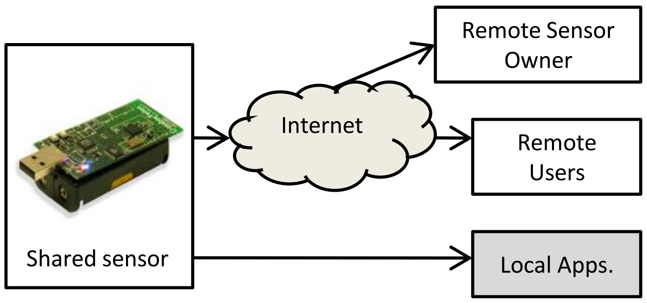

Figure 1: Shared sensor scenario for which security solution is implemented.

Implementing such a direct connection from sensor nodes to multiple clients in practice further entails providing *end to end security* for data communication. The need for security directly from the source node also arises in scenarios where the sensor node is deployed by one entity but it sends its data over a network device provided by an untrusted entity. For instance, a power metering node for real time demand-response pricing may be deployed by the utility company (shown as Remote Sensor Owner in the figure) in a home and the node may use the home Internet router supplied by the home owner. The utility company trusts only the sensor node but not the intermediate network device. The sensor node must supply authenticated data directly, rather than relying on the Internet router. The data may also be accessed by other interested clients such as the home user locally, or the building landlord remotely, who may all want authenticated data. Our implementation complements the network layer and application layer protocols available for these scenarios with support for security.

The fundamental communication pattern here is from a sensor node to multiple users. A first requirement for security is that multiple users be able to authenticate the data they receive from a sensor node. Authentication is arguably the most important security property in securing WSN communication [5] . We describe our implementation to achieve authentication and provide experimental evaluations that help decide among key design choices involved. The design is optimized to provide authentication for node to multi-user communication from resource constrained sensor nodes. We explore the design space and show that a digital signature based approach is the most efficient choice for this scenario, especially since the resource constrained sensors may not be able to establish shared keys with a changing set of multiple users. We compare different digital signature schemes for authenticating node to multi-user communication. The goal of the paper is not to invent new cryptography primitives but rather to close the gap between research and practice by evaluating certain key deign choices in realizing the security implementation for a shared sensor scenario.

To sum up, our key contributions are:

1. We demonstrate how node-to-multiuser communication can be authenticated in WSN's, as well as identify and evaluate the design choices involved.

2. We provide a comparison between Boneh-Lynn-Shacham (BLS) short signature scheme [6] and Elliptic Curve Digital Signature Algorithm (ECDSA) in the context of WSN's (note that ours is the first implementation of BLS on sensor platforms and the comparison results differ from prior comparisons performed on a Pentium-III processor [7]).

3. We provide an authenticated web service communication solution, named Secure Tiny Web Service or *Secure-TWS*, for shared and interoperable sensor nodes, based on integration of our security implementation with the tiny web service stack from [2] along with its underlying IP stack [3]. The end result is a complete solution for node-to-multiuser communication, leveraging the existing lower layer functionality for this scenario.

The remainder of this work is organized as follows. Section 2 discuss the overall design motivation for the Secure-TWS security implementation along with its usage setup. A detailed discussion of the implementation issues that were evaluated in our experiments is provided in Section 3 and the evaluation results are discussed in Section 4. Related work and conclusions appear in Sections 5 and 6, respectively.

2. AUTHENTICATION FOR SHARED SENSOR SCENARIO

Authentication consists of two properties: i) *source authentication*, which ensures a receiver that the message did in fact originate from the claimed sender; and ii) *data authentication*, that guarantees a receiver that the message received is "fresh" (i.e. it is not a replay attack) and its content was not changed since it left the sender. A second aspect of data security is privacy, typically ensured using data encryption. The implementation presented in this paper does not address the privacy aspect; privacy might not even be needed for many sensors shared over the Internet.

In this section, we identify the key security primitives that are applicable for providing authentication and select a small number of appropriate design options that need to be evaluated for enabling efficient implementations. The system to provide the required security features is described. The selected design options and parameters are then discussed and evaluated with their implementations in subsequent sections.

2.1 Design Space

Authentication can be achieved using one of two types of security primitives: *using message authentication codes (MAC's)* and using *digital signatures.*

2.1.1 Using Message Authentication Codes

Methods that use MAC's begin with an initialization step that involves distributing the shared keys in a secure manner. Once keys are distributed, authentication in a pairwise communication pattern is straightforward: sender uses the shared key to generate a message authentication code that may be verified by the receiver. Computation overhead for symmetric key cryptography is very low [8] and many embedded processors have hardware support for it, such as a hardware implementation of AES.

The use of MAC's for a node to multi-user scenario is possible, such as demonstrated in μTESLA [8] and LEAP [9]. The μTESLA scheme was described for authentication of messages from a base station to multiple nodes and the LEAP scheme for single hop communication. Both schemes use a one way key chain (a sequence of keys $k_1, \ldots k_n$, where k_{i+1} is generated from k_i by applying a one-way hash function $f()$, i.e., $k_{i+1} = f(k_i)$) to achieve authentication. All nodes who are supposed to receive the authenticated messages must be supplied with a group key k_n in a secure manner. This group key is common to all nodes. In μTESLA all nodes also have to maintain a synchronization with the sender regarding which key in the sequence is currently valid. Clearly, these requirements are not well-suited for our scenario where a sensor node is shared over the Internet by a changing set of multiple clients.

