Discrete Applied Mathematics 216 (2017) 622–626

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Permutation polynomials and a new public-key encryption

Gurgen Khachatrian [a,*], Melsik Kyureghyan [b]

[a] American University of Armenia, Yerevan, Armenia

[b] Institute for Informatics and Automation Problems, National Academy of Sciences of Armenia, Yerevan, Armenia

ARTICLE INFO

Article history:
Received 13 February 2015
Accepted 1 September 2015
Available online 21 October 2015

Dedicated to the memory of Levon Khachatrian

Keywords:
Permutation polynomials
Public-key encryption
White box reduction

ABSTRACT

In this paper a new public key encryption system based on permutation polynomials is developed. The permutation polynomial $P(x)$ is declared to be a public polynomial for encryption. A public key encryption of given $m(x)$ is the evaluation of polynomial $P(x)$ at point $m(x)$ where the result of evaluation is calculated via so called White box reduction, which does not reveal the underlying secret polynomial $g(x)$. It is shown that for the new system to achieve a comparable security with conventional public key systems based on either Discrete logarithm or Integer factorization problems, substantially less processing length n is required resulting in a significant acceleration of public key operations.

1. Introduction

Let $GF(q)$ be the finite field with q elements, where q is a prime or power of a prime. A polynomial $f(x)$ over $GF(q)$ is called a permutation polynomial if an equation $f(x) = r$ for any $r \in GF(q)$ has only one root in $GF(q)$. Permutation polynomials have been studied intensively in the past (see for example [2,3,6,7]) and have important applications in coding theory [2] and cryptography [6]. In this paper we will introduce a new class of permutation polynomials and will show how they can be used to design a public key system. Public-key cryptography started in 1976 with the publication of pioneering work of Diffie and Hellman [1] in which DH key exchange was presented, and in 1978 with another fundamental work by Rivest, Shamir and Adleman [5] called RSA cryptosystem. DH key exchange is based on discrete logarithm problem (DLP) and RSA is based on integer factorization problem. The current state of the art requires that public modulus size must be 2048 bits. Another important development for public key cryptosystems was the invention of Elliptic curve cryptosystems [4] which are also based on DLP problem, but require significantly less number of modular size, but more complex operations. In this paper a novel public key encryption system based on binary permutation polynomials is presented and its security relays on the problem of solving a polynomial equation over $GF(2^n)$ when the field representation polynomial is unknown.

The paper is organized as follows: In Section 2 a new method of construction of permutation polynomials is introduced. In Section 3 a new public key encryption system based on permutation polynomials constructed in Section 2 is presented. In Section 4 White box implementation of polynomial evaluation is represented. In Section 5 a security analysis of the proposed system is given. In Section 6 implementation aspects of the proposed system are discussed. Section 7 concludes the paper.

2. New Construction method of permutation polynomials

Let $GF(q)$ be a finite field of characteristic p. For every permutation polynomial $f(x)$ of $GF(q)$, there exists a unique polynomial $f^{-1}(x)$ of $GF(q)$ such that $f(f^{-1}(x)) = f^{-1}(f(x)) = x$ called the compositional inverse of $f(x)$.

* Corresponding author.
E-mail addresses: gurgenkh@aua.am (G. Khachatrian), melsik@ipia.sci.am (M. Kyureghyan).

http://dx.doi.org/10.1016/j.dam.2015.09.001
0166-218X/

Definition 2.1. A polynomial of the form $P(x) = \sum_{i=0}^{n} a_i x^{q^i}$ with coefficients in an extension field $GF(q^n)$ of $GF(q)$ is called a q-polynomial or linearized polynomial over $GF(q^n)$.

Theorem 2.2 ([3, Theorem 7.9]). *Let $GF(q)$ be finite field of characteristic p. Then the polynomial $P(x) = \sum_{i=0}^{n} a_i x^{p^i}$, $a_i \in GF(q)$ is a permutation polynomial of $GF(q)$ if and only if $P(x)$ has only one root equal to 0 in $GF(q)$.*

Theorem 2.3 ([7]). *Let $F(x) = \sum_{u=0}^{n} a_u x^u \in GF(q)$ and $P(x) = \sum_{u=0}^{n} a_u x^{q^u}$ be its linearized q-associate. Then the polynomial $x^{-1}P(x) = \sum_{u=0}^{n} a_u x^{q^u-1}$ is irreducible over $GF(q)$ if and only if $F(x)$ is a primitive polynomial over $GF(q)$.*

Based on Theorem 2.3 we propose a new construction method of permutation polynomials of $GF(q^n)$. Let $F(x) = \sum_{u=0}^{n} a_u x^u$, $a_u \in GF(q)$ be a primitive polynomial. In accordance with Theorem 2.3 it can be concluded that polynomial $x^{-1}P(x) = \sum_{u=0}^{n} a_u x^{q^u-1}$ is an irreducible polynomial. Hence $P(x) = \sum_{u=0}^{n} a_u x^{q^u}$ has only one root equal to 0 in $GF(q^n)$. According to Theorem 2.2 $P(x)$ will be a permutation polynomial of $GF(q^n)$.

