

Performance of Additive Homomorphic EC-ElGamal Encryption for TinyPEDS

Osman Ugus
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
ugus@netlab.nec.de

Alban Hessler
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
hessler@netlab.nec.de

Dirk Westhoff
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
westhoff@netlab.nec.de

ABSTRACT

Data aggregation is a popular approach employed in wireless sensor networks (WSNs) to minimize data transmission and storage. With aggregation techniques, the monitored data is expressed in a condensed form: Therefore, instead of storing all data sensed by several nodes, the network stores a condensed value only such as the sum of these values. However, data aggregation becomes problematic when the data to be aggregated is encrypted. As a solution, we apply an additive homomorphic encryption scheme, namely the elliptic curve ElGamal (EC-ElGamal) cryptosystem, and present the performance results of our implementation for the prominent sensor platform MicaZ mote.

1. INTRODUCTION

Regarding the frequency of transmission wireless sensor networks (WSNs) may be divided into two subgroups, namely asynchronous and synchronous [7]. In synchronous WSNs the monitored data transmitted to a reader device is real-time responsive. In asynchronous WSNs, however, the monitored data is transmitted to a reader device only seldomly. Therefore, asynchronous WSNs need to store the data in the network in a distributed manner. However, the implementation of distributed data storage for WSNs is very challenging.

Firstly, due to the limited storage capacity of the nodes, the storage capacity of the whole WSN is restricted. Thus, it is necessary to reduce the amount of the data being processed, e.g. stored or transmitted, in the network without losing relevant information. Secondly, the nodes have limited power capacity. Distributed data storage requires transmission between sensor nodes. Since the transmission affects the power consumption, techniques for minimizing it are mandatory. Finally, the nodes are in general equipped with non-tamper-resistant hardware. Since WSNs are usually employed in a public environment, data must be protected and concealed.

One popular method employed for minimizing the data transmission and storage is *in-network data aggregation*. This means that the monitored data is expressed in a condensed form such that instead of storing all the data sensed by several nodes, the network stores only condensed values such as the sum or the average of these values. In order to secure the data stored in the network, data encryption techniques are employed. However, in-network data aggregation becomes challenging when the data is encrypted. In such case the encryption scheme needs to allow homomorphic addition of ciphertexts. As solution for these problems, *tiny persistent encrypted data storage* (TinyPEDS) was proposed, see [7].

In this work we introduce the *additive homomorphic elliptic curve ElGamal* (EC-ElGamal) encryption scheme, which is employed for securing distributed storage and transmission of aggregated data in TinyPEDS. Furthermore, the performance results of the implemen-

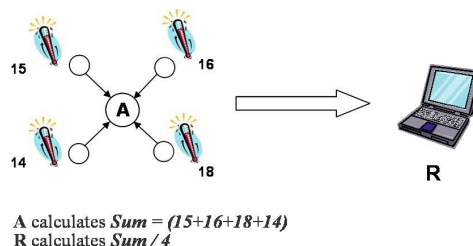


Figure 1: Data aggregation in WSNs

tation on MicaZ mote are presented.

2. CONCEALED DATA AGGREGATION

As resource consumption of the nodes is a critical factor for the overall lifetime of a wireless sensor network, it is necessary to employ techniques to reduce it. The in-network data aggregation works toward this goal by reducing the amount of the data being stored and transmitted, while still providing an appropriate degree of information to the reader device. Figure 1 depicts a simple example for the in-network data aggregation process where the sensed values are not encrypted.

However, as WSNs are usually employed in a public environment, the data has to be encrypted in order to limit damage caused by possible attacks. In this case, the aggregator node *A* should calculate the $Sum = [Enc(15) + Enc(16) + Enc(18) + Enc(14)]$ of the encrypted values. This scenario is depicted in Figure 2.

