ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2011. No. 10(116)
ELEKTRONIKA IR ELEKTROTECHNIKA

T 120 SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Microprocessor Implementation of Key Agreement Protocol over the Ring of Multivariate Polynomials

A. Katvickis, E. Sakalauskas, N. Listopadskis
Department of Applied Mathematics, Kaunas University of Technology,
Studentu str. 50, Kaunas, Lithuania, phone: +370 37 300300, e-mails: arturas.katvickis@ktu.lt, eligijus.sakalauskas@ktu.lt, narimantas.listopadskis@ktu.lt

crossref http://dx.doi.org/10.5755/j01.eee.116.10.893

Introduction

Nowadays more and more day-by-day works and problems move to electronic and mobile environment, more and more services we can access on internet with various small devices, such as cell phones. Using these devices and applications, we faced to user authentication, data integrity and data confidentiality problems. Various cryptographic algorithms can be used to solve them. Traditional asymmetric cryptography algorithms, used for data integrity or key distribution, require high computational resources and thus, they are not suitable for small electronic devices wherein no specialized co-processors installed. In this case, low-cost cryptographic algorithms are needed.

Another aspect of cryptographic algorithms implementations in embedded systems is reduction of energy consumption. Such analyses were made in [1, 2].

Key agreement protocols (KAP) are one of the basic asymmetric cryptography algorithms. KAP allows two or more parties negotiate a common secret key using insecure communications.

The most widely used KAP algorithms are based on hard number theory problems such as discrete logarithm or integer factorization problems. These algorithms perform operations with large integers (for example 1024-bit integers) and such algorithm implementation request specialized co-processors to speed up computations.

On the other hand, a polynomial time algorithm for integer factorization and computation of discrete logarithm presented in 1999 [3]. This algorithm can be realised on quantum computers and thus these computers will create a potential thread to existing cryptographic algorithms in future.

In this paper, we present the implementation analysis of a KAP in computational resources restricted microprocessor's. This algorithm was proposed in [4]. The problem of effective realization of different cryptographic protocols is actual in embedded systems based on microprocessors.

Moreover, algorithm is based on solution of multivariate quadratic equations (MQ) system and it is believed that the solution of randomly generated MQ system is hopeless when system consists of $n \geq 80$ equations with $s \geq 80$ variables [5]. Moreover, it is proved that MQ system's solution is NP-Complete problem over any field.

Key agreement protocol

The protocol works with the set of matrices M of order m over the multivariate polynomials ring $Z_2[t_1, \ldots, t_p]$ with p arguments $t_1, \ldots, t_p$. The set $Z_2 = \{0, 1\}$ is the binary field with exclusive OR (XOR) addition operation and logical multiplication (AND) operation. These operations are the simplest bitwise operations existing in microprocessors and hence we expect that proposed KAP can be efficiently implemented in them.

The set M of these matrices is a non-commutative matrix ring which more formally can be denoted by $M(m, Z_2[t_1, \ldots, t_p])$. Let M_L and M_R are the subsets consisting of mutual commuting matrices. This means that for any $L_1, L_2 \in M_L$ and $R_1, R_2 \in M_R$ the following commuting condition holds $L_1L_2 = L_2L_1$, $R_1R_2 = R_2R_1$.

The KAP is performed by the following steps.

0. Alice and Bob agrees on publicly known data, used in further steps of KAP: matrices L, R and Q of order m over the multivariate polynomials ring $Z_2[t_1, \ldots, t_p]$. Users chose all these matrices randomly:

1. Alice randomly generates two secret matrix polynomials of degree k represented by the randomly chosen bit sequences $\{b_{xi}\}$, $\{b_{yi}\}$, $i = 0, 1, \ldots, k$ and computes two matrices:

$$X = \sum_{i=0}^{k} b_{xi} L^i = b_{x0} I + b_{x1} L + \ldots + b_{xk} L^k, \qquad (1)$$

$$Y = \sum_{i=0}^{k} b_{yi} R^i = b_{y0} I + b_{y1} R + \ldots + b_{yk} R^k . \quad (2)$$

Alice computes intermediate value $K_A = XQY$, and sends K_A to Bob.

