State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks *

Gunnar Gaubatz, Jens-Peter Kaps, Erdinç Öztürk, Berk Sunar
Cryptography & Information Security Lab, Worcester Polytechnic Institute, U.S.A.
E-mail: {gaubatz,kaps,erdinc,sunar}@wpi.edu

Abstract

Security in wireless sensor networks is currently provided exclusively through symmetric key cryptography. In this paper we show that special purpose ultra-low power hardware implementations of public key algorithms can be used on sensor nodes. The reduced protocol overhead due to public key cryptography (PKC) translates into less packet transmissions and hence, power savings. We provide an in-depth comparison of three popular public key implementations and describe how four fundamental security services benefit from PKC.

1. Introduction

Security in *Wireless Sensor Networks* (WSN) has seen a recent surge in interest. Most publications, however, seem to preclude that public key cryptography (PKC) is not feasible on severely resource constrained sensor nodes, and therefore revert to emulation of asymmetry using symmetric key techniques [9]. Most, if not all, implement cryptographic primitives in software on general purpose micro-controllers. While intuition might support this notion of infeasibility, we are not aware of any studies that have actually analyzed the cost of PKC in sensor networks, apart from [1].

In this paper we show that PKC tremendously simplifies the implementation of many typical security services and additionally reduces transmission power due to less protocol overhead. Moreover, the capture of a single node would not compromise the entire network, since no globally shared secrets are stored on it. Our approach to overcome the difficulty in implementing PKC in sensor nodes is based on providing a custom-designed low-power co-processor that can be embedded in the node and that handles all of the compute-intensive tasks.

* This material is based upon work supported by the National Science Foundation under Grants No. ANI-0133297 (NSF CAREER Award) and No. ANI-0112889.

In the following section we identify four fundamental security services that would particularly benefit from PKC. In Section 2 we select three low-complexity PKC schemes, for which we have developed three basic encryption architectures in TSMC 0.13μ CMOS standard cell technology. Based on the analysis of these architectures in conjunction with the full algorithm descriptions we estimate the overall power and bandwidth requirements of encryption and signature primitives in Section 3. The analysis of the results in Section 4 with respect to the previously mentioned security services can serve protocol designers as a guideline for incorporating public key-based services into their WSN protocols.

2. Security Services

In this section we state our assumptions regarding the structure of the WSN and define an exemplary subset of four security services that would particularly benefit from the use of PKC.

Sensor networks typically consist of a number of tiny nodes communicating with a base station [9]. The base station collects the data from the sensors and communicates with the outside world. The sensor nodes only have limited power and can therefore communicate directly only with nodes in close proximity. They establish a routing tree with the base station at its root. The base station is assumed to have sufficient power for all computations and communications with the nodes and the outside world.

Broadcast Authentication In this scenario, a base station would, for example, broadcast a set of commands, to all sensor nodes at once. Each sensor node would need to verify that this message originated from the trusted base station and not from an adversary. This scenario is a typical application for PKC. All nodes would need to have the base station's public key embedded. Data recovered from captured nodes would not be helpful to the adversary in forging messages. Previously published schemes either require large amounts of data to be sent or a complicated symmetric key release scheme [9].

Data Encryption Data encryption using PKC is much more expensive than using secret key cryptography. However, in certain cases where no secret key is established, it can be useful. One case is node-to-node key distribution described below. Another scenario could be, that a sensor node has to send some data all the way to the base station. The node just needs the base station's public key for this operation.

Node-to-Node Key Distribution Another typical PKC application is key distribution and key agreement. Key agreement refers to a protocol where two parties jointly establish a key, whereas key distribution is defined as a protocol where one party securely transmits a key to another party. Here we are assuming that each node knows the public key of its neighbors. The private key could be distributed during the routing setup phase or by querying the base station. If two nodes want to establish a session key a node simply encrypts it using its neighbors public key and sends it. Unlike other schemes, the base station does not need to get involved, which saves transmission power.

Addition of new Nodes New legitimated nodes might need to be added to a WSN at any point in time. These nodes must be included into the security scheme of the WSN. Again, PKC offers an elegant solution. Each sensor node has its own public/private key pair and additionally the base stations public key. The public keys of the new nodes can be sent via the outside communications link to the base station. Now the base station can trust the new nodes and vice versa. The base station can encrypt a secret session key with the node's public key and send it, or the node can announce its arrival in the network by sending a signed message to the base station. There is no need for additional bootstrapping information.