Our main focus will be permutation polynomials of $GF(2^n)$. Let $F(x) = \sum_{u=0}^{n} a_u x^u$ be a primitive polynomial over $GF(2)$, then $P(x) = \sum_{u=0}^{n} a_u x^{2^u}$ will be a permutation polynomial of $GF(2^n)$. Elements of $GF(2^n)$ will be represented through a primitive polynomial $g(x)$ over $GF(2)$. Now we will describe an algorithm for finding an inverse polynomial of $P(x)$ denoted by $P^{-1}(x) = \sum_{u=0}^{n} \lambda_u x^{2^u}$.

Note that

$$x^{2^n} \equiv x \bmod g(x) \tag{2.1}$$

then we will have

$$\begin{aligned} P^{-1}(P(x)) &= \sum_{v=0}^{n} \lambda_v \left(\sum_{u=0}^{n} a_u x^{2^u} \right)^{2^v} \bmod g(x) \\ &= \sum_{v=1}^{n-1} \lambda_v \left(\sum_{u=1}^{n-1} a_u x^{2^u} \right)^{2^v} \bmod g(x) \\ &= \sum_{v=1}^{n-1} \lambda_v \left(\sum_{u=1}^{n-1} a_u x^{2^{u+v}} \right) \bmod g(x). \end{aligned}$$

Let

$$\varphi_v(x) = \sum_{u=1}^{n-1} a_u x^{2^{u+v}} \bmod g(x), \quad v = 1, 2, \ldots, n-1.$$

It is easy to see that

$$\varphi_i(x) \equiv \varphi_{i-1}^2(x) \bmod g(x) = \varphi_1^{2^{i-1}}(x) \bmod g(x) \tag{2.2}$$

for $i = 2, \ldots, n-1$. Then

$$\begin{aligned} \sum_{v=1}^{n-1} \lambda_v \left(\sum_{u=1}^{n-1} a_u x^{2^{u+v}} \right) \bmod g(x) &= \sum_{v=1}^{n-1} \lambda_v \left(\sum_{u=1}^{n-1} a_u x^{2^{u+1}} \bmod g(x) \right)^{2^{v-1}} \bmod g(x) \\ &= \sum_{v=1}^{n-1} \lambda_v \varphi_v(x). \end{aligned}$$

Now we will describe how to find $\varphi_1(x)$. Implying that $u+1 = ln + r$ and (2.1) we will have $x^{2^{u+1}} = x^{2^{ln+r}} = x^{2^r}$ where $0 \le r \le n-1$. Let $R_0(x) \equiv x \bmod g(x)$, $R_1(x) \equiv x^2 \bmod g(x)$, $R_2(x) \equiv x^{2^2} \bmod g(x)$, $\ldots$, $R_{n-1}(x) \equiv x^{2^{n-1}} \bmod g(x)$. Hence

$$\varphi_1(x) \equiv \sum_{u=1}^{n-1} a_u x^{2^{u+1}} \bmod g(x) = \sum_{r=1}^{n-1} a_r x^{2^r} \bmod g(x) = \sum_{r=1}^{n-1} a_r R_r(x).$$

By (2.2) we can find all $\varphi_v(x)$ for $v = 2, \ldots, n-1$.

According to the definition of the compositional inverse of the permutation polynomial

$$P^{-1}(P(x)) = \sum_{v=1}^{n-1} \lambda_v \varphi_v(x) \bmod g(x) \equiv x \bmod g(x). \tag{2.3}$$

Then in order to determine λ_i, $i = 1, 2, \ldots, n-1$ coefficients the system of linear equations of $n-1$ variables should be solved based on the equality (2.3).

3. New public key encryption scheme based on permutation polynomials

Let a primitive polynomial $g(x)$ of degree n over $GF(2)$ be the public key encryption private (secret) parameter and a permutation polynomial $P(x)$ be the public key encryption public parameter.

