A Public Key Cryptoscheme Using Bit-pair Shadows*  

Shenghui Su1, Shuwang Lü2, and Maozhi Xu3  

1 College of Computers, Beijing University of Technology, Beijing 100124  
2 Graduate School, Chinese Academy of Sciences, Beijing 100039  
3 School of Mathematics, Peking University, Beijing 100871  

Abstract: The authors give the definition and property of a bit-pair shadow, and design the three algorithms of a public key cryptoscheme that is based on a multivariate permutation problem (MPP) and an anomalous subset product problem (ASPP) to which no subexponential time solutions are found so far, and regards a bit-pair as an operation unit. Further, demonstrate that the decryption algorithm is correct, deduce the probability that a plaintext solution is nonunique is nearly zero, dissect the running times of the three algorithms, analyze the security of the new scheme against extracting a private key from a public key and recovering a related plaintext from a ciphertext by LLL lattice basis reduction, meet-in-the-middle dichotomy, and adaptive-chosen-plaintext attack. The analysis shows that the bit-pair method increases the density of a related ASSP knapsack to \(
\frac{864}{7} < 0.7778 < 1 \) for \( n = 864 \), \( \frac{696}{7} < 0.7612 < 1 \) for \( n = 696 \), and decreases the modulus length of the new scheme to \( \lceil \frac{lg M}{} \rceil = 464, 544, or 640 \) corresponding to \( n = 80, 96, or 112 \) separately.

Keywords: Public key cryptoscheme; Semantical security; Bit-pair shadow; Random padding

Anomalous subset sum problem; Compact sequence

1 Introduction

In [1], we propose a prototypal public key cryptosystem called REESSE1+ which is based on the three new provable problems, contains the five algorithms, and is used for data encryption and digital signing.

In REESSE1+, a ciphertext is defined as \( \mathcal{G} = \prod_i \mathcal{G}_i^{b_i} \) (\( \% M \)), an anomalous subset product problem (ASPP), where \( b_i \) is the bit shadow of a bit \( b_i \) [1] (also see Section 2.2), and \( n \) is the bit-length of a plaintext block.

Let \( C_1 = g^{u_1} \) (\( \% M \)), \( \ldots, C_n = g^{u_n} \) (\( \% M \)), and \( \mathcal{G} = g^v \) (\( \% M \)), where \( g \) is a generator of \( (\% M) \) which can be found in tolerable subexponential time when the modulus \( M < 2^{1024} \) can be factorized [2]. Then solving \( \mathcal{G} = \prod_i \mathcal{G}_i^{b_i} \) (\( \% M \)) for \( b_1, \ldots, b_n \) is equivalent to solving

\[
b_1 u_1 + \ldots + b_n u_n = v \mod M.
\]

where \( v \) may be substituted with \( v + k M \) along with \( k \in [0, n - 1] \) [3].

Equation (1) is called an anomalous subset sum problem (ASSP) due to every \( b_i \in [0, n] \) [1]. Likewise, due to every \( b_i \in [0, n], \{ u_1, \ldots, u_n \} \) is called a compact sequence [4].

May convert an ASSP into a subset sum problem (SSP) through converting \( u_i \) to a binary number, and thus the density of an ASSP knapsack is defined as

\[
D = \sum_i \lceil \frac{lg n}{} \rceil / \lceil \frac{lg M}{} \rceil = n \lceil \frac{lg n}{} \rceil / \lceil \frac{lg M}{} \rceil.
\]

Evidently, the parameters \( \lceil \frac{lg M}{} \rceil \) and \( n \) have an important influence on the value of \( D \).

In REESSE1+, there are \( n = 80, 96, 112, \) or \( 128 \), and correspondingly \( \lceil \frac{lg M}{} \rceil = 696, 864, 1030, \) or \( 1216 \). Substituting the parameters with concrete values yields

\[
D = 80 \times 7 / 696 = 0.8046 < 1 \text{ for } n = 80 \text{ and } \lceil \frac{lg M}{} \rceil = 696, \\
D = 96 \times 7 / 864 = 0.7778 < 1 \text{ for } n = 96 \text{ and } \lceil \frac{lg M}{} \rceil = 864, \\
D = 112 \times 7 / 1030 = 0.7612 < 1 \text{ for } n = 112 \text{ and } \lceil \frac{lg M}{} \rceil = 1030, \\
D = 128 \times 8 / 1216 = 0.8421 < 1 \text{ for } n = 128 \text{ and } \lceil \frac{lg M}{} \rceil = 1216.
\]

The above values mean that the original solution to an ASSP may possibly be found through the LLL lattice basis reduction algorithm [5][6]. However, it is uncertain to find the original solution to the

*This work is supported by MOST with Project 2007CB311100 and 2009AA01Z441. Corresponding email: reesse@126.com.
ASSP since the density $D < 1$ of an ASSP only assure that the shortest vector is unique in a related lattice, and cannot assure that the vector of the original solution is just the shortest vector or an approximately shortest vector which will occur in the final reduced lattice basis.

The LLL reduction algorithm is famous for it has a fatal threat to the classical MH knapsack cryptosystem [7] which produces a ciphertext in the form of a subset sum problem.

To avoid the low density of a knapsack from an ASPP and to decrease the modulus length of a cryptoscheme, on the basis of REESSE1+, we propose a new cryptoscheme called JUNA which treats a bit-pair as an operation unit and introduces random ingredients when a bit string is encrypted, and moreover it is proved to be semantically secure.

Throughout this paper, unless otherwise specified, $n \geq 80$ is the bit-length of a plaintext block, $b \geq 144$ is the item-length of a public key sequence, the sign % denotes “modulo”, $M$ does “$M - 1$” with $M$ prime, $\lg x$ means the logarithm of $x$ to the base 2, $\neg$ does the opposite value of a bit, $\mathbf{P}$ does the maximal prime allowed in coprime sequences, $|x|$ does the absolute value of a number $x$, $|S|$ does the order of an element $x \ % \ M$, $|S|$ does the size of a set $S$, and $\gcd(a, b)$ represents the greatest common divisor of two integers. Without ambiguity, “% $M$” is usually omitted in expressions.

2 Several Definitions

The following definitions lay the stone foundation for the new public key encryption scheme.

2.1 A Coprime Sequence

**Definition 1:** If $A_1, \ldots, A_n$ are $n$ pairwise distinct positive integers such that $\forall A_i, A_j \ (i \neq j)$, either $\gcd(A_i, A_j) = 1$ or $\gcd(A_i, A_j) = F \neq 1$ with $(A_i / F) \ \forall A_k \ \forall k \neq i, j \in [1, n]$, these integers are called a coprime sequence, denoted by $\{A_1, \ldots, A_n\}$, shortly $\{A_n\}$.

