PPC : A primitive polynomial cryptosystem based on ElGamal

Abstract

In this paper we introduce new structure based on primitive polynomial and new cryptosystem from the structure. Our cryptosystem's significant property is that the speed of encryption and the security of cryptosystem can be selected without changing the size of key. Also based on this structure we offered some applications about Digital Signature Algorithm and Diffie-Hellman key exchange scheme.

Contents

1. Introduction

Since Diffie and Hellman [1] introduce the public key cryptography, the modern cryptosystems relies on the one-way function, which is easy to compute with the function but hard to find the inverse. For example, The ElGamal cryptosystem by [2] depends on the Discrete Logarithm Problem (DLP), and the RSA cryptosystem by [3] on the hardness of factorization of a large number. These cryptosystems are known that today there does not exists efficient algorithms to break them so they are used widely on many areas such as secure communication or digital authentication, *etc*.

In this paper we introduce a new cryptosystem *primitive polynomial cryptosystem* (PPC) by using the relation of primitive polynomial and the periodic properties of the linear homogeneous recurrence relation on finite field. Our cryptosystem relies on the hardness to find index with the given sequence term generated from a special recurrence relation, which has some similarity with the DLP when we consider the exponentiation as a geometric sequence. In fact we found out that there are some sequences satisfying special property which fits to the notion primitive roots on the multiplicative group $\mathbb{Z}_p^{\times}$.

In section 2, we first describe the periodic properties of linear homogeneous recurrence relation with constant coefficients on finite field $GF(p)$ and define *primitive recurrence relation*. Then we offer the properties of primitive recurrence which will be used for the algorithm to find the recurrence. In section 3, we introduce the entire cryptosystem scheme and offer a simple example. For the rest sections, we discuss the security of our system and give some applications of our cryptosystem.

2. Preliminaries

2.1. Motivation

It is well known that Fibonacci sequence has some periodic property on any finite field $GF(p)$, which is called the Pisano period. Based on this fact, we conjectured that a general sequence which has a linear homogeneous recurrence relation with constant coefficients has some period corresponding to Pisano period of Fibonacci sequence when we embed it on $GF(p)$. In fact, we found out that every general sequence must have some period and there is a somewhat non-intuitive relation between the length of period and the recurrence relation of given sequence. Let $a : \mathbb{N} \cup \{0\} \to \mathbb{Z}$ defined as $n \mapsto a_n$ be a sequence and suppose that a_n has a linear homogeneous recurrence relation of degree m with integer coefficients below:

$$a_{n+m} = c_1 a_{n+m-1} + c_2 a_{n+m-2} + \cdots + c_m a_n, \ c_m \neq 0$$

Also, let $GF(p)$ be a finite field of the order prime p.

Definition 1. A vector defined as $V_n = \begin{bmatrix} a_n & a_{n+1} & \cdots & a_{n+m-1} \end{bmatrix}^T$ is called a *sequential vector* of $\{a_n\}$. Then it is trivial that the $m \times m$ matrix C defined as below satisfies $V_{n+1} = CV_n$.

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ c_m & c_{m-1} & c_{m-2} & c_{m-3} & \cdots & c_1 \end{bmatrix}$$

In this sense we define the matrix C be the *characteristic matrix* of $\{a_n\}$ and the characteristic polynomial[2] of C be the *characteristic polynomial* of $\{a_n\}$.

Remark that it is natural to observe that V_n has the same structure with the original sequence a_n, so we will call the sequence $\{V_n\}$ as the sequential vector structure. We will use this as the main structure for our cryptosystem. Also, we will use the term "characteristic matrix of $f(x)$" as the same meaning of characteristic matrix of $\{a_n\}$ where $\{a_n\}$ is a sequence derived from $f(x)$.

Analyzing the characteristic matrix gives many significant information about its sequence. For example, we can easily derive that the characteristic polynomial is exactly $x^m - (c_1 x^{m-1} + c_2 x^{m-2} + \cdots + c_m)$ with respect to variable x and the general relation between the index n and the sequence term a_n is $a_n = k_1 {\gamma_1}^n + k_2 {\gamma_2}^n + \cdots + k_m {\gamma_m}^n$, where γ_1, γ_2, $\cdots$, γ_m are the roots of the characteristic polynomial and k_1, k_2, $\cdots$, k_m are constants determined by the initial condition of $\{a_n\}$.

When we embed the sequence $\{a_n\}$ on the finite field $GF(p)$ with the mapping defined as $a_n \mapsto a_n \bmod p$, we can expect that the embedding of $\{a_n\}$ may have the recurrence relation with the same coefficient as the original ones modulo p, so it is natural to carry out the theorem below.

Theorem 2. Let the sequence $\{b_n\}$ be an embedding of $\{a_n\}$ on $GF(p)$. Then there exists an positive integer k_p and $N \in \mathbb{N} \cup \{0\}$ such that for every $n \in \mathbb{N}$ with $n \geq N$, $b_n = b_{n+k_p}$. We call the least k_p as the *period* of b_n on $GF(p)$ and $p^m - 1$ is an upper bound of the period.

Definition 3. Let $f(x) \in GF(p)[x]$ be a polynomial. The *order* of $f(x)$, denoted by $\mathrm{ord}_p(f(x))$, is the least positive integer e such that $f(x)$ divides $x^e - 1$ on $GF(p)[x]$.

---

2) Characteristic polynomial which is used to compute the eigenvalues of given matrix.

This definition implies that we can find out that there is a significant relation between the period of given sequence on the finite field $GF(p)$ and the order of characteristic polynomial of the sequence. In this relation, we observe that there is a condition on which the period of sequence becomes the upper bound described on the **Theorem 2**. We call this sequence as the *Primitive Recurrence Relation*. The properties of this sequence is discussed thoroughly on the next section.

Theorem 4. Let $P(x)$ be a characteristic polynomial of $\{a_n\}$ and k_p be the period of $\{b_n\}$ on $GF(p)$. Then $k_p = \mathrm{ord}_p(P(x))$.

