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

Leonardo B. Oliveira[1], Aman Kansal[2], Conrado P. L. Gouvêa[1], Diego F. Aranha[1], Julio López[1], Bodhi Priyantha[2], Michel Goraczko[2] and Feng Zhao[2]

[1]*University of Campinas*
[2]*Microsoft Research*
Email: leob@ic.unicamp.br, kansal@microsoft.com,
{conradoplg,dfaranha,jlopez}@ic.unicamp.br, {bodhip,michelg,zhao}@microsoft.com

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 of four suitable signature schemes – the Elliptic Curve Digital Signature Algorithm (ECDSA), the (elliptic curve) Schnorr signature, and the Boneh-Lynn-Shacham (BLS) and Zhang-Safavi-Naini-Susilo (ZSS) short signature schemes. We implement these schemes on two popular sensor node architectures (based on AVR ATmega128L and MSP430 processors with 802.15.4 radios) and experimentally characterize relevant trade-offs.

Wireless Sensor Networks; Security; Cryptography; Authentication; 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 reduce 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.

Implementing such a direct connection from sensor nodes to multiple clients in practice further entails providing *end-to-end security* for data communication.

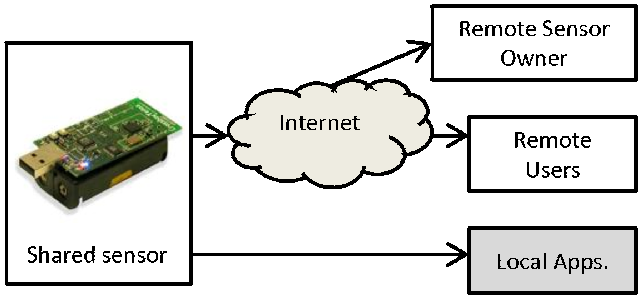

FIGURE 1. Shared sensor scenario for which security solution is implemented.

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 must 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 also 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) [6] and Zhang-Safavi-Naini-Susilo (ZSS) [7] short signature schemes with the Elliptic Curve Digital Signature Algorithm (ECDSA) and (elliptic curve) Schnorr signature in the context of WSN's (note that ours is the first implementation of short signature schemes on sensor platforms).
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 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 specified by NIST in FIPS 186-3. Its security relies on the Discrete Logarithm Problem (DLP) and the best known algorithm to solve it has a subexponential running time. As as result, the parameters for DSA are rather large, making it ill suited for

constrained environments such as sensor nodes.

Elliptic Curve DSA (ECDSA): ECDSA is the elliptic curve analogue of DSA. However, its security relies on the Elliptic Curve DLP which the best known algorithm to solve it runs in fully exponential time. As consequence, one can use smaller parameters with the same security level of DSA. In addition, small key sizes offer potential reduction in processing power, memory, bandwidth, and energy. In TinyECC [14], for example, is shown that the generation of signatures using ECDSA on an MSP430-based sensor platform, at the 80-bit security level, is only 1.6s. ECDSA's signature length, however, is as long as DSA's. For instance, the signature schemes DSA-1024 and ECDSA-160, at 80-bit security level, produce a 320-bit signature.

Schnorr Signature: This signature scheme can be instantiated with the same efficient parameters as ECDSA, producing signatures of the same bit-length. The main difference between both is that Schnorr signatures do not require the computation of modular inverses for generating signatures. It is considered the simplest digital signature scheme to be provably secure in a random oracle model and is covered by U.S. Patent 4,995,082, which has already expired.

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.

Zhang-Safavi-Naini-Susilo (ZSS) Scheme: ZSS [7] is another certificate-based signature scheme relying on pairings. In comparison with BLS, ZSS requires smaller computation overhead under the same random oracle hardness assumption. The ZSS scheme is closely related to a scheme independently proposed by Boneh and Boyen (BB) [16], but proven secure in a slightly weaker complexity assumption.[3]

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 *scalar* or *point multiplication* (denoted using S 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* (denoted as P) is a computable, non-degenerate function that has a special property known as bilinearity. Variants of the Tate pairing [15] are used in our implementation. An *exponentiation* (denoted E in the table) is a modular exponentiation. i.e., a computation of the form $a^b \bmod c$.

Note that in the above schemes point multiplication works with 160-bit parameters while the pairings and exponentiation work with much larger 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. Note that the procedure does not involve any processing at the sensor that is 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.

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

[3] It is worth noting that BB has one variant that does not rely on the *random oracle model* of a hash function at the expense of increasing the signature length. We have not employed this one for efficiency purposes.