Notice that the elements of a coprime sequence are not necessarily pairwise coprime, but a sequence of which the elements are pairwise coprime is a coprime sequence.

For example, $\{13, 2, 23, 11, 17, 21, 15\}$ and $\{29, 7, 11, 23, 19, 13, 5, 17\}$ are two coprime sequences separately.

**Property 1:** Let $\{A_1, \ldots, A_n\}$ be a coprime sequence. If randomly select $k \in [1, n]$ elements $A_{x_1}, \ldots, A_{x_k}$ from the sequence, then the mapping from a subset $\{A_{x_1}, \ldots, A_{x_k}\}$ to a subset product $G = \prod_{i=1}^{k} A_{x_i}$ is one-to-one, namely the mapping from $b_1\ldots b_k$ to $G = \prod_{i=1}^{k} A_{x_i}^{b_i}$ is one-to-one, where $b_1\ldots b_k$ is a bit string.

Refer to [1] for its proof.

2.2 A Bit Shadow

**Definition 2:** Let $b_1\ldots b_n \neq 0$ be a bit string. Then $b_i$ with $i \in [1, n]$ is called a bit shadow if it comes from such a rule: $\circ$ $b_i = 0$ if $b_i = 0$, $\triangledown$ $b_i = 1 + $ the number of successive 0-bits before $b_i$ if $b_i = 1$, or $\Box$ $b_i = 1 + $ the number of successive 0-bits before $b_i$ if $b_i = 0$ after the rightmost 1-bit if $b_i$ is the leftmost 1-bit.

Notice that $\circ$ of this definition is slightly different from that in [1].

For example, let $b_{10} = 1001000010011000$ or $0010010011000100$, then $b_{10} = 3003000006003100$ or $00500300031000400$.

**Fact 1:** Let $b_1\ldots b_n$ be the bit shadow string of $b_1\ldots b_n \neq 0$. Then there is $\sum_{i=1}^{n} b_i = n$.

**Proof:**

According to Definition 2, every bit of $b_1\ldots b_n$ is considered into $\sum_{i=1}^{n} b_i$, where $k \leq n$, and $b_{x_1}, \ldots, b_{x_k}$ are 1-bit shadows in the string $b_1\ldots b_n$, and thus there is $\sum_{i=1}^{n} b_i = n$.

On the other hand, there is $\sum_{i=1}^{n} b_i = 0$, where $b_{x_1}, \ldots, b_{x_n}$ are 0-bit shadows.

In total, there is $\sum_{i=1}^{n} b_i = n$. \hfill \Box

**Property 2:** Let $\{A_1, \ldots, A_n\}$ be a coprime sequence, and $b_1\ldots b_n$ be the bit shadow string of $b_1\ldots b_n \neq 0$. Then the mapping from $b_1\ldots b_n$ to $G = \prod_{i=1}^{n} A_{x_i}^{b_i}$ is one-to-one.

**Proof:**

Let $b_1\ldots b_n$ and $b'_{1\ldots b_n}$ be two different nonzero bit strings, and $b_1\ldots b_n$ and $b'_1\ldots b'_n$ be the two corresponding bit shadow strings.

$\circ$ If $b_1\ldots b_n = b'_1\ldots b'_n$, then by Definition 2, there is $b_1\ldots b_n = b'_1\ldots b'_n$. \hfill \Box
Again, for any arbitrary bit shadow string $b_1…b_n$, there always exists a preimage $b_1…b_n$, namely the mapping from $b_1…b_n$ to $b_1…b_n$ is surjective.

Thus, the mapping from $b_1…b_n$ to $b_1…b_n$ is one-to-one.

$\bigcirc$. Obviously the mapping from $b_1…b_n$ to $\Pi_{i=1}^{n} A_i^\mathbf{b}$ is surjective.

Again, presuppose that $\Pi_{i=1}^{n} A_i^\mathbf{b} = \Pi_{i=1}^{n} A_i^\mathbf{b'}$ for $b_i \neq b_i'$.

Since $\{A_i, \ldots, A_n\}$ is a coprime sequence, and $A_i^\mathbf{b}$ either equals 1 with $b_i = 0$ or contains the same prime factors as those of $A_i$ with $b_i \neq 0$, we can obtain $b_1…b_n = b_1…b_n'$ from $\Pi_{i=1}^{n} A_i^\mathbf{b}$ = $\Pi_{i=1}^{n} A_i^\mathbf{b'}$. It is in direct contradiction to $b_1…b_n \neq b_1…b_n'$, which indicates that the mapping from $b_1…b_n$ to $\Pi_{i=1}^{n} A_i^\mathbf{b}$ is injective [8].

Thus, the mapping from $b_1…b_n$ to $\Pi_{i=1}^{n} A_i^\mathbf{b}$ is one-to-one.

By transitivity, the mapping from $b_1…b_n$ to $\Pi_{i=1}^{n} A_i^\mathbf{b}$ is also one-to-one. $\Box$

### 2.3 A Bit-pair Shadow

It is well understood that a public key cryptosystem is mainly used for transmitting a symmetric key. Assume that $b_1…b_n$ is a symmetric key. At present, to guard against exhaustive search namely brute force attack, $n$ should be no less than 80 [9].

To make the modulus $M$ of the new cryptoscheme comparatively small, we will utilize the idea of a bit-pair string with 2 bits to 3 items.

In this way, the length of a coprime sequence is changed to $3n/2$, namely $\{A_1, \ldots, A_n\}$ is substituted with $\{A_1, A_2, A_3, \ldots, A_{3n/2-2}, A_{3n/2-1}, A_{3n/2}\}$ that may be logically orderly partitioned into $n/2$ triples of which each comprises 3 elements: $A_{3j-2}, A_{3j-1}, A_{3j}$ with $j \in [1, n/2]$. Likewise, a non-coprime sequence $\{C_1, \ldots, C_n\}$ is substituted with $\{C_1, C_2, C_3, \ldots, C_{3n/2-2}, C_{3n/2-1}, C_{3n/2}\}$, where $(C_{3j-2}, C_{3j-1}, C_{3j})$ with $j \in [1, n/2]$ is acquired from $(A_{3j-2}, A_{3j-1}, A_{3j})$ and other private parameters.