Definition 5. Let $a_{n+m} = c_1 a_{n+m-1} + c_2 a_{n+m-2} + \cdots + c_m a_n$ be the linear homogeneous recurrence relation of a sequence a_n. Then this relation is called a *Primitive Recurrence* if the embedding $\{b_n\}$ of $\{a_n\}$ on $GF(p)$ has a period $p^m - 1$.

2.2. Properties of Primitive Polynomial

In this section we introduce some properties of primitive recurrence which will be used to find it. First, we define the primitive polynomial on $GF(p)[x]$.

Definition 6. Let $f(x) \in GF(p)[x]$ be a polynomial with degree m. Then $f(x)$ is a primitive polynomial if every elements of $GF(p^m)^\times$ except 0 can be expressed as the power of one of every roots of $f(x)$ on $GF(p^m)^\times$.

From this definition, the reader may want to verify the existence of primitive polynomial in $GF(p)[x]$. Note that the existence of primitive root on $GF(p^m)$ gives an explicit formula to construct a primitive polynomial.

Theorem 7. Primitive polynomial of degree m exists in $GF(p)[x]$ for any prime p and integer m. We can construct $P(x)$ by using one generator λ of $GF(p^m)^\times$ such as below.

$$P(x) = (x-\lambda)(x-\lambda^p)(x-\lambda^{p^2})\cdots(x-\lambda^{p^{m-1}})$$

Thus it is natural to consider the relation of a primitive polynomial and a primitive recurrence because of the periodic properties of them. In fact, we found out that the characteristic polynomial of a primitive recurrence must be a primitive polynomial. Also, when we construct a recurrence relation from given primitive polynomial, we observed that the sequence induced from the constructed recurrence relation is primitive recurrence.

Theorem 8. A linear homogeneous recurrence relation is primitive recurrence if and only if its characteristic polynomial is primitive polynomial.

Thus, finding primitive recurrence on $GF(p)$ is equivalent to find primitive polynomial in $GF(p)[x]$. So we aimed to find the algorithm for finding primitive polynomial. In fact, there is an algorithm from the generative property of primitive polynomial so we end this section with the theorems for this algorithm.

Theorem 9. Let $\{a_n\}$ be a sequence on $GF(p)$ which has a linear homogeneous recurrence relation of degree m and C be a $m \times m$ characteristic matrix of $\{a_n\}$. Then for the set $F := \{r_1, r_2, \cdots, r_k\}$ of prime factors of $p^m - 1$, and if C satisfies the property

$$C^{(p^m-1)/r} \bmod p \neq I_m \text{ for all } r \in F, \text{ and } C^{p^m-1} \bmod p = I_m,$$

where I_m is a $m \times m$ identity matrix, then the recurrence relation of $\{a_n\}$ is a primitive recurrence.

Theorem 10. There exists exactly $\phi(p^m - 1)/m$ primitive polynomials of degree m in $GF(p)[x]$, where $\phi(x)$ is the Euler's phi function.

3. Cryptosystem Description

First, we will review the ElGamal cryptosystem [2] and Decisional Diffie-Hellman Problem [7]. In ElGamal, choose a large prime p, its primitive root g and integer $x \in \{0, 1, \cdots, p-1\}$, and compute $y = g^x \bmod p$. The public key is p, g and y, and the private key is x. For encryption let the message space $\mathfrak{B}$ be $\{1, 2, \cdots, p-1\}$. The sender choose an random integer $k \in \{1, 2, \cdots, p-1\}$ for each plaintext $m \in \mathfrak{B}$ and compute the ciphertext (c_1, c_2) where $c_1 := g^k$, $c_2 := m \cdot y^k$. On the decryption process, the receiver compute $m = {c_1}^{-x} c_2$ with the given ciphertext (c_1, c_2) and the private key x.

Suppose that the eavesdropper can choose some of plaintexts and get their ciphertexts. Then for the eavesdropper to distinguish two different plaintext from its ciphertext, he must solve the Decisional Diffie-Hellman Problem (DDHP): even if g^x and g^k are known, the efficient algorithm to distinguish g^{kx} from g^z does not exist, where z is randomly chosen integer from $\mathbb{Z}_p$. So he cannot distinguish two ciphertexts from two different plaintexts even if he knows the original plaintexts.

Note that the index on our sequence vector structure can be understood as matrix exponentiation. So from our structure there are matrix-related problems [9] which are similar to DLP and DDHP: Suppose that the primitive polynomial of degree m is given and let $\{V_n\}$ be a sequential vector structure on $GF(p)^m$ induced from this polynomial. Also, assume that the initial vector V_0 is given. Then there are no such efficient algorithm to find x when V_x is given. Also, even if the tuple (V_x, V_y) is given, V_{xy} and V_z cannot be distinguished, where z is a random integer chosen from $\{1, 2, \cdots, p^m - 1\}$. So we will con-

sider that the notions DLP and DDHP are the same as the index-wise problems we described above.

From the decryption process of ElGamal, note that by knowing x and g^k the receiver can easily compute g^{kx} and find the plaintext m. On the similar way, we aimed to find a method to compute sequential vector V_{kx} from x and V_k. So, we find an algorithm for do that and described on the encryption scheme.

3.1. Setup

First, we must choose a prime p and a primitive polynomial of degree m in $GF(p)[x]$. The algorithm to find this polynomial is offered on Appendix B. From this, we construct a sequential vector structure $\{V_n\}$ and choose a step size $s \in \{1,2,\cdots,p^m-1\}$ such that $\gcd(s,p^m-1)=1$. Since a structure $\{A_n\}$ defined as $A_n := V_{sn}$ has the same period as $\{V_n\}$, we will use it for our cryptosystem. Now we choose an initial vector $A_0 \in GF(p)^m \setminus \{\vec{0}\}$, an integer $x \in \{1, 2, \cdots, p^m-1\}$ and compute A_x. Since by **Definition 5** any sequential vector V satisfies the property $C^iV = C^jV$ implies $i \equiv j \pmod{p^m-1}$ for the characteristic matrix C of the given recurrence, the initial vector can be chosen arbitrarily. In this situation x be a secret key and A_x, A_0, $S := C^s$ and p be a public key. The further discussion about setting parameters is given on the section 6. The rest of our paper we will call S as a *shift matrix*. Also we will use a function $B_{p,m} : GF(p)^m \to \mathbb{Z}_{p^m-1}$ which converts a vector to a p-adic integer, defined as $B_{p,m}(v) = [p^{m-1}\ p^{m-2}\ \cdots\ 1]v$.

