Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. 61, *1*, pp. 84–88, Bucarest, 2016

CRYPTOGRAPH KEY DISTRIBUTION WITH ELLIPTIC CURVE DIFFIE-HELLMAN ALGORITHM IN LOW-POWER DEVICES FOR POWER GRIDS

RADEK FUJDIAK[1], JIRI MISUREC[1], PETR MLYNEK[1], LEONARD JANER[2]

Key words: Low-power device, Cryptography, Elliptic curves, Finite field, Prime field, Binary field, Key distribution, Power grid, Diffie-Hellman algorithm.

Power grid networks, that use symmetric ciphers for secured communication, need some system for key distribution. It might be a special secured channel or a method which allows key-distribution via public channel. The public channel method could be done with asymmetric ciphers, but the growing computing and power requirements of modern ciphers are problematic for low-power devices, which are used in this kind of networks and which should provide sufficient security for communication. This article deals with the implementation of the Diffie-Hellman algorithm over elliptic curves for ultra-low-power devices, used in power grid and smart grid networks. The algorithm uses elliptic curves over the prime field. Our method might be used for key-distribution via public channels without any other equipment or devices.

1. INTRODUCTION

Elliptic curve cryptography (ECC) was first introduced by N. Koblitz [1] and V. Miller [2]. ECC is today an independent scientific field, which starts from asymmetric cryptography (the public key cryptography). The security of ECC is based on the elliptic curve discrete logarithm problem (ECDL) [3]. The plane curve over a finite field (Galois field) algebra is used (more about finite field algebra in [4]). Compared with public-key cryptography the ECC offers smaller key-size with the same security level (Table 1 [5]). This is an advantage in particular for applications with limited resources.

Table 1
ECC and Rivest-Shamir-Adleman algorithm (RSA) key-size NIST comparison

Symmetric key-size [bits]	ECC key-size [bits]	Asymmetric key-size [bits]
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

There exist many software implementations and libraries of ECC for devices with non-limited resources (Crypto++ [6], OpenSSL [7], Bouncy castle [8], FlexiProvider [9], and many others), but those with limited resources are in a different situation. Different platform architectures, limited memory and limited computing power are usually the main problem as regards compatibility with common libraries. The works in this field concentrate on ECC tests on 8/16/32 bit [10–12] low-power microcontrollers. These works mostly provide solution for low-size key with ECC. Current work [13, 14] concentrate on the new Curve25519 (more about this curve in [15]). These works try to provide the fastest possible solution or implementation, which might be used for low-power devices. This could be a solution for networks or places where the time is crucial (sensor networks, wireless networks) and where, for example, key-establishment is performed repeatedly. Power grid networks only need sufficient speed, but here the memory requirements or the final memory requirements of the implementation with a concrete cryptosystem are also very important, because the memory is also used for the data correction, communication protocols, data collection, etc.

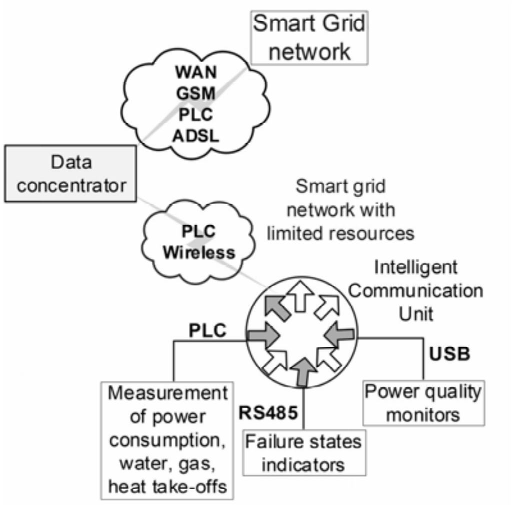
Smart Grid network
WAN
GSM
PLC
ADSL
Data concentrator
Smart grid network with limited resources
PLC
Wireless
Intelligent Communication Unit
PLC
USB
Measurement of power consumption, water, gas, heat take-offs
RS485
Failure states indicators
Power quality monitors

Fig. 1 – Power grid network for remote data acquisition [21].