Signature Scheme	Computation		Communication (bytes)
	Generation	Verification	
Identity-based[17]	$2S$	$1S + 1P$	40
Certificateless[13]	$1S + 1P$	$4P$	40
DSA	$1E$	$2E$	40
ECDSA	$1S$	$2S$	40
Schnorr	$1S$	$2S$	40
BLS	$1S$	$2P$	20
ZSS	$1S$	$1S + 1P$	20

TABLE 1. Signature schemes's requirements and costs

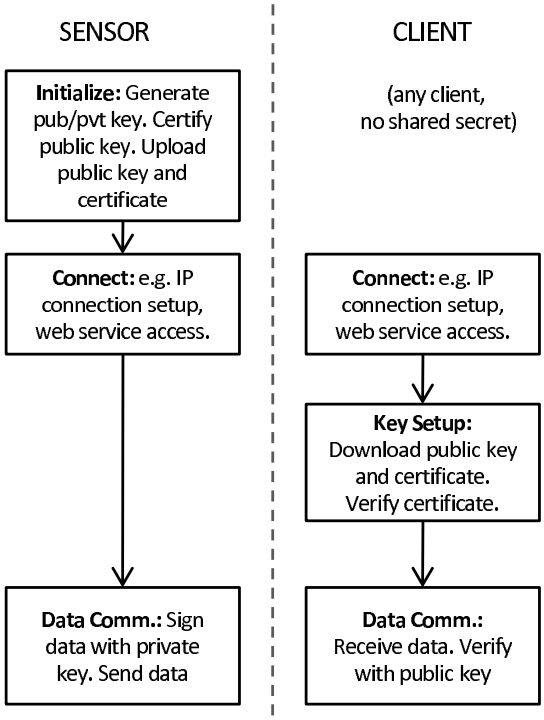

FIGURE 2. Overview of the authentication procedure implemented in Secure-TWS.

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, BLS, or ZSS algorithms. DSA is clearly worse than ECDSA for sensor nodes due to its large parameter sizes and then we compare ZSS, 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 resource-constrained. Considering the computation overhead of signature generation in Table 1, we see that all three schemes involve one point multiplication. ZSS and BLS do have a lower communication overhead. Considering the table, therefore, complexity, i.e, the order of computation alone, one may expect that ZSS and BLS are better. However in actual implementation, it turns out that exact computation costs vary greatly among the three choices due to the specific computation and parameter values involved. The choice between these three options is explored in greater depth in the next section using actual implementations of the 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 digital signature schemes 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 utilize two commonly used low-power processors to implement the Secure-TWS authentication solution: the MSP430-F2418 (16-bit 16MHz core, 8KB

RAM, 116KB ROM) and AVR ATmega128L (8-bit, 7.3728MHz core, 4KB RAM, 128KB ROM). The former is present on the mPlatform [18] and the latter on the MICA mote family of nodes [19]. A block diagram of the software implementation is shown in Figure 3.

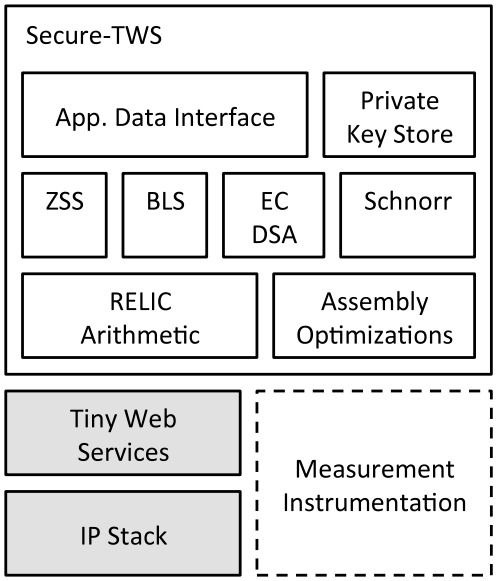

FIGURE 3. Block diagram of the security solution implementation.

As shown, the Secure-TWS implementation has been integrated with the Tiny Web Service [2] stack along with its IP stack [3]. This integration has been tested for the MSP430 processor only.

The key elements of the implementation are the ZSS, BLS, and ECDSA 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 three 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 arithmetic is implemented using the RELIC library from [20]. RELIC 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.

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 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.

The application data interface uses C function calls. It allows using our implementation easily with the underlying Tiny Web Services 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.

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.

