

# Energy Analysis of Public-Key Cryptography for Wireless Sensor Networks

Arvinderpal S. Wander<sup>†</sup>, Nils Gura<sup>‡</sup>, Hans Eberle<sup>‡</sup>, Vipul Gupta<sup>‡</sup>, Sheueling Chang Shantz<sup>‡</sup>  
<sup>†</sup>University of California, Santa Cruz <sup>‡</sup>Sun Microsystems Laboratories  
awander@soe.ucsc.edu {Nils.Gura, Hans.Eberle, Vipul.Gupta, Sheueling.Chang}@sun.com

## Abstract

*In this paper, we quantify the energy cost of authentication and key exchange based on public-key cryptography on an 8-bit microcontroller platform. We present a comparison of two public-key algorithms, RSA and Elliptic Curve Cryptography (ECC), and consider mutual authentication and key exchange between two untrusted parties such as two nodes in a wireless sensor network. Our measurements on an Atmel ATmega128L low-power microcontroller indicate that public-key cryptography is very viable on 8-bit energy-constrained platforms even if implemented in software. We found ECC to have a significant advantage over RSA as it reduces computation time and also the amount of data transmitted and stored.*

## 1. Introduction

The availability of inexpensive radio transceivers has enabled new types of applications and created new research challenges. Applications range from wireless sensor networks (WSNs) that can be used for health monitoring, industrial control, and building automation to smart cards that enable seamless user authentication and signing of digital documents. Depending on the application, these devices may not only exchange information locally with peers, but also globally with entities on the Internet. They are expected to be deployable in large quantities and in environments where they may be exposed to tampering, eavesdropping, and attempts to modify transmitted data and insert unauthorized messages into the network. To counter such threats, flexible and effective mechanisms for secure communication are essential.

For many applications and devices under consideration, energy is a critical and limited resource. To assess the energy demands of public-key cryptography, we quantify the energy cost<sup>1</sup> of optimized software implementations of public-key algorithms. Since the true benefits and accurate analysis of any particular algorithm is closely tied to how it is used within a security protocol, we also consider protocols that provide mutual authentication. In addition, we consider the impact of public-key cryptography on battery life and compare

public-key cryptography to other factors influencing energy consumption, such as idle listening, data reception and transmission, symmetric cryptography, etc. Our analysis can be generalized to estimate the cost of public-key cryptography in other security protocols and applications with varying characteristics.

Security in WSNs has lately received increased attention. However, in almost all cases, non-public-key-based key distribution and authentication schemes have been presented [1], with the underlying assumption that public-key cryptography exceeds the computational capabilities of devices suitable for WSNs. Unfortunately, none of these schemes are capable of providing the flexibility offered by public-key-based solutions. Earlier work by Gura et al. [2] showed public-key cryptography to be computationally feasible on 8-bit devices. However, this paper does not analyze energy cost or the application of public-key operations in security protocols. Energy numbers for bulk encryption and hash functions have been presented in [3,4,5].

## 2. Public-key Authentication

Secure communication can be achieved by employing strong cryptography to ensure confidentiality (non-disclosure of secret information), integrity (prevention of data alteration), authentication (proof of identity), and non-repudiation (unique, non-contestable message origin). These goals can be accomplished through a combination of symmetric-key algorithms (e.g. AES, DES, RC4), public-key algorithms (e.g. RSA, ECC), and cryptographic hash functions (e.g. MD5, SHA). RSA [6] is by far the most widely used public-key algorithm on the Internet today. However, ECC [7] offers equivalent security at much smaller key sizes. ECC is especially attractive for constrained wireless devices because the smaller keys result in memory, bandwidth and computational savings. A well-known key exchange algorithm for ECC is the Elliptic Curve Diffie-Hellman (ECDH) algorithm and ECC-based signatures can be generated and verified with the Elliptic Curve Digital Signature Algorithm (ECDSA) [7]. Digital certificates are commonly used to verify the identity of a party sending a message, and to provide the recipient with the means to encode an encrypted reply. RSA with 1024-bit keys (RSA-1024) provides the currently accepted security level, and is equivalent in strength to ECC with 160-

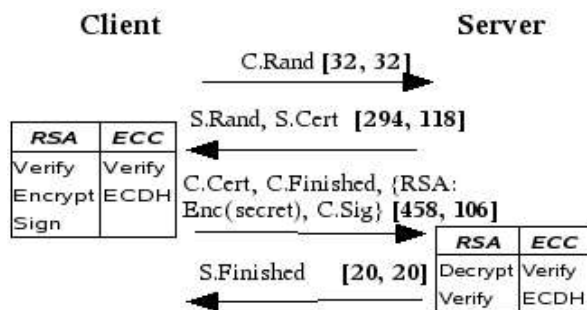
<sup>1</sup> In this paper, cost always refers to the energy cost, unless stated otherwise.

bit keys (ECC-160). To protect data beyond the year 2010, RSA Security recommends RSA-2048 (equivalent to ECC-224) as the new minimum key size [8].