3. Implementation

Several public key schemes can be used to provide the security services described above. We take a closer look at Rabin's Scheme, NtruEncrypt and Elliptic Curve Cryptosystems (ECC) as the most promising candidates for low-power implementations. Our architectures for the encryption function of Rabin's Scheme, NtruEncrypt and ECC point multiplication were reported in [2, 8].

In order to be able to qualitatively compare these inherently different algorithms and their suitability for ultra-low power implementation we chose algorithm specific parameter sets that provide approximately the same level of security. In this section we talk about the rationale behind our selections, followed by a brief description of their implementations.

3.1. Parameter Selection

When we talk about matching levels of security, we base our assumptions on the widely recognized analysis by Lenstra and Verheul [6]. They relate the selection of key sizes of various types of cryptosystems to the anticipated progress of cryptanalysis and cost of computation. They distinguish between key sizes of classical asymmetric systems (RSA, Rabin's Scheme, ElGamal, etc.), Subgroup Discrete Logarithm (DL) based schemes, and Elliptic Curve (EC) based systems. However, their analysis does not include a definition of equivalent security for a lattice based scheme like NtruEncrypt. For our purpose of finding parameters for NtruEncrypt, which offer a level of security comparable to the other two systems, we therefore refer to the analysis of Hoffstein, Silverman and Whyte [5].

While in practice certain classes of applications might require a higher level of security than others, we regard our designs simply as proof of concept and hence choose to implement them at a comparatively low level of security. It should, however, be relatively straightforward to estimate the cost of higher security level implementations based on the analysis that we give at the end of this paper. For Rabin's Scheme we selected a modulus of 512 bits, which according to Lenstra and Verheul [6] provides a security level of around 60 bits. Our ECC architecture performs arithmetic in a prime field of 100 bits in size, which provides a security level between 56 and 60 bits depending on the confidence level one puts into the assumption that no significant cryptanalytic progress has been made. In the case of NtruEncrypt we chose the system parameters as $(N, p, q) = (167, 3, 128)$, based on findings in [5], offering a security level of around 57 bits.

3.2. Rabin's Scheme

Rabin's Scheme was introduced in 1979 in [10]. It is based on the factorization problem of large numbers and is therefore similar to the security of RSA with the same sized modulus. Rabin's Scheme has asymmetric computational cost. The encryption operation is faster than decryption, which is comparable to both operations of RSA with similar parameters. Its asymmetry makes Rabin's Scheme an interesting choice for sensor network scenarios in which nodes and base stations have disparate computational capabilities. A detailed description of Rabin's Scheme is contained in [10, 7].

The encryption function of Rabin's Scheme is $\mathrm{E}_{n,b}(x) \equiv x(x+b) \mod n$ where $0 \leq \mathrm{E}_{n,b}(x) < n$, $0 \leq x < n$, $0 \leq b < n$. If we set $b = 0$ this function becomes a simple squaring operation $\mathrm{E}_n(x) = x^2 \mod n = y$. Rabin's Scheme requires only one squaring for encryption.

Decryption involves finding the roots of y. The decryption function is $\mathrm{D}_n(x) \equiv \sqrt{y} \mod n$ and yields four results. In order to determine the correct solution, sufficient redundancy has to be included in x.

A more detailed description of our implementation of the encryption function can be found in [2]. We built a squarer as a bit-serial multiplier, operating on the entire width of the 512-bit multiplicand and on a single bit of the multiplier at a time. This approach has the advantage, that this unit can easily be converted to perform the exponentiations needed for the decryption function of Rabin's Scheme. The multiplier circuit consumes a chip area of less than $17,000$ gates with its accompanying average power consumption of $148.18\mu W$ (Table 1).

3.3. NtruEncrypt and NtruSign

NtruEncrypt and its associated signature scheme NtruSign are crypto systems based upon the hardness of the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) in lattices of high dimensions ($N = 167..503$).

NtruEncrypt in particular is highly efficient and suitable for embedded applications such as smart cards or RFID tags, while claiming to provide a level of security comparable to that of other established schemes.