The special case of one-to-one communications when the connection is between a resource constrained sensor node and a single client connected via the Internet was implemented in [10] using SSL, and named Sizzle. The conventional public key cryptography (PKC) mechanisms used in SSL for exchange of shared keys were replaced with a resource efficient version of PKC, based on elliptic curve cryptography (ECC), leading to a lighter version of SSL suitable for resource constrained sensor nodes. Applying Sizzle directly for node to multi-user communication communication is inefficient as the sensor node would then have to run the SSL handshake with every one of its clients on the Internet, including short term and transient ones. Also, the data would have to be sent multiple times to each client since a different session key would be used by each client, and no multi-cast or sharing of the same data message would be feasible.

2.1.2 Digital Signature Based Authentication

The alternative option of using digital signatures leads to certain advantages over the approach of using MAC's in this node to multi-user scenario:

1. When signatures are used, the node is not required to establish shared secret keys with each client who wishes to receive authenticated data.

2. Shared keys need not be managed or stored at the resource constrained node.

3. The same authenticated message can be sent to multiple users and forwarded or multi-cast to other users retaining its authentication properties. This allows the network layer to optimize service to multiple simultaneous clients using multi-cast without requiring the sensor node to send multiple packets.

The design options for implementing digital signatures for sensor networks are discussed next.

A digital signature allows a sender (signatory) to generate a signature on a message. The receiver can verify the authenticity of the signature to ensure that the message indeed originated from the claimed sender and has not been modified since. The signature is generated using a private key known only to the sender and verified using a public key known publicly to everyone including the receivers. An adversary cannot forge a sender's signature without the sender's private key. Implementing digital signatures thus involves providing two components – a method to obtain [public, private] key pairs used for signing and verification, and efficient computation of the signatures.

The [public, private] key pairs can be obtained either using certificate based schemes or certificate-free schemes. In certificate based schemes, the key pair is associated with a particular user by a mutually trusted entity, sometimes referred to as a certification authority (CA). The trusted entity signs a user's credentials using its own private key and everyone who trusts this entity can associate the given public key to that user.

Among certificate-free schemes, many are based on identity-based methods [11]. In this approach, some unique information that correctly identifies the user (such as an email address or an IP address) is used to derive their public key. No certification is thus needed to bind the public key to the user. The keys are generated by an unconditionally trusted authority (TA) for each user. While the identity based scheme has lower overheads in managing the issuance and verification of certificates, it has the drawback that the TA in this case knows everyone's private keys and can impersonate any user. While such a trusted authority is easy to implement within a single sensor network, such as when all nodes fully trust the base station [12], it is not easy to provide in an Internet based sharing scenario with many different types of users.

A certificate-free scheme that does not require a TA has been proposed in [13], named *certificateless*. However this scheme has a very high computation complexity and is not suitable for resource constrained sensor nodes. On the other hand, CA's are already a part of the existing infrastructure. Note that CA's are easier to provide in the Internet since the users only trust the CA to reliably bind public keys to themselves, and do not have to allow the CA to be able to impersonate as themselves. Hence, in our implementation, we use the certificate based scheme.

While obtaining the key pair is a one time overhead, the second component, the computation using the keys for signing and verification, is involved in every message exchange. We now consider the computation and communication overheads of digital signatures, for optimizing the design of this step. The signature computation choices available are described below.

Digital Signature Algorithm (DSA): DSA is a commonly used certificate based signature scheme in Internet communication. Its security relies on the Discrete Logarithm Problem (DLP). The DLP is not a fully exponential problem and hence the size of computation parameters required in signature computation is rather large, making it ill suited for sensor nodes.

Elliptic Curve DSA (ECDSA): ECDSA is the Elliptic Curve based counterpart of DSA. However, its security relies on the Elliptic Curve DLP which is a fully exponential problem. The size of parameters needed is much smaller and the signature computation becomes significantly faster than when using DSA. TinyECC [14], for example, showed the generation of signatures on an MSP430 based sensor platform in only 1.6s. ECDSA's signature length, however, is as long as DSA's. For instance, for a RSA-1024 security level, both protocols produce a 320-bit signature.

Boneh-Lynn-Shacham (BLS) Scheme: BLS [6] is a certicate-based signature generation and verification scheme that relies on pairings [15]. It has the advantage that its signature bit-length is half that of DSA's and ECDSA's for RSA-1024 security level. For this reason it is also referred to as a short signature scheme. Its computation overhead is asymmetric: heavier computation is needed on the receiver side but this is not a major concern for the scenario of interest.

The resource overheads for the above options are listed in Table 1. The exact cost may vary by specific implementation; the table accounts for the fundamental steps involved, using specific examples. Examples of identity-based and certificateless schemes are also included for completeness, though they are clearly not strong contenders for our design choices.

The communication overhead is quantified using the extra bits needed for security, in addition to the data and protocol headers. The computation overhead is listed in terms of the computationally intensive operations involved. A *point multiplication* (denoted using 'pt multiply' in the table) is a multiplication operation of a point, $\mathbf{P}$, on the elliptic curve by a scalar, k, to obtain $\mathbf{Q} = k\mathbf{P}$. This represents $\mathbf{P}$ added to itself k times where the addition is as defined in the elliptic curve group. A *pairing* is a computable, non-degenrate function that has a special property known as bilinearity. The Tate pairing [15] is used in our implementation. An *exponentiation* (denoted 'exp' in the table) is a modular exponentiation. i.e., a computation of the form a^b mod c.

Note that in the above schemes point multiplication works with 160-bit parameters while the pairings and exponentiation work with 1024 bit parameters, making the point multiplication here relatively less computationally intensive than the other two operations. The parameter sizes come from the underlying problems of the cryptosystems, for achieving RSA-1024 equivalent security.

2.2 Authentication Setup

Based on the discussion of design options above, we can now select the overall authentication procedure for the security scenario of interest. To summarize, a signature based scheme is preferred over MAC based ones. Among signature based schemes, a method based on certificates is preferred over certificate-free ones. Such an authentication scheme is implemented in Secure-TWS. It operates in shared wireless sensors as shown in Figure 2 and described below. The procedure does not involve any processing at the sensor specific to the client identity and the approach is hence scalable to any number of short term or long term users of the sensor node.