## 2.1. Mutual Authentication and Key Exchange: WSNs

Two parties operating in an open environment that potentially includes adversaries seeking to impersonate or eavesdrop on communication, will want a mechanism to separate friends from foes and keep the content of their conversations secret. Mutual authentication and key exchange using public-key cryptography is a well-known mechanism that enables exactly that.

The mutual authentication and key exchange protocol presented below is a simplified version of the SSL handshake [9]. Using SSL terminology, we refer to the party initiating the communication as the client and to the responding party as the server. Figure 1 shows the exchange of data between the client and the server during an SSL handshake process for both RSA-based and ECC-based algorithms. To conserve energy, we reduced the amount of data exchanged in a typical Internet-based SSL handshake. We assume the WSN administered by a single organization such that many of the parameters can be fixed instead of negotiated (e.g. cipher suite and protocol version) and that abbreviated X.509 certificates containing only a unique device ID, a public key and a signature can be used. While a traditional X.509 RSA-1024 certificate is on the order of 700 bytes long, the simplified certificate is only 262 bytes long and an ECC-160 certificate can be reduced from approximately 530 bytes to 86 bytes.



**Figure 1: Simplified RSA and ECC-based SSL handshake with mutual authentication. Payload message sizes in bytes [RSA, ECC].**

## 3. Energy Analysis

We conducted our experiments on the Berkeley/Crossbow motes platform, specifically on the Mica2dots [10], which are a popular platform for WSN research. The major energy consumers on these sensor devices are the Atmel ATmega128L 8-bit microcontroller and the Chipcon CC1000 low-power wireless trans-

ceiver. The Atmega128L runs at a clock frequency of 4 MHz. Our program code was written in a combination of nesC, C and assembly language, where optimized assembly code was used to implement RSA and standardized ECC over integer fields GF(p), as presented in [2].

We found that while the ATmega128L is designed for low power applications, it does not employ techniques such as gated clocks to shut down processor modules that are temporarily unused. Therefore, we approximated the energy consumption for individual cryptographic algorithms and other activities such as data transmission by measuring the current drawn from the power supply. An earlier, more accurate approach using an oscilloscope and a sense resistor showed the error to be less than 5%.

Table 1 presents characteristic data for the Mica2dot platform that we measured and calculated. It is interesting to note that the power required to transmit 1 bit is equivalent to roughly 2090 clock cycles of execution on the microcontroller alone. This confirmed our assumption that the energy cost of computation is small compared to data transmission. The cost of receiving one byte (28.6 $\mu$ J) is roughly half of that required to transmit a byte (59.2  $\mu$ J). During transmit and receive, the microcontroller is powered on along with the wireless transceiver. We used a packet size of 41 bytes, 32 for the payload and 9 bytes for the header. The header, ensuing a 8-byte preamble, consists of source, destination, length, packet ID, CRC, and a control byte. Receiving one 41-byte packet (including 8-byte preamble) costs  $49 \times 28.6 \mu\text{J} = 1.40 \text{mJ}$  and transmitting one 41-byte packet costs  $49 \times 59.2 \mu\text{J} = 2.90 \text{mJ}$ .

Field	Value
Effective data rate	12.4 kbps
Energy to transmit	59.2 $\mu$ J/byte
Energy to receive	28.6 $\mu$ J/byte
ATmega128L active mode	13.8 mW
ATmega128L power down mode	.0075 mW
ATmega128L MIPS/Watt	289 MIPS/W

**Table 1: Characteristic data for the Mica2dot sensor platform at 3V, 4MHz, 915 MHz transceiver, transmit power 3 mW(5 dBm).**

### 3.1. Energy Cost of Primitive RSA and ECC Operations

Table 2 compares the energy consumed by RSA and ECC for generating and verifying signatures and the energy cost of key exchanges not including authentication and certificate verification. While the cost of an RSA verify is small, it is overshadowed by the more expensive sign operation, both of which are required for authentication. In comparison, ECDSA<sup>2</sup> signatures are

<sup>2</sup> We estimated the energy numbers for ECDSA based on our measurements of ECC point multiplication and integer inversion. In addition, ECDSA requires hashing, integer

significantly cheaper than RSA signatures and ECDSA verifications are within reasonable range of RSA verification. Note that when transitioning from RSA-1024 to RSA-2048 the energy cost of signing increases by a factor of more than seven, while ECDSA-224 signing is less than three times as expensive as ECDSA-160 signing. To put these numbers into perspective, one RSA-1024 sign operation is equivalent to transmitting 5,132 bytes, compared to 385 bytes for an ECDSA-160 sign operation. The RSA-based key exchange protocol relies on party A to encrypt a randomly generated secret key with party B's public key, and party B decrypting the key using its private key. With ECC, both parties perform a single ECDH operation to derive the secret key.