The arithmetic of both schemes is built upon cyclic convolution in a polynomial ring $R = \mathbb{Z}(x)/(x^N - 1)$. Various security levels can be selected by appropriately choosing the parameter set (N, p, q), where p and q are short moduli by which the polynomial coefficients are reduced.

We base our performance estimates for NtruSign on data we obtained from the implementation of our scalable NtruEncrypt architecture, which is described in more detail in [2]. It is scalable with regards to the number of arithmetic units working in parallel, allowing a trade-off between area and performance. Our smallest implementation of NtruEncrypt with a single arithmetic unit takes up a chip area of less than $3,000$ gates consuming less than $20\,\mu$W, while a highly parallelized variant with 84 arithmetic units uses up to $16,200$ gates and approximately $120\,\mu$W at $500\,\mathrm{kHz}$.

3.4. Elliptic Curve Architecture

Elliptic Curve Cryptography (ECC) is the collective term for a multitude of different asymmetric cryptographic key exchange and agreement protocols, e.g. ECDH, ECDSA, ECMV, etc. Scalar point multiplication serves as the basic building block of these and is the computationally most expensive operation.

For the purpose of establishing the feasibility of an ECC based public key primitive in a pervasive security context, we selected the ECDSA as a protocol for signature generation / verification, and ECMV as a key transport protocol. Based upon the use of scalar point multiplication in the protocols and on the data we obtained from our low-power ECC base architecture, we estimate their computational complexity and performance.

Different types of finite fields can be used for the construction of elliptic curve groups. The most common ones are Galois Fields with prime characteristic or binary extension fields, e.g. $\mathrm{GF}(p)$ and $\mathrm{GF}(2^k)$. Efficient arithmetic in these fields is the key to low-power implementations of ECC in hardware.

In our ECC scalar point multiplication architecture [8] we perform operations on points of an elliptic curve given by the equation $y^2 = x^3 + ax + b$, defined over the field $\mathrm{GF}(p)$, where $p = (2^{101} + 1)/3$. We make use of the special scaled modulus $m = 2^{101} + 1$, with a scaling factor of $s = 3$, which allows us to efficiently implement modular reduction. Fast inversion can be achieved by using a variation of Thomas' et al. Algorithm X [11]. All arithmetic primitives such as addition, subtraction, multiplication and division (inversion) are implemented in a bitserial fashion. The architecture occupies a chip area equivalent to $18,720$ gates and consumes just under $400\,\mu\mathrm{W}$ of power at a clock frequency of $500\,\mathrm{kHz}$ (see Table 1).

3.5. Analysis

Table 1 shows a direct comparison of various metrics we obtained from the basic encryption architectures for Rabin's Scheme and both variants of NtruEncrypt, as well as the basic ECC scalar point multiplication architecture.

We can see that NtruEncrypt has the smallest circuit size in its simple variant and as such uses the least amount of power for computation, but it also has the worst message expansion factor[1]. The highly parallelized variant of NtruEncrypt consumes power comparable to that of Rabin's Scheme, but at the same time is more energy efficient.

In terms of power and energy consumption our ECC architecture is not comparable, neither to Rabin's Scheme nor NtruEncrypt. This is mainly due to the algorithm's heterogeneous arithmetic structure. The order of magnitude difference in delay between simple NtruEncrypt and ECC is caused by the highly bitserial implementation of the latter.

In the following we will use the data obtained from these implementations to extrapolate the power and energy requirements of decryption, signature and signature verification primitives for each of the PKC schemes under consideration.

---

1 Note that message expansion would be worse if we chose binary over ternary message representation: $p = X + 2$ instead of $p = 3$