3.2. Encryption

Let the message space $\mathfrak{E}$ be $\{0, 1, 2, \cdots, p^m-1\}$ and the plaintext be $M \in \mathfrak{E}$. For each plaintext the sender choose a random integer k from the set $\{1, 2, \cdots, p^m-1\}$. From this setting the sender encrypts plaintext M by the process below:

(1) Compute A_k by $A_k = S^kA_0$.

(2) Compute A_{kx} with A_0, A_x, k using an algorithm below.

(3) The ciphertext will be $(c_1, c_2) = (B_{p,m}(A_k), B_{p,m}(A_{kx}) + M \bmod p^m-1)$.

On the step (2) the sender must compute A_{kx} with A_0, A_x and k without knowing x. It seems that the sender have to solve DLP, but by using the algorithm called *Imitation Matrix Algorithm* (IMA), the sender can compute A_{kx} efficiently.

Algorithm. Imitation Matrix Algorithm

Input: Sequential vectors $A_0, A_x \in GF(p)^m$, an integer $y \in \{0, 1, \cdots, p^m - 1\}$, a $m \times m$ shift matrix S.

Output: Sequential vector A_{xy}

Initialize $m \times m$ zero matrices V and P

For $t = 1, 2, \cdots, m$ do

t-th column of $P \leftarrow A_x$

t-th column of $Q \leftarrow A_0$

$A_x \leftarrow SA_x$

$A_0 \leftarrow SA_0$

Find the inverse Q^{-1} of Q

Compute $(PQ^{-1})^y A_0 = S^{xy} A_0 = A_{xy}$

return A_{xy}

Remark that by using the notion of shift matrix, $S^x A_n = A_{n+x}$ for all $n \in \mathbb{Z}$. In IMA, we can find an *imitation matrix* $X = S^x$ without finding x, by solving a linear system

$$\begin{cases} XA_0 = A_x \\ XA_1 = A_{x+1} \\ \vdots \\ XA_{m-1} = A_{x+m-1} \end{cases}.$$

From this system, we have that

$$X[\, A_0 \;\; A_1 \;\; \cdots \;\; A_{m-1} \,] = [\, A_x \;\; A_{x+1} \;\; \cdots \;\; A_{x+m-1} \,] \text{ or } XQ = P.$$

Since every eigenvectors of S is not in $GF(p)^m$, the set $\{A_0, A_1, \cdots, A_{m-1}\}$ is linearly independent. So Q is invertible, which implies we can compute the unique X from

$$X = [\, A_x \;\; A_{x+1} \;\; \cdots \;\; A_{x+m-1} \,][\, A_0 \;\; A_1 \;\; \cdots \;\; A_{m-1} \,]^{-1} = PQ^{-1}.$$

Thus $A_{xy} = S^{xy} A_0 = (S^x)^y A_0 = X^y A_0 = (PQ^{-1})^y A_0$.

Also note that IMA needs approximately $O(m^3 \log y)$ multiplication and $O(m^2 \log p)$ storage, where almost of the multiplication comes from the process for computing X^y.

3.3. Decryption

Suppose that the receiver retrieves the ciphertext (c_1, c_2). Since the receiver knows the secret key x and A_k from the ciphertext c_1, by applying Imitation matrix algorithm once again the receiver can easily get the plaintext from the given ciphertext.

(1) Compute A_{kx} by using IMA with x, $A_k = B_{p,m}^{-1}(c_1)$.

(2) Compute $M_r = c_2 - B_{p,m}(A_{kx}) \bmod p^m - 1$, M_r is the plaintext.

3.4. Implementation example

Consider the case when we encrypt and decrypt the message with the prime $p = 7$, the degree $m = 4$, the primitive polynomial $f(x) = x^4 + 6x^3 + 5x^2 + 5x + 3 \equiv x^4 - (4 + 2x + 2x^2 + x^3)$, the shift matrix S with step size $s = 79$, initial vector A_0 and A_x with $x = 419$ defined as

$$S = C^{79} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 4 & 2 & 2 & 1 \end{bmatrix}^{79} = \begin{bmatrix} 3 & 1 & 1 & 5 \\ 6 & 6 & 4 & 6 \\ 3 & 4 & 4 & 3 \\ 5 & 2 & 3 & 0 \end{bmatrix}, \; A_0 = \begin{bmatrix} 4 \\ 2 \\ 2 \\ 5 \end{bmatrix},$$

$$A_x = S^x A_0 = \begin{bmatrix} 3 & 1 & 1 & 5 \\ 6 & 6 & 4 & 6 \\ 3 & 4 & 4 & 3 \\ 5 & 2 & 3 & 0 \end{bmatrix}^{419} \begin{bmatrix} 4 \\ 2 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 5 & 2 & 2 & 6 \\ 3 & 3 & 0 & 1 \\ 4 & 5 & 5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 2 \\ 6 \end{bmatrix}.$$

The receiver makes public key (p, m, S, A_0, A_x) and keeps x as the secret key. The message is encoded with two letters per block, substituting to (blank)=00, A=01, ⋯, Z=26:

I HATE EVE is encoded to 0900 0801 2005 0005 2205

For example, suppose that we want to encrypt the message $M =$ 0900. Assume that the random integer process $k = 253$ is chosen. Then $A_k = [5\ 3\ 6\ 1]^T$ is computed from the fact $A_k = S^{253} A_0$ and $B_{p,m}(A_k) = 1905$ is used as ciphertext c_1. Note that k is chosen for each message, so other messages will be encrypted with different ks.