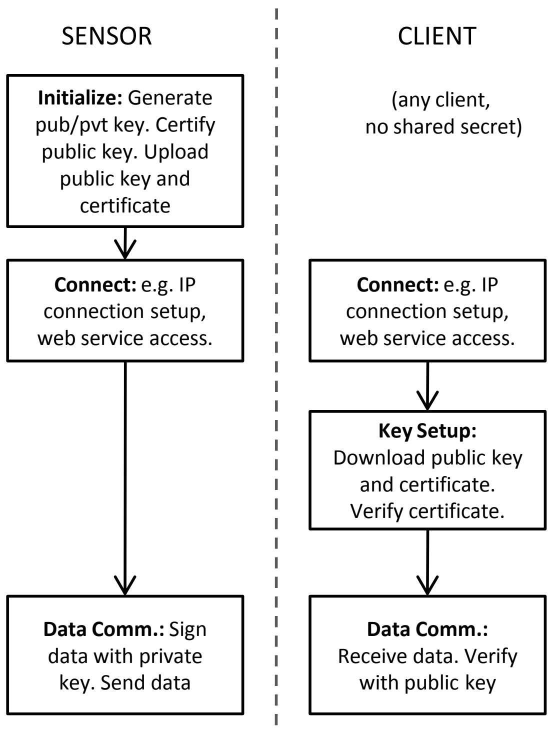
SENSOR
CLIENT
Initialize: Generate pub/pvt key. Certify public key. Upload public key and certificate
(any client, no shared secret)
Connect: e.g. IP connection setup, web service access.
Connect: e.g. IP connection setup, web service access.
Key Setup: Download public key and certificate. Verify certificate.
Data Comm.: Sign data with private key. Send data
Data Comm.: Receive data. Verify with public key

Figure 2: Overview of the authentication procedure implemented in Secure-TWS.

Signature Scheme	**Computation**		**Communication** (bytes)
	generation	**verification**	
Identity-based[16]	2 pt multiply	1 pt multiply + 1 pairing	40
Certificateless[13]	1 pt multiply + 1 pairing	4 pairings	40
DSA	1 exp	2 exp	40
ECDSA	1 pt multiply	2 pt multiply	40
BLS	1 pt multiply	2 pairings	20

Table 1: Signature schemes's requirements and costs

Initialization: Prior to deployment, such as when the application code is loaded into the sensor node, a [public,private] key pair, $[p_{pub}(i), p_{pvt}(i)]$, is generated for each node $x(i)$. The private key $p_{pvt}(i)$ is loaded onto the node. The private key may optionally be stored in a sensor node management database by the sensor owner but these need not be shared with the certification authority or anyone else. The public key, $p_{pub}(i)$, should be provided to a certification authority. The CA will verify the deploying entity's identity and issue a certificate that binds $p_{pub}(i)$ to $x(i)$. Subsequently, the public key and the certificate are uploaded to key servers from where clients may download them directly as needed. This helps alleviating the need for nodes to transmit public keys and certificates themselves.

Connection Setup: Client applications that wish to use data from a sensor node $x(i)$ may establish a connection to the sensor node using the tiny web services stack [2] included in Secure-TWS, or a lower layer Internet Protocol stack [3, 4]. Multi-cast may be used to support multiple simultaneous clients when available.

Authenticated Data Access: The client now downloads the sensor node's public key, $p_{pub}(i)$ and the corresponding certificate from a key server. The CA's signature on the certificate is verified by the client. Node $x(i)$ signs the application layer data using one of the certificate based signature generation schemes and sends it to the client. The clients can authenticate the data using signature verification based on $p_{pub}(i)$.

2.3 Key Design Choices

The certificate based digital signature scheme in the Secure-TWS setup described above could be based on either the DSA, ECDSA or the BLS algorithms. DSA is clearly worse than ECDSA for sensor nodes due to its large parameter sizes and exponential computation, and hence we compare BLS and ECDSA. The computation performed at the sensor node only involves signature generation since the verification is performed at the client which is not as resource constrained. Considering the computation overhead of signature generation in table 1, we see that both ECDSA and BLS involve one point multiplication. BLS does have a lower communication overhead. Considering theoretical computation complexity, i.e, the order of computation alone, one may expect that BLS is better. However in actual implementation, it turns out that exact computation costs vary greatly among the two choices due to the specific computation and parameter values involved. The choice between these two options is explored in greater depth in the next section using actual implementations of both schemes on two common sensor platforms.

3. SECURE-TWS IMPLEMENTATION

We now discuss in depth the design parameters and resource overheads that affect the choice of BLS vs. ECDSA for shared sensor node usage. We also highlight specific implementation issues we faced in implementing these security primitives on two commonly used sensor platforms.

3.1 Platform and Software

We use the mPlatform [17] hardware to implement the Secure-TWS authentication solution on two commonly used low power processors: the MSP430-F2418 (16-bit 16MHz core, 8kB RAM, 116kB ROM) and the ARM LPC2138 (32-bit 60MHz core, 32kB RAM, 512kB ROM). The mPlatform has hardware boards for both these processors as well as an IEEE 802.15.4 radio. This allows us to evaluate both computation and communication. A block diagram of the software implementation is shown in Figure 3.

As shown, the security methods of Secure-TWS are implemented on top of the Tiny Web Service [2] stack along with its IP stack [3]. This makes Secure-TWS a complete solution for the usage context of direct communication from a sensor node to multiple applications or users since tiny web services already provide the necessary network layer and application semantics functionality for that scenario. Secure-TWS can of course be used with other types of lower layer network protocols. The integration with tiny web services has been tested for the MSP430 processor only while the Secure-TWS layer has been tested on both the MSP430 and ARM processors.