Algorithm	Signature		Key Exchange	
	Sign	Verify	Client	Server
RSA-1024	304	11.9	15.4	304
ECDSA-160	22.82	45.09 <sup>3</sup>	22.3	22.3
RSA-2048	2302.7	53.7	57.2	2302.7
ECDSA-224	61.54	121.98 <sup>3</sup>	60.4	60.4

**Table 2: Energy cost of digital signature and key exchange computations [mJ].**

We do not present the cost of key generation. However, we note that for ECC, key generation only involves generating a random number, which becomes the user's private key, and executing an ECDH operation to compute the corresponding public key. RSA key generation is much more time consuming as it requires the generation of large prime numbers. For details, we refer the reader to [11].

### 3.2. Energy Cost of Symmetric-Key and Hash Algorithms

For our analysis we chose to focus on AES with 128-bit keys for data encryption/decryption and SHA-1 for hashing. We used an assembly-optimized AES implementation [12] and a C-implementation of SHA-1 [13]. Table 3 contains data for both implementations. For a more comprehensive discussion of energy consumption of symmetric ciphers and hash algorithms see [3, 5].

Algorithm	Energy
SHA-1	5.9 $\mu$ J/byte
AES-128 Enc/Dec	1.62/2.49 $\mu$ J/byte

**Table 3: Energy numbers for AES and SHA-1. AES numbers include key-setup. The numbers were averaged over inputs ranging from 64 to 1024 bytes.**

addition and multiplication operations, all of which we expect to consume only a minimal amount of energy.

3 While our ECDSA numbers assume two point multiplications, known optimizations can be made at the cost of more memory.

### 3.3. Authentication Scenarios

Next, we consider the mutual authentication and key exchange handshake for nodes operating in a WSN. In order to realistically quantify the impact of public-key cryptography on battery life, we first start by determining the factors that influence energy consumption and then present our analysis.

For applications that do not require secure communication, three factors dominate the overall energy consumed; idle listening, application-specific operations and communication. Secure communication through public-key-based authentication and key exchange introduces two new factors: bulk data encryption/decryption including hashing and public-key operations including communication.

Considerable research has gone into designing low-power media access control (MAC) protocols (e.g. [14]), specifically towards duty cycle mechanisms seeking to eliminate idle listening. Duty cycles (i.e. receive time / sleep time ratios) as low as .1% are often used [15], while still being flexible enough to efficiently transfer different workloads and adapt to changing network conditions.

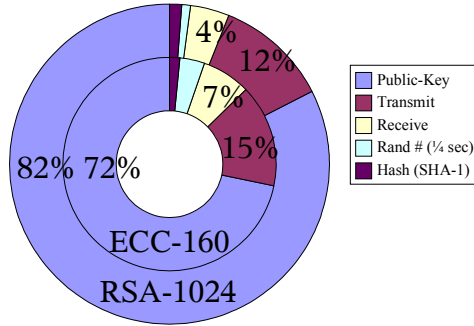
We analyze the energy usage of the simplified SSL handshake based on RSA-1024 and ECC-160. The amount of combined energy spent by client and server is determined by public-key computation, transmitting and receiving handshake messages, hash computation, and random number generation<sup>4</sup>. The first three factors depend on the public-key algorithm used. Table 4 presents the energy consumed by the entire handshake for RSA-1024 and ECC-160. Figure 2 shows the percentage of energy spent on each major part of the handshake process. For both RSA-1024 and ECC-160, the public-key computation dominates, consuming 82% and 72% of the energy, respectively. Communication costs are second, while random number generation and hashing costs are negligible.

The computational benefits of ECC-160 over RSA-1024 are apparent, where the RSA-1024 computations consume 4.9 times the energy of ECC-160 computations. Compared to ECC-160, an RSA-1024 handshake also consumes 2.7 times the energy in transmitting and receiving data. Communication costs for RSA-1024 are higher because of the longer key sizes, thus making the certificates larger as well. With RSA-1024, the entire handshake requires the client to transmit 490 bytes of payload and the server to transmit 314 bytes of payload. With ECC-160, both parties transmit the same amount of payload data, 138 bytes.