In [10–14] the memory tests are missing, the same as tests with a concrete cryptosystem. In our article, we focus on the security between the measuring point and the data concentrator in the power grid network. Figure 1 shows our concrete network, where the Intelligent Communication Unit belongs to the MSP430 low-power devices family (PLC – power line communication, WAN – wide area network, GSM – mobile communication, ADSL – asymmetric digital subscriber line, RS485 – serial link). We try to implement with this microcontroller the concrete full-size secured elliptic curves from a different kind of standards (*e.g.* National Institute of Standards and Technology –

[1] Brno University of Technology, Department of Telecommunications, Faculty of Electrical Engineering and Communication, Czech Republic, xfujdi00@stud.feec.vutbr.cz.
[2] Escola Superior Politècnica, TCM1 planta 2 Despatx num. 9 / TCM3 planta 6 CTT, Fundació TecnoCampus Mataró-Maresme, 08302 Mataró, Spain

NIST [16], Standards for Efficient Cryptography – SEC [17], Standard for HASH function and signature – ANSI [18], Wireless transport layer security standards – WTLS [19]). These curves could be used for Advanced Encryption Standards (AES) key-establishment with the elliptic curve Diffie-Hellman cryptosystem (ECDH) [20]. We provide experimental measurement from the time and the memory points of view.

Our previous work has provided a random number generator for cryptography on low-power devices [22], big number representation and arithmetic for low-power devices [23] and also elliptic curves over binary field for low-power devices [24]. This article shows our latest work with prime field curves and also a comparison with our previous work with binary field curves. The article is divided into the following parts: experimental background and presets (where we give all the necessary information that was used for our experiment and implementation), experimental results (commented results are shown), discussions (answering the impact value question of our research), and conclusions.

2. EXPERIMENTAL BACKGROUND AND PRESETS

We used the MSP430f5438A ultra-low-power micro-controller from Texas Instruments. Its strong points are real-time capability with ultra-low-power consumption, digitally controlled oscillator (DCO) stability, stack processing capability, application operating modes, no external crystal need, low wake-up time (less than 5 μs), 16-bit operations, up to 32 MHz crystals, 32-bit hardware multiplier, 256 Kb FLASH, 16 kB RAM and 16-bit registers (all these are the main attributes for our problematic more about MSP430 in [25]). The microcontroller presets for the experiment were: the digital controlled oscillator was used as the source for CPU. We used default DCO frequency ~1 MHz. That means ~100 ns for one single cycle ($T_{\mathrm{cycle}} = 1/f_{\mathrm{CPU}}$). V_{cc} was 3000 mV and the operating mode was active mode (AM). The I_{cc} was 300 μA for our V_{cc} and f_{CPU}.

Texas Instruments has its own development software for their devices based on eclipse core, namely the code composer studio (CCS). We worked with CCS version 5.2. CCS[*] also has its own options for software compilation, which impact on the final results of measurement. Our presets for CCS were as follow: for the processor setup we used the mspx silicon version (20 bits wide for registers and buses), the application binary interface coffabi (the coffabi shows in the settings better memory requirements in tens of a percentage point compared with the eabi binary interface), large model for data and code (this allows placing the data and code anywhere in flash memory) while the indication of near data was not used. The debug model was a full symbolic debug and the full optimization in the presence of debug directives was on (these settings allow good debugging, but they could also impact a little bit on the size or the speed). No special software optimization options or advanced optimization options were used. The *msp430f5438.h* library was used for the registers and basic microcontroller settings.

The elliptic curves (EC) over finite field were chosen for their relatively simple computation and small key-size, which is critical for low-power devices such as MSP430. The elliptic curve cryptography mostly uses finite fields. There are two basic types of EC finite fields, *i.e.* prime field curves (GF_p) and binary field curves (GF_2m). The EC over binary field should be simpler than prime field curves, but with devices that have a good hardware multiplier and a good CPU the prime field curves should be much faster. Based on elliptic curves, the elliptic curve Diffie-Hellman was chosen as the key-distribution algorithm. The algorithm is based on discrete logarithm problem and has been developed for key-distribution purpose.

