Towards Acceptable Public-Key Encryption in Sensor Networks

Erik-Oliver Blaß and Martina Zitterbart

Institut für Telematik, Universität Karlsruhe
Zirkel 2, 76128 Karlsruhe, Germany
{blass, zit}@tm.uka.de
http://www.tm.uka.de

Abstract. One of the huge problems for security in sensor networks is the lack of resources. Based on microcontroller architectures with severe limited computing abilities, strong public-key cryptography is commonly seen as infeasible on sensor devices. In contrast to this prejudice this paper presents an efficient and lightweight implementation of public-key cryptography algorithms relying on elliptic curves. The code is running on Atmels popular 8Bit ATMEGA128 microcontroller, the heart of the MICA2[15] platform. To our knowledge this implementation is the first to offer acceptable encryption speed while providing adequate security in sensor networks.

1 Introduction

Wireless sensor networks are one of the key technologies of the ubiquitous computing visions. Physically small sensor devices are able to cooperate with each other using radio interfaces. Furthermore there is a large number of scenarios where the data exchanged between sensor nodes is critical eg. in health or military applications. To protect sensible data against threats and to ensure security properties like integrity, authenticity or confidentiality traditional network protocols rely on cryptographic primitives like encryption and decryption as well as signature schemes. The question is whether these primitives can be used in sensor networks as well.

As cost and energy savings are of paramount importance, most commonly utilized sensor hardware is based around a battery powered microcontroller like e.g. Atmels 8Bit ATMEGA128 offering only 7 MIPS with 4 KByte RAM. Popular systems like UC Berkleys MICA platform [15] are built around this core. Because of these limited sensor hardware strong cryptography is commonly considered as a delicate problem throughout the community. While there is consensus that symmetric ciphers might work (see related work), there is a prejudice against the feasibility of well known asymmetric methods based on the RSA-problem or the Diffie-Hellman-problem. This is due to the fact that a pleasing implementation of asymmetric algorithms giving satisfying performance together with minimal memory consumption is yet missing. As cryptographic primitives are the fundamental building blocks of every secure protocol the knowledge of algorithm usability is crucial for the design of new protocols for sensor networks.

The contribution of this work is an implementations of asymmetric encryption and signature generation schemes for the 8Bit ATMEL sensor platform that features acceptable run-time and memory consumption while preserving a level of acceptable security for sensor networks. In contrast to the public opinion this allows the design of new security protocols utilizing public-key techniques even for sensor networks.

Hereby our implementation is focused on elliptic curve cryptography (ECC). Algorithms like Diffie-Hellman [2], El-Gamal [13], DSA [14] based on ECC offer the same security than traditional based algorithms but consume a lot less of memory and computing power [8]. As an example: RSA with a key size of 622 Bits offers the same security against attacks as an elliptic curve with only 105 Bit key size while consuming a lot more computing time and memory. This is what makes elliptic curves attractive for wireless sensor networks.

Our fast implementation is based on the precomputation of *points* on the one hand and handcrafted optimization on the other[2].

2 Related work

As no satisfying implementation of efficient asymmetric cryptography on microcontroller based sensor hardware exits there is apparently the need to look for alternatives. Publications like [12] mimic asymmetric signatures schemes by a relatively complex scheme of two party hash chains, so do [10] and [11]. Other works like [4] try to establish pairwise secret keys to avoid public and private key schemes or Diffie-Hellman like key exchanges. In [5] and [6] the authors implement elliptic curve cryptography for sensor networks. However the underlying hardware is quite sophisticated consisting of 16 Bit microcontrollers with 16 MHz clock frequency. Therefore the results are only of limited value as typical sensor hardware does not dispose of such powerful computing resource. In [3] a high-performance microcontroller offerings 24 MIPS, i.e. 3 times more than the usual ATMEGA 128, is utilized. The work is also based on very special Galois fields called *optimal extension fields* where field multiplication can be done quite efficiently but the security of this idea is yet to be proven. The authors of [1] try to implement elliptic curves on 8 Bit ATMEGA128 chips but reach poor results: for a signature generation over 1:08min of expensive computing and battery time has to be spent, which surely is not affordable.