The key elements of the implementation are the ECDSA

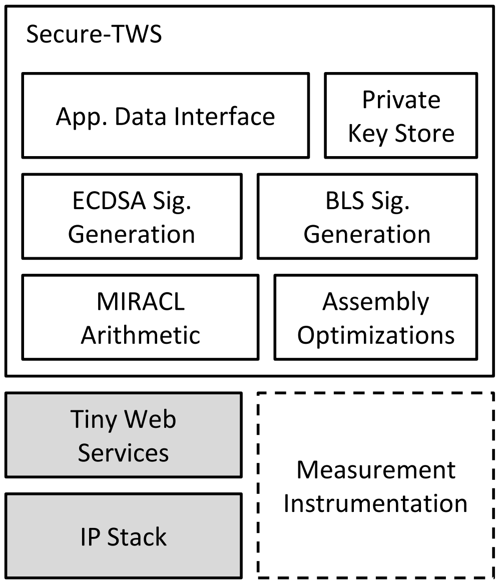

Figure 3: Block diagram of the security solution implementation.

and BLS signature generation methods. Note that only one of these is required in a specific instantiation and we only compile the software to produce the complete system containing one of these two schemes. Assembly language optimizations have been included to improve the computation performance of signature generation.

The finite field arithmetic and big number arithmetic used for the ECC based computation is implemented using the Multi-precision Integer and Rational Arithmetic C/C++ Library (MIRACL) from [18]. MIRACL is a publicly available C library that implements all the arithmetic primitives required in our implementation and has support for several popular platforms including the ones used in our implementation. In addition, MIRACL allocates memory exclusively from the stack, which is a useful feature for development on resource-constrained platforms. The crucial operation used in the prime finite field arithmetic: multiplication – a big integer multiplication followed by a reduction modulo p, used MIRACL's implementation of the Comba [19] and Montgomery [20] methods for these operations.

Note that the private key is stored locally on the node as it is used for signature generation. If the node is compromised, this key may be stolen. However, the public private key pair for the node is exclusive to that node itself and there is no shared key that may compromise other nodes if this node is compromised.

Our prototype implementation also includes certain measurement instrumentation in our code that sends signals on external processor pins to allow measuring the computation and communication times involved in various key stages. The measurement process is described in Section 4. This instrumentation is not considered a part of Secure-TWS but is included only for evaluation purposes.

The application data interface uses C function calls. It allows using our implementation easily with the underlying tiny web service stack [2]. Sample usage is shown in Figure 4, with simply a function call made for signature generation, followed by commands to send the message.

```
// Application data collection and processing
...

// generate signature and signed message
ec_sign(msg,sig); //generate signature
strcpy(sigmsg,msg); //copy orig msg
strcat(sigmsg,sig); //append signature

// prepare to send a signed message
s->send_data_ptr = (char *) sigmsg;
s->send_data_len = sizeof(sigmsg)-1;
```

Figure 4: Secure-TWS interface usage example

Next, we consider the choice of parameter values required for the various security modules in the implementation.

3.2 Parameter Choices

In certain sensor network scenarios, where security is limited to within the network, a 64-bit security level has sometimes been used [8] to help reduce security overheads. However, in our scenario, since the authentication must interface with the Internet, we choose to use a 80-bit security level (or 1024-bit RSA) for greater security.

An important choice involved in the implementation is that of the finite fields and elliptic curves used for ECDSA and BLS, for achieving the RSA-1024 security level. The two most popular finite fields used in ECC implementation are the prime and binary fields. Prime fields are preferred due to two reasons. Firstly, binary field multiplication is not presently supported by most processors and its evaluation must thus be performed in software, increasing the computation time. Secondly, the fact that the DLP in prime fields is believed to be harder than the DLP in binary fields allows one to work with smaller parameters that, in turn, speeds up the computation. In the particular case of ECDSA there is one more reason for using prime fields. The protocol can use a *Mersenne prime* (i.e., a number of the form 2^p-1) and leverage the prime's special form for computing modular *reduction* efficiently, an essential operation in the underlying primary field arithmetic. Thus, for ECDSA, using prime fields is an accepted choice. We use the elliptic curve $y^2 = x^3 - 3x + 157$ with the prime based on $p = 2^{160} - 2^{112} + 2^{64} + 1$.

The selection between primary and binary fields for BLS, however, depends on numerous aspects and is not so straightforward. The verification process in BLS requires the computation of pairings. This restricts the choice of the underlying finite field's parameters to pairing friendly curves. It is hard to find such curves and one may no longer expect the suitable curve to allow using a special prime that would speed up the signature generation computation. Curves that do help speed up the pairing computation have been found, and are used in our implementation, but since the verification is performed only at the client, this does not help the computation at the sensor.

To help speed up the reduction operation of the signature generation phase, one may however use binary fields, where the reduction is carried out modulo a polynomial, rather than a prime, and is hence faster. However, with binary fields, finding a pairing friendly curve leads to larger parameter sizes, increasing computation overhead, and larger signature sizes, increasing the communication overhead, for ensuring the RSA-1024 equivalent security. This happens because to achieve short signature sizes an embedded degree, $k = 6$, has to be used for the pairings. However, pairing friendly curves over binary fields found so far have $k = 4$ at the most, leading to an increased signature size of 256 bits, with no savings in computation complexity at sensor side. Hence, we use prime fields which allow shorter signature length, while accepting the overhead of not using a special prime. We use the nonsupersingular MNT curve $y^2 = x^3 - 3x + B$ with $k = 4$ for the BLS with Tate pairings and the underlying parameters given as:

```
p = 7DDCA613A2E3DDB1749D0195BB9F14CF44626303
b = 21C3F3AC7864D1F1F99273D0F828D3657D8CFD4E
r = BB94C2745C7BB62E93BBADB1D6BFF48B23BE531
```

3.3 Point Multiplication

As mentioned in Table 1, elliptic curve scalar point multiplication is the most expensive operation in ECDSA and BLS signature generation. However, the computation complexity of this operation varies between ECDSA and BLS.