We worked with cryptographic OpenSSL libraries (*openssl-fips-ecp-2.0.6* [7], *openssl-fips-ecp-2.0.7* [7] and *openssl-1.0.1i* [7]). Their algorithms are written in the C Language (this was also a precondition for simpler imple-mentation and future compatibility among different micro-controller version and other platform versions (PC)). The concept, name of the headers, functions and files stayed the same or at least very similar. This should provide for a transparent code and also a better compatibility with the original libraries. The used libraries provide the following 27 prime-field curves [7]:

- F_{p112}: secp112r1, secp112r2, wap_wsg_idm_ecid_wtls6, wap_wsg_idm_ecid_wtls8.
- F_{p128}: secp128r1, secp128r2.F_{p160}: secp160k1, secp160r1, secp160r2, wap_wsg_idm_ecid_wtls7, wap_wsg_idm_ecid_wtls9.
- F_{p192}: secp192r1, secp192k1, X9_62_prime192v1, X9_62_prime192v2, X9_62_prime192v3.
- F_{p224}: secp224k1, secp224r1, secp224r2, wap_wsg_idm_ecid_wtls12.
- F_{p239}: prime239v1, X9_62_prime239v2, X9_62_prime239v3.
- F_{p256}: secp256k1, X9_62_prime256v1.
- F_{p384}: secp384r1.
- F_{p521}: secp521r1.

These curves are recommended by different kinds of the standards SECG (Standards for Efficient Cryptography Group), WTLS (Wireless Transport Layer Security), NIST (National Institute of Standards and Technology) or ANSI X9.62 (standard for HASH and signatures). All these curves were also implemented in the microcontroller. These curves use the non-adjective form of multiplication (*w-NAF* function), where it is first necessary to compute the multiplicand n of $Q = nP$ (Algorithm 1), where Q and P are points on the elliptic curve and the n is scalar multiplicand.

```
i = 0
while (n > 0) do
 if ((n mod 2) == 1) then
   $n_i$ = n mod $2^w$
   n -= $n_i$
 else
   $n_i$ = 0
 n /= 2
 i++
return ($n_{i-1}$, $n_{i-2}$, ... $n_0$)
```

Algorithm 1 – *w-NAF* algorithm (computation of *d* multiplicand)

The modulo function is defined as follow Algorithm 2:

```
if ((n mod $2^w$) >= ($2^w$-1))
  return ((n mod $2^w$) – 2)
 else
  return (n mod $2^w$)
```

Algorithm 2 – w-NAF algorithm (the modulo function definition)

[*] http://www.ti.com/tool/CCSTUDIO

and the *nP* point multiplication is in Algorithm 3:

```
Q = 0
for (j = i-1 down to 0) do
  Q *= 2
  if ($n_j$ != 0)
   Q = Q + $n_j$P
return Q
```

Algorithm 3 – *w-NAF* algorithm (final *dP* multiplication)

For the curve computation purpose, the three main different functions were implemented.

The first function is *ec_gfp_simple_meth*. It is a basic function for curve representation, it contains the simplest possible algorithms and it is also the lowest common denominator implementation for the prime field curves. Two other functions *ec_gfp_mont_meth* and *ec_gfp_nist_meth*, use this simple function as their starting point.

The *ec_gfp_mont_meth* adding the Montgomery multiplication and methods to the following functions:

- *group_init* (initialization function for curves)
- *group_finish* (ending function)
- *group_clear_finish* (clearing the group of curves)
- *group_copy* (copying the group of curves)
- group_set_curve (choosing the concrete curves)
- *field_mul* (field multiplication)
- *field_sqr* (field squaring)
- *field_encode* (field encoding)
- *field_decode* (field decoding)

and

- *field_set_to_one* (special function for internal purposes).

This Montgomery implementation should optimize the rather difficult computation functions.

The *ec_gfp_nist_meth* optimizes the computation process with the NIST standard. This function is only for the NIST standardized functions and it optimizes the following functions: *group_copy group_set_curve*, *field_mul* and *field_sqr*.

3. EXPERIMENTAL RESULTS

For low-power-devices the important attributes for real-event implementation are not only speed but also memory requirements. The speed impacts on the time that the microcontroller needs to be awake in the active mode. This is critical for the battery lifetime.

Another problem is that in the time of hard computation it is not possible to use the microcontroller for anything else (for example communication, data sending, *etc*.).

The memory size impacts on what we will still be able to implement into the microcontroller. If the memory is fully filled, it will not be possible to implement another mechanism for communication, data storage or any other purpose.