The choice of finite fields and elliptic curves are crucial for the overall performance of ECDSA and pairing-based schemes. The most commonly finite fields used in ECC schemes are the prime field $\mathbb{F}_p$ and the binary field $\mathbb{F}_{2^m}$. For some field operations such as squaring, square-root, addition and reduction, binary fields present some computational advantage over prime fields. On the other hand, for some computational platforms, prime fields can take advantage of a native multiplication instruction, for accelerating a field multiplication. Both curves over prime or binary fields offer some optimizations for point multiplication; the special family of binary curves, called Koblitz curves [21], have the property that a point multiplication can be accelerated by exploiting the Frobenius endomorphism $\pi : (x, y) \rightarrow (x^2, y^2)$. For example, the wTNAF [22] method for point multiplication, replaces the computation of point doubling $2P$ by $\pi(P)$, a much faster operation; on the other hand, for certain ordinary elliptic curves defined over prime fields ($p > 3$), that have an efficiently-computable endomorphism, the technique of Gallant, Lambert and Vanstone (GLV)[23] can be used to speed

```
// 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

point multiplication on these curves. We describe this method in Section 3.4.2 with a small modification to make it faster in embedded architectures with expensive division instructions.

For our scenario, we obtain a good performance using standardized curves for the software implementation of ECDSA at the 80-bit security level. Our implementation was based on the binary Koblitz curve `sect163k1` and on the prime curve `secp160k1` defined by [24]. The selection between primary and binary fields for the pairing-based schemes, however, depends on numerous aspects and is not so straightforward. The verification process in BLS and ZSS requires the computation of pairings. This restricts the choice of parameters to pairing-friendly curves:

- When using binary fields, the only pairing friendly elliptic curves known are supersingular, whose embedding degree are at most $k = 4$. This requires a 353-bit binary field in order to provide the required 80-bit level of security, which in turn increases the signature size to 353 bits. We conclude that curves over binary fields are not adequate for implementing BLS or ZSS in this scenario since they provide worst performance (which we have demonstrated experimentally) while requiring larger signatures; this was also pointed out in [6]. Implementations of pairing-based key agreement schemes have suggested using a 271-bit binary field [12], but due to Coppersmith's attack [25], the more conservative choice of 353 bits is a better suggestion for attaining the 80-bit security level.
- When using prime fields, there is a choice between supersingular and ordinary curves. Supersingular prime curves are limited to an embedding degree of $k = 2$ (which is too low for the 80-bit level and would require a 512-bit prime field); therefore, an ordinary prime curve was chosen. There are multiple families of pairing-friendly ordinary prime curves, and for the required security level the MNT family with $k = 6$ seems the most appropriate. However, there are some disadvantages with this family — namely, it is a sparse family, which prevents the use of prime modulus with a special form; and it provides quadratic twists only [26]. An alternative is the BN family, which is specially appropriate for the 128-bit level of security due to its embedding degree $k = 12$ and sextic twists [27]. While it seems ill-suited for the 80-bit level of security, it actually provides a faster implementation in this scenario due to many optimizations tailored for the family [28]. In particular, the use of sparse primes greatly improves its performance, as will be described later. We use the BN curve $y^2 = x^3 + 3$ with parameter $x = 2^{38} + 2^5 + 2^4 + 1$ and the Optimal Ate pairing [29].

3.3. Algorithmic Choices

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

ECDSA and Schnorr: As mentioned before, there is no restriction on curve selection, so the most efficient curves can be used. For this reason, Koblitz binary curves are employed due to the fast point multiplication using the wTNAF method. Furthermore, 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 pre-computation at the expense of some storage overhead. When using the wTNAF method, for example, $2^{w-2} - 1$ points are precomputed offline, where w is the number of bits processed at once. In our ECDSA and Schnorr implementation, we have used $w = 5$ resulting in 7 precomputed points stored in ROM. Each precomputed value requires 44 bytes for storage resulting a storage overhead of 308 bytes. 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 value is dependent of the message being signed and thus assumes a different value each time a signature is generated. Thus, for the random point multiplication, we have used a right-to-left wNAF method, with $w = 4$, requiring the online precomputation of 3 points. The reason for this was that right-to-left approaches are faster when point doubling is cheaper, as it is the case with the employed supersingular binary curve. Over a binary field, it is also possible to use an optimization know as short exponent [30]: by using a private key k with 160 bits instead of the full 353 bits, the security provided stays the same but the multiplication kP is much faster. Point multiplication in the prime pairing-friendly curve was implemented according to the GLV method [23], with the resulting simultaneous multiplication computed as the standard left-to-right interleaving of 4-width NAFs [31].