Encryption/Decryption		Rabin	NtruEncrypt	NtruEncrypt parallel	ECMV
- Message Payload		< 512 bits	< 265 bits	< 265 bits	< 200 bits
- Ciphertext (Packets of 30 bytes)		512 bits (3)	1,169 bits (5)	1,169 bits (5)	400 bits (2)
Encryption	Time per Message	2.88 ms	58.45 ms	0.87 ms	817.7 ms
	Avg. Power	148.18 μW	19.13 μW	118.7 μW	394.4 μW
	Energy per Message	426.76 nJ	1,118.15 nJ	102.79 nJ	322.5 μJ
Decryption	Time per Message	1.089 s	116.9 ms	1.732 ms	411.54 ms
	Avg. Power	191.5 μW	58.73 μW	158.3 μW	394.4 μW
	Energy per Message	208.64 μJ	6,865.54 nJ	274.18 nJ	162.31 μJ
Sign / Verify		Rabin	NtruSign	NtruSign parallel	ECDSA
- Signature Length (Packets of 30 bytes)		512 bits (3)	1,169 bits (5)	1,169 bits (5)	200 bits (1)
Sign	Time per Message	1.089 s	233.8 ms	3.464 ms	410.45 ms
	Avg. Power	191.5 μW	58.73 μW	158.3 μW	394.4 μW
	Energy per Message	208.64 μJ	13.73 μJ	548.35 nJ	161.88 μJ
Verify	Time per Message	2.88 ms	58.45 ms	0.87 ms	822.5 ms
	Avg. Power	148.18 μW	19.13 μW	118.7 μW	394.4 μW
	Energy per Message	426.76 nJ	1,118.15 nJ	102.79 nJ	324.39 μJ

Table 1. Comparison of PKC Functions

4. Feasibility Study

In Section 2 we identified four security services that would benefit from an efficient ultra-low power PKC implementation. Here we identify which PKC function is needed for each of these services. *Broadcast Authentication* uses signature verification on the node using the base station's public key. In order to provide the *Data Encryption* service for sending data to the base station, the node has to encrypt data, again using the base station's public key. *Node-to-Node Key Distribution* requires encryption as well as decryption. *Addition of New Nodes* is based on a node having a private key. The node would either sign a message and send it to the base station, or decrypt a message received from the base station. Table 1 provides an overview of these functions for the three PKC systems that we consider.

4.1. Public Key Schemes

Now we are showing which PKC can support the above mentioned services and provide estimates on the power consumption and throughput.

Rabin's Scheme as defined in [10] can be used for all four methods. Section 3 shows how it can be used for data encryption. This is the same function as signature verification. Data decryption, as well as signature generation, requires solving the equation $\mathrm{D}_n(x) \equiv \sqrt{y} \mod n$. If we set $p \equiv q \equiv 3 \mod 4$ then the square root can be computed elegantly using Euler's criterion and Garner's algorithm as follows. We compute the solution for $y^{(p+1)/4} = c_1 \mod p$ and $y^{(q+1)/4} = c_2 \mod q$ separately. Using a slightly modified Garner's algorithm we compute the result x as $x = \pm c_1 + \left[(\pm c_2 \pm c_1) \cdot p(p^{-1} \mod q)\right] \mod n$. The exponents $(p+1)/4$ and $(q+1)/4$ as well as the factor $p(p^{-1} \mod q)$ can be precomputed and hard wired, just like the keys. A decryption takes two exponentiations with an exponent of at most 255 bits and one multiplication $\mod n$. We ignore the cost of the additions as it is negligible compared to the multiplication cost. The decryption will need 762 multiplications on average with 255-bit coefficients and one 512-bit multiplication. If we employ the same circuit as we use for encryption (which is not an optimal solution) then one decryption would take on average 544,753 clock cycles. That means a single decryption or signing would take 1.09 seconds. This new circuit would require more storage for c_1 and c_2, additional multiplexers for the precomputed constants and a more complex control logic. Conservative estimates result in a total power consumption of 191.5μW.

NtruEncrypt and NtruSign Our basic NtruEncrypt encryption primitive provides us with representative data from which we can extrapolate power and energy requirements for the decryption, signature generation and signature verification procedure. In our original architecture we fixed the public key as a constant in a very compact look-up table. For our estimates of the other primitives we therefore add an overhead of around $40\,\mu$W of static power to our simulation results that caters for the additional storage requirements. We base our estimates further on the number of cyclic convolutions that are required by the respective primitive, since that is the central arithmetic operation in all Ntru schemes.

Based on [4, 3] we found the number of convolution operations to be 1, 2, 4 and 1 for encryption, decryption, signature generation and verification, respectively. Convolution

is by far the most complex operation in NtruEncrypt and NtruSign, so it is safe to assume that time and energy are proportional to the number of convolutions. The figures for energy consumption are the products of the time and power estimates.