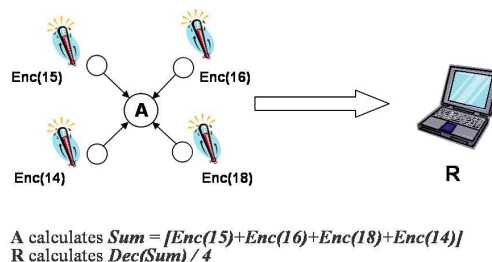


Figure 2: Concealed data aggregation in WSNs

In general, the decrypted sum of ciphertexts $Dec[Enc(15) + Enc(16) + Enc(18) + Enc(14)]$ will not be equal to the sum of the plaintexts $(15+16+18+14)$. Thus, the encryption scheme employed needs to support the property such that the following equa-

tion holds:

$$Enc(a_1 + a_2 + \dots + a_n) = Enc(a_1) \diamond Enc(a_2) \diamond \dots \diamond Enc(a_n),$$

where $Enc(a)$ denotes the encryption of a message a and \diamond represents an addition performed on ciphertexts from a public encryption scheme. An encryption scheme with this property is called *additive homomorphic*. TinyPEDS proposes to employ both a symmetric and a public key encryption scheme. However, in this work we introduce only the asymmetric additive homomorphic encryption EC-ElGamal. An example for a symmetric additive homomorphic encryption scheme is [4].

3. ELLIPTIC CURVE ELGAMAL ENCRYPTION SCHEME

The original ElGamal encryption scheme, see [6], is *multiplicative homomorphic*. In order to get the desired additive homomorphic property, it can be implemented in elliptic curves. The methods for EC-ElGamal encryption and decryption is shown in Algorithm 1 ([13]).

The function $map()$ is a deterministic mapping function used to map values $m_i \in GF(p)$ into plaintext curve points $M_i \in E$ such that

$$map(m_1 + \dots + m_n) = \underbrace{map(m_1)}_{M_1} + \dots + \underbrace{map(m_n)}_{M_n}$$

holds. The function $map()$ is necessary, because the addition over an Elliptic Curve is only possible with points on that curve, thus, integers have to be mapped to corresponding points. For this purpose, each integer m is mapped to a curve point M , which is the m -multiple of the generator point G , i.e. $M = mG$. The reverse mapping function $rmap()$ then extracts m from a given point mG . The mapping function

$$map : m \rightarrow mG \text{ with } m \in GF(p)$$

fulfills the required homomorphic property, because

$$\begin{aligned} M_1 + M_2 + \dots + M_n &= map(m_1 + m_2 + \dots + m_n) \\ &= (m_1 + m_2 + \dots + m_n)G \\ &= m_1G + m_2G + \dots + m_nG \end{aligned}$$

holds, where $m_1, m_2, \dots, m_n \in GF(p)$.

The mapping function is not security relevant, i.e. it neither increases nor decreases the security of the EC-ElGamal encryption scheme. Note that the reverse mapping function is the same as solving the *discrete logarithm problem* over an elliptic curve and, therefore, a weakness of this scheme. However, since the reverse mapping function is only performed on the powerful reader device and the maximum length of the final aggregation is assumed to be small for realistic WSNs, see [13], the m can be obtained by performing a brute force attack on $M = mG$ on the reader device.

Algorithm 1 Elliptic curve ElGamal encryption scheme

Private key: $x \in GF(p)$

Public key: E, p, G, Y , whereby $Y = xG$ and elliptic curve E over $GF(p)$ with $G, Y \in E$

Encryption: For a given plaintext $m \in [0, p - 1]$ and random $k \in [1, n - 1]$, where n is the order of E .

$$M = map(m)$$

$$\text{ciphertext } C = enc(m) = (R, S) = (kG, M + kY)$$

Decryption:

$$M = dec(C) = dec(R, S) = -xR + S$$

$$m = rmap(M)$$

4. IMPLEMENTATION

The software architecture of the EC-ElGamal cryptosystem depicted in Figure 3 consists of three abstraction levels. The lowest level contains finite field arithmetic operations such as *modular addition*, *modular subtraction*, and *modular multiplication*. Since the operations on higher levels are based on the operations on this level, the finite field level is the most critical level for the performance. Elliptic curve arithmetic operations such as *point addition*, *point doubling*, and *scalar point multiplication* are grouped on the second level. Finally, on the application level, by using the operations from lower levels any elliptic curve cryptosystem such as EC-ElGamal may be implemented.

The performance of a cryptosystem is influenced by the employed algorithms. Therefore, it is important to choose the best combination carefully. In the following the algorithms employed in each abstraction level are introduced.