ZSS: In this scheme, the point multiplication again involves a fixed basis. However, this point multiplication is computed in the same pairing-friendly curves required by BLS, providing a

performance middle-ground between the schemes above. A *comb* approach [32] with 8 precomputed points stored in ROM was used for computing the fixed point multiplication in the binary case, while a combination of the *comb* approach and the GLV method was employed in the prime case. The implementation could use more precomputation to accelerate this operation, but for fairness, the schemes must be evaluated with the same memory footprint.

3.4. Strategies for Point Multiplication

In the next section, we briefly describe our simple modifications to the conventional approaches for computing point multiplication in the pairing-friendly curves employed.

3.4.1. The Right-to-left approach

Left-to-right window-based approaches for computing scalar multiplications kP require some amount of precomputation in the form of small multiples of a point P depending on the window length w. The wNAF approach, for example, requires recoding the integer k such that it can be represented in the digit set $\{3, 5, \ldots, (2^{w-1} - 1)P\}$ and also computing the set of small multiples $\{3P, 5P, \ldots, (2^{w-1} - 1)P\}$. These precomputed points must be represented in affine coordinates to allow faster mixed-coordinate addition and minimize storage overhead [31]. A straightforward way to compute this set of multiples is to double P as $2P$ and successively add $2P$ as $3P = P + 2P, 5P = 3P+2P, \ldots, (2^{w-1}-1)P = (2^{w-1}-3)P+2P$, requiring in total one point doubling and $2^{w-2} - 1$ point additions. In order to minimize the number of field inversions computed in this precomputation stage, curve arithmetic can be done in projective coordinates and the affine coordinates can be recovered through an expensive simultaneous inversion [33].

If the elliptic curve supports a fast inversion-free doubling algorithm in affine coordinates, as the pairing-friendly supersingular binary curves discussed, this can be avoided with a simple alternate strategy: a set of accumulators $\{Q_1, Q_3, Q_5, \ldots, Q_{(2^{w-1}-1)}\}$ are used to accumulate point $2^i P$ in the i-th iteration of the algorithm. The final result can then be recovered as $Q = \sum_{j=1}^{(2^{w-1}-1)} jQ_j$. This strategy will be faster whenever these final $(2^{w-1} - 1)$ point additions are faster than the simultaneous inversion operation and also allow further savings depending on how efficient is doubling a point in affine coordinates.

3.4.2. The GLV method

For the ECDSA, BLS and ZSS signatures, it is possible to exploit the efficient endomorphism $\phi\colon E \rightarrow E$ such that $\phi(P) = \lambda P$ for some λ. Using it, the point multiplication kP can be written as $k = k_0 + k_1\phi(P)$, which can be computed more efficiently using an interleaving method. In order to split k, it is necessary to solve the equation $(k, 0) = \beta_1 v_1 + \beta_2 v_2$ for β_1, β_2 where $v_1 = (v_{10}, v_{11})$ and $v_2 = (v_{20}, v_{21})$ are short vectors precomputed as described in [23]. We can then write

$$d = v_{1,0}v_{21} - v_{11}v_{2,0}$$
$$\beta_1 = k\frac{v_{21}}{d}$$
$$\beta_2 = -k\frac{v_{11}}{d}$$

and then round β_1, β_2 to the nearest integer. However, this approach requires a long division by d, which is costly. In order to avoid it, simply precompute $c_1 = \frac{2^t v_{21}}{d}$ and $c_2 = \frac{2^t v_{11}}{d}$, rounded to the nearest integer, where t is the number of bits of the curve order plus one. Then, for example, one can compute $\beta_1 = \frac{kc_1}{2^t}$, since division by 2^t is cheap. The last bit discarded in the right shift by t decides if β_1 should be rounded up or not. This precomputation technique was hinted at in [31], but not actually described.

While the application of the GLV method for the ECDSA and BLS protocols is straightforward, it is not that clear for the ZSS protocol, since it requires a fixed point multiplication, usually computed with the *comb* method [32]. It can be done, however, by an adaptation of the two-table variant of the *comb* method. In this approach, the multiplier k is written as $k = k_0 + 2^{\frac{|k|}{2}} k_1$ and certain multiples of the fixed point P are precomputed in one table and multiples of $2^{\frac{|k|}{2}} P$ are precomputed in a second table. This approach is faster since kP can then be computed with an interleaving method, halving the number of point duplications, but it requires twice the storage size. However, when the GLV method is available, write $k = k_0 + k_1\phi(P)$ as usual and the elements of the second table can be computed from the first table using the endomorphism as needed. Therefore, we achieve a faster multiplication with the same storage overhead.

3.5. Implementation on the ATmega128 8-bit processor

The MICAz Mote sensor node is equipped with an ATmega128 8-bit processor clocked at 7.3728MHz. The program code is stored in a 128KB EEPROM chip and data memory is provided by a 4KB RAM chip [19]. The ATmega128 processor is a typical RISC architecture with 32 registers, but six of them are special pointer registers. Since at least one register is needed to store temporary results or data loaded from memory, 25 registers are generally available for arithmetic. The instruction set is also reduced, as only 1-bit shift/rotate