ECDSA: In ECDSA, as mentioned before a computation friendly prime can be used. Further, the point P to be multiplied is a fixed public parameter that is known *a priori*. In this case, point multiplication can be accelerated significantly using precomputation [19] at the expense of some storage overhead. By using the Comb method [19], for example, $2^w - 1$ points are precomputed offline (i.e. $2P$, $3P$, …, and $(2^w - 1)P$), where w is the number of bits processed at once. In our ECDSA implementation, we have used $w = 4$ resulting in 16 precomputed points stored in ROM. Each precomputed value requires 40 bytes for storage resulting a storage overhead of 640B. Such an overhead is acceptable in most situations given the ROM sizes of current sensor platforms.

BLS: In BLS, on the other hand, the point P's value is dependent of the message being signed and thus assumes a different value each time a signature is generated. Precomputation is hence not possible, leading to a higher computation complexity for the same point multiplication operation.

4. EVALUATION AND RESULTS

This section presents the performance and resource overheads of the Secure-TWS solution implemented using the above parameters and compares these overheads for ECDSA and BLS. Specifically, the measurements evaluate i) storage, ii) computation, and iii) communication overheads.

The experimental system used for performing the measurements is shown in Figure 5. The hardware used is

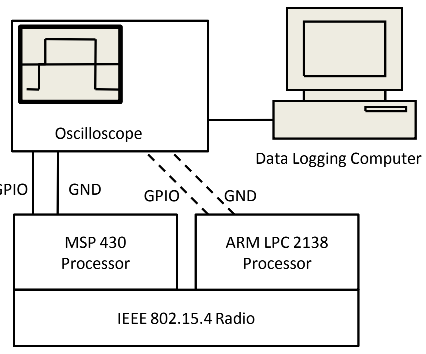

Figure 5: Experiment setup to measure resource overheads of implemented security schemes.

from the mPlatform kit [17]. The setup consists of the mPlatform MSP430 board (that has the MSP430 processor and an IEEE 802.15.4 radio), and an mPlatform ARM board. The Secure-TWS implementation is compiled for both these processor platforms. The software has been instrumented to generate a signal on one of the GPIO pins at each crucial computation and communication step. The GPIO pin and a ground pin are brought out using jumper wires and connected to an oscilloscope for measuring the timing between the crucial steps. The power drawn by each board is measured separately by measuring the voltage across a sense resistor connected to the supply rail of the processors. The measurements of ECDSA and BLS overheads for the two platforms are discussed below.

4.1 MSP430 Platform

The MSP430 is representative of a class of very low power sensor nodes that have been currently prototyped. The three types of overheads studied are as below.

4.1.1 Storage

Table 2 summarizes storage requirements for BLS and ECDSA schemes on the MSP430-F2418. Note that virtually all the RAM memory is allocated from the stack, which means that once cryptographic operations are completed the memory becomes available for the sensor applications. Since this RAM is only used before sending a message, the memory is available for data collection and processing operations at other times for the applications.

	Algorithm	**Memory Used**
RAM	BLS	2.9 kB
	ECDSA	2.9 kB
	ECDSA+Precomputation	2.9 kB
ROM	BLS	47.0 kB
	ECDSA	31.3 kB
	ECDSA+Precomputation	31.9 kB

Table 2: Secure-TWS's memory overheads on the MSP430-F2418 processor.

Adding the numbers in the table to the ROM overhead of the underlying tiny web services module, the total ROM required is 47.6kB for ECDSA with precomputation and 62.7k for BLS.

4.1.2 Computation

The ECDSA and BLS schemes are often referred to as having similar computation requirements for signature generation. Theoretically, this is not incorrect: the computationally intensive step that both schemes require is a point multiplication which in practice incurs costs of the same order of magnitude. In power machines, these costs are therefore equivalent [7]. In practice, however, costs of the same order are not necessarily equivalent, especially in resource constrained sensor platforms. Having implemented BLS on a real sensor platform allows measuring these differences in a more precise manner.

Figure 6 shows the computation costs for ECDSA, ECDSA with pre-computation (denoted ECDSA-PRE in the figure), and BLS on the MSP430 platform.

On a Pentium-III 1GHz processor, BLS was found to have lower computation overhead than ECDSA [7]. The figure shows that this is not the case for an MSP430 processor, re-emphasizing the fact that insights from measurements on a different class of platforms may not apply directly to sensor networks. One of the likely causes for this variation is that our implementation used careful parameter selection for ECDSA to reduce its overheads without affecting its security properties. The focus of the work in [7] was on the receiver side (i.e., on signature verification) and the computation of signature generation for ECDSA and BLS were presented mainly as an illustration, without dwelling on optimization of its parameter selection. Another possible cause of the variation is the availability of computation acceleration hardware built into the Pentium processor that the lower power processors do not have.

The ECDSA computation shown here is more than twice as fast as a prior implementation [14] for a similar platform primarily because the clock frequency of the MSP430 used in our hardware was 16MHz instead of 8MHz as in [14], and also because more efficient libraries and compilers were used.