(a) Public key encryption: any message $m(x)$ of the length n as an input (plaintext) and evaluation of the polynomial $P(m(x)) = c(x) \ mod \ g(x)$ as an output (ciphertext). The evaluation operation will be implemented via "White box" evaluation without revealing the polynomial $g(x)$. The output of public key encryption will be $c'(x)$ based on "White box" tables explained in the next section.
(b) Private key decryption: Given a ciphertext $c'(x)$ calculate $c(x)$. Compute $P^{-1}(c(x)) = P^{-1}(P(m(x))) = m(x) \ mod \ g(x)$.

4. White box implementation of polynomial evaluation

The main purpose of the white box polynomial evaluation is to calculate an evaluation result of given polynomial by the client in such a way, that only the "owner" of the scheme can correctly obtain an evaluation result without revealing modulo reduction polynomial $g(x)$. We will consider as a primary setting the degree of polynomial to be equal to 128. There are approximately 2^{120} primitive polynomials of degree 128, as such that makes brute force of all polynomials infeasible. Let t be the weight of $P(x)$. Then the evaluation result for any $P(m(x))$ before modulo $g(x)$ reduction will have not more than $t \times 127$ nonzero members. The evaluation procedure will be as follows: all possible residues $x^N \equiv R_N(x) \ mod \ g(x)$ for $N = 2^i r$, where $i = 1, \ldots, 128$, $r = 2k + 1$, $k = 0, 1, \ldots, 63$ are biased by using random 64 secret polynomials based on another secret polynomial $L(X)$ for example they can be any residues modulo a primitive polynomial $L(X)$ denoted as $L_0, L_1, \ldots, L_{63}$ which are only known to the "owner" of the system. All biased values for residues modulo polynomial $g(x)$ are provided to the public in the following manner:

$$B_N(x) = ((R_N(x) \times L_0(x)) \ mod \ L(x)) \oplus L_{k+1}(x) \tag{4.1}$$

for any $N = 2^i(2k + 1)$. Based on above explanation an encoding procedure will be as follows: The user calculates an evaluation result of the polynomial $P(m(x))$ without any reduction, takes the polynomial $R(x)$ that contains all terms of evaluation for the degrees not exceeding 127, and calculates the modulo two sum of nonzero terms $B_N(x)$ denoted by $\sum B_N(x)$ corresponding to nonzero terms of evaluation result exceeding $N = 127$.

An encrypted message $c'(x)$ then contains two 16 byte vectors including $R(x)$, $\sum B_N(x)$ and another 8 byte vector $B = (b_0, b_1, \ldots, b_{63})$, where $b_k = 0$ if the number of nonzero terms with the same k in evaluation result is even and $b_k = 1$ otherwise for $N = 2^i(2k + 1)$, $k = 0, \ldots, 63$. An Example how $B_N(x)$ and a vector $(b_0, \ldots, b_{63})$ are defined.

Let $P(x) = x + x^{16} + x^{64}$ and $m(x) = x^3 + x^9 + x^{19}$. We have that

$$\begin{aligned} P(m(x)) &= x^3 + x^9 + x^{19} + \left(x^3 + x^9 + x^{19}\right)^{16} + \left(x^3 + x^9 + x^{19}\right)^{64} \\ &= x^3 + x^9 + x^{19} + x^{48} + x^{144} + x^{192} + x^{304} + x^{576} + x^{1216}. \end{aligned}$$

We have that $\sum B_N(x) = B_{144}(x) \oplus B_{192}(x) \oplus B_{304}(x) \oplus B_{576}(x) \oplus B_{1216}(x)$, where $B_N(x)$ are defined according to (4.1). Also we have that nonzero terms exceeding 128 in $P(m(x))$ evaluation can be presented as $144 = 9 \cdot 16$, $192 = 3 \cdot 64$, $304 = 19 \cdot 16$, $576 = 9 \cdot 64$, $1216 = 19 \cdot 64$. Thus we have that for the vector $B = (b_0, b_1, \ldots, b_{63})$ in this case $b_1 = 1$ and $b_i = 0$ otherwise.