Table 2
Number of cycles for different kinds of curves (only one-side operation)

Curve Name	Type	Number of cycles (*gfp_mont*)	Number of cycles for whole ECDH	Heap Size [B]	Standard
secp112r1	F_{p112}	139 276	271 675	2 219	SECG/WTLS
secp112r2	F_{p112}	161 508	311 511	2 219	SECG
wap_wsg_idm_ecid_wtls6	F_{p112}	155 225	286 675	2 219	SECG/WTLS
wap_wsg_idm_ecid_wtls8	F_{p112}	154 206	348 474	2 219	WTLS
secp128r1	F_{p128}	171 116	313 053	2 219	SECG
secp128r2	F_{p128}	183 768	331 110	2 219	SECG
secp160k1	F_{p160}	213 400	375 573	2 425	SECG
secp160r1	F_{p160}	184 673	331 379	2 450	SECG
secp160r2	F_{p160}	197 593	331 520	2 470	SECG/WTLS
wap_wsg_idm_ecid_wtls7	F_{p160}	184 404	331 152	2 470	SECG/WTLS
wap_wsg_idm_ecid_wtls9	F_{p160}	277 533	510 505	2 470	WTLS
secp192k1	F_{p192}	335 016	411 531	2 550	SECG
X9_62_prime192v1 secp192r1	F_{p192}	359 441	451 296	2 650	NIST/X9.62/SECG
X9_62_prime192v2	F_{p192}	317 978	433 735	2 645	X9.62
X9_62_prime192v3	F_{p192}	342 049	441 291	2 665	X9.62
secp224k1	F_{p224}	435 447	540 010	2 800	SECG
secp224r1	F_{p224}	382 366	486 052	2 825	NIST/SECG
wap_wsg_idm_ecid_wtls12	F_{p224}	376 861	484 792	2 800	WTLS
X9_62_prime256v1 secp256r1	F_{p256}	506 953	611 219	2 905	X9.62/SECG

The requirements for memory include the big number arithmetic and representation, the cryptographic secure random number generator, the elliptic curve cryptography (prime and binary fields), the three computation methods (simple, NIST and Montgomery) and the ECDH cryptosystem. The RAM memory requirements are 4291 kB (26 %), the FLASH memory requirements are 58kB (23 %).

The RAM only contains the necessary parts of the program and configuration, because in our settings (large-code model) we use primarily the FLASH memory for data (non-text section as *.reset*, *.const*) and text (text section as code and C-assembled libraries).

The stack for the whole concept was constantly 1 kB. The heap size for the curves changed with the growing key-size (Table 2). The main measurement is given in Table 2. It is given the heap size, curve standardization, final speed requirements for single curves and also for the implemented ECDH cryptosystem with these curves.

The Table 2 is only for the *gfp_mont_meth*. The *gfp_simple_meth* should be used only for the simplest curves.

The *gfp_nist_meth* was also tested for the NIST curves but the attributes were mostly worse than the Montgomery implementation.

In the example with the secp224r1 curve, the computing requirement was 1.3 million cycles higher than with the *gfp_mont_meth* and the heap size requirement was 400 B lower (in general the NIST function is much slower and it has lower memory requirements).

The comparison of the heap size in our solution with binary field curves (for different key-sizes) is shown in Figure 2. The heap size requirements of the prime field are really similar compared with the binary field. The heap size would not be the main reason for choosing the prime field curves in the low power microcontroller. The time requirement comparison of the prime field with binary field curves (with different key-sizes) is shown in Figure 3.

The main parameters of the implemented prime field elliptic curve secp256r1 might be found in [16] as NIST P-256 curve.

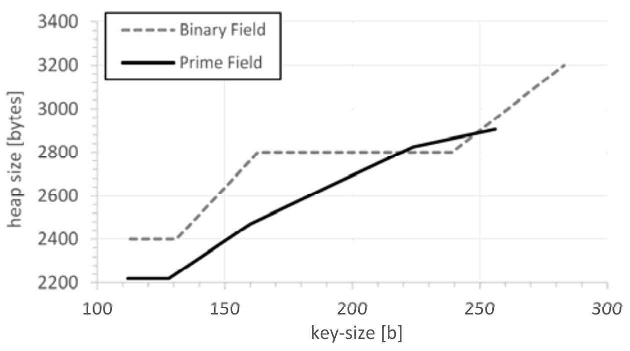

Fig. 2 – The heap size based on key-size for different kinds of field.