3 Elliptic Curve Cryptography

Elliptic curves are an algebraic structure whose use for cryptography was first mentioned in [9]. They feature properties which allow the setup of a problem similar to the well known discrete logarithm problem of finite (Galois) fields. An elliptic curve K is a set points over a field that satisfies a certain equation. Is a curve defined over the field $\mathbb{F}$ all of its points (x, y) with $x, y \in \mathbb{F}$ satisfy the so called *Weierstraß*-equation

$$y^2 + a_1xy + a_3y \equiv x^3 + a_2x^2 + a_4x + a_6,\ a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$$

Here a_i are the parameters of the curve.

[2] A comprehensive and detailed version of this work has been publish as a technical report at `http://doc.tm.uka.de/tr/TM-2005-1.pdf`.

Elliptic Curve Discrete Logarithm Problem: ECDLP The problem to find logarithms is difficult to compute in finite fields. For a large prime p, a finite field $\mathbb{F}_p$ and the equation $a = b^c \bmod p$, $a, b \in \mathbb{F}_p$, $0 \leq c \leq p - 2$ the task is to find c with given a and b. There is no known algorithm to compute a so called *discrete* logarithm in polynomial time. Something similar holds for elliptic curves over finite fields. Without going into details you can *add* and *subtract* two points $A, B \in K$ from a curve K which results in a new point $C \in K$. Together with a point 0 at infinity the addition of points on elliptic curves gives an algebraic structure called a group. Within this group a problem analog to the discrete logarithm problem can be introduced. A multiplication $Q = kP$ of a point P with an integer k can be seen as multiple additions of point P resulting in a product $Q \in K$. Given a curve K over F_{2^p}, a point $P \in K$ and a product $Q \in K$ it is a problem to find $k \in \mathbb{N}$ that holds $Q = kP$. This problem is called *Elliptic Curve Discrete Logarithm Problem ECDLP* and hard to solve. For example with a finite field $\mathbb{F}_{2^p}$ you need about $O(2^{\frac{p}{2}})$ operations to find k [8]. On top of ECDLP certain algorithms from section 5 can be set up.

4 ECC Implementation Details

This section deals with implementation internals and describes all the details done to achieve maximum performance while preserving main memory. The reader may skip to section 5 to learn about the results of the implementation.

One of the first and most important decision to take is regarding user key size. In ECC key size means the size of the underlying finite field, i.e. if you want to use 53 Bit keys, the elliptic curve has to be over $F_{2^{53}}$. Smaller keys mean better performance but offer less security. The smallest but secure key size has to be found. The largest broken ECDLP yet had 109 Bit key size ie. over the finite field $\mathbb{F}_{2^{109}}$ and it took 17 months[7] to break it. Therefore security of 109 Bit keys size is now debatable and one should choose larger keys for securing very sensible information even in sensor networks. The next greater than 109 Bit possible key size/curve that is suitable for ECC is 113 Bit, ie. a curve over $\mathbb{F}_{2^{113}}$. We choose this curve as it offers about 16 times more security than 109 Bit which seems enough security for todays hardware.

RAM to ROM One key point to save main memory is moving all larger unchangeable data from RAM to flash-ROM or EEPROM which is generally supported on the Atmel platform. Later on ROM regions can be copied temporarily back from ROM to RAM using specials commands.

Without giving details ECC multiplication utilizes a large but constant multiplication matrix λ_{ij0}. It was therefore very reasonable to precompute this table offline and distribute it to every sensor node prior to deployment as well as to move λ_{ij0} out of valuable main memory to (flash-)ROM. This saved about 908 Bytes of valuable RAM which does not sound much but means 22% of entire main memory. The same goes for field inversion operations in $F_{2^{113}}$ which need two arrays of constants for its work. Moving these to ROM, additional 1164 Bytes (28%) were saved.