2. Analogously, Bob randomly generates two randomly chosen secret bit sequences $\{b_{ui}\}$, $\{b_{vi}\}$, $i = 0, 1, \ldots, k$ and computes two matrices:

$$U = \sum_{i=0}^{k} b_{ui} L^i = b_{u0} I + b_{u1} L + \ldots + b_{uk} L^k , \quad (3)$$

$$V = \sum_{i=0}^{k} b_{vi} R^i = b_{v0} I + b_{v1} R + \ldots + b_{vk} R^k . \quad (4)$$

Bob computes intermediate value $K_B = UQV$, and sends K_B to Alice.

3. Parties compute the following common secret key K

$$K = XK_BY = UK_AV = XUQVY = UXQYV. \quad (5)$$

The last identity holds since $X, U \in M_L$ and $Y, V \in M_R$.

The publicly known security parameters are integers m, p and k. The public key are matrices $Q \in M$, $L \in M_L$ and $R \in M_R$. All publicly known matrices should be non-invertible. The private key are matrices $X \in M_L$, $Y \in M_R$ (Alice) and $U \in M_L$, $V \in M_R$ (Bob).

Implementation analysis

This algorithm performs operations with matrices of multivariate polynomials in $Z_2[t_1, \ldots, t_p]$. Every polynomial can be represented as a bit string of length 2^p. This bit string can be interpreted as unsigned 2^p-bit integer. This fact gives us some advantages. First, we can use standard data type (i.e. unsigned integer) for storing one multivariate polynomial in computer memory. Secondly, it leads to wide range of implementation in any kind of 8, 16 or 32-bits microprocessors without any hardware accelerators, special co-processors for arithmetic operations with large integers as take place in the case of traditional El Gamal or Elliptic Curve based KAP. In 8-bits microprocessor can be used multivariate polynomial ring $Z_2[t_1, t_2, t_3]$, in 16-bits microprocessors – ring $Z_2[t_1, t_2, t_3, t_4]$ and in 32-bits microprocessor – ring $Z_2[t_1, t_2, t_3, t_4, t_5]$.

All operations (addition and multiplication) between two multivariate polynomials in $Z_2[t_1, \ldots, t_p]$ can be performed using simple bit operations (bitwise XOR, bitwise AND, SHIFT).

Let us consider a toy example: multivariate polynomial representation in $Z_2[x, y]$. Every polynomial has general form $\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy$, $\alpha_i \in \{0, 1\}$ and corresponds to bit string $\alpha_0\alpha_1\alpha_2\alpha_3$.

Addition of polynomials $(\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy)$ and $(\beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy)$ can be applied as a bitwise XOR between $(\alpha_0\alpha_1\alpha_2\alpha_3)$ and $(\beta_0\beta_1\beta_2\beta_3)$.

Two polynomials multiplication is more complicated – every term of first polynomial has to be multiplied with every term of second polynomial. Grouping similar terms allow us to rewrite multiplication in this way

$$(\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy)(\beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy) =$$
$$= \alpha_0(\beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy) + \alpha_1((\beta_0 + \beta_1)x + (\beta_2 + \beta_3)xy) +$$
$$+ \alpha_2((\beta_0 + \beta_2)y + (\beta_1 + \beta_3)xy) + \alpha_3(\beta_0 + \beta_1 + \beta_2 + \beta_3)xy. \quad (6)$$

Depending on a term of first polynomial, the different result of multiplication takes place, i.e. with a term α_0 multiplication result gives all terms and with a term $\alpha_3 xy$ – just one term. Hence the multiplication operation should be supplied by conditional operation in the pseudo-code presented below.