**Definition 3:** Let $\{A_{3j-2}, A_{3j-1}, A_{3j}\} \mid j = 1, \ldots, n/2$ be a coprime sequence. Orderly partition a bit string $b_1…b_n$ into $n/2$ pairs $B_1, \ldots, B_{n/2}$, where $B_i$ with $j \in [1, n/2]$ has four states: 00, 01, 10, and 11 which correspond to 1, $A_{3j-1}, A_{3j-1},$ and $A_{3j}$ respectively. Then $B_1, \ldots, B_{n/2}$ is called a bit-pair string, shortly $B_1…B_{n/2}$.

**Property 3:** Let $\{A_{3j-2}, A_{3j-1}, A_{3j}\} \mid j = 1, \ldots, n/2$ be a coprime sequence, and $B_1…B_{n/2}$ be a nonzero bit-pair string. Then the mapping from $B_1…B_{n/2}$ to $G = \Pi_{i=1}^{n/2} (A_{3(i-1)+j})^{\mathbf{b}_i/3}$ with $A_0 = 1$ is one-to-one, where $|B_i/3| = 0$ or 1, and $G$' is called a coprime subsequence product.

Its proof is parallel to that of Property 1 in [1].

For example, let $n = 8$, $\{A_1, A_2, A_3, \ldots, A_{10}, A_{11}, A_{12}\} = \{13, 7, 23, 6, 11, 29, 17, 31, 37, 41, 19, 5\}$, and $b_1…b_{12} = 11011000$, then $B_1…B_4 = 11 01 10 00$, and $G' = \Pi_{i=1}^{n/2} (A_{3(i-1)+1})^{b_i/3} = A_4A_4A_8 = 23 \times 6 \times 31 \times 1 = 4278$.

**Definition 4:** Let $B_1…B_{n/2}$ be a nonzero bit-pair string. Then $B_i$ with $i \in [1, n/2]$ is called a bit-pair shadow if it comes from such a rule: $\bigcirc$ $B_i = 0$ if $B_i = 00$, $\bigcirc$ $B_i = 1$ + the number of successive 00-pairs before $B_i$ if $B_i \neq 00$, or $\bigcirc$ $B_i = 1$ + the number of successive 00-pairs before $B_i$ + the number of successive 00-pairs after $B_i$ is the leftmost non-00-pair if $B_i$ is the leftmost non-00-pair.

For example, let $n = 16$, and $B_1…B_8 = 100100000101000$ or 0010010011001001, then $B_1…B_8 = 21003020$ or 03102020.

**Fact 2:** Let $B_1…B_{n/2}$ be the bit-pair shadow string of $B_1…B_{n/2}$ ≠ 0. Then there is $\sum_{i=1}^{n/2} b_i = n/2$.

**Proof:**

According to Definition 4, every pair of $B_1…B_{n/2}$ is considered into $\sum_{i=1}^{n/2} b_i$, where $k \leq n/2$, and $B_{n/2}, \ldots, B_1$ are non-00-pair shadows in the string $B_1…B_{n/2}$, and thus there is $\sum_{i=1}^{n/2} b_i = n/2$.

On the other hand, there is $\sum_{i=1}^{n/2-k} b_i = 0$, where $B_{n/2}, \ldots, B_{n/2-k}$ are 00-pair shadows.

In total, there is $\sum_{i=1}^{n/2} b_i = n/2$. $\Box$

**Property 4:** Let $\{A_{3j-2}, A_{3j-1}, A_{3j}\} \mid j = 1, \ldots, n/2$ be a coprime sequence, and $B_1…B_{n/2}$ be the bit-pair shadow string of $B_1…B_{n/2}$ ≠ 0. Then the mapping from $B_1…B_{n/2}$ to $G = \Pi_{i=1}^{n/2} (A_{3(i-1)+j})^\mathbf{b_i}$ with $A_0 = 1$ is one-to-one, where $G$ is called an anomalus coprime subsequence product.

Its proof is parallel to that of Property 2 in Section 2.2.

For example, let $n = 8$, $\{A_1, A_2, A_3, \ldots, A_{10}, A_{11}, A_{12}\} = \{13, 7, 23, 6, 11, 29, 17, 31, 37, 41, 19, 5\}$,
and $B_1…B_4 = 11\ 01\ 10\ 00$, then $B_1…B_4 = 2110$, and $G = \prod_{i=1}^{\ell/4} (A_{4(i-1)+i})^{\ell/3} = (A_3)^2A_4A_1 = 23^2\times 6\times 31\times 1 = 98394$.

Property 3 and 4 manifest that $G'$ or $G$ may act as a component of a trapdoor function under bit-pair string circumstances.

2.4 A Lever Function

**Definition 5**: The secret parameter $\ell(i)$ in the key transform of a public key cryptoscheme is called a lever function, if it has the following features:

- $\ell(.)$ is an injection from the domain $\{1, \ldots, n\}$ to the codomain $\Omega \subset \{5, \ldots, M\}$, where $M$ is large;
- the mapping between $i$ and $\ell(i)$ is established randomly without an analytical expression;
- an attacker has to be faced with all the arrangements of $n$ elements in $\Omega$ when extracting a related private key from a public key;
- the owner of a private key only needs to consider the accumulative sum of $n$ elements in $\Omega$ when recovering a related plaintext from a ciphertext.

The latter two points manifest that if $n$ is large enough, it is infeasible for the attacker to search all the permutations of elements in $\Omega$ exhaustively while the decryption of a normal ciphertext is feasible in some time being polynomial in $n$. Thus, there are the large amount of calculation on $\ell(.)$ at “a public terminal”, and the small amount of calculation on $\ell(.)$ at “a private terminal”.

Notice that $\ell(\ell(i))$ in modular $M$ arithmetic, $-x$ represents $M - x$; $\ell$ the number of elements of $\Omega$ is not less than $\ell$; $\ell$ considering the speed of decryption, the absolute values of all the elements should be comparatively small; $\ell$ the lower limit 5 will make seeking the root $W$ from $W^{\ell(i)} = A_2^{-1}C_i (\% M)$ face an unsolvable Galois group when the value of $A_2 \leq 1201$ is guessed [10].

Concretely to the new cryptoscheme, $\ell(i)$ in the key transform $C_i = (A_iW^{\ell(i)} , (\% M))$ with $i \in [1, n]$ is an exponent.