ECMV and ECDSA The elliptic curve based encryption and signature algorithms we selected are all based upon scalar point multiplications. According to the description of these algorithms in several standards and publications, the encryption and signature verification primitives of ECMV and ECDSA each require two scalar point multiplications, while for decryption and signature generation a single scalar point multiplication is sufficient. We base our time estimates on these findings, coupled with the performance figures for our baseline ECC architecture.

4.2. Comparison

Table 1 compares the PKC functions with regards to speed, power, energy and message length. The transmission power for the messages is not included as we did not consider a particular transmission system. However, the length of the ciphertext and signature can be used for an estimate for a particular transmitter. For encryption and decryption the ratio of payload length vs. ciphertext length is important. Signatures are transmitted in addition to the original message.

Typical packet sizes on WSN are 30 bytes [9] and 56 bytes. Due to its asymmetry, Rabin's scheme is particularly suitable if only encryption and signature verification are performed on the node. Otherwise it is comparable to ECC. Ntru has the smallest average power consumption, but the largest message size of 5 packets. In environments where transmission power is not the most dominant part, Ntru has an advantage. ECC has a small message expansion for encryption and a high power consumption but requires the smallest number of packets. Also, the message content (key material) rarely exceeds 200 bits. On most WSN nodes, transmitting a single bit costs as much power as executing 1000 instructions. Small message sizes and low overhead is of utmost importance which is a feature of ECC.

Broadcast Authentication can benefit greatly from using a PKC. With ECC only one additional packet needs to be sent to authenticate a message from the base station. Protocols like μTESLA [9] require a complicated delayed key disclosure scheme which requires constant key updates, the nodes have to store keys, and be time synchronized. Bootstrapping a new node becomes especially difficult. *Node-to-Node Key Distribution* can now be done with only two (ECC) or three (Rabin) packets between the nodes. Involving the base station in this key setup becomes especially expensive if the communicating nodes are many hops away. The scheme presented in [9] requires at least four messages, three of which involve the base station. The details of when *Data Encryption* is advantageous and how the *Addition of new Nodes* is handled is dependent on the specific protocol. However, our results indicate, that only very few packets are necessary with PKC.

5. Conclusions

In this paper we have provided an in-depth comparison of three different PKC implementations particularly targeted at wireless sensor networks. We have shown that the use of PKC can actually reduce the amount of traffic overhead due to key management in WSN. The computational cost is within acceptable limits and sufficiently fast. Our estimates are based upon implementations of the encryption function of three representative, but inherently different algorithms.

References

[1] D. W. Carman, P. S. Kruus, and B. J. Matt. Constraints and approaches for distributed sensor network security. Technical report, NAI Labs, Security Research Division, 2000.

[2] G. Gaubatz, J.-P. Kaps, and B. Sunar. Public key cryptography in sensor networks—revisited. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, 2004.

[3] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J. Silverman, and W. Whyte. NTRUSign: Digital signatures using the NTRU lattice. In *Topics in Cryptology–CT-RSA 2003*, volume 2612 of *LNCS*, pages 122–140. Springer Verlag, 2003.

[4] J. Hoffstein, J. Pipher, and J. Silverman. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory (ANTS III)*, volume 1423 of *LNCS*, pages 267–288, Berlin, 1998. Springer-Verlag.

[5] J. Hoffstein, J. Silverman, and W. Whyte. NTRU report 012, version 2. estimated breaking times for NTRU lattices. Technical Report 12, NTRU Cryptosystems, Inc., 2003.

[6] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[7] A. J. Menezes, P. C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press Inc., 1997.

[8] E. Öztürk, B. Sunar, and E. Savaş. Low-power elliptic curve cryptography using scaled modular arithmetic. In *CHES 2004*, volume 3156 of *LNCS*, pages 92–106. Springer, 2004.

[9] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[10] M. O. Rabin. Digitalized signatures and public key functions as intractable as factorization. MIT/LCS/TR-212, Massachusetts Institute of Technology, 1979.

[11] J. Thomas, J. Keller, and G. Larsen. The calculation of multiplicative inverses over GF(p) efficiently where p is a mersenne prime. *IEEE Transactions on Computers*, 5(35):478–482, 1986.