Then, multiplication of two polynomials $(\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy)$ and $(\beta_0 + \beta_1 x + \beta_2 y + \beta_3 xy)$ can be written in pseudo-code as

```
result = 0000;
if α0 = 0
    result = result XOR β0β1β2β3
if α1 = 1
    result = result XOR (((β0β1β2β3) XOR (SHIFT1
(β0β1β2β3))) AND 0011)
if α2 = 1
    result = result XOR ((β0β1β2β3) XOR (SHIFT2
(β0β1β2β3))) AND 0011
if α3 = 1
    temp = (β0β1β2β3) XOR (SHIFT1 (β0β1β2β3) );
    result = result XOR ((temp XOR (SHIFT2 temp))
AND 0001)
```

To complete multiplication it has to be done 4 conditional operations mentioned above and this number is comparable with 4 SHIFT, 3 bitwise AND, 7 bitwise XOR operations (in average, we have to half these numbers).

Analogously, multiplication operation can be generalized to other multivariate polynomial rings over Z_2 with a larger number of variables.

In summary, polynomial multiplication in $Z_2[t_1, \ldots, t_p]$ request 2^p conditional operations, $(p+2)2^{p-1}$ bitwise XOR, $p2^{p-1}$ SHIFT and 2^p-1 bitwise AND operations.

The total number of operations increases then the number of matrix elements increases and there are $m(m-1)$ extra bitwise XOR operations for two matrices of order m multiplication over $Z_2[t_1, \ldots, t_p]$. Thus, in total for multiplication of matrices of order m over $Z_2[t_1, \ldots, t_p]$ is needed to perform $m^2 2^p$ conditional operations, $(p+2)m^2 2^{p-1} - m(m-1)$ bitwise XOR, $pm^2 2^{p-1}$ SHIFT and $(2^p-1)m^2$ bitwise AND operations.

Comparison of efficiency with Diffie-Hellman and Elliptic Curve KAP

We choose the security parameters values of our KAP proposed in [4]. These parameters are matrices order m, number of multivariate polynomial indeterminates p and secret bits string length k. For 32-bits microprocessor we choose the following values $m = 4$, $p = 5$, $k = 255$. For 16-bits microprocessor parameters values are $m = 6$, $p = 4$, $k = 255$, and for 8-bits microprocessor they are $m = 9$, $p = 3$, $k = 255$.

The total number of operations to perform two matrices multiplication, in case of 8-, 16- and 32-bits microprocessor implementation, presented in the Table 1 below. Counting total number of operations, we assumed

that one conditional operation performs at the same cost as two bitwise operations, i.e. are taking two microprocessors clock cycles.

Table 1. Number of operations depends on microprocessor

Operation	8-bits	16-bits	32-bits
XOR	3096	3516	3608
SHIFT	1944	2304	2560
AND	1134	1020	992
conditional	648	576	512
total	7470	7992	8184

Diffie-Hellman (DH) KAP was presented in [6]. In this protocol, users have to perform two exponentiation operations with large integers to obtain common secret key.

Another traditional KAP is Elliptic Curve DH (ECDH) [7]. This protocol performs operations with elliptic curve over field GF(p) or GF(2^n) points (points addition and doubling). ECDH uses smaller key size and operates with smaller integers than DH KAP at the same security level. Therefore, ECDH is faster and requires less computational resources than DH KAP.

Both DH and ECDH use arithmetic with large integers. To speed up an arithmetic operation the specialized co-processors is used hence increasing the cost of realization.

In our protocol, users have to perform four matrix multiplication operation over $Z_2[t_1, \ldots, t_p]$. Thus, we can compare two matrices multiplication with one modular exponentiation or one elliptic curve point multiplication.

As we can see from Table 1, total number of operations is roughly the same ($\sim 8 \cdot 10^3$) for all implementations.