**property 5 (Indeterminacy of $\ell(.)$)**: Let $\delta = 1$ and $C_i = (A_iW^{\ell(i)} , (\% M))$ with $\ell(i) \in \Omega = \{5, 6, \ldots, n + 4\}$ and $A_i = A = \{2, 3, \ldots, P\}$ $P \leq 1201$ for $i = 1, \ldots, n$. Then $\forall W (|W| \neq M)$ is in $(1, M)$ and $\forall x, y, z (x \neq y \neq z) \in [1, n]$,

- $\bigcirc$ when $\ell(x) + \ell(y) = \ell(z)$, there is $\ell(x) + |W| + \ell(y) + |W| = \ell(z) + |W| (\% M)$;
- $\bigcirc$ when $\ell(x) + \ell(y) = \ell(z)$, there always exist $C_i = A_i'W^{\ell(i)} (\% M)$, $C_y = A_y'W^{\ell(i)} (\% M)$, and $C_z = A_z'W^{\ell(i)} (\% M)$ such that $\ell'(x) + \ell'(y) = \ell'(z) (\% M)$ with $A_i' \leq P$.

Refer to [1] for its proof.

Notice that according to the proof of Property 5 in [1], it is not difficult to understand that when $\Omega = \{5, 6, \ldots, n + 4\}$ is substituted with $\Omega = \{\pm/-5, \pm/-7, \ldots, \pm/-2(n + 3)\}$, where “$\pm/-”” means the selection of the “$+\”$ or “$-\”$ sign, Property 5 still holds.

3 Design of the New Cryptoscheme

Due to $L_2[1/3, 1.923] = 2^{20}$ with $\rho$ prime and $|\lg \rho| = 1024$ [11], the shortest bit-length of a plaintext block should be 80 which makes the time complexity of an exhaustive search attack on the plaintext block reach $2^{80}$.

In the new scheme, to acquire provable semantical security, 16 random bits are appended the terminal of a plaintext block of $n$ bits when it is encrypted.

Let $\bar{n} = n + 16$ with $n = 80, 96, \text{or} 112$.

Additionally, two adjacent bits are orderly treated as a unit, namely a bit-pair string $B_1…B_{n/2}$ is used to represent a plaintext block $b_1…b_n \neq 0$.

3.1 Key Generation Algorithm

Considering decryption speed, the absolute values of elements of $\Omega$ should be as small as possible, and every three successive elements of $\Omega$ should be in a triple conforming with 2 bits to 3 items.

Let $A = \{2, \ldots, P\}$, where $P = 937, 991, \text{or} 1201$ corresponds to $\bar{n} = 96, 112, \text{or} 128$ separately.

Let $I = [\lg M] = 464, 544, \text{or} 640$ also corresponds to $\bar{n} = 96, 112, \text{or} 128$ separately.

Assume that $\bar{A}$ is the maximum in $(A_{y-2}, A_{y-1}, A_y) \forall y \in [1, \bar{n}/2]$.

The following algorithm is generally employed by the owner of a key pair.
INPUT: the integer \( n \); the integer \( i \); the prime \( P \).

S1: Let \( \hat{n} \leftarrow n + 16 \), \( A \leftarrow \{2, \ldots, P\} \).
Yield the first \( \hat{n} \) primes in the natural number set \( p_1, \ldots, p_{\hat{n}} \).
Yield \( \Omega \leftarrow \{(+/-(6j-1), +/-(6j+1), +/-(6j+3)) \mid j = 1, \ldots, \hat{n}/2 \} \).

S2: Produce an odd coprime sequence \( \{A_1, \ldots, A_{3\hat{n}/2} \mid A_i \in A \} \).
Formally \( \{A_1, \ldots, A_{3\hat{n}/2} \} = \{A_{3j-2}, A_{3j-1}, A_{3j} \mid j = 1, \ldots, \hat{n}/2 \} \).
Arrange \( A_1, \ldots, A_{3\hat{n}/2} \) to \( A_{\hat{n}/2}, \ldots, A_3 \) in descending order.

S3: Find a prime \( M > \hat{A}_1^A \hat{A}_2 \hat{A}_3 \hat{A}_4 \hat{A}_5 \) making \( [\lg M] = \hat{i} \) and
\( \prod_{i=1}^{\hat{n}/2} p_i \mid M \), where \( k \) meets \( \prod_{i=1}^{\hat{n}/2} e_i \geq 2^{10} \) and \( p_k < \hat{n} \).

S4: Produce pairwise distinct \( (\ell(3j-2), \ell(3j-1), \ell(3j)) \in \Omega \)
for \( j = 1, \ldots, \hat{n}/2 \).

S5: Stochastically pick \( W \in (1, \hat{M}) \) making \( |W| \geq 2^{n-20} \).
Stochastically pick \( \delta \in (1, \hat{M}) \) making \( \gcd(\delta, \hat{M}) = 1 \).

S6: Compute \( C_i \leftarrow (A_i W^{\ell(i)})^{\hat{M}} \times M \) for \( i = 1, \ldots, 3\hat{n}/2 \).

OUTPUT: a public key \( \{C_1, \ldots, C_{3\hat{n}/2}, M\} \); a private key \( \{A_1, \ldots, A_{3\hat{n}/2}, W, \delta, M\} \).
The lever function \( \ell(1), \ldots, \ell(\hat{n}/2) \) is discarded but must not be divulged.

Notice that
\( \prod \) at S1, \( \Omega = \{(+/-(6j-1), +/-(6j+1), +/-(6j+3)) \mid j = 1, \ldots, \hat{n}/2 \} \) indicates that \( \Omega \) is one of \((3!)^{3/2} \hat{n}/2 \) potential sets consisting of 3-tuple elements, where “+/−” means the selection of the “+” or “−” sign, and the subscript \( P \) means that \((+/-(6j-1), +/-(6j+1), +/-(6j+3)) \) is a permutation of \((+/-(6j-1), +/-(6j+1), +/-(6j+3)) \).
\( \prod \) at S2, \( \gcd(A_{3j-2}, A_{3j-1}, A_{3j}) \neq 1 \) (i.e., \( [3, 3, 3] \)) is allowed — \((3^2, 3, 3) \) for example since only one of three elements will occur in the product \( G \).
\( \prod \) at S3, the inequation \( M > \hat{A}_1^A \hat{A}_2 \hat{A}_3 \hat{A}_4 \hat{A}_5 \) assures that a ciphertext can be decrypted correctly;
\( \prod \) at S5, let \( W \in \mathbb{E}^{|W|} (\% M) \), then \( |W| = \hat{M} / \gcd(\hat{M}, \hat{M}/w) \) [10], where \( F \geq 2^{n-20} \) is a factor of \( \hat{M} \) and \( g \) is a generator by Algorithm 4.80 in Section 4.6 of [11].