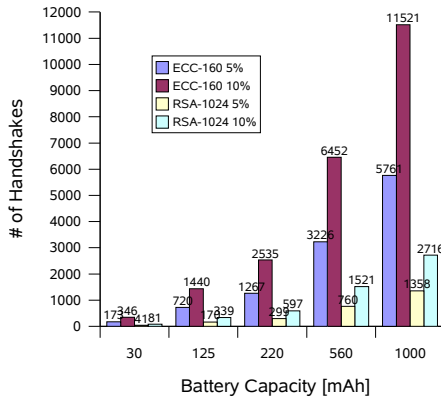
4 We did not implement a random number generator. Based on commonly known algorithms we assume that a number that meets the randomness criteria can be generated in a quarter of a second.

Algorithm	Client	Server
RSA-1024	397.7	390.3
ECC-160	93.7	93.9

**Table 4: Energy consumption of handshake protocol based on RSA-1024 and ECC-160.**



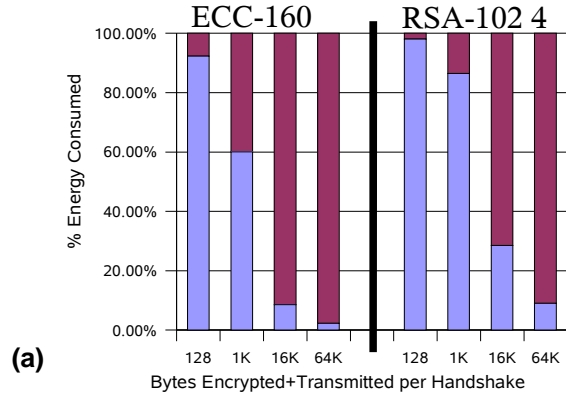
**Figure 2: Decomposition of energy for mutual authentication and key exchange on the client side.**



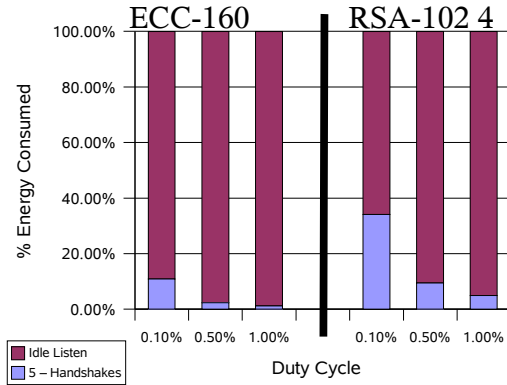
**Figure 3: Number of handshakes when only a 5% or 10% fraction of the battery capacity is available.**

Figure 3 shows the number of handshakes possible as we assume that only a certain fraction, 5% or 10%, of the overall battery capacity is available for handshakes. The capacities reflect those of common coin-based watch and general purpose batteries [16]. From the figure we can see that even with 5% of the energy available from a miniature 30 mAh battery, a node can perform 173 ECC-160 and 41 RSA-1024 handshakes. Next, we examine the cost of the handshake in a likely usage scenario. Consider a newly deployed WSN consisting of an arbitrary number of nodes. Each node has been assigned a public and private key pair along with a digital certificate issued by a trusted certificate authority (CA) and the CA's public key. After deployment, nodes authenticate themselves to their immediate neighbors

and exchange secret keys for data encryption/decryption, for example, using the simplified SSL handshake described earlier.



**(a)**



**(b)**

**Figure 4: (a) Comparison of energy for one handshake and subsequent bulk data encryption and transmission. (b) Comparison of energy spent on idle listening and on five handshakes over a 24 hour period.**

After completing the handshakes, nodes use the agreed-upon secret keys for encrypting/decrypting all or parts of the application data. The secret keys may be used for the lifetime of the nodes or while the nodes remain in communication range. Figure 4(a) plots the percentage of energy consumed by a single handshake versus the number of bytes that have been transmitted after being encrypted by the secret key. The initial cost of the ECC-160 handshake falls below 10% after transmitting 16kB and is almost negligible (~2%) after 64kB have been transmitted. As expected, the relative cost of an RSA-1024 handshake takes longer to fall. While the number of bytes encrypted and transmitted per handshake will vary based on the application, nodes near the base station will generally experience more traffic, thus the cost of handshakes at these nodes will likely be small. As was mentioned earlier, the fraction of energy spent on listening over an idle channel can be signifi-

cant. Ignoring other factors such as application-specific computation and communication, Figure 4(b) considers three nodes with varying duty cycles of .1%, .5% and 1%, where each node performs five handshakes over a one day period. In all three cases, the five ECC-160 handshakes represent less than 11% of the energy consumed per day.