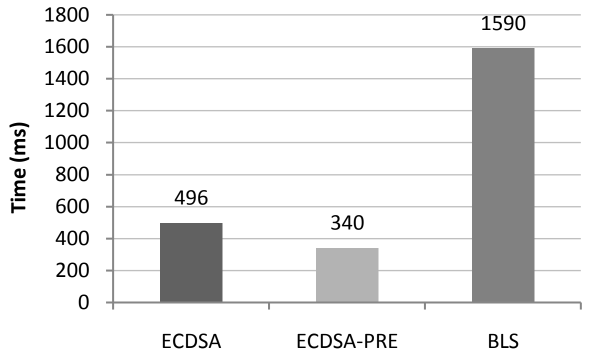

Figure 6: Computation overhead of Secure-TWS options on the MSP430 platform

Using precomputation allows ECDSA to speed up by 31.4%, which is significant. ECDSA-PRE is approximately 4.7 times faster than BLS.

As mentioned before (Section 3), ECDSA can make use of a special prime to speed up the computation. Further, during signature generation, the BLS method needs to map messages into elliptic curve points, an operation that does consume a non-negligible amount of time. In our measurement we found that 28% ofx the time is spent in mapping the message to the curve, and 70% in the the actual point multiplication, in BLS.

4.1.3 Communication

In our implementation of BLS, the signature bit-length is exactly half that of ECDSA's signature bit-length. Table 3 shows how this affects the communication energy cost of transmitting the signature portion of a data packet. The size of the signature does not depend on the size of data sent. The size of the data is not affected by the choice of the signature scheme and is hence not included in the comparison. The total communication cost would of course depend on the size of data and packet headers used.

Note that the energy savings in BLS are not exactly half that of ECDSA because of radio start-up energy costs that are fixed for both cases. Radio costs are discussed in more detail in [21]. The signature length is not affected by precomputation in ECDSA and the energy cost is hence not shown separately.

Signature Scheme	Energy (mJ)
ECDSA	0.23
BLS	0.15

Table 3: Communication energy consumption for signature schemes implemented in Secure-TWS (MSP430)

Another small advantage of a shorter signature is in the delay: while the ECDSA signature requires 2.6ms for transmission, the BLS signature takes only 1.7ms.

4.1.4 *Combined Resource Overhead*

As can be seen from the above data, the computation energy dominates, while the size of the signature is not a significant issue. While it is often the case that on a sensor node the energy to communicate dominates the energy to computation and some additional computation that reduces data size is usually helpful to save energy, in the case of digital signature computation, the computation overhead is the dominant drain on energy.

The overall costs are shown in Figure 7. In fact, the cost of computation is so much higher that even when the transmission of the signed data to multiple recipients and over multiple wireless hops is considered, the one time cost of computing the signature is still dominant for a reasonable number of clients. The comparisons show that Secure-TWS should be used with the ECDSA signature option in most scenarios.

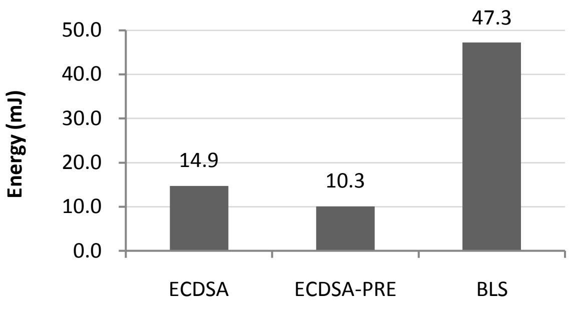

Figure 7: Energy overhead of different signature schemes in SecureTWS on the MSP430 platform

4.2 ARM

The ARM7 architecture represents more capable sensor nodes than the MSP430 but is also a low powered processor used in embedded systems.

4.2.1 *Storage*

Table 4 summarizes storage requirements for BLS and ECDSA schemes on the ARM. Given the ROM available in this processor to be 512kB, the ROM overheads are likely to acceptable in most situations.

	Algorithm	Memory Used
RAM	BLS	9.22 kB
	ECDSA	9.22 kB
	ECDSA+Precomputation	9.22 kB
ROM	BLS	61.63 kB
	ECDSA	48.59 kB
	ECDSA+Precomputation	49.23 kB

Table 4: Secure-TWS's memory overheads on the ARM processor.

Note that the numbers in Table 4 are obtained when assembly optimizations are included to speed up the multiplication related operations. Using assembly does limit the portability of the code, and in some situations it may hence be desirable to use only the C source without such optimizations. The non-optimized (more portable) code requires 5.9kB of additional ROM and uses the same amount of RAM, again allocated only from the stack.

It may further be noted that the RAM usage differs from that on the MSP since the cryptography primitives are implemented using platform specific assembly libraries, and even though the algorithms are the same, the actual implementation code differs.

4.2.2 *Computation and Communication*

Figure 8 shows the computation time overheads for this platform. As seen on the MSP430, the computation costs of ECDSA and BLS vary significantly even on the ARM. The absolute time taken is of course significantly lower, almost by a factor of 16 for both signature generation options. Another difference is that for BLS on the ARM, the mapping of the message to the elliptic curve consumed 47% of the time and the the point multiplication itself took 52%, making the point mapping even more significant than on the MSP.

The communication energy costs are shown in Table 5.

The radio used on the ARM platform is the same as used on the MSP based platform. However, the energy cost of communication is different from that shown in Table 3 because the processor is also powered up during the communication and the two processors operate at

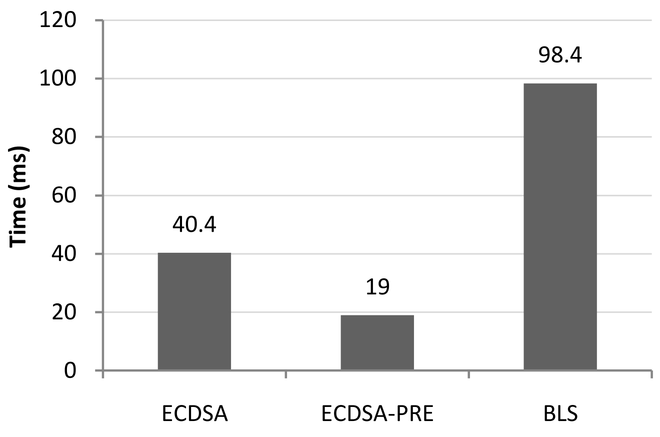

Figure 8: Computation time overhead on the ARM platform.