**Definition 6:** Given \( \{C_1, \ldots, C_{3\hat{n}/2}, M\} \), seeking the original \((A_1, \ldots, A_{3\hat{n}/2}, \ell(1), \ldots, \ell(\hat{n}/2)), W, \delta \) from \( C_i \leftarrow (A_i W^{\ell(i)})^{\hat{M}} \times M \) with \( A_i \in A = \{2, \ldots, P \} \mid P \leq 1201 \) and \( \ell(i) \) from \( \Omega = \{(+/-(6j-1), +/-(6j+1), +/-(6j+3)) \mid j = 1, \ldots, \hat{n}/2 \} \) for \( i = 1, \ldots, 3\hat{n}/2 \) is referred to as a multivariate permutation problem (MPP).

### 3.2 Encryption Algorithm

This algorithm is employed by a person who wants to encrypt plaintexts.

INPUT: a public key \( \{C_1, \ldots, C_{3\hat{n}/2}, M\} \);
the bit-pair string \( B_1 \ldots B_{\hat{n}/2} \) of a plaintext block \( b_1 \ldots b_{\hat{n}/2} \).

Notice that if the number of 00-pairs in \( B_1 \ldots B_{\hat{n}/2} \) is larger than \( n/4 \), let \( b_1 \ldots b_{\hat{n}/2} = -b_1 \ldots -b_{\hat{n}/2} \) in order that a related ciphertext can be decrypted conforming to the constraint on \( M \).

S1: Yield a random bit string \( b_{\hat{n}/2+1} \ldots b_n \) appended to \( b_1 \ldots b_{\hat{n}/2} \),
and form \( B_1 \ldots B_{\hat{n}/2} \) with the number of 00-pairs \( \leq \hat{n}/4 \).

S2: Set \( C_0 \leftarrow 1, k \leftarrow 0, i \leftarrow 1, s \leftarrow 0 \).

S3: If \( B_i = 00 \) then
S3.1: let \( k \leftarrow k + 1, B_i \leftarrow 0 \)
else
S3.2: let \( B_i \leftarrow k + 1, k \leftarrow 0 \);
S3.3: if \( s = 0 \) then \( s \leftarrow i \) else null.

S4: Let \( i \leftarrow i + 1 \).
If \( i \leq \hat{n}/2 \) then goto S3.

S5: If \( k \neq 0 \) then let \( B_{i} \leftarrow B_{i} + k \).

S6: Stochastically produce \( r_1 \ldots r_{\hat{n}/2} \in \{0, 1\}^{\hat{n}/2} \),
and set \( r_i \leftarrow 1 \).

S7: Compute \( \hat{G} \leftarrow \prod_{i=1}^{\hat{n}/2} (C_{(3j-1)+\hat{n}/2} + r_{(3j-1)+\hat{n}/2})^b \times M \).

OUTPUT: a ciphertext \( \hat{G} \).
Obviously, a different ciphertext will be outputted every time an identical plaintext is inputted repeated.

The identical plaintext may correspond to at most \(2^{2^n-2^n} \) different ciphertexts because \( n/2 \) bit-pairs may be interlaced by a 00-pair and a non-00-pair, and \( b_{n+1} \ldots b_n \) is produced randomly. It will take the running time of \( O(n^2 \cdot 2^n \cdot \log^2 M) \) bit operations exhaustively to search all the possible ciphertexts of a plaintext.

Notice that a JUNA ciphertext \( \prod_{i=1}^{b_n/2} (C_i^{(3(i-1)+B) + \cdots + (3(-1) - B)})^\delta \) \( (\% M) \) is different from a Naccache-Stern ciphertext \( c = \prod_{i=1}^{b_n/2} v_i^\delta \) \( (\% M) \) with \( \delta \) prime is a public key.

**Definition 7:** Given \( (C_1, \ldots, C_{b_n/2}, M) \) and \( \delta \), seeking \( B_1 \ldots B_{b_n/2} \) from \( \delta = \prod_{i=1}^{b_n/2} (C_i^{(3(i-1) + B)})^{\delta/3} \) \( (\% M) \) with \( C_0 = 1 \) is referred to as a subset product problem (SPP), where \( B_1 \ldots B_{b_n/2} \) is the bit-pair string of \( b_1 \ldots b_n \neq 0 \).

**Definition 8:** Given \( (C_1, \ldots, C_{b_n/2}, M) \) and \( \delta \), seeking \( B_1 \ldots B_{b_n/2} \) from \( \delta = \prod_{i=1}^{b_n/2} (C_i^{(3(i-1) + B)})^{\delta} \) \( (\% M) \) or \( \delta = \prod_{i=1}^{b_n/2} (C_i^{(3(i-1) + B)})^{\delta} \) \( (\% M) \) with \( C_0 = 1 \) and \( r_1 \ldots r_{b_n/2} \) a random bit string is referred to as an anomalous subset product problem (ASPP), where \( B_1 \ldots B_{b_n/2} \) is the bit-pair shadow string of \( B_1 \ldots B_{b_n/2} \) corresponding to \( b_1 \ldots b_n \neq 0 \).

### 3.3 Decryption Algorithm

This algorithm is employed by a person who wants to decrypt ciphertexts.

**INPUT:** a private key \( (A_1, \ldots, A_{b_n/2}) \), \( W, \delta, M \); a ciphertext \( \delta \).

It should be noted that due to \( 2 \left| \sum_{i=1}^{b_n/2} B_i \right| + 2 \left( \ell(r_{3(i-1) + B}) + \cdots + r_{3(-1) + B} \right) \) for \( i \in [1, n/2] \) with \( \ell(0) = 0, \delta = \sum_{i=1}^{b_n/2} \ell(r_{3(i-1) + B} + \cdots + r_{3(-1) + B}) \) must be even.

**S1:** Compute \( Z_0 \leftarrow \delta^{\delta/3} \% M \).

**S2:** If \( \delta Z_0 \) then do \( Z_0 \leftarrow Z_0 \cdot W^{2^{i+1} \delta} \% M \), goto S2.

**S3:** Set \( B_1 \ldots B_{b_n/2} = 0, j \leftarrow 0, k \leftarrow 0, l \leftarrow 0, i \leftarrow 1, G \leftarrow Z_0 \).