$$S^{253} A_0 = \begin{bmatrix} 3 & 1 & 1 & 5 \\ 6 & 6 & 4 & 6 \\ 3 & 4 & 4 & 3 \\ 5 & 2 & 3 & 0 \end{bmatrix}^{253} \begin{bmatrix} 4 \\ 2 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 6 & 5 \\ 6 & 3 & 1 & 4 \\ 2 & 0 & 4 & 5 \\ 6 & 5 & 3 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 2 \\ 5 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 6 \\ 1 \end{bmatrix}.$$

By IMA we can compute $A_{kx} = [3\ 2\ 1\ 1]^T$ and compute the ciphertext $c_2 = B_{p,m}(A_{kx}) + M$ $= B_{p,m}([3\ 2\ 1\ 1]^T) + 900 = 1135 + 900 = 2035$, thus the encrypted message for 0900 is

$$c_1 = 1905,\ c_2 = 2035.$$

In the same way, the entire ciphertext is

$$c_1 = 1905\ 0092\ 1791\ 2380\ 1385$$

$$c_2 = 2035\ 0139\ 1625\ 2364\ 1815$$

The receiver can decrypt our ciphertext by first computing $A_{kx} = [3\ 2\ 1\ 1]^T$ using IMA with the secret key $x = 419$ and $A_k = [5\ 3\ 6\ 1]^T$ from the given ciphertext c_1. Then compute $M_s = c_2 - B_{p,m}(A_{kx}) \bmod 2400 = 2035 - 1135 = 900$, which is a plaintext.

4. Feasible attacks

To attack our cryptosystem, the eavesdropper may want to recover the secret key, restore

the whole or partial plaintext from given ciphertext without secret key or distinguish two messages from two ciphertexts. In this section we assume the eavesdropper mainly focus on the first and second objectives above and examine security against some attack methods.

4.1. Brute-force attack

First the eavesdropper may try to find A_{kx} from the given information: A_k and A_x. To achieve this, the eavesdropper may compute all of the sequential vectors to find x, k or both. Since to compute the next term we have to do m multiplication, the eavesdropper must compute mp^m multiplication to compute the whole structure, which depends on the size of p^m. So our cryptosystem is secure against brute-force attack when we choose sufficiently large p^m.

4.2. Subsequence attack

Let $\langle S \rangle$ be a cyclic group generated by a shift matrix S. Then by **Theorem 9**, the order of $\langle S \rangle$ is exactly $p^m - 1$. Note that for any vector A_i in sequential vector structure and $r \in \{1, 2, \cdots, p-1\}$, there exists another vector A_j such that $A_i = rA_j$. This implies $S^{j-i} \equiv rI_m \mod p$ because IMA yields the unique solution.

Based on this fact, we have that the least integer satisfying the condition above for $i = 0$ is $d := \frac{p^m - 1}{p-1}$, so let $X := S^d \equiv gI_m \mod p$ for some integer $g \in \{1, 2, \cdots, p-1\}$. Then for $k \in \{0,1,\cdots,d-1\}$, $A_k = gA_{d+k}$, which implies we can construct the whole structure by multiplying a power of g on each elements of the subsequence $\{A_n\}_{n=0}^{n=d-1}$.

Now assume that the A_0, S and A_x is given and the eavesdropper want to find x. Then he can try to find $x' \in \{0, 1, \cdots, d\}$ and $r \in \{1, 2, \cdots, p-1\}$ such that $C^{x'}A_0 = rA_x$. In this situation he can find g defined above and solve the equation $g^i \equiv r \mod p$ with respect to i. So the eavesdropper can compute $x = x' + id$, but he must compute with approximately $O(mp^{m-1})$ multiplication operations, which implies our system is secure against this attack.

4.3. Birthday attack

One may consider the probabilistic method to restore the secret key x. In birthday attack, the eavesdropper want to find x' satisfying $A_x = A_{x'}$, which is called a collision. So until find the collision, the eavesdropper first chooses a set $F \subset GF(p)^m$ with n elements uniformly, compute $S = \{A_s \mid s \in F\}$ and compare A_x to the elements of S. We can compute probability p_c when the collision occurs by using the fact $1 - x \simeq e^x$ for $x \simeq 0$:

$$p_c = 1 - \prod_{k=1}^{n} \left(1 - \frac{k}{p^{m-1}}\right) \simeq 1 - e^{-\frac{n^2}{2p^{m-1}}}.$$

So if the eavesdropper wants to find collision with the probability $p_c = p_t$, he must choose the set of size

$$n = \sqrt{2p^{m-1}\ln\frac{1}{1-p_t}}.$$

For example, when $p_t = 0.5$, $n = \sqrt{2p^{m-1}\ln 2} \simeq 1.18p^{(m-1)/2}$. In this example to find the collision we need approximately $O(m^4 p^{(m-1)/2}\log p)$ multiplication and $O(p^{(m-1)/2})$ for storage. So for the suggested parameter setting above, it is hard to find the collision efficiently.

4.4. Differential cryptanalysis attack

Differential cryptanalysis is a method to analyze the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs [6]. Suppose that the eavesdropper have access to plaintexts m_1 but does not know the other plaintext m_2. And their ciphertexts (c_{11}, c_{12}), (c_{21}, c_{22}) are given. In this situation the eavesdropper wish to restore m_2 from (c_{21}, c_{22}). If $c_{11} = c_{21}$, then by using the fact $c_{22} - c_{12} = m_2 - m_1$ the eavesdropper can find m_2, otherwise it is hard to find any information about m_2 from the given data. Even though the eavesdropper successfully restored m_2, the key restoration is much more harder. But the probability to the case $c_{11} = c_{21}$ occur is $1/p^m$, which implies our cryptosystem is secure against this attack.