instructions are natively supported. Bitwise shifts by arbitrary amounts can then be implemented with combinations of shift/rotate instructions and other instructions. In particular, shifts by 1, 4 and 7 bits can be implemented very efficiently [34]. The processor pipeline has two stages and memory instructions always cause pipeline stalls. Arithmetic instructions with register operands cost 1 cycle and memory instructions or memory addressing cost 2 processing cycles [35]. Minimizing the number of executed memory operations in low-level arithmetic is thus an evident necessity.

For the standardized binary field $\mathbb{F}_{2^{163}}$ used in the ECDSA algorithm, we followed the polynomial-basis implementation described in [34]. For the binary field $\mathbb{F}_{2^{353}}$ used in BLS/ZSS, we have selected the square-root friendly [36] trinomial $f(x) = x^{353} + x^{69} + 1$. This pentanomial has two important features: modular reduction only requires shifts by 1, 4 or 7 bits which are fast in this platform; square-root extraction does not require expensive shifts in processors with word size of 8 or 16 bits. The implementation of this field closely follows [12], with the difference that the multiplier now first features an instance of the Karatsuba algorithm before the direct López-Dahab method can be used. This was required due to the increase in parameter size from 271 to 353 bits, quickly exhausting the number of registers available for performing arithmetic.

For the prime curve used in BLS/ZSS, multiplication and squaring were implemented with a hybrid comba approach as described in [37]. This method was specifically designed to be implemented in embedded architectures with expensive memory acess. The multiple-precision multiplication required for computing Montgomery reduction [38] was specially optimized to take into account the sparse form of the prime modulus, where 6 of the 20 bytes required to store it are zero, allowing the elimination of several word multiplications inside the reduction algorithm.

3.6. Implementation on the MSP430 16-bit processor

The mPlatform provides an MSP430F2418 16-bit processor clocked at 16 MHz. It contains 116KB of program flash memory and 8KB of RAM. The MSP430 family provides 12 general purpose registers and a small instruction set with 27 instructions including 1-bit-only shifts (it is possible to use up to 4-bit shifts, but with the same speed of 4 distinct shifts). In particular, 15-bit shifts can be implemented with the left-shift/rotate-through-carry instructions. Operands may be located in registers or in memory. Since there is no cache, determining the number of cycles taken by each instruction is simple (with a few exceptions): one cycle to fetch the instruction, one cycle to fetch each offset word (if any), one cycle for each memory read and one cycle for each memory write. Small constants (-1, 0, 1, 2, 4 and 8) are generated by using some special registers and do not require offset words when used.

In the binary curve used for ECDSA, the standardized pentanomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ was used. Multiplication in $\mathbb{F}_{2^{163}}$ was implemented with the LD algorithm, while squaring was implemented with a 512-byte lookup table storing the square of all 8-bit polynomials in ROM. Field elements are stored as an array of $n = 11$ 16-bit digits. For the binary curve used in BLS/ZSS, we have chosen the slightly different trinomial $f(x) = x^{353} + x^{95} + 1$ which allows reduction with only 1- and 2-bit shifts. Multiplication was implemented with one level of Karatsuba, as described for the ATmega128.

For the prime curve used in BLS/ZSS, multiplication was implemented with the Comba multiplication algorithm, using the MAC operation of the MSP430 hardware multiplier as described in [39]. Modular reduction was implemented with the Montgomery algorithm (also using the MAC operation), which avoids costly divisions. It was also possible to speed up the reduction by exploiting the format of the prime modulus, since two of its 16-bit digits are zero. Since the Montgomery reduction has the same structure of a multiple precision multiplication (with one of the operands being the prime modulus), it is possible to skip the steps involving multiplication by those zero digits.

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 ZSS, BLS, ECDSA, and Schnorr signature schemes on the MSP and AVR platforms. Specifically, the measurements evaluate i) storage, ii) computation, and iii) communication overheads.

4.1. Storage

Table 2 summarizes storage requirements for the signature schemes. In general, ZSS requires more ROM and global RAM, while BLS requires more RAM from the stack.

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.

The ZSS protocol requires the largest global RAM space due to the precomputation table for the fixed point multiplication. As a tradeoff, this table

Memory	Algorithm	MSP		AVR	
		Prime	Binary	Prime	Binary
RAM (global)	ZSS	1.804	2.002	2.073	1.968
	BLS	1.292	0.818	1.239	1.412
	ECDSA	1.390	1.260	1.361	1.492
	Schnorr	1.390	1.260	1.361	1.492
RAM (stack)	ZSS	1.438	2.596	1.471	2.280
	BLS	2.086	2.830	1.942	2.533
	ECDSA	1.923	1.834	1.783	1.326