**S4:** If \( (A_{3i-j})^{\delta/3} \% G \) then let \( l \leftarrow l + 1 \), goto S4.

**S5:** Let \( j \leftarrow j + 1 \).

If \( l = 0 \) and \( j \leq 2 \) then goto S4.

**S6:** If \( l = 0 \) then

| S6.1 | Let \( k \leftarrow k + 1, i \leftarrow i + 1 \) else
| S6.2 | Compute \( G \leftarrow G/\left( A_{3i-j} \right)^\delta \).
| S6.3 | If \( k > 0 \) or \( l \geq i \) then
| S6.4 | Let \( B_i \leftarrow 3 - j, i \leftarrow i + 1 \) else
| S6.5 | Let \( B_{i+1} \leftarrow 3 - j, i \leftarrow i + 1 \;
| S6.6 | Set \( l \leftarrow 0, k \leftarrow 0 \).

**S7:** If \( l \leq n/2 \) and \( G \neq 1 \) then set \( j \leftarrow 0 \), goto S4.

**S8:** If \( G \neq 1 \) then set \( h \leftarrow -h \), do \( Z_0 \leftarrow Z_0 \cdot W^{2^{i+1} \delta} \% M \), goto S2.

**S9:** Separate \( B_1 \ldots B_{b_n/2} \) from \( B_1 \ldots B_{b_n/2} \).

**OUTPUT:** a related plaintext \( B_1 \ldots B_{b_n/2} \), namely \( b_1 \ldots b_n \).

Notice that only if \( \delta \) is a true ciphertext, can the algorithm always terminate normally, and \( b_1 \ldots b_n \) will be original although \( r_1 \ldots r_{b_n/2} \) is brought into an encryption process.

### 4 Correctness, Uniqueness, and Complexity

In this section, we will discuss whether a ciphertext can be decrypted correctly, a plaintext solution is unique, and a decryption process can be finished in polynomial time.

#### 4.1 Correctness of the Decryption Algorithm

Because \( \mathbb{Z}_M^* \) is an Abelian group, namely a commutative group, \( \forall k \in [1, M] \), there is

\[
W^k (W^{-1})^k = W^k (W^k)^{-1} = 1 \quad (\% M),
\]

where \( W \in [1, M] \) is any arbitrary integer.
Fact 3: Let $k = \sum_{i=1}^{b_2} \mathcal{B}_i \ell(r_i(3(i-1)+B)+-r_i(3(i-B)+B)) \bmod M$ with $\ell(0) = 0$, where $\mathcal{B}_1,\ldots,\mathcal{B}_{b_2}$ is the bit-pair shadow string of $B_1,\ldots,B_{b_2}$ corresponding to $b_1,\ldots,b_2 \neq 0$, and $r_1,\ldots,r_{b_2}$ is a random bit string. Then $G^{-k} (W^{-1})^k = \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta \bmod M$.

Proof:
Let $b_1,\ldots,b_2$, namely $B_1,\ldots,B_{b_2}$ be a plaintext of $n$ bits. Additionally, let $A_0 = 1$.

According to the key generator, the encryption algorithm, and $\sum_{i=1}^{b_2} \mathcal{B}_i = n/2$, there is

$$G = \prod_{i=1}^{b_2} (C_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta = \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta W^{(r_i(3(i-1)+B)+-r_i(3(i-B)+B))}^\theta \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta = W^k \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta \bmod M.$$

Further, raising either side of the above congruence to the $\delta^{-1}$-th power yields

$$G^\delta = (W^k)^{\delta^{-1}} \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta (W^{-1})^k \bmod M.$$

Multiplying either side of the just above congruence by $(W^{-1})^k$ yields

$$G^{-k} (W^{-1})^k = W^k \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta (W^{-1})^k = \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta \bmod M.$$

Clearly, the above process also gives a method of seeking $G$ at one time. Notice that in practice, $k$ is unknowable in advance.

However, because $|k| < \delta 2(3 \delta/2 + 3/2) = 3\delta(n+1)/2$ is comparatively small, we may search $k$ heuristically by multiplying $W^{-1}$ or $W^k \bmod M$ and judging whether $G = 1$ after it is divided exactly by some $(A_{r_1,\ldots})$. It is known from the decryption algorithm that the original $B_1,\ldots,B_{b_2}$ will be acquired at the same time the condition $G = 1$ is satisfied.

4.2 Uniqueness of a Plaintext Solution

Because the public key $\{C_1,\ldots,C_{b_2}\}$ is a non-coprime sequence, the mapping from $B_1,\ldots,B_{b_2}$ to $G = \prod_{i=1}^{b_2} (C_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta \bmod M$ is one-to-one. It might possibly result in the nonuniqueness of a plaintext solution $B_1,\ldots,B_{b_2}$ when $G$ is being unveiled.

Suppose that a ciphertext $G$ can be obtained respectively from two different bit-pair strings $B_1,\ldots,B_{b_2}$ and $B_1',\ldots,B_{b_2}'$. Then,

$$G = \prod_{i=1}^{b_2} (C_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta = \prod_{i=1}^{b_2} (C_{r_i(3(i-1)+B')+-r_i(3(i-B')+B')})^\theta \bmod M.$$

That is,

$$\prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta W^{r_i(3(i-1)+B)+-r_i(3(i-B)+B)}^\theta = \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B')+-r_i(3(i-B')+B')})^\theta W^{r_i(3(i-1)+B')+-r_i(3(i-B')+B')}^\theta \bmod M.$$

Further, there is

$$W^{k_1} \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta = W^{k_2} \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B')+-r_i(3(i-B')+B')})^\theta \bmod M,$$

where $k = \sum_{i=1}^{b_2} \mathcal{B}_i \ell(r_i(3(i-1)+B)+-r_i(3(i-B)+B))$, and $k' = \sum_{i=1}^{b_2} \mathcal{B}_i \ell(r_i(3(i-1)+B')+-r_i(3(i-B')+B')) \bmod \ell(0) = 0$.

Raising either side of the above congruence to the $\delta$-th power yields

$$W^{k_1} \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B)+-r_i(3(i-B)+B)})^\theta = W^{k_2} \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B')+-r_i(3(i-B')+B')})^\theta \bmod M.$$

Without loss of generality, let $k \geq k'$. Because $(\sum_{i=1}^{b_2} \mathcal{B}_i \ell(r_i(3(i-1)+B')+-r_i(3(i-B')+B'))) \bmod \ell(0) = 0$.

$$\theta = \prod_{i=1}^{b_2} (A_{r_i(3(i-1)+B')+-r_i(3(i-B')+B')})^\theta \bmod M,$$

namely $\theta = W^k - k' \bmod M$.