Signature Scheme	Energy (mJ)
ECDSA	0.59
BLS	0.40

Table 5: Communication energy consumption on the ARM processor.

different power levels.

The total energy costs are as shown in Figure 9. Note that even though the ARM is a higher power processor, the energy cost of adding security is significantly lower than on the MSP430. Again, the energy cost of ECDSA is better, making it the preferred option for most cases.

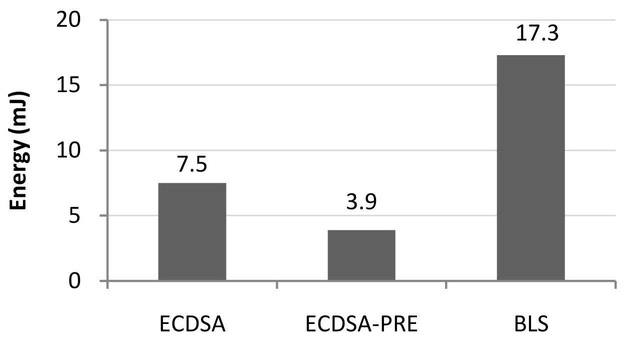

Figure 9: Energy cost of Secure-TWS authentication on the ARM platform.

5. RELATED WORK

Several works have addressed the problem of providing security in wireless sensor networks. Communication in WSNs exhibits a number of different patterns. To be effective and efficient, a solution needs to be tailored to the particular communication pattern at hand. This has lead to several methods for security for different scenarios:

1. Node to node [1] (e.g. [8, 22, 9, 23, 24] among others);
2. Node to multiple nodes within same sensor network (e.g. [25]);
3. User to node (e.g.[8])
4. Node to user (e.g.[10]).
5. User to multiple nodes (e.g. [8]);

In this work we consider a communication pattern from a sensor node to multiple Internet connected users. This has not been sufficiently addressed before. The most closely related works are those in one-to-many communication within sensor networks. Majority of these proposals make use of authenticated broadcasts, based on symmetric cryptosystems, such as μTESLA [8]. The μTESLA approach has been studied and improved for specific contexts in follow up works (e.g [26, 5, 27, 28]). A slightly different approach, specifically targeted to local broadcasts, was proposed by Zhu *et al.* [9]. As discussed in Section 2.1.2, those strategies, albeit very effective for the scenarios they are designed for, are not adequate for authenticating node to multi-user interactions.

The authentication in the reverse direction, from a user to multiple sensor nodes has also been considered before. Ren *et al.* [29] have combined Merkle trees, Bloom filter, and PKC based signature schemes for this purpose.

Our implementation makes use of digital signature schemes, based on PKC. PKC has already been shown to be feasible in resource-constrained sensor nodes [30, 31, 32, 14]). For instance, Gura *et al.* [30] reported results for ECC and RSA primitives on the ATmega128L and demonstrated the advantages of ECC. We use ECC based PKC in our implementation. The ECC implementation in [30] is based upon arithmetic in prime finite fields. Malan *et al.* [31], on the other hand, presented the first ECC implementation over binary fields for sensor nodes. We have used prime finite fields in our implementation, as discussed in Section 3.2.

Liu *et al.* [14] have previously demonstrated the ECDSA signature scheme in resource constrained sensor node platforms. We compare the ECDSA performance to

[1]Sometimes, this problem has been addressed indirectly, i.e., by providing a key agreement protocol. Keys established can further be employed to generate message authentication codes

BLS, demonstrate its use for a shared sensor authentication scenario, and integrate it with end-to-end network and application layer protocols from [2, 3].

One of the signature schemes we have used in our proposal is the BLS, which is based on pairing based cryptography (PBC), a relatively recent addition to ECC. PBC has previously been proposed for use in WSNs [32, 12] but it has a high computation overhead of several seconds. Szczechowiak *et al.* [32] developed the η_T pairing over binary and prime fields. Their implementation uses the Karatsuba's multiplication method and takes 10.96s on an ATmega128L based platform. This performance has further been improved in [12], achieving the η_T computation in 5.5s on the ATmega128L by using López-Dahab field multiplication [33]. The same η_T has been implemented in the work of Ishiguro et al [34] using ternary fields and evaluates pairings in 5.79s. These works are complimentary to our implementation as in our approach, the pairings are not required to be computed on the resource constrained sensors, thus avoiding the high computation overhead of PBC on the sensors.

6. CONCLUSION

We developed a solution, Secure-TWS, for authenticated communication for the scenario of sensor nodes that are shared as a common deployed substrate among multiple applications and users both locally and through the Internet. Our implementation was tested with existing web service layer and IP layer implementations for resource constrained sensor nodes, thus providing a useful system that can be used for deployment scenarios where authentication is necessary.

We also discussed the numerous design choices that were considered in our implementation. We found that a digital signature based approach is the most efficient choice for this type of interaction and then compared different digital signature schemes for realizing it. The ECDSA and BLS schemes were the most desirable ones and were implemented on two popular sensor platforms. This implementation provides one of the first experimental characterization of BLS resource overheads on resource constrained sensor nodes. The performance resource overheads were experimentally measured and the factors involved in the choice among these schemes were discussed. The implementation clearly shows that for this authentication scenario, computation costs largely dominate over communication, as opposed to symmetric key based schemes where communication is the dominant cost. The comparison also showed that ECDSA is faster than BLS on the ARM7 and MSP430 processors, while in prior comparisons on the Pentium-III 1GHz processor, BLS had been found to be faster for signature generation. In addition to providing a practically usable system, the implementation effort has also provided valuable insights into relevant challenges and design choices.