It is knows that modular exponentiation with large integers can be performed with the repeated square-and-multiply algorithm. Bit complexity of this algorithm is $O((\lg n)^3)$ [8]. To reduce computation cost it is recommended to use smaller exponent size. In case of 1024-bits integer recommended exponent size is 160 bits [9]. Then, modular exponentiation will require about 10^6 simple bit operations and that is hundreds of times more than total number of simple bits operations in case of KAP with matrices over $Z_2[t_1, \ldots, t_p]$.

ECDH with 160 bits keys produce the same security level as DH with 1024 bits keys [10].

To compute elliptic curve point multiplication with 160-bits integers, it is needed to perform 160 point doubles and on average 80 point addition. To perform modular exponentiation of 160 bits exponent size with repeated square-and-multiply algorithm it is needed to perform 160 modular squarings and on average 80 modular multiplications. Since, we can compare one elliptic curve operation with one modular multiplication.

The dominant costs in an elliptic curve point doubling are two field multiplications and one field inversion. The cost of a point-add is roughly the same. Field inversion costs about 2.5 field multiplications [8]. Therefore, for a rough estimate let us say that an elliptic curve point operation costs us the same as five 160 bits integers modular multiplication.

The bit complexity of modular multiplication is $O((\lg n)^2)$ [8]. Thus, a 1024-bits modular multiplication takes roughly $(1024/160)^2 = 40$ times as long as a 160-bits modular multiplication. Since an elliptic curve point operation requires 5 modular multiplications, we expect that a 1024-bit modular multiplication will cost 40/5 = 8 times as much as a 160-bits elliptic curve operation. Squaring can be a little faster than modular multiplication, but not more than twice as fast and expected that ECDH is about 5 times faster than DH KAP.

We consider protocol implementation in ordinary 8-bits and 16-bits microprocessors.

To compare KAP over multivariate polynomial ring to DH and ECDH we used data from [11], [12].

In reference [11], elliptic curve cryptography (ECC) is compared to RSA on 8-bit Atmel ATmega128 processor clocked at 8 MHz.

In reference [12], practical implementation of ECC over GF(p) is described on 16-bit single-chip microcomputer M16C designed by Mitsubishi Electric Corporation clocked at 10 MHz.

Assuming exponentiations time is linear to exponents bit size and knowing RSA-1024 execution time with public key $2^{16}+1$ (17 bits) we say that DH-1024 should take at least about 9 times longer.

The expected time of our KAP realization in different processors is presented in Table 2.

Table 2. Comparison of KAPs execution times

Algorithm	Execution time in seconds	
	8-bit Atmel ATmega128	M16C
ECC-160	0.81	0.15
RSA-1024 public key $2^{16}+1$	0.43	0.4
DH-1024	~ 4	~ 3.6
Our KAP	~ 0.04	~ 0.03

Conclusions

The efficiency analysis of KAP over the multivariate polynomial ring is presented. The secure protocol parameters are defined and secure they values are determined depending on implementation..

The effective realization with respect to computation time is considered. The number of simple microprocessor operations is estimated and expected realization time is computed for 8-bit and 16-bit microprocessors.

The comparison between DH and ECDH KAPs is presented for the same microprocessors. Comparison with DH and ECDH KAP shows, that KAP over the ring of multivariate polynomials is suitable for highly resource-restricted areas, where there are no specialized co-processors required and low-cost cryptographic algorithms are needed.