5. Applications

In this section we apply our sequential vector structure to Diffie-Hellman key exchange scheme and Digital Signature Algorithm. We motivated from the applications on Elliptic Curve Cryptosystem (ECC) [3] and since these schemes are based on DLP, we thought that also we can harness the applications because of the similarity of DLP and our structure. Let p be a prime and assume that the primitive polynomial $f(x) \in GF(p)[x]$ of degree m is given. Also, let A_n be a sequential vector induced from $f(x)$ and S be its shift matrix.

5.1. Primitive Polynomial Diffie-Hellman key exchange scheme

Diffie-Hellman key exchange scheme [1] is a method for the participants to share the common by using their secret on public channel. In this section suppose that Alice and Bob wish to share the common. By the method below, Alice and Bob share the common:

(1) Alice and Bob share p, S and A_0.

(2) Alice picks a positive integer j from $\{1, 2, \cdots, p^m - 1\}$ and sends A_j to Bob.

(3) Bob picks a positive integer i from $\{1, 2, \cdots, p^m - 1\}$ and sends A_i to Alice.

(4) Alice and Bob compute A_{ij} by IMA.

Note the eavesdropper has access to the whole integer transmitted between Alice and Bob. But it is hard for the eavesdropper to compute A_{ij} from A_i, A_j A_0 because this is equivalent to solve the DLP. Thanks to that fact Alice and Bob can use A_{ij} as a symmetric key securely.

5.2. Public Key Concealing

Remark that our cryptosystem can be attacked by some feasible attack if the eavesdropper knows all of the public key, but if he does not know some of the public key, it is extremely hard for him to construct the sequential vector structure so the attack methods we offered are useless even the Brute-force attack. Based on this fact, in this section we introduce two methods to conceal the initial vector A_0 and the shift matrix S of the public key, which can be applied at the same time and iteratively for the better security. Suppose that Alice and Bob share a prime p, S and A_0 previously and wish to create new shift matrix and initial vector which will be used for the new cryptosystem.

5.2.1. Shift Matrix Concealing

On our cryptosystem, the information of shift matrix is vital for entire encryption and decryption scheme because without this, the entire sequential vector structure cannot be constructed. So if the shift matrix is well concealed, then the eavesdropper must find S by solving DLP. Thus we propose the method for Alice and Bob to conceal the shift matrix by using the primitive polynomial Diffie-Hellman key exchange scheme:

(1) By applying Primitive Polynomial Diffie-Hellman key exchange to the shared shift matrix S, Alice and Bob have the shared S^{ij}. On the exchange process, assert that $\gcd(i,p^m-1)=\gcd(j,p^m-1)=1$.
(2) Alice and Bob discard i, j, S^i, and S^j which are used for key exchange.
(3) Now S^{ij} is a new shift matrix and Bob chooses the secret key x.
(4) Bob makes a public key (p, A_0, A_x) without including the shift matrix S^{ij}.

5.2.2. Initial Vector Concealing

Remark that our cryptosystem works well regardless of the initial vector. So we propose the method to conceal the initial vector of public key by using primitive polynomial Diffie-Hellman key exchange scheme. By the process below, Alice and Bob can construct new initial vector and conceal it.

(1) Using Primitive Polynomial Diffie-Hellman key exchange, Alice and Bob have the shared A_{ij}.
(2) Alice and Bob discard i, j, A_i, and A_j which are used for key exchange.

(3) Now A_{ij} is a new initial vector and Bob chooses the secret key x.

(4) Bob makes a public key (p, S, A_x) without including the initial vector A_{ij}.

Remark that from our method it is hard for the eavesdropper to compute A_{ij} from A_i, A_j A_0 because this problem is equivalent to DLP, which means our method successfully conceals the initial vector. Even if the eavesdropper computes A_{ij} once by Brute-force attack, by processing key concealing iteratively we can make the eavesdropper compute more initial vectors from he computed previously. Plus, the eavesdropper must compute the initial vector in accordance with order of the concealing process.

To ensure that our method makes the system more secure, consider the case that the eavesdropper obtained the plaintext M and its ciphertext (c_1, c_2) and wants to restore the secret key x. Then he can compute A_{kx} from $c_2 = M + B_{p,m}(A_{kx})$. In this case if he knows the initial vector V, then he can compute S^k by IMA from c_1 and try Brute-force attack to find x with $O(p^m)$ of multiplication operation from the fact $A_{kx} = S^{kx}V$, where S is the given shift matrix. But if V is concealed by our method, then he must try Brute-force attack to find the triplet (k, x, V) which fits to the given information. In this situation, he has to solve DLP approximately $O(p^{2m})$ times. So it is the most efficient way for the eavesdropper to attack our cryptosystem is to find V by solving DLP on the public key concealing process.

5.3. Primitive Polynomial Digital Signature Algorithm

Digital Signature Algorithm (DSA) [5] aims to provide assurance that the sender signed the information by generating signature for each message with his secret key. Also, the generated signature can be verified from the receiver's public key. The first version of DSA is based on DLP [2], but from [5] Elliptic Curve Digital Signature Algorithm (ECDSA) is suggested, which is based on the structure constructed by Elliptic curve [4]. In this section we propose the Primitive Polynomial Digital Signature Algorithm (PPDSA) which based on our sequential vector structure from primitive polynomial.

On PPDSA, first we need parameters described as below:

(1) p : Any prime.

(2) $f(x)$: A primitive polynomial in $GF(p)[x]$ of degree m.

(3) q : A N-bit prime which divides $p^m - 1$.

(4) n : An integer satisfying $nq = p^m - 1$.

(5) A_0 : An arbitrary initial vector.