While this implementation has addressed the authentication problem, other security aspects such as privacy may also be relevant in certain shared sensor node scenarios and end-to-end solutions, that are integrated with sensor node network stack implementations, may be addressed in future work.

7. ACKNOWLEDGMENTS

The authors are grateful to Piotr Szczechowiak, Michael Scott, Fredrik Österlind, and Diego Aranha for useful discussions and helpful tips from their implementation experiences.

8. REFERENCES

[1] Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao. SenseWeb: An infrastructure for shared sensing. *IEEE MultiMedia*, 14(4):8–13, 2007.

[2] Nissanka Bodhi Priyantha, Aman Kansal, Michel Goraczko, and Feng Zhao. Tiny web services: Design and implementation of interoperable and evolvable sensor networks. In *6th ACM Conference on Embedded Networked Sensor Systems (Sensys'08)*, Raleigh, NC, USA, November 2008.

[3] Adam Dunkels. Full TCP/IP for 8-bit architectures. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 85–98, New York, NY, USA, 2003. ACM.

[4] Jonathan W. Hui and David E. Culler. IP is dead, long live IP for wireless sensor networks. In *Proceedings of the 6th international Conference on Embedded Networked Sensor Systems (ACM Sensys'08)*, Raleigh, North Carolina, USA, November 2008.

[5] Mark Luk, Adrian Perrig, and Bram Whillock. Seven cardinal properties of sensor network broadcast authentication. In *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 147–156, New York, NY, USA, 2006. ACM.

[6] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[7] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Efficient implementation of pairing-based cryptosystems. *Journal of Cryptology*, 17(4):321–334, 2004.

[8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002. Also in MobiCom'01.

[9] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM conference on Computer and communication security (CCS'03)*, pages 62–72. ACM Press, 2003.

[10] Vipul Gupta, Matthew Millard, Stephen Fung, Yu Zhu, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle: A

standards-based end-to-end security architecture for the embedded internet (best paper). In *PERCOM'05: 3rd IEEE International Conference on Pervasive Computing and Communications*, pages 247–256, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, pages 47–53. Springer-Verlag, 1984.

[12] Leonardo B. Oliveira, Michael Scott, Julio López, and Ricardo Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *5th International Conference on Networked Sensing Systems (INSS'08)*, pages 173–180, Kanazawa/Japan, 2008.

[13] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473, 2003.

[14] An Liu and Peng Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 245–256, Washington, DC, USA, 2008. IEEE Computer Society.

[15] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS'00)*, pages 26–28, Jan 2000.

[16] Paulo Barreto, Benoît Libert, Noel Mccullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *Advances in Cryptology - ASIACRYPT 2005*, pages 515–532, 2005.

[17] Dimitrios Lymberopoulos, Nissanka B. Priyantha, and Feng Zhao. mPlatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes. In *IPSN '07: 6th international conference on Information processing in sensor networks*, pages 128–137, New York, NY, USA, 2007. ACM.

[18] Michael Scott. *MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library*. Shamus Software Ltd, Dublin, Ireland, 2003. Available at http://indigo.ie/ mscott.

[19] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer, 2004.

[20] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[21] Dimitrios Lymberopoulos, Nissanka B. Priyantha, Michel Goraczko, and Feng Zhao. Towards energy efficient design of multi-radio platforms for wireless sensor networks. In *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pages 257–268, Washington, DC, USA, 2008. IEEE Computer Society.

[22] Laurent Eschenauer and Virgil D. Gligor. A key management scheme for distributed sensor networks. In *9th ACM conf. on Computer and communications security (CCS'02)*, pages 41–47, 2002.

[23] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. on Info. and System Security*, 8(1):41–77, 2005. Also in ACM CCS'03.

[24] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM Trans. on Info. and System Security*, 8(2):228–58, 2005. Also in ACM CCS'03.

[25] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *11th IEEE Inter'l Conference on Network Protocols (ICNP'03)*, pages 326–335, Atlanta, Nov 2003.

[26] Donggang Liu, Peng Ning, Sencun Zhu, and Sushil Jajodia. Practical broadcast authentication in sensor networks. In *MOBIQUITOUS '05: 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 118–132, Washington, DC, USA, 2005. IEEE Computer Society.

[27] Qi Dong, Donggang Liu, and Peng Ning. Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks. In *WISE'08: 1st ACM Conference on Wireless Network Security*, pages 2–12, 2008.

[28] Peng Ning, An Liu, and Wenliang Du. Mitigating dos attacks against broadcast authentication in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(1):1–35, 2008.

[29] Kui Ren, Kui Ren, Wenjing Lou, Wenjing Lou, Yanchao Zhang, and Yanchao Zhang. Multi-user broadcast authentication in wireless sensor networks. In *SECON'07 4th Sensor, Mesh and Ad Hoc Communications and Networks*, pages 223–232, 2007.

[30] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*, pages 119–132, 2004.

[31] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, 2004.

[32] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. NanoECC: Testing the limits of elliptic curve cryptography in sensor networks. In *European conference on Wireless Sensor Networks (EWSN'08)*, volume 4913, pages 305–320, Bologne/Italy, 2008.

[33] Julio López and Ricardo Dahab. High-speed software multiplication in GF(2^m). In *Progress in Cryptology - INDOCRYPT'00*, pages 203–212, 2000. Lecture Notes in Computer Science.

[34] T. Ishiguro, M. Shirase, and T. Takagi. Efficient implementation of pairings on sensor nodes. In *Applications of Pairing-Based Cryptography – NIST*, pages 96–106, 2008.