This congruence signifies when the plaintext $B_1,\ldots,B_{b_2}$ is not unique, the value of $W$ must be relevant to $\theta$. The contrapositive assertion equivalent to it is that if the value of $W$ is irrelevant to $\theta$, $B_1,\ldots,B_{b_2}$ will be unique. Thus, we need to consider the probability that $W$ takes a value relevant to $\theta$.

If an adversary tries to attack an 80-bit symmetric key through the exhaustive search, and a computer
can verify trillion values per second, then it will take 38334 years for the adversary to verify all the potential values. Hence, currently 80 bits are quite enough for the security of a symmetric key.

$B_1 \ldots B_{\theta/2}$ contains $\theta$ bits which indicates $\prod_{i=1}^{\theta/2} (C_{\theta}(3(\theta-1)+B_i)+\neg r(3(\theta-1)+B_i))$ has $2^\theta$ potential values, and thus the number of potential values of $\theta$ is at most $2^\theta \times 2^\theta$. Notice that because $A_1^{-1}, \ldots, A_{\theta/2}^{-1}$ are not necessarily coprime, some values of $\theta$ may possibly occur repeatedly.

Because $|k-k'| < 3\theta \theta + 1) \leq 47601 \approx 2^{16}$ with $\theta \leq 128$, and $W$ has at most $2^{16}$ solutions to every $\theta$, the probability that $W$ takes a value relevant to $\theta$ is at most $2^{16} \times 2^\theta / M$.

When $\theta \geq 96$, there is $2^{16} \times 2^\theta / M \leq 2^{208} / 2^{464} = 1 / 2^{256}$ (notice that when $\theta = 96, 112, \text{ or } 128$, there is $\lfloor \log_2 M \rfloor = 646, 544, \text{ or } 640$), which is close to zero. The probability will further decrease when $W$ is a prime since the solutions to $W$ lean to being composite integers in the average case.

In addition, if you please, resorting to $\sum_{i=1}^{\theta/2} B_i = \theta/2$, may exclude some unoriginal plaintext solutions.

4.3 Running Times of the Algorithms

The running time of an algorithm is measured in the number of bit operations [11], and it has an asymptotic implication. According to [11], a modular addition will take $O(\log M)$ bit operations (shortly, bos), and a modular multiplication will take $O(2 \log^2 M)$ bos.

4.3.1 Running Time of the Key Generator

In the key generator, the steps which exert dominant effects on running time are S5 and S6.

At S5, seeking $W$ namely seeking $g$ will take $O(6(\log \log M) \log^3 M)$ bos [11]. For every $i$, S6 contains one modular power $(A_i W^{(i)} s)$, where $W^{(i)}$ will take $2\log(3\theta/2)$ modular multiplications which is subject to one modular power since $\theta$ is far smaller than $M$. Thus, the running time of the key generator is about $O(6(\log \log M + 4(3\theta/2))\log^2 M) = O(6(\log \log M + n)\log^2 M)$ bos.

4.3.2 Running Time of the Encryption Algorithm

In the encryption algorithm, the dominant step is S7.

Due to $\sum_{i=1}^{\theta/2} B_i = \theta/2$, S7 contains at most $\theta/2$ modular multiplications. Thus, the running time of the encryption algorithm is $O(\theta \log^2 M)$ bos.

4.3.3 Running Time of the Decryption Algorithm

In the decryption algorithm, the dominant step is S2 which pairs with S8 to form a loop. S1 of a modular power is subject to the loop S2 $\leftrightarrow$ S8.

It is easy to see that the number of times of executing the loop S2 $\leftrightarrow$ S8 which mainly contains a modular multiplication is $k = \sum_{i=1}^{\theta/2} B_i \ell(r(3(i-1) + B_i) + \neg r(3(i-1) + B_i))$, where $B_i$ is relevant to a plaintext block, and $\ell(r(3(i-1) + B_i) + \neg r(3(i-1) + B_i))$ is relevant to the set $\Omega = \{(+/(6j + 1), +/-(6j + 3))| j = 1, \ldots, \theta/2 \}$ which is indeterminate. Therefore, it is very difficult accurately to know the value of $k$.

When a plaintext block $B_1 \ldots B_{\theta/2}$ contains $\theta/4$ successive 00-pairs, we may obtain the maximal value of $k$ on condition that every $\ell(r(3(i-1) + B_i) + \neg r(3(i-1) + B_i))$ has the same operating sign.

The maximal value of $k$ is

\[
k = (\theta/4 + 1)2(3\theta/2) + 2(3\theta/2) - 3 + (23\theta/2) - 9 + \ldots + (23\theta/2) + 3 - 6(\theta/4 - 1) = (3/4)(\theta + 4)(\theta + 1) + (3/16)(3\theta + 4)(\theta - 4) = (3/16)\theta(7\theta + 12).
\]

Considering that $\ell(r(3(i-1) + B_i) + \neg r(3(i-1) + B_i))$ may be positive or negative, the minimal value of $k$ will be 0 with a suitable plaintext block $B_1 \ldots B_{\theta/2}$.

Thus, the simply expected value of $k$ is $(3/32)\theta(7\theta + 12) \approx (21/32)\theta^2$.

Again considering that $W^{-2}$ and $W^2$ is multiplied alternately every time, the simply expected value of $k$ should be $2(21/32)\theta^2 \approx \theta^2$.

In summary, the simply expected time of the decryption algorithm is $O(2\theta^2 \log^2 M)$ bos.

In practice, because the possible values of $k$ (including the repeated) will distribute at $(3/32)\theta(7\theta + 12)$ integral points 0, 2, 4, ..., $(3/16)\theta(7\theta + 12)$ which is the maximal possible range, and the probability that $k$ takes large integers is comparatively small, the concrete running time of a decryption process will be far smaller than $O(2\theta^2 \log^2 M)$ bos.
5 Analysis of Security of a Private Key

In this section, we will analyze the security of the new cryptoscheme against extracting a related private key from a public key.

In cryptanalysis, we suppose that the integer factorization problem (IFP) \( N = pq \) with \( \lceil \log N \rceil < 1024 \) [2], the discrete logarithm problem (DLP) \( y = g^x \) with \( \lceil \log y \rceil < 1024 \) [13][14], and the subset sum problem of low density (SSP) \( s = \sum r_i c_i \) with \( D \approx \log M \) can be solved in tolerable subexponential time or in polynomial time [15].