	Schnorr	2.010	1.860	1.811	1.356
ROM	ZSS	32.3	28.3	39.2	35.7
	BLS	30.3	25.2	37.6	30.7
	ECDSA	27.4	27.3	36.9	34.5
	Schnorr	26.9	27.4	36.1	33.6

TABLE 2. Secure-TWS's memory overheads (KB).

could be moved to ROM. This also applies to the precomputed constants from the GLV method, which is the reason ZSS requires more global RAM than ECDSA. The BLS protocol uses more stack RAM due to the precomputation table used in random point multiplication.

The ZSS protocol also requires more ROM in comparison to BLS due to the inversion modulo n present in the signature; this routine is fairly large and is not required for BLS. The same effect appears when comparing ECDSA and Schnorr.

4.2. Computation and Communication

Signature schemes based on ECC 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 practice, however, costs of the same order are not necessarily equivalent, especially in resource-constrained sensor platforms. Having implemented schemes on a real sensor platform allows measuring these differences in a more precise manner.

Table 3 shows the computation costs for ZSS, BLS, ECDSA, and Schnorr signature schemes. In our implementation, ZSS is faster than BLS due to the fixed point multiplication using precomputation. ZSS is not faster than ECDSA since the latter i) makes use of a special prime enabling fast reduction in the prime case and ii) uses a smaller finite field in the binary case. Also, concerning ECDSA, its is worth noting that the binary case faster than the prime case – due to the Koblitz curve optimizations –, but BLS and ZSS are faster in the prime field – due to the smaller field required. The binary ZSS is much slower than the binary BLS because it can not use the short exponent optimization. Schnorr is slightly faster than ECDSA because the computation of a modular inversion is not required. Also note that the results for the MSP and for the AVR are not equivalent, since there are differences in clock speed and word size that favors the former – MSP is usually 3 or 4 times faster than AVR.

Table 4, in turn, shows how signature length affects the communication energy cost of transmitting the signature. The cost is higher for binary fields since the signature size depends on the field size, and pairing-based protocols need a larger field in the binary case due to the small embedding degree of the supersingular binary curves [6]. It is worth noting that the term "short signatures" is only applicable to ZSS and BLS when those are implemented under prime fields – in fact, under binary fields, ZSS' and BLS' signature sizes are larger even than ECDSA's.

Finally, note that the energy consumption does not depend solely on the signature length because of existing radio start-up energy costs. Radio costs are discussed in more detail in [40].

4.3. Combined Resource Overhead

Overall costs are shown in Table 5. As can be seen from its data, the computation energy dominates, while the size of the signature is not a significant issue.

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.

Based on these comparisons, it seems that Secure-TWS should be used with Schnorr when only time and overall energy efficiency is required. Whenever there is the additional requirement of interoperability, ECDSA must rather be considered since it has also performed well on the evaluated scenarios and is a standardized algorithm.

Field	Algorithm	MSP		AVR	
		Time (ms)	Energy (mJ)	Time (ms)	Energy (mJ)
prime	ZSS	229	6.8	710	18.0
	BLS	302	9.0	1130	25.7
	ECDSA	134	4.0	680	17.5
	Schnorr	121	3.6	620	14.9
binary	ZSS	703	20.9	2490	59.8
	BLS	391	11.6	1180	28.3
	ECDSA	128	3.8	370	8.9
	Schnorr	114	3.4	330	7.9

TABLE 3. Secure-TWS's computations costs.

Field	Algorithm	MSP		AVR	
		Bit-length	Energy (mJ)	Bit-length	Energy (mJ)
prime	ZSS	161	0.15	161	0.14
	BLS	161	0.15	161	0.14
	ECDSA	320	0.23	320	0.21
	Schnorr	320	0.23	320	0.21
binary	ZSS	354	0.24	354	0.23
	BLS	354	0.24	354	0.23
	ECDSA	320	0.23	320	0.21
	Schnorr	320	0.23	320	0.21

TABLE 4. Communication signature lengths and energy consumption for signature schemes implemented in Secure-TWS

It is worth noting that ZSS and BLS may also be a good choice despite their results in the evaluated scenarios. They indeed have smaller signatures and then are more appropriate for scenarios where communication dominates energy consumption (e.g. for underwater sensor networks [41]). Besides, BLS is able to aggregate signatures [42] and then is also more adequate for environments where receivers are storage-constrained.

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 [4] (e.g. [8, 43, 9, 44, 45, 46] among others);
2. Node to multiple nodes within same sensor network (e.g. [47]);
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.

[4]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

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 [48, 5, 49, 50]). 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.* [51] have combined Merkle trees, Bloom filters, 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 [52, 53, 54, 14]). For instance, Gura *et al.* [52] 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 [52] is based upon arithmetic in prime finite fields. Malan *et al.* [53], on the other hand, presented the first ECC implementation over binary fields for sensor nodes. We have used prime and binary fields in our implementation, as discussed in Section 3.2.

Field	Algorithm	MSP	AVR
		Energy (mJ)	Energy (mJ)
prime	ZSS	6.95	18.14
	BLS	9.15	25.84
	ECDSA	4.23	17.71