References

1. **Toldinas J., Štuikys V., Ziberkas G., Naunikas D.** Power Awareness Experiment for Crypto Service–Based Algorithms // Electronics and Electrical Engineering. – Kaunas: Technologija, 2010. – No. 5(101). – P. 57–62.
2. **Toldinas J., Stuikys V., Damasevicius R., Ziberkas G., Banionis M.** Energy Efficiency Comparison with Cipher

Strength of AES and Rijndael Cryptographic Algorithms in Mobile Devices // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 2(108). – P. 11–14.
3. **Shor P. W.** Polynomial–Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer // SIAM J.Sci.Statist.Comput., 1999. – No. 41(2). – P. 303–332.
4. **Sakalauskas E., Katvickis A., Dosinas G.** Key Agreement Protocol over the Ring of Multivariate Polynomials//, Information Technology And Control. – Kaunas: Technologija, 2010. – Vol. 39. – No. 1. – P. 51–54.
5. **Faugere J.–C., Joux A.** Algebraic Cryptanalysis of Hidden Field Equations (HFE) Cryptosystems Using Grobner Bases // Crypto, 2003. – P. 44–60.
6. **Diffe W., Hellman M.** New Directions in Cryptography, // In IEEE Transaction on Information Theory, 1976. – Vol. IT–22. – P – 644–654.
7. **Koblitz N.** Algebraic Aspects of Cryptography. – Berlin Heildelberg, Springer–Verlag, 1999.
8. **Menezes A., van Oorschot P. and Vanstone S.** Handbook of applied Cryptography. – 2001. Online: http://www.cacr.math.uwaterloo.ca/hac/.
9. **Raymond J.–F., Stiglic A.** Security Issues in the Diffie–Hellman Key Agreement Protocol. – 2000. Online: http://crypto.cs.mcgill.ca/~stiglic/Papers/dhfull.pdf.
10. **NIST.** Recommendation for Key Management, Special Publication 800–57 Part 1 Rev. 3. – 2011. Online: http://csrc.nist.gov/groups/ST/toolkit/documents/SP800–57Part1–Revision3_May2011.pdf
11. **Gura N., Patel A., Wander A., Eberle H., Shantz S. C.** Comparing Elliptic Curve Cryptography and RSA on 8–bit CPUs // Proc. Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, LNCS 3156, 2004. – P. 119–132.
12. **Hasegawa T., Nakajima J., Matsui M.** A Practical Implementation of Elliptic Curve Cryptosystems over *GF*(*p*) on a 16–Bit Microcomputer // PKC '98 Proc. of the 1st International Workshop on Practice and Theory in Public Key Cryptography. – Springer–Verlag London, 1998. – P. 182–194.

Received 2011 05 30
Accepted after revision 2011 10 10

A. Katvickis, E. Sakalauskas, N. Listopadskis. Microprocessor Implementation of Key Agreement Protocol over the Ring of Multivariate Polynomials // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 10(116). – P. 95–98.

Efficiency analysis of key agreement protocol with matrices over multivariate polynomial ring is presented. We consider protocol implementation for highly resource-restricted areas. Total number of simple bit operations requires computing common secret key is determined for different microprocessors types and estimated expected protocol run time. Comparison to traditional key agreement protocols (Diffie-Hellman and Elliptic Curve Diffie-Hellman) shows that key agreement protocol over multivariate polynomial ring is more suitable for areas where there are no specialized co-processors installed. Bibl. 12, tabl. 2 (in English; abstracts in English and Lithuanian).

A. Katvickis, E. Sakalauskas, N. Listopadskis. Rakto apsikeitimo protokolo virš baigtinio kelių kintamųjų daugianarių žiedo vykdymas mikroprocesoriuose // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2011. – Nr. 10(116). – P. 95–98.

Darbe pateikta rakto apsikeitimo protokolo virš baigtinio kelių kintamųjų daugianarių žiedo efektyvumo analizė. Nagrinėjama, kaip vykdomas protokolas srityse su ribotais skaičiavimo ištekliais. Įvertintas bazinių pobitinių operacijų skaičius, reikalingas protokolui vykdyti, atsižvelgiant į galimas realizacijas skirtinguose mikroprocesoriuose. Atliktas nagrinėjamo rakto apsikeitimo protokolo palyginimas su tradiciniais rakto apsikeitimo protokolais. Bibl. 12, lent. 2 (anglų kalba; santraukos anglų ir lietuvių k.).