Decoding procedure by the "owner" of the system will be as follows: based on the value $B = (b_0, b_2, \ldots, b_{63})$ and vector $\sum B_N(x)$ the "owner" calculates:

$$R(x) \oplus \left(\sum B_N(x) \oplus \sum_{i=1}^{63} (b_i \times L_i(x))\right) \times (L_0(x))^{-1} = c(x) \ mod \ L(x). \tag{4.2}$$

Now we will prove the following lemma:

Lemma 4.1. *Decoding procedure according to the formula (4.2) gives a correct result of the ciphertext $c(x)$ which is the evaluation of $P(m(x)) \ mod \ g(x)$.*

Proof. The objective is to calculate the value of $c(x)$ based on public values $\sum B_N(x)$, the vector $B = (b_0, b_1, \ldots, b_{63})$ and $R(x)$ without revealing the value of the polynomial $g(x)$. According to the formula (4.1) each value of $R_N(x)$ is obfuscated by two polynomials $L_0(x)$ and $L_{k+1}(x)$. Note that $\sum B_N(x)$ is the modulo 2 sum of all nonzero coefficients in evaluation of $P(m(x))$ for N's exceeding 127. If the ith component of the vector B is 1 then the contribution of corresponding $L_{i+1}(x)$ vectors in $\sum B_N(x)$ will be just $L_{i+1}(x)$ otherwise it will be zero. In order to get $\sum R_N(x)$ from $\sum B_N(x)$ after removing respective values of $L_{i+1}(x)$ one will simply need to divide the obtained value by $L_0(x)$ which is reflected in formula (4.2). As a result we will get the value of $R(x) \oplus \sum R_N(x)$ which in fact is the correct result of evaluation of $P(m(x)) \ mod \ g(x)$. Thus Lemma 4.1 is proved.

After calculating $c(x) = P(m(x)) \ mod \ g(x)$ the owner of the system can decrypt the message: $P^{-1}(c(x)) = m(x)$, where P^{-1} is the compositional inverse of the polynomial $P(x)$.

5. Security analysis

As follows from the previous paragraph the decryption procedure is actually the determination of $c(x)$ from $c'(x)$, which in turn requires the knowledge of polynomials $g(x)$ and $L(x)$ as well as all values of $L_i(x), i = 0, \ldots, 63$. Next we will analyze how difficult will be to determine secret polynomials $L(x)$ and $g(x)$ as well as all values of $L_i(x), i = 0, \ldots, 63$ based on publicly available information $B_N(x)$.

Let us prove the following lemma:

Lemma 5.1. *For any given set $B_N(x)$ to uniquely determine polynomials $L_i(x), i = 0, \ldots, 63$ without the knowledge of the polynomial $g(x)$ would be computationally infeasible. Moreover for every possible $g(x)$ there will be as many possibilities for polynomials $L_i(x)$ as many different polynomials $g(x)$.*

Proof. Let us prove that for a given set $B_N(x)$ and a polynomial $g(x)$ polynomials $L_i(x)$ can be determined uniquely. For example let us assume that $i = 7, r = 1$. According to (4.1) we have that

$$B_{128}(x) = ((R_{128}(x) \times L_0(x)) \mod L(x)) \oplus L_1(x) \tag{5.1}$$

and

$$B_{256}(x) = \left(\left(\left(R_{128}(x) \times x^{128}\right) \mod g(x)\right) \times L_0(x)\right) \mod L(x)\right) \oplus L_1(x). \tag{5.2}$$

From (5.1) and (5.2) we get that

$$(B_{128}(x) \oplus B_{256}(x)) \times \left(x^{128} + 1\right)^{-1} = (R_{128}(x) \times L_0(x)) \mod L(x)$$

and

$$L_1(x) = B_{128}(x) \oplus (B_{128}(x) \oplus B_{256}(x)) \times \left(x^{128} + 1\right)^{-1},$$

$$L_0(x) = (B_{128}(x) \oplus L_1(x)) \times (R_{128}(x))^{-1}.$$

It should be noted that an operation $\left(x^{128} + 1\right)^{-1}$ is implemented modulo $g(x)$ but multiplication with $B_{128}(x) \oplus B_{256}(x)$ is implemented modulo $L(x)$. We use the fact that there is a unique reverse element for any polynomial modulo irreducible polynomial $g(x)$.

Without knowledge of the polynomial $g(x)$ it would be computationally infeasible to determine $\left(x^{128} + 1\right)^{-1}$ as well as $(R_{128}(x))^{-1}$.

Thus Lemma 5.1 is proved.

An option to attack the system is to try to remove unknown polynomials from the system of equation (4.1). For example based on equations $N = 128, 256$ and 512 we can get that

$$B_{512}(x) \oplus B_{128}(x) = \left(\left(\left(R_{128}(x) \times \left(1 + x^{384}\right)\right) \mod g(x)\right) \times L_0(x)\right) \mod L(x),$$

and

$$B_{128}(x) \oplus B_{256}(x) = \left(\left(\left(R_{128}(x) \times \left(1 + x^{128}\right)\right) \mod g(x)\right) \times L_0(x)\right) \mod L(x).$$

If we divide these equations by each other trying to remove all unknown polynomials we will get the equality

$$\frac{B_{512}(x) \oplus B_{128}(x)}{B_{128}(x) \oplus B_{256}(x)} = \frac{1 + x^{384}}{1 + x^{128}}. \tag{5.3}$$

However it should be noted that in this case left side of the equality (5.3) is modulo $L(x)$ while the term $\frac{1+x^{384}}{1+x^{128}}$ is modulo $g(x)$. This means it is still impossible to extract an information concerning to polynomial $g(x)$ from (5.3).