Point Multiplication As ECDLP is based on a lot of point multiplications $Q = kP$, this is the most crucial operation in ECC. With the description of the different implemented algorithms from section 5 you notice that point multiplication can generally be classified into two different types. The first one is point multiplications with an always *fixed* point P, the *base point*, as well as point multiplications with arbitrary non-fixed varying points P.

Class 2 multiplications are slower as the ones from class 1 and are implemented using an ECC version of the popular square-and-multiply algorithm for large number exponentiation. Class 1 multiplications do have an advantage of allowing the use of precomputation – which is our key to speed here. Consider the binary representation of k as:

$$k = k_{112}2^{112} + k_{111}2^{111} + \cdots + k_1 2^1 + k_0 2^0,\ k_i \in \{0,1\}.$$

As P is considered as a fixed point here for all communication we can set up a buffer table where all products $2^i P$ with $0 \leq i < 113$ are stored. This one time initial computation of the buffer can again be done offline and written to sensor nodes prior to deployment. It has a size of a 3616 Bytes and perfectly fits into program memory regions of flash-ROM.

For a new multiplication $Q = k'P$ this means for every Bit k'_i that is set to 1: add $2^i P$ to Q. Instead of adding P together $k', 1 \leq k \leq 2^{113}$-times expensively, we can simply use our table of precomputed points and simply add no more than 113 times to obtain Q. As we will see in table 1 our class 1 multiplication is faster by a factor of 2.56 than class 2 multiplication.

Further optimizations Another way to gain more speed is handcrafting a source to the target platform which is often underestimated. Using run-time profiles execution times of heavily used and expensive functions could be halved by e.g. sophisticated loop-unrolling. This comes with the cost of a larger ROM image as seen in section 5.

5 Results of implemented algorithms

This section describes implementation results of various ECC-based algorithms that run on our sensor hardware to prove the feasibility of asymmetric cryptography. The implemented algorithms were chosen because of their popularity throughout the community. All results are summarized in table 1 and offer serious performance gains compared to the outputs of [1].

Elliptic Curve Diffie-Hellman ECDH This well known algorithm from [2] is quite important in modern protocols as a key exchange and can be adopted for ECC. ECDH needs two point multiplications. One multiplication is with a fixed base point P and the other one with the received peers public key. Thus a complete ECDH takes an average time of 24.02sec. In most cases one of the point multiplications can be omitted in a way that a complete ECDH would take only one class 2 point multiplication with a received public key in 17.28sec.

Operation	Time[s]	Standard derivation/s	Estimated results as of [1][s]
Point multiplication (fixed)	6.74	0.67	≈34
Point multiplication (random)	17.28	0.47	≈34
Key generation	6.74	0.67	≈34
Complete Diffie-Hellman key exchange	17.28 (24.02)	0.57	≈68
El-Gamal encryption	24.07	0.94	≈68
El-Gamal decryption	17.87	0.03	≈34
ECDSA signature	6.88	0.46	≈34
ECDSA verification	24.17	0.72	≈68

Table 1. Average times for different operations

El-Gamal Taher ElGamal described a popular asymmetric encryption algorithm in [13] back in 1985 which relies on traditional DLP and that has been adopted to elliptic curves and the ECDLP. Encryption uses one (fast) multiplication with base point P, one (slow) multiplication with a random point and a few other operations for data embedding or point addition. It takes 24.07sec as a whole. Decryption consumes 17.87sec of time.

Elliptic Curve Digital Signature Algorithm Finally the *Digital Signature Algortihm* [14] (DSA) algorithm was implemented on our target platform as it can be transformed to use ECDLP. Signature generation consists of only one point multiplication of the fixed point P taking only around 6.88sec. A signature verification step takes about 24.17sec.