4.1 Design decisions at finite field level

4.1.1 Selecting the underlying field

In [3] Branovic *et al.* studied the performance and memory requirements of several elliptic curve algorithms over the prime $GF(p)$ and binary field $GF(2^n)$. They found that the number of memory accesses and the code sizes of elliptic curve algorithms for binary fields is higher than those for prime fields. Moreover, the binary field arithmetic, particularly multiplication, is not sufficiently supported by usual microprocessors. Thus, the use of binary field would lead to lower performance [9].

On the other hand, the main disadvantage of the prime field arithmetic is that the inversion over $GF(p)$ is computationally much more expensive than the inversion over a binary field. However, by using different coordinate systems, the number of inversions required by finite field and elliptic curve arithmetic can be reduced to only one, which is needed for converting the final results back to affine coordinates. Since the implemented cryptosystem in this work allows those final conversions on the external reader device, the performance drawback of prime fields stemming from expensive inversions can be ignored.

In contrast to the inversion, *modular reduction* can be performed over prime fields faster than over binary fields when special *Pseudo-Mersenne primes* [18] are used. Therefore, the arithmetic operations in this work were implemented over $GF(p)$.

4.1.2 Multi-precision multiplication

The efficiency of the multi-precision multiplication dominates the overall performance. In [9] Gura *et al.* show that on resource constrained devices such as MicaZ mote, 85% of the execution time is spent on the multi-precision multiplication for a typical point multiplication. That means that the performance of the multi-precision multiplication is critical for the entire cryptosystem and therefore, it is especially necessary to concentrate on its optimization possibilities.

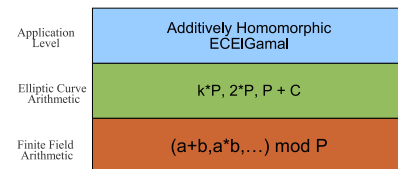


Figure 3: Elliptic curve ElGamal (EC-ElGamal) design architecture

Additionally to the well-known *Schoolbook* and *Karatsuba* multiplication, see [11], [17] analyzed the *Comba* multiplication, see [8], and the *Hybrid* multiplication from [9]. Our analysis showed that the Hybrid method is the most promising one, because it combines the advantages of the Schoolbook and Comba method. Therefore, it requires a small amount of registers and memory accesses.

4.1.3 Modular reduction

After each finite field operation the modular reduction is necessary whenever the result is bigger than the modulus. Therefore, the performance of the modular reduction is as important as multi-precision multiplication for the overall performance of the cryptosystem.

In [17] we analyzed three methods to perform modular reduction: *Montgomery reduction*, *Barrett reduction*, and the *pseudo-Mersenne prime reduction* [11]. Our analysis showed that the pseudo-Mersenne prime reduction is superior to the other methods.

4.2 Design decisions at elliptic curve level

4.2.1 Selecting the coordinate system

Elliptic curve points may be represented in several coordinate systems. For each of such systems, the performance and the memory requirements of the elliptic curve operations are different. Therefore, a good choice for the coordinate systems is an important factor for the successful implementation of the elliptic curve cryptosystems on resource limited devices. In [17] we compared six popular coordinate systems in terms of performance and memory requirements, namely *Affine*, *Projective*, *Jacobian*, *Chudnovsky-Jacobian*, *modified Jacobian*, and *mixed* coordinate systems.

We showed that *AJJ* coordinates are superior to the other mixed coordinate systems. Thereby, *A*, *J* and *M* denotes the affine, Jacobian and modified Jacobian coordinates, respectively. Note that *AJJ* denotes the point addition, whereby the result is in Jacobian coordinates, while the input points are in affine and Jacobian coordinates. Similarly, *MJ* denotes the point doubling with the input point is in modified Jacobian and the result is in Jacobian coordinates. The performance of the point doublings in *MJ* coordinates is the highest, whereas the point doublings in those coordinates require more temporary storage than the point doublings in *JJ* coordinates. However, since the results of the point additions in the application level are used as input for the point doublings during the scalar point multiplication, the use of *MJ* coordinates needs one more additional conversion from Jacobian to modified Jacobian coordinates. This means the computational efficiency of point doublings in *MJ* coordinates is equal to the point doublings in *JJ* coordinates. Thus, *AJJ* and *JJ* coordinates were chosen in the implementation of the point addition and point doubling, respectively.