From the above setting the sender chooses an integer x in $\{1, 2, \cdots, q-1\}$ and uses it as

secret key. Also, the public key will be p, $f(x)$, q, A_0 and A_{nx}. For the message m the sender wishes to send to receiver, the signature for m is generated by the process below:

(1) Compute $e = HASH(m)$ and define z as the leftmost N bits of e.
(2) Choose a random integer k from $\{1, 2, \cdots, q-1\}$.
(3) Compute A_{nk} by using IMA and define $r := B_{p,m}(A_{nk}) \bmod q$.
(4) Compute $s := k^{-1}(z+xr) \bmod q$, where k^{-1} is an inverse of k on $\mathbb{Z}_q$.
(5) The signature is the pair (r, s). If r or s is 0, then choose another k and try again.

The signature (r, s) will be transmitted along with the message to the receiver. Note that the verification can be performed by anyone who has access to the signature and it's public key. The receiver can perform message verification from the received message m and its signature (r, s) by the method below:

(1) Verify that both of r and s are integers in $\{1, 2, \cdots, q-1\}$.
(2) Compute $e = HASH(m)$ and define z as the leftmost n bits of e.
(3) Compute $u_1 := zs^{-1} \bmod q$ and $u_2 := rs^{-1} \bmod q$, where s^{-1} is an inverse of s on $\mathbb{Z}_q$.
(4) Compute $V_1 := (C^n)^{u_1} \bmod q$ and $V_2 := (C^{nx})^{u_2} \bmod q$ by IMA, where C is the characteristic matrix from $f(x)$.
(5) Compute $V \equiv (V_1 V_2)A_0 \equiv A_{n(u_1+xu_2)} \pmod q$ and $v := B_{p,m}(V) \bmod q$.
(6) If $r = v$, then the message is well verified.

6. Conclusion and Discussion

In short, we introduced new cryptosystem based on the periodic properties of the primitive polynomial and its application. Now we will compare our cryptosystem to the other cryptosystems, for example, ElGamal and ECC, which use some cyclic group structure. Assume that m be a degree and p and q be a prime such that the bit size of q is equal to that of p^m. From above q is a parameter used for ElGamal and ECC, and p, m are used for our system. Then our system needs $O(m^3 \log(p^m))$, ElGamal needs $O((\log q)^2)$, and ECC needs $O((\log q)^3)$ multiplication operation. From above we can observe that when m goes smaller, then the computation of our system become faster. Explicitly, we can derive the fact that if $m^3 < \log p$, then our system is faster than ElGamal, and if $m^{3/2} < \log p$, then our system is faster than ECC. On the other hand, when m goes bigger our system requires more multiplication, so the process become slower, but as we described on section 4, if we fix the bit size of structure as p^m, the number of multiplication needed for Brute-force attack and Birthday attack depend on m and m^3, respectively, which implies

our system become more secure. Therefore, our system is much more tangible to choose p and m for the trade-off between process speed and security.

Unfortunately, our cryptosystem has a problem on the setup process: On the setup our cryptosystem requires the integer factorization and finding a primitive polynomial, which are excessively time-consuming task. Even though it takes a lot of time for the initial setup inevitably, there is a simple method to change the sequential vector structure when the size of structure is fixed: The user of our cryptosystem can change s of the shift matrix $S = C^s$ to s' satisfying $\gcd(s', p^m - 1) = 1$. This method works well because the shift matrix determines the sequential vector structure. We expect that by using the Shor's Algorithm [8] on quantum computer the problems on the setup can be solved, but it is questionable whether the Shor's Algorithm breaks our cryptosystem.

7. References

[1] W. Diffie and M. Hellman, “New directions in cryptography,” IEEE Transactions on Inform Theory, vol. IT-22, 1976, pp. 644-654.

[2] Taher ElGamal, “A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” IEEE Transactions on Information Theory, vol. IT-31, 1985, pp. 469–472.

[3] R. Rivest, A. Shamir and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Communications of the ACM, vol. 21, 1978, pp. 120-126.

[4] N. Koblitz, “Elliptic curve cryptosystems,” Mathematics of Computation, vol. 48, 1987, pp. 203–209.

[5] C. Kerry, P. Gallagher, “Digital Signature Standard (DSS),” NIST, MD 20899-8900, 2013.

[6] E. Biham and A. Shamir, “Differential Cryptanalysis of the Data Encryption Standard,” Springer Verlag, 1993, pp. 11-32, ISBN 0-387-97930-1, ISBN 3-540-97930-1.

[7] D. Boneh, “The Decision Diffie-Hellman Problem,” Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, vol. 1423, 1998, pp. 48-63, ISBN 978-3-540-64657-0.

[8] P. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” SIAM Journal on Computing, vol. 26, 1997, pp. 1484-1509.

[9] D. Freeman, “The Discrete Logarithm Problem in Matrix Groups,” 2004, available online at http://theory.stanford.edu/~dfreeman/papers/discretelogs.pdf.

8. Appendix

A. The proofs of theorems

In this section we offer the proofs of theorems on section 2. We will use some notations below:

$|S|$: Cardinal number of a set S.
$\mathrm{diag}(a_1, a_2, \cdots, a_n)$: A diagonal matrix (c_{ij}) whose diagonal elements are a_1, a_2, $\cdots$, a_n, *i.e.*, $c_{ii} = a_i$ for $i = 1, 2, \cdots, n$ and other elements are zero.

Theorem 2. Let the sequence $\{b_n\}$ be an embedding of $\{a_n\}$ on $\mathbb{Z}_p$. Then there exists an positive integer k_p and $N \in \mathbb{N} \cup \{0\}$ such that for every $n \in \mathbb{N}$ with $n \geq N$, $b_n = b_{n+k_p}$. We call the least k_p as the *period* of b_n on $\mathbb{Z}_p$ and $p^m - 1$ is an upper bound of the period.

proof)
Let B_n be the sequential vector for $\{b_n\}$. If $B_n = \vec{0}$, then for any $x \in \mathbb{N}$, $B_{n+x} = \vec{0}$, which implies the period of b_n is 1. So assume that $B_n \neq \vec{0}$ for any $n \in \mathbb{N} \cup \{0\}$. Then by pigeonhole principle, there exists at least one vector $V \in \mathbb{Z}_p{}^m$ such that $B_x = B_y = V$ for some $x, y \in \mathbb{N} \cup \{0\}$ with $x > y$. When we denote $k_p = x - y$ and $N = y$, we have that $B_{n+x} = B_{n+y} \Leftrightarrow B_n = B_{n+(x-y)} = B_{n+k_p}$ for any $n \geq N$, which proves the existence of the period.