Memory Consumption Besides computing time memory consumption is an important criteria for the use in sensor networks. All implemented algorithms together consumed a total of 208 Bytes RAM (164 Bytes for `.bss`, 44 Bytes for `data`). A total of 208 Bytes of main RAM (=0.05%) is what a sensor node has to spent for ECC permanently. As there is no recursion in the code the stack is only slightly utilized for function calls.

The ATMEGA128 features 4 KByte of EEPROM which is not used in our current implementation, but can be accessed in a similar way as flash-ROM is. While ROM-memory use is not quite as critical as main RAM memory it is interesting to see how much space is consumed due to the use of loop-unrolling and inlining of functions for speed optimization. A total of 73 KBytes of flash-ROM is permanently utilized for ECC operations which is about 57% of available ROM. This leaves 55 KByte for normal sensor code which is still quite a lot and should not make any problem.

6 Conclusion

This work concludes that public-key cryptography *is* possible in sensor networks – quite contrary to popular related work. Existing security protocols for sensor networks detouring asymmetric primitives with complicated symmetric constructs have to be reconsidered as there is now the chance to develop new security protocols for sensor networks which might be based on more elegant asymmetric or hybrid techniques. The key for efficiency in our work are memory optimizations as well as a precomputation of *base points* for faster execution.

This does of course not solve the general trust or key bootstrapping problem in sensor networks, i.e. how to initially distribute trust or keys in an ad-hoc formed network without (public-key) infrastructures. But future work can now tackle this problem without turning public key cryptography aside.

References

1. Malan D. J., Welsh, M., Smith, M. D.: A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography. First IEEE International Conference on Sensor and Ad Hoc Communications and Networks, 2004
2. Diffie, W., Hellman, M. E.: New Directions in Cryptography. IEEE Transactions on Information Theory, 1976
3. Kumar, S., Girimondo, M., Weimerskirch, A., Paar, C., Patel, A., Wander, S.: Embedded End-to-End Wireless Security with ECDH Key Exchange. The 46th IEEE Midwest Symposium On Circuits and Systems, 2003.
4. Liu, D., Ning, P.: Establishing Pairwise Keys in Distributed Sensor Networks. 10th Computer and Communications Security, 2003
5. Huang Q., Cukier J., Kobayashi, H., Liu B., Zhang, J.: Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks. International Conference on Wireless Sensor Networks and Applications, 2003
6. Huang Q., Kobayashi, H.: Energy/security scalable mobile cryptosystem. IEEE Personal, Indoor and Mobile Radio Communications, 2003
7. Certicom: Press Release – Certicom Announces Elliptic Curve Cryptosystem (ECC) Challenge Winner, 1997
   `http://www.certicom.com`
8. Lenstra, A. K., Verheul, E. R.: Selecting Cryptographic Key Sizes. Journal of Cryptology: the journal of the International Association for Cryptologic Research, 2001
9. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation, Vol. 48, 1987
10. Weimerskirch, A., Westhoff, D.: Identity Certified Authentication for Ad-hoc Networks. 10th Workshop on Security of Ad Hoc and Sensor Networks, 2003
11. Balfanz, D., Smetters, D., Stewart, P., Wong, H.: Talking to strangers: Authentication in adhoc wireless networks. Symposium on Network and Distributed Systems Security, 2002
12. Anderson, R., Bergadano F., Crispo, B., Lee, J., Manifavas C., Needham R.: A New Family of Authentication Protocols. ACMOSR: ACM Operating Systems Review, 1998
13. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. Proceedings of CRYPTO 84 on Advances in cryptology, 1985
14. Federal Information Processing Standards Publication 186: Digital Signature Standard (DSS). National Institute of Standards and Technology, 1994
15. University of California Berkeley: Tiny OS Hardware Designs, 2004
   `http://www.tinyos.net/scoop/special/hardware`