4.2.2 Scalar point multiplication

The main operation in elliptic curve cryptosystems is the scalar point multiplication. Since the scalar point multiplication is a time critical operation, many efficient implementations have been proposed. We evaluated the *Left-to-Right* and *Right-to-Left* binary methods, see [11], as a core algorithm for scalar multiplication.

The examination in [17] showed that the Left-to-Right method is superior due to lower storage requirements and the fact that the Right-to-Left method does not allow *AJJ* point additions.

Several techniques may be employed to further speed up the scalar point multiplication on elliptic curves. One of them is the *Interleave method* [12], which aims at reducing the required number of point doublings. The Interleave method was originally proposed for the multi-scalar multiplications of the form $(k_1 \cdot P_1 +$

Table 1: Comparison of the scalar point multiplication implementations

160-bit (sec160r1)	#Precom. points	Ex. time [s]
This work (L2R)	0	1.23
This work (2MOF)	0	1.03
[9]	0	0.81
[18]	0	1.35
TinyECC-0.2	0	1.78
This work	1	0.69
This work	2	0.57
[18]	15	1.24

$k_2 \cdot P_2 + \dots + k_n \cdot P_n$). However, this idea can be used to perform single scalar multiplications. This is due to the fact that the scalar multiplication kP with a n -bit scalar k and a point P may also be represented as a sum of t partial multiplications with t scalars k_i and t points P_i as follows

$$kP = (k_t \cdot 2^{(t-1)n/t} + \dots + k_2 \cdot 2^{n/t} + k_1) \cdot P \quad (1)$$

$$= \sum_{i=1}^t k_i \cdot P_i \quad (2)$$

Thereby, each scalar k_i is a n/t -bit long part of k and $P_i = 2^{(i-1)n/t} P$. Note that n is padded with 0's from left until it is a multiple of t . In order to increase the performance the points P_t, P_{t-1}, \dots, P_2 are ideally precomputed and stored off-line.

Another way for accelerating scalar multiplication is to reduce the required number of point additions. The point additions are performed, if the corresponding bit of the scalar is 1. This means that the number of point additions may be further reduced, if the scalar is represented with a low *Hamming weight*, which is the number of the non-zero bits. In general the Hamming weight of a scalar is low, if it is represented in a signed representation. [17] analyzed both the *non-adjacent form* (NAF) and the *mutual opposite form* (MOF), see [15] and [14], respectively. Although both lead to the same Hamming weight, MOF is superior to NAF, because it exhibits lower memory requirements.

5. IMPLEMENTATION RESULTS

The implementation was done on the MicaZ mote [2], which is a typical device for WSNs and equipped with 8-bit processor. The operating system employed in the implementation was TinyOS-2.0 [1], an open-source operating system designed for wireless embedded sensor networks. The timing results are the average over 500 executions with random numbers and were generated using the curve parameters *secp160r1* from [16]. The operations on the lowest abstraction level were realized using Assembler, while those on higher levels were written in *NesC*.

Table 1 compares the performance of the point multiplication with existing solutions, namely TinyECC-0.2 [10] and the solution from [9] and [18]. Note that in Table 1 the results in the first row imply that the point multiplication employed in the ECEI/Gamal was realized by using only the Left-to-Right binary method. The values in the second row of the table present the point multiplication accelerated by using 2MOF. The results presented in the remainder of the table were obtained by employing the Interleave method. Therefore, the first five rows imply *general point multiplication*, while the rest of the table represents *fixed point multiplication*.