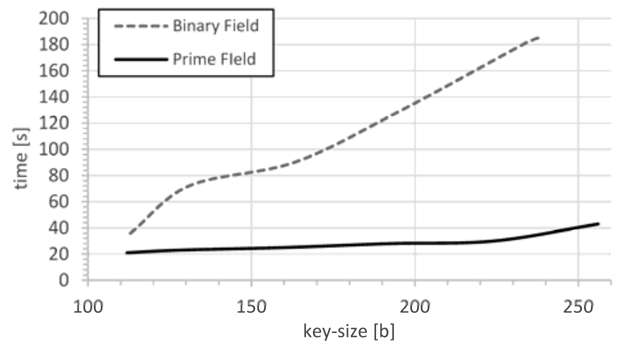

Fig. 3 – The time requirements based on key-size for different kinds of field.

4. DISCUSSION

The important attributes for the real events and environments are the speed and the memory requirements. As described previously the speed is important, because, for example, in power grids the speed of the cryptography algorithm for securing the communication impacts on the delay until the two sides can communicate.

The memory requirements, on the other hand, have an impact on the possibilities to implement communication protocols, etc.

Compared with binary field curves, our results with prime field implementation show better attributes, both in speed and also in memory requirements. Compared with other works, we provide secured 256-bit elliptic curve for ECDH (or a different algorithm). Our best resulted curve is standardized by the SECT and X.92 standards.

The 256 key-size of elliptic curve corresponds to the AES−128 (which were chosen for communication). The AES−128 should be sufficient till 2030+, the same as the implemented 256-bit curve [26].

Not many solutions with a 256-bit curve for limited resources are provided; for this problematic there exist two related works dealing with the 256-bit curve – Curve25519 and NIST P−256. Compared with the Curve25519 implementation with ca. 9 million cycles [15], our solution has 1.2 million cycles (double side, ECDH implementation).

Similar work to ours was done with the NIST P-256 prime elliptic curve, 1.1 million cycles were obtained with ephemeral key elliptic curve Diffie-Hellman (ECDH) (more in [27]). These speed results are comparable.

The related works did not provide any memory requirements that are critical for a real environment. Our solution still leaves 75 % of free memory. This should also leave sufficient space for other communication protocols, coding and other methods, which are necessary for communication.

5. CONCLUSIONS

The implementation of prime field curves was described and the measurements were provided. The prime field curves were also measured with ECDH implementation and tested in real network.

The comparison with current research shows comparable results, but we provide complete measurements from the time and memory points of view, suitable for the power grid network. The binary field curves show poor efficiency in the microcontroller with hardware multiplier compared with the prime field solution and it seems that the prime field curves should be the solution for low-power devices such as MSP430.

The future work will concentrate on lower memory requirements and also on curves with higher secure-bit key-size. We would like to provide highly secured curves for low-power devices also for the far future (later than 2030).

It will also be interesting to try the implementation of Curve25519 on the MSP430 with hardware multiplier.

ACKNOWLEDGMENTS

Research described in this paper was financed by the National Sustainability Program under grant No LO1401. For the research, the infrastructure of the SIX Centre was used.

Received on April 9, 2015

REFERENCES

1. N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, **48**, *177*, pp. 203 – 209, p.138, 1987.
2. V. Miller, *Use of elliptic curves in cryptography*, CRYPTO. ISBN 978-3-540-1646-0. Lecture note in Computer Science, **85**, pp. 417–426, 1985.
3. B. Oualid, *Elliptic Curve Discrete Logarithm*, GEMAN, **15**, *1*, 2013.
4. R.A. Raval, S.A. Patel, D.B. Patel, *Galois Field and Security for Online Data Storage in Cloud Computing*, IJESRT, **3**, *5*, pp. 424–427, 2014.