Given the set of $B_N(x)$ there exist a unique polynomials $g(x)$, $L(x)$ and polynomials $L_i(x), i = 0, \ldots, 63$ for which congruences of the type (4.1) are satisfied for all N's.

But due to Lemma 5.1 and the reasonings presented above the only way to find a right $g(x)$ is to brute force all primitive polynomials of degree 128, which is computationally infeasible to implement.

Summarizing our security analysis we can state the following:

(a) Determination of $R_N(x)$ from $B_N(x)$ by using relationships given by (4.1) will require the calculation of reverse elements both modulo $g(x)$ and modulo $L(x)$. As Lemma 5.1 states for each $g(x)$, and $B_N(x)$ we will get respective values for $R_N(x)$. This means that in order to get right values for $R_N(x)$ we will need to brute force all primitive polynomials of degree 128 which is infeasible to do.

(b) If we try to remove unknown coefficients based on polynomial $L(x)$ by dividing different appropriate equalities (4.1) we would end up with the situation when there is an equality with the right side modulo $L(x)$ and a left side modulo $g(x)$. Such kind of relation will not give any clue about polynomial $g(x)$ since polynomial $L(x)$ is also unknown. In this case the only option left is to brute force all primitive polynomials $L(x)$ in order to get an information regarding to polynomial $g(x)$.

Thus our analysis shows the infeasibility of calculating the correct value of $c(x)$. This fact provides one-wayness of the white box based encryption function. More detailed security analysis against chosen plaintext attack and other type of attacks will be provided in the followup paper.

6. Implementation aspects of the system

Public key encryption of the proposed system requires evaluation of the polynomial of weight t at any point of the field $GF(2^N)$ regarded as a polynomial of degree less than N with no modular reduction, and then modular reduction using white box implementation. The first operation will require not more than $N \times t$ polynomial squaring's, which is almost a "free" operation without modular reduction. Further processing of evaluation result before sending it to the receiver requires just a few XOR operation to get $\sum B_N(x)$ and a creation of a vector B as it was explained in Section 2. The decryption will require one or two modulo two additions and one multiplication of polynomials as explained in Section 2. Final decryption operation will require to compute $P^{-1}(P(m(x))) = m(x)$, where P^{-1} is a compositional inverse of polynomial $P(x)$, which can be shown to require not more than $(N-1)^3$ operations. Modular reduction operation will require in the case of $N = 128$ and if $t = 64$ the most typical average size for evaluation polynomial to store 127×64 replacement $B_N(x)$ values. The memory required for each replacement term will be 16 bytes resulting overall $64 \times 127 \times 16$ bytes $=$ 130 Kbytes on average and 260 Kbytes in the worst case.

We actually plan to implement the construction presented in this paper and will represent more detailed implementation result that time.

7. Conclusion

In this paper a novel white box encryption scheme based on permutation polynomials has been presented. It was shown that public key encryption schema based on permutation polynomials is a one-way function. Implementation aspects have also been briefly discussed.

Acknowledgment

Authors express their gratitude to anonymous referee for very valuable comments which helped to improve the content of the paper.

References

[1] W. Diffie, M.E. Hellman, New Directions in Cryptography, in: IEEE Transactions on Information Theory, vol. IT-22, 1976, pp. 644–654.
[2] Y. Laigle-Chapuy, Permutation polynomials and applications to coding theory, Finite Fields Appl. 13 (2007) 58–70.
[3] R. Lidl, Niederreiter, Finite Fields, Addison Wesley, reading, MA, 1983.
[4] V.S. Miller, Use of Elliptic curves in cryptography, in: Advanced in Cryptology-Crypto-85 Proceedings, Springer-Verlag, 1986, pp. 417–426.
[5] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM 21 (2) (1978) 120–126.
[6] J. Schwenk, K. Huber, Public key encryption and digital signatures based on permutation polynomials, Electron. Lett. 34 (1998) 759–760.
[7] N. Zeirler, Linear recurring sequens, J. Soc. Ind. Appl. Math. 7 (1959) 31–48.