	Schnorr	3.83	15.12
binary	ZSS	21.14	60.03
	BLS	11.84	28.53
	ECDSA	4.03	9.11
	Schnorr	3.63	8.15

TABLE 5. Energy overhead of different signature schemes in SecureTWS.

Liu *et al.* [14] have previously demonstrated the ECDSA signature scheme in resource-constrained sensor node platforms. We compare the ECDSA performance to 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 [54, 12] but it has a high computation overhead of several seconds. Szczechowiak *et al.* [54] developed an implementation of pairings 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 pairing computation in 5.5s on the ATmega128L by using López-Dahab field multiplication [55]. The same η_T has been implemented in the work of Ishiguro et al [56] 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 ZSS, BLS, ECDSA, and Schnorr schemes were the most desirable ones and were implemented on two popular sensor platforms. This implementation provides one of the first experimental characterization of ZSS's and BLS's 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, among the three schemes, Schnorr is fastest on the AVR and MSP430 processors. 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. These may be addressed in future work.

7. ACKNOWLEDGMENTS

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

REFERENCES

[1] Kansal, A., Nath, S., Liu, J., and Zhao, F. (2007) SenseWeb: An infrastructure for shared sensing. *IEEE MultiMedia*, **14**, 8–13.

[2] Priyantha, N. B., Kansal, A., Goraczko, M., and Zhao, F. (2008) Tiny Web Services: Design and implementation of interoperable and evolvable sensor networks. *Proceedings of 6th ACM Conference on Embedded Networked Sensor Systems (Sensys'08)*,

Raleigh, NC, November, pp. 253–266.

[3] Dunkels, A. (2003) Full TCP/IP for 8-bit architectures. *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, San Franciso, CA, May, pp. 85–98. ACM.

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

[5] Luk, M., Perrig, A., and Whillock, B. (2006) Seven cardinal properties of sensor network broadcast authentication. *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, New York, NY, USA, October, pp. 147–156. ACM.

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

[7] Zhang, F., Safavi-Naini, R., and Susilo, W. (2004) An efficient signature scheme from bilinear pairings and its applications. In Bao, F., Deng, R. H., and Zhou, J. (eds.), *Public Key Cryptography (PKC 2004)*, LNCS, **2947**, pp. 277–290. Springer.

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

[9] Zhu, S., Setia, S., and Jajodia, S. (2003) LEAP: efficient security mechanisms for large-scale distributed sensor networks. *10th ACM conference on Computer and communication security (CCS'03)*, New York, NY, USA, October, pp. 62–72. ACM Press.

[10] Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., Gura, N., Eberle, H., and Chang Shantz, S. (2005) Sizzle: A standards-based end-to-end security architecture for the embedded internet. *Pervasive Mob. Comput.*, **1**, 425–445. Also appeared in PERCOM'05.

[11] Shamir, A. (1984) Identity-based cryptosystems and signature schemes. In Blakley, G. and Chaum, D. (eds.), *CRYPTO'84*, Santa Barbara, California, August, pp. 47–53. Springer-Verlag.

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

[13] Al-Riyami, S. S. and Paterson, K. G. (2003) Certificateless public key cryptography. In Laih, C.-S. (ed.), *Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security ASIACRYPT'03*, Taipei,Taiwan, November, pp. 452–473. Springer.

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

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

[16] Boneh, D. and Boyen, X. (2008) Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, **21**, 149–177.

[17] Barreto, P., Libert, B., Mccullagh, N., and Quisquater, J.-J. (2005) Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In Roy, B. K. (ed.), *Advances in Cryptology - ASIACRYPT 2005*, Chennai, December, pp. 515–532. Springer-Verlag.

[18] Lymberopoulos, D., Priyantha, N. B., and Zhao, F. (2007) mPlatform: a reconfigurable architecture and efficient data sharing mechanism for modular sensor nodes. *IPSN '07: 6th international conference on Information processing in sensor networks*, New York, NY, USA, pp. 128–137. ACM.

[19] Hill, J. L. and Culler, D. E. (2002) Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, **22**, 12–24.

[20] Aranha, D. F. and Gouvêa, C. P. L. (2010). RELIC Cryptographic Toolkit. Available at http://code.google.com/p/relic-toolkit.

[21] Koblitz, N. (1991) CM-Curves with Good Cryptographic Properties. *11th Annual International Cryptology Conference (CRYPTO 1991)*, Santa Barbara, California, August, LNCS, **576**, pp. 279–287. Springer.

[22] Solinas, J. A. (2000) Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography*, **19**, 195–249.

[23] Gallant, R., Lambert, R., and Vanstone, S. (2001) Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. In Kilian, J. (ed.), *21st Annual International Cryptology Conference (CRYPTO 2001)*, Santa Barbara, California, August, LNCS, **2139**, pp. 190–200. Springer.

[24] Standards for Efficient Cryptography Group (2000). Sec 2: Recommended elliptic curve domain parameters. SECG2.

[25] Coppersmith, D. (1984) Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, **30**, 587–593.

[26] Miyaji, A., Nakabayashi, M., and Takano, S. (2001) New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *Transactions on Comm./Elec./Information and Systems*, **E84A**, 1234–1243.