5. M. Muni Babu, S. Mp. Quebeb, V. Sunil Babu, *A Comparative study of elliptic curve cryptography and RSA to Kerberos authentication protocol*, International Journal of Advances in Science Engineering and Technology, **1**, *3*, pp. 43–45, 2014.
6. W. Dai, *Crypto+^{TM} Library: a Free C++ Class Library of Cryptographic Schemes*, 2013, http://www.cryptopp.com/.
7. OpenSSL, *OpenSSL Library: a Project of Open Source SSL and TLS protocols implementation*, https://www.openssl.org/., 2015.
8. The Legion of Bouncy Castle, *Java and C# Crypto Libraries*, Collection of APIs used in cryptography, 2014, https://www.-bouncycastle.org.
9. Theoretical Computer Science Research Group, *Flexiprovider: a powerful toolkit for the Java Cryptography Architecture (JCA/JCE)*, Technische Universitat Darmstadt, Germany, 2012, http://www. flexiprovider.de/.
10. N. Gura; A. Patel; A. Wander; H. Eberle, Ch. Shantz, *Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs*, Sun Microsystems Laboratories, Springer, Lecture Notes in Computer Science, **3156**, pp. 119–132, 2004.
11. E. Wenger, M. Werner, *Evaluating 16-bit Processors for Elliptic Curve Cryptography*, Springer, Lecture Notes in Computer Science, **7079**, pp. 166–181, 2011.
12. E. Wenger, T. Unterluggauer, M. Werner, *8/16/32 Shades of Elliptic Curve Cryptography on Embedded Processors*, Springer. Lecture Notes in Computer Science, **8250**, pp. 244–261, 2013.
13. G. Hinterwalder, A. Morad, M. Hutter, P. Schwabe, Ch. Paar, *Full-Size High-Security ECC Implementation on MSP430 Microcontrollers*, 2014.
14. P. Sasdrich, T. Guneysu, *Efficient Elliptic-Curve Cryptography using Curve 25519 on Reconfigurable Devices*, Reconfigurable Computing: Architectures, Tools, and Applications, Springer International Publishing, pp. 25–36, 2014.
15. D.J. Bernstein, *Curve25519: new Diffie-Hellman speed records,* Public Key Cryptography-PKC 2006, Springer Berlin Heidelberg, pp. 207–228, 2006.
16. NIST, *Recommended Elliptic Curves for Federal Government Use*, Technical Report, National Institute of Standards and Technology (NIST), 1999.
17. Certicom Corp, *SEC 1: Elliptic Curve Cryptography*, Certicom Research, Standard for efficient Cryptography, Ver. 2, May 2009.
18. American National Standard Institute, *The elliptic curve digital signature algorithm*, ANSI X9.62-1998, 1998.
19. WTLS, *Wireless application protocol, wireless transport layer security specification*, Wireless Application Forum, 1999.
20. NIST, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, NIST Special Publication 800–56 (Revised), 2007.
21. P. Mlynek, J. Misurec, M. Koutny and O. Raso, *Design of Secure Communication in Network with Limited Resources*, Proceedings of the 4th European Innovative Smart Grid Technologies (ISGT), 2013, pp. 1–5.
22. R. Fujdiak, P. Mlynek, J. Misurec, O. Raso, *Random Number Generator in MSP430 × 5xx Families*, Elektrorevue, **4**, *4*, pp. 70–74, 2013.
23. O. Raso, *Mod_arithm v1-01: a static library with mathematical computation functions for big numbers*, Brno University of Technology, Czech Republic, 2013.
24. P. Mlynek, O. Raso, R. Fujdiak, L. Pospichal, P. Kubicek, *Implementation of Elliptic Curve Diffie Hellman in Ultra Low Power Microcontroller*, Proceeding of the 37th International Conference on Telecommunications and Signal Processing (TSP), Berlin, Germany, pp. 267–271, 2014.
25. Texas Instruments, *Datasheet: MSP430F543x and MSP430F541x Mixer-Signal Microcontrollers*, Technical Documentation (SLAS612E), Aug. 2009 (revised Aug. 2014).
26. BlueKrypt, *The Cryptography Key Length Recommendation Project*, Feb. 2015, http://www.keylength.com/.
27. S. Gueron, V. Krasnov, *Fast Prime Field Elliptic Curve Cryptography with 256 bit Primes*, Journal of Cryptographic Engineering, Nov., pp. 1–11, 2014.