The performance of the point multiplication from TinyECC-0.2 is not presented in [10]. Thus, the TinyECC-0.2 was integrated in the test suite employed for evaluating this work. Note that the finite

Table 2: Implementation results for EC-ElGamal

#Precom. points	Ex. time [s]
0 (L2R)	2.48
0 (2MOF)	2.16
2	1.42
4	1.19

field inversion is required for converting the elliptic curve points from Jacobian to affine coordinates during the decryption process. Since that needs to be performed only on the reader device, the finite field inversion was not implemented. However, since one inversion and four multiplications are needed for the conversion, some assumptions are necessary. According to [5], the ratio between finite field multiplications and inversions $Inv/Mult$ is 30. Hence, together with the additional four modular multiplications needed for the conversion, this results in $34 \cdot 0.532ms = 18ms$, where $0.532ms$ is the execution time of one modular multiplication, see [17]. For a fair comparison, this value is already added to the results in Table 1. Finally, Table 2 shows the performance of the different realizations of the EC-ElGamal, which contains two point multiplications with n -bit scalar k and one short point multiplication with the sensed data m , see Algorithm 1. Note that for testing purposes m was chosen to be 8-bit.

6. CONCLUSION

We implemented elliptic curve and finite field arithmetic operations on MicaZ mote which is a typical device employed in wireless sensor networks. Our tests showed that scalar point multiplication with a random base takes 1.03s, while it takes only 0.57s in the case of fixed point multiplication, when 2 precomputed points are employed. According to our knowledge, our implementation is the second fastest for general point multiplication. Moreover, compared to the best previous result, our implementation is at least 44% faster for fixed point multiplication. The implementation of the elliptic curve EC-ElGamal encryption showed that the encryption operation only takes 1.19s, when in total 4 precomputed points are utilized.

7. ACKNOWLEDGMENTS

The work presented in this paper is supported in part by the European Commission within the STREP UbiSec&Sens. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the UbiSec&Sens project or the European Commission.

8. REFERENCES

- [1] <http://www.tinyos.net>.
- [2] Crossbow Technology Inc. <http://www.xbow.com>.
- [3] I. Branovic, R. Giorgi, and E. Martinelli. Memory Performance of Public-Key cryptography Methods in Mobile Environments. In *ACM SIGARCH Workshop on Memory performance: DEaling with Applications, systems and architecture (MEDEA-03)*, pages 24–31, New Orleans, LA, USA, September 2003.
- [4] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient Aggregation of Encrypted Data in Wireless Sensor Networks. *3rd Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Sensor Networks, Italy*, April 2005.
- [5] H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65, London, UK, 1998. Springer-Verlag.
- [6] T. E. Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in cryptology - Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag New York, Inc., 1985.
- [7] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks. *Elsevier Ad Hoc Journal*, 5(7):1073–1089, September 2007.
- [8] J. Großschädl, R. M. Avanzi, E. Savas, and S. Tillich. Energy-efficient software implementation of long integer modular arithmetic. In *Cryptographic Hardware and Embedded Systems*, volume 3659 of *Lecture Notes in Computer Science*, pages 75–90, 2005.
- [9] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems - CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132, Cambridge, MA, USA, August 2004.
- [10] A. Liu and P. Ning. *TinyECC: Elliptic Curve Cryptography for Sensor Networks*. September 2006.
- [11] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [12] B. Möller. Algorithms for Multi-exponentiation. In *Selected Areas in Cryptography - SAC 2001*, volume 2259 of *Lecture Notes In Computer Science*, pages 165–180, London, UK, 2001. Springer-Verlag.
- [13] E. Mykletun, J. Girao, and D. Westhoff. Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks. In *IEEE Int. Conference on Communications - ICC*, Istanbul, Turkey, June 2006.
- [14] K. Okeya, K. Schmidt-Samoa, C. Spahn, and T. Takagi. Signed Binary Representations Revisited. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 123–139, 2004.
- [15] G. W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.
- [16] Standards for Efficient Cryptography Group (SECG). *Specification of Standards for Efficient Cryptography — SEC 2: Recommended Elliptic Curve Domain Parameters*, September 2000.
- [17] O. Ugus. Asymmetric Homomorphic Encryption Transformation for Securing Distributed Data Storage in Wireless Sensor Networks. Master’s thesis, Technische Universität Darmstadt - in Cooperation with NEC Europe Ltd., Heidelberg, Darmstadt, 2007.
- [18] H. Wang and Q. Li. Efficient Implementation of Public Key Cryptosystems on Mote Sensors. In *Eighth International Conference on Information and Communications Security (ICICS 2006)*, volume 4307 of *Lecture Notes in Computer Science*, pages 519–528, December 2006.