The above results indicate that for applications where handshakes are relatively infrequent the cost of the handshakes is negligible. The fraction of energy spent on idle listening, application-specific computation and communication will likely dominate over the initial cost of the handshake. When considering which public-key algorithm to use, ECC is a better fit for this class of devices. In addition to consuming significantly less energy, the execution time and memory requirements for ECC are also much lower compared to RSA. For example, an ECC-160 point multiplication takes just 1.61 seconds and 282 bytes of data memory, while an RSA-1024 private-key modular exponentiation takes nearly 22 seconds and 930 bytes of data memory [2].

#### 4. Conclusions

Contrary to widely held beliefs, our results indicate that authentication and key exchange protocols using optimized software implementations of public-key cryptography are very viable on small wireless devices. Depending on the frequency of public-key computations, its relative energy cost may even be negligible. Furthermore, our analysis suggests that the use of ECC over RSA can lead to significant energy savings. In addition to the computational benefits of ECC, its smaller keys and certificates lead to significant savings in public-key communication costs.

With a given amount of energy, we were able to perform 4.2 times the number of key exchange operations (including mutual authentication) with ECC-160 compared to RSA-1024. While such absolute numbers are platform-specific, we expect the computational cost to fall faster than the cost to transmit and receive. For example, ultra-low-power microcontrollers such as the 16-bit Texas Instruments MSP430 [17] can execute the same number of instructions at less than half the power required by the 8-bit ATmega128L. A similar analysis conducted on such platforms is likely to show communication costs representing a larger fraction of the overall energy spent on authentication and key exchange protocols. The benefits of transmitting smaller ECC keys and certificates will in turn be more significant.

In addition to flexible key exchange and peer authentication, public-key cryptography can be the enabling technology for numerous other WSN applications, including securely connecting pervasive devices to the Internet and distributing signed software patches.

We thank Matt Millard for setting up the Mica2 motes and Arun Patel for providing AES and SHA-1

numbers.

An extended version of this paper can be found at <http://research.sun.com/projects/crypto>

#### References

- [1] A. Perrig, J. Stankovic and D. Wagner, "Security in Wireless Sensor Networks", Communications of the ACM, Vol. 47 No. 6, June 2004.
- [2] N. Gura, A. Patel, A. Wander, H. Eberle, S. Chang Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs", CHES, August 2004.
- [3] D. W. Carman, P. S. Kruus, and B. J. Matt, "Constraints and Approaches for Distributed Sensor Security", Network Associates Labs Tech. Rep. 2000.
- [4] L. Yuan and G. Qu, "Design Space Exploration for Energy-Efficient Secure Sensor Network", ASAP 2002.
- [5] N. Potlapally, S. Ravi, A. Raghunathan and N. K. Jha, "Analyzing the Energy Consumption of Security Protocols", ISLPED 2003.
- [6] C. K. Koc, "High-speed RSA implementation", Tech. Rep. TR 201, RSA Laboratories, November 1994.
- [7] D. Hankerson, A. Menezes, S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag New York, Inc. 2004. ISBN 0-387-95273-X.
- [8] B. Kaliski, "TWIRL and RSA Key Size", RSA Laboratories Technical Note, May 2003.
- [9] A. Freier, P. Karlton and P. Kocher, "The SSL Protocol Version 3.0", <http://home.netscape.com/eng/ssl3/>
- [10] Crossbow Technology Inc., *Processor/Radio Modules*, <http://www.xbow.com/>
- [11] A. Juels and J. Guajardo, "RSA Key Generation with Verifiable Randomness", RSA Laboratories, <http://www.rsasecurity.com/rsalabs/node.asp?id=2041>
- [12] C. Röpke, W. Urowski and K. Tellman, "AES assembly implementation for the AVR instruction set", [http://www.christianroepke.de/praktikum\\_b.html](http://www.christianroepke.de/praktikum_b.html)
- [13] D. Eastlake, P. Jones, "US Secure Hash Algorithm 1 (SHA1)", IETF Request for Comments 3174, 2001.
- [14] J. Polastre, J. Hill, D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", SenSys, 2004.
- [15] M. Hamilton, M. Allen, D. Estrin, J. Rottenberry, P. Rundel, M. Srivastava, and S. Soatto. "Extensible Sensing System: An advanced Network Design for Microclimate Sensing", <http://www.cens.ucla.edu>
- [16] Matsushita Electric Industrial Co., Ltd. "Manganese dioxide lithium batteries (CR series)"
- [17] Texas Instruments Inc., "MSP430 Family of Ultra-low-power 16-bit RISC Processors", <http://www.ti.com>