From the assumption we have that $S := \{B_n | n \in \mathbb{N} \cup \{0\}\} \subseteq GF(p)^m \setminus \{\vec{0}\}$. Note that $|S| = N + k_p$ because when we denote $S_N := \{B_n | n \in \mathbb{N} \cup \{0\} \wedge n \leq N\}$, $|S_N| = N$ and $S_N \cap S = \varnothing$. From this fact, we have that $N + k_p \leq p^m - 1$, which implies the upper bound of k_p is $p^m - 1$ when $N = 0$ and $k_p = p^m - 1$. ■

Theorem 4. Let $P(x)$ be a characteristic polynomial of $\{a_n\}$ and k_p be the period of $\{b_n\}$ on $\mathbb{Z}_p$. Then $k_p = \mathrm{Ord}_p(P(x))$.

proof)
Let C be the $m \times m$ characteristic matrix of $\{a_n\}$, $C = S\Lambda S^{-1}$ be the diagonalization of C on $GF(p^m)$, and $A_n \in GF(p)^m$ be the n-th term of sequential vector. To find the period of $\{a_n\}$ on $GF(p)$, we must find k_p satisfying $C^{k_p} A_0 = A_0 \Leftrightarrow S\Lambda^{k_p} S^{-1} = I_m$, where I_m is the $m \times m$ identity matrix. By **Definition 3**, we have that $P(x) \mid x^{\mathrm{ord}_p(P(x))} - 1 \Leftrightarrow \lambda_k{}^{\mathrm{ord}_p(P(x))} = 1$ for $k = 1, 2, \cdots, m$, where λ_1, λ_2, $\cdots$, λ_m are the roots of $P(x)$ on $GF(p^m)$. Since

$\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_m)$, we can derive that

$$\Lambda^{\mathrm{ord}_p(P(x))} = \mathrm{diag}(\lambda_1^{\mathrm{ord}_p(P(x))}, \lambda_2^{\mathrm{ord}_p(P(x))}, \lambda_m^{\mathrm{ord}_p(P(x))}) \equiv I_m \pmod{p}$$

$$\Leftrightarrow S\Lambda^{\mathrm{ord}_p(P(x))}S^{-1} \equiv I_m \pmod{p},$$

which implies $k_p \le \mathrm{ord}_p(P(x))$.

Assume that $k_p < \mathrm{ord}_p(P(x))$. Then there exists some integer $t < \mathrm{ord}_p(P(x))$ such that $\lambda_k^{\,t} \equiv 1 \pmod{p}$ for $k = 1, 2, \cdots, m$. This implies $P(x) \mid x^t - 1$ but by the minimality of order this is impossible, so $k_p = \mathrm{ord}_p(P(x))$. ■

Theorem 7. Primitive polynomial of degree m exists in $GF(p)[x]$ for any prime p and integer m. We can construct $P(x)$ by using one generator of $GF(p^m)^\times$ such as below.

$$P(x) = (x-\lambda)(x-\lambda^p)(x-\lambda^{p^2})\cdots(x-\lambda^{p^{m-1}})$$

proof)
Since $GF(p^m)^\times$ is a finite multiplicative subgroup of a field $GF(p^m)^\times$, $GF(p^m)$ is cyclic, so we can choose one generator λ of $GF(p^m)^\times$. Also, $\lambda^i = \lambda^j$ implies $(i-j) \mid p^m - 1$ and $\gcd(p^k, p^m - 1) = 1$ for $k = 0, 1, \cdots, m-1$. Thus by **Definition 6**, $P(x)$ defined above must be a primitive polynomial. ■

Theorem 8. A linear homogeneous recurrence relation is primitive recurrence if and only if its characteristic polynomial is primitive polynomial.

proof)
First, assume that the primitive recurrence $\{a_n\}$ on $GF(p)$ and its $m \times m$ characteristic matrix C is given. Then by **Theorem 4** and **Definition 5**, when we denote $P(x)$ as the characteristic polynomial of $\{a_n\}$ and K as the splitting field of $P(x)$ over $\mathbb{Z}_p$, the order of $P(x)$ and the period of $\{a_n\}$ on $GF(p)$ is equal to $p^m - 1$. This implies that $P(x) \mid x^{p^m-1} - 1$, so for one root λ on $GF(p^m)$, $\lambda^{p^m-1} = 1$. Due to the minimality of order, the order of cyclic group G generated by λ is $p^m - 1$ and $G \subseteq GF(p^m)^\times$. Thus, $G = K$, *i.e.*, $P(x)$ is primitive polynomial.

Conversely, now assume that the primitive polynomial $f(x) \in \mathbb{Z}_p[x]$ and its splitting field $GF(p^m)$ is given and let $\{b_n\}$ be a sequence induced from $f(x)$. Then by **Definition 6** and the definition of splitting field, for one root λ of $f(x)$ on $GF(p^m)$, the least positive integer t satisfying $\lambda^t = 1$ is $t = p^m - 1$. Since for every solution of $f(x)$ this condition is satisfied, $f(x) \mid x^{p^m-1} - 1$, which implies $\{b_n\}$ has a primitive recurrence by **Theorem 4**. ■

Theorem 9. Let $\{a_n\}$ be a sequence on $GF(p)$ which has a linear homogeneous recurrence

relation of degree m and C be a $m\times m$ characteristic matrix of $\{a_n\}$. Then for the set $F:=\{r_1,r_2,\cdots,r_k\}$ of prime factors of p^m-1, if C satisfies the property

$$C^{(p^m-1)/r} \bmod p \neq I_m, \text{ for all } r\in F \text{ and } C^{p^m-1} \bmod p = I_m,$$

where I_m is a $m\times m$ identity matrix, then the recurrence relation of $\{a_n\}$ is a primitive recurrence.