[27] Barreto, P. S. L. M. and Naehrig, M. (2005) Pairing-Friendly Elliptic Curves of Prime Order. In Preneel, B. and Tavares, S. E. (eds.), *12th International Workshop on Selected Areas in Cryptography (SAC 2005)*, Kingston, Canada, August, LNCS, **3897**, pp. 319–331. Springer.

[28] Pereira, G., Simplício, M. A., Jr., Naehrig, M., and Barreto, P. S. L. M. (2011) A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, **84**, 1319–1326.

[29] Vercauteren, F. (2010) Optimal pairings. *IEEE Transactions on Information Theory*, **56**, 455–461.

[30] Koblitz, N. and Menezes, A. (2005) Pairing-based cryptography at high security levels. In Smart, N. P. (ed.), *IMA Int. Conf.*, Cirencester, UK, December, LNCS, **3796**, pp. 13–36. Springer.

[31] Hankerson, D., Menezes, A., and Vanstone, S. (2004) *Guide to elliptic curve cryptography*. Springer.

[32] Lim, C. H. and Lee, P. J. (1994) More Flexible Exponentiation with Precomputation. In Desmedt, Y. (ed.), *14th Annual International Cryptology Conference (CRYPTO 1994)*, London, UK, August, LNCS, **839**, pp. 95–107. Springer.

[33] Montgomery, P. (1987) Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, **48**, 243–264.

[34] Aranha, D. F., Oliveira, L. B., López, J., and Dahab, R. (2010) Efficient implementation of elliptic curve cryptography in wireless sensors. *Advances in Mathematics of Communications*, **4**, 169–187.

[35] Atmel (2004) *8 bit AVR Microcontroller ATmega128(L) manual*, 2467m-avr-11/04 edition.

[36] Avanzi, R. M. (2007) Another look at square roots (and other less common operations) in fields of even characteristic. In Adams, C. M., Miri, A., and Wiener, M. J. (eds.), *Selected Areas in Cryptography*, Berlin, Heidelberg, March, pp. 138–154. Springer-Verlag.

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

[38] Montgomery, P. L. (1985) Modular multiplication without trial division. *Mathematics of Computation*, **44**, 519–521.

[39] Gouvêa, C. P. L. and López, J. (2009) Software Implementation of Pairing-Based Cryptography on Sensor Networks Using the MSP430 Microcontroller. In Roy, B. K. and Sendrier, N. (eds.), *INDOCRYPT*, New Delphi, India, December, LNCS, **5922**, pp. 248–262. Springer.

[40] Lymberopoulos, D., Priyantha, N. B., Goraczko, M., and Zhao, F. (2008) Towards energy efficient design of multi-radio platforms for wireless sensor networks. *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, Washington, DC, USA, April, pp. 257–268. IEEE Computer Society.

[41] Cui, J.-H., Kong, J., Gerla, M., and Zhou, S. (2006) Challenges: Building scalable mobile underwater wireless sensor networks for aquatic applications. *IEEE Network, Special Issue on Wireless Sensor Networking*, **20**, 12–18.

[42] Boneh, D. and Gentry, C. (2003) Aggregate and verifiably encrypted signatures from bilinear maps. *Proceedings of Eurocrypt 2003*, Warsaw, Poland, May, pp. 416–432. Springer-Verlag.

[43] Eschenauer, L. and Gligor, V. D. (2002) A key management scheme for distributed sensor networks. *9th ACM conf. on Computer and communications security (CCS'02)*, Washington, DC, November, pp. 41–47. ACM.

[44] Oliveira, L. B., Wong, H. C., Bern, M., Dahab, R., and Loureiro, A. A. F. (2006) SecLEACH – a random key distribution solution for securing clustered sensor networks. *5th IEEE International Symposium on Network Computing and Applications (NCA'06)*, July. p. 145-154.

[45] Liu, D., Ning, P., and Li, R. (2005) Establishing pairwise keys in distributed sensor networks. *ACM Trans. on Info. and System Security*, **8**, 41–77. Also in ACM CCS'03.

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

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

[48] Liu, D., Ning, P., Zhu, S., and Jajodia, S. (2005) Practical broadcast authentication in sensor networks. *MOBIQUITOUS '05: 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Washington, DC, July, pp. 118–132. IEEE Computer Society.

[49] Dong, Q., Liu, D., and Ning, P. (2008) Preauthentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks. *WiSec'08: 1st ACM Conference on Wireless Network Security*, New York, NY, USA, March, pp. 2–12. ACM.

[50] Ning, P., Liu, A., and Du, W. (2008) Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks*, **4**, 1–35.

[51] Ren, K., Lou, W., and Zhang, Y. (2007) Multi-user broadcast authentication in wireless sensor networks. *SECON'07 4th Sensor, Mesh and Ad Hoc Communications and Networks*, Kiev, Ukraine, July, pp. 223–232. IEEE.

[52] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. (2004) Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In Marc Joye, J.-J. Q. (ed.), *Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*, Cambridge,MA, August, pp. 119–132. Springer.

[53] Malan, D. J., Welsh, M., and Smith, M. D. (2008) Implementing public-key infrastructure for sensor networks. *ACM Transactions on Sensor Networks*, **4**, 22:1–22:23. Also in SECON'04.

[54] Szczechowiak, P., Oliveira, L. B., Scott, M., Collier, M., and Dahab, R. (2008) NanoECC: Testing the limits of elliptic curve cryptography in sensor networks. *European conference on Wireless Sensor Networks (EWSN'08)*, Bologne/Italy, February, pp. 305–320. Springer-Verlag.

[55] López, J. and Dahab, R. (2000) High-speed software multiplication in GF(2^m). In Bimal K. Roy, E. O. (ed.), *Progress in Cryptology - INDOCRYPT'00*, London, UK, December, pp. 203–212. Springer-Verlag. LNCS.

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