proof)
Let $P(x)\in GF(p)[x]$ be a characteristic polynomial of $\{a_n\}$ and assume that the characteristic matrix C of $\{a_n\}$ satisfies the property on the statement. By **Theorem 6**, it suffices to show that $P(x)$ is a primitive polynomial. From the proof of **Theorem 4** for the eigenvalue matrix Λ of C we have that $\Lambda^k=\mathrm{diag}(\lambda_1{}^k, \lambda_2{}^k, \cdots, \lambda_m{}^k)$, where λ_i for $i=1, 2, \cdots, m$ is a root of $P(x)$ on $GF(p^m)$. Also, from the proof of **Theorem 7** when we denote λ as a generator of $GF(p^m)$ there exists an integer e_i such that $\lambda_i=\lambda^{e_i}$ for $i=1, 2, \cdots, m$, so we can observe that $\Lambda^k=\mathrm{diag}(\lambda^{e_1k}, \lambda^{e_2k}, \cdots, \lambda^{e_mk})$. Since $\lambda^{e_ik}=1$ if and only if $p^m-1 \mid e_ik$, $C^k \bmod p=I_m$ occurs when k is a factor of p^m-1 or exactly p^m-1. Note that on the former case we have that $\lambda_i^k=1$ for $k<p^m-1$ so in this case $P(x)$ is not a primitive polynomial. Thus, by assumption we can conclude that $P(x)$ is a primitive polynomial. ■

Theorem 10. There exists exactly $\phi(p^m-1)/m$ primitive polynomials of degree m in $GF(p)[x]$, where $\phi(x)$ is the Euler's phi function.

proof)
By **Theorem 7**, for a generator λ of $GF(p^m)$ one primitive polynomial $f(x)$ can be expressed as

$$f(x)=(x-\lambda)(x-\lambda^p)\cdots(x-\lambda^{p^{m-1}}).$$

Since each element $e\in GF(p^m)$ can be expressed as $e=\lambda^k$ for some integer k in $\{1,2,\cdots,p^m-1\}$, we can observe that e can be a generator of $GF(p^m)$ if and only if $\gcd(k,p^m-1)=1$, so the number of generators of $GF(p^m)$ is exactly $\phi(p^m-1)$.

Note that if two polynomial of the same degree is equal if and only if every roots of these polynomial are same. Based on this fact, assume that the $g(x)$ defined below is equal to $f(x)$:

$$g(x)=(x-\mu)(x-\mu^p)\cdots(x-\mu^{p^{m-1}}),$$

where $\mu\neq\lambda$ is a generator of $GF(p^m)$. Since μ can be expressed as $\mu=\lambda^q$ for some integer q in $\{1, 2, \cdots, p^m-1\}$, we have that $\mu^{p^i}=(\lambda^q)^{p^i}=\lambda^{qp^i}$, so by assumption q must be a power of p. Thus, there are $m-1$ primitive polynomials which is equal to $f(x)$, so the number of primitive polynomial in $GF(p)[x]$ is exactly $\phi(p^m-1)/m$, completing the proof. ■

B. Algorithms for setting up cryptosystem

In this section we offer an algorithm to find primitive polynomial of degree m. Our algorithm is based on **Theorem 9** and **Theorem 10**, which give the criteria for primitive polynomial and its exact number in $GF(p)[x]$.

Algorithm. Primitive Polynomial Algorithm

Input: Degree of polynomial m, prime p.

Output: Primitive polynomial $t(x)$

Compute p^m-1 and find the set F of prime factors of p^m-1.

Until find the primitive polynomial do

Construct polynomial $t(x)$ with random integer coefficients in $\{0, 1, \cdots, p-1\}$ $\cdots$ (a)

Compute characteristic matrix C of $t(x)$

If $C^{p^m-1} \not\equiv I_m \pmod p$ then

Go to (a)

For f in F do

Compute $C^{(p^m-1)/f} \bmod p$

If $C^{(p^m-1)/f} \equiv I_m \pmod p$ then

Go to (a)

End the loop

return $t(x)$

C. Benchmark on various parameter settings

We offer benchmarks for various parameter settings. We used Intel i3-7020U CPU and python 3.7.4 code based on our theorems. The entire code is available online on GitHub: https://github.com/zadkel/PPC

bit of p	degree m	average time (ms)					
		setup				enc	dec
		prime	factorization	primitive polynomial	total		
64	1	14	73	22	111	8	2
32	2	2	3	36	44	7	1
16	4	<1	2	176	187	5	3
8	8	<1	6	557	573	16	6
4	16	<1	5	1765	1808	68	37
1	64	<1	5	247445	248799	3046	1327

Table 1. Average computation time for our cryptosystem. The key size is fixed to 64 bits.

bit of p	degree m	average time (ms)					
		setup				enc	dec
		prime	factorization	primitive polynomial	total		
4	4	<1	1	19	26	4	3
8	4	<1	1	33	36	5	2
16	4	<1	2	176	187	5	3
32	4	2	33	455	496	10	3
64	4	9	9354	433	9813	27	8

Table 2. Average computation time for our cryptosystem. Degree is fixed to 4.

bit of p	degree m	average time (ms)					
		setup				enc	dec
		prime	factorization	primitive polynomial	total		
8	1	<1	<1	2	5	1	1
8	2	<1	<1	7	11	6	2
8	4	<1	1	33	36	5	2
8	8	<1	6	557	573	16	6
8	16	<1	465	9389	9931	161	90

Table 3. Average computation time for our cryptosystem. Size of prime is fixed to 8 bits.

Empirically, we observed that when we fix the size of key, the setup process is efficient when p is large, because the effect of increasing degree on time to find a primitive polynomial is larger than the effect of increasing prime on time to factorize $p^m - 1$. Also, the time for encryption and decryption follows our theoretical computational cost referred on section 6.