5.1 A Property of the MPP

In the new cryptoscheme, there is the MPP \( C_i = (A_i, W^{(t, 0)} \delta) \) with \( A_i \in A \) and \( t \) from \( \Omega = \{+/-(6j-1), +/-(6j+1), +/-(6j+3) \}_{j=1, \ldots, \bar{n}/2} \) for \( i = 1, \ldots, 3\bar{n}/2 \).

According to Definition 6, the MPP has the following property.

Property 6: The MPP \( C_i = (A_i, W^{(t, 0)} \delta) \) with \( A_i \in A = \{2, \ldots, \mathcal{P} \mid \mathcal{P} \leq 1201\} \) and \( t \) from \( \Omega = \{+/-(6j-1), +/-(6j+1), +/-(6j+3) \}_{j=1, \ldots, \bar{n}/2} \) for \( i = 1, \ldots, 3\bar{n}/2 \) is computationally equivalent to the DLP in the same prime field.

Refer to Section 4.1 of [1] for its proof.

Notice that the structure of the set \( \Omega \) consisting of triples has no change in essence compared with the \( \Omega \) in [1].

5.2 Attack by Interaction of the Key Transform Items

In the key transform \( C_i = (A_i, W^{(t, 0)} \delta) \) with \( A_i \in A = \{2, 3, \ldots, \mathcal{P} \mid \mathcal{P} \leq 937, 991, \ldots, 1201\} \) and \( t \) from \( \Omega = \{+/-(6j-1), +/-(6j+1), +/-(6j+3) \}_{j=1, \ldots, \bar{n}/2} \)

5.2.1 Eliminating \( W \) through \( t(x_1) + t(x_2) = t(y_1) + t(y_2) \)

\( \forall x_1, x_2, y_1, y_2 \in \{1, 3\bar{n}/2\} \), assume that \( t(x_1) + t(x_2) = t(y_1) + t(y_2) \).

Let \( G_i = C_i(C_i^2(C_i^2C_i)^{-1})^{-1} \) namely \( G_i = (A_i, A_i^{-1}A_i^{-1})^{-1} \).

If an adversary divides the values of \( A_{x_1}, A_{x_2}, A_{y_1}, A_{y_2} \in A \), he may compute \( \delta \) through a discrete logarithm in \( L_n[1/3, 1.923] \) time, where \( M < 2^{640} \).

However, a concrete \( \Omega \) is one of \( (2^{331})^{\bar{n}/2} \) potential sets, indeterminate, and unknown due to \( |\Omega| = \bar{n}/2 \) and \( \Omega = \{+/-(6j-1), +/-(6j+1), +/-(6j+3) \}_{j=1, \ldots, \bar{n}/2} \).

For example, assume that \( t(x_1) + t(x_2) = 5 + 11 \) and \( t(y_1) + t(y_2) = -7 + 9 \), then there is \( t(x_1) = t(x_2) \neq t(y_1) = t(y_2) \). Therefore, among \( t(1), \ldots, t(3\bar{n}/2) \), there does not necessarily exist \( t(x_1) + t(x_2) = t(y_1) + t(y_2) \).

The above example illustrates that to determine the existence of \( t(x_1) + t(x_2) = t(y_1) + t(y_2) \), the adversary must first determine the constitution of the set \( \Omega \), which will have \( O((2^{331})^{\bar{n}/2}) \) time complexity.

5.2.2 Eliminating \( W \) through the \( |W| \)-th Power

Owing to \( |\log M| = 512 \) or 640, \( |W| \) can be factorized in tolerable subexponential time. Again owing to \( \prod^{\bar{n}/2} p_i \mid |W| \) and \( \prod^{\bar{n}/2} p_i \geq 2^{10} \) with \( p_i < \sqrt{|W|} \), \( |W| \) can be divided in the time of about the \( 2^{10} \) magnitude.

Raising either side of \( C_i = (A_i, W^{(t, 0)} \delta) \% M \) to the \( |W| \)-th power yields \( C_i^{\mid W \rangle} = (A_i)^{\mid W \rangle} \% M \).

Let \( C_i = g^{v_i} \% M \) and \( A_i = g^{v_i} \% M \), where \( g \) is a generator of \( (\mathbb{Z}_M^*, \cdot) \). Then \( u_i \% |W| = v_i \% |W| \delta \% |M| \) for \( i = 1, \ldots, 3\bar{n}/2 \) Notice that \( u_i \neq v_i \% |M| \) due to \( |W| \% |M| \).

The above congruence looks to be the MH transform [7]. Actually, \( \{v_i \% |W|, \ldots, v_{3\bar{n}/2} \% |W|\} \) is not a super increasing sequence, and moreover there is not necessarily \( \log(u_i \% |W|) = \log |M| \).

Because \( v_i \% |W| \in [1, |M|] \) is stochastic, the inverse \( \delta^{-1} \% |M| \) not need be close to the minimum \( |M|/(u_i \% |W|), 2|M|/(u_i \% |W|), \ldots, (u_i \% |W| - 1)|M|/(u_i \% |W|) \).

Namely \( \delta^{-1} \) may lie at any integral position in the interval \( k|M|/(u_i \% |W|), (k + 1)|M|/(u_i \% |W|) \), where \( k = 0, 1, \ldots, u_i \% |W| - 1 \), which illustrates the accumulation points of minima do not exist. Further observing, in this case, when \( i \) traverses the
interval \([2, 3\beta/2]\), the number of intersections of the intervals including \(\delta^{-1}\) is likely the max of 
\((u_2|W|, ..., u_{3\beta/2}|W|)\) which is promisingly close to \(\mathcal{M}\). Therefore, the Shamir attack by the
accumulation point of minima is fully ineffectual \([16]\).

Even though find out \(\delta^{-1}\) by the Shamir attack method, because each of \(v_i\) has \(|W|\) solutions, the
number of potential sequences \(\{g^{1}, ..., g^{3\beta/2}\}\) is up to \(|W|^{3\beta/2}\). Because of needing to verify whether
\(\{g^{1}, ..., g^{3\beta/2}\}\) is a coprime sequence for each different sequence \(\{v_1, ..., v_{3\beta/2}\}\), the number of coprime
sequences is in direct proportion to \(|W|^{3\beta/2}\). Hence, the initial \(\{A_1, ..., A_{3\beta/2}\}\) cannot be determined in
subexponential time. Further, the value of \(W\) cannot be computed, and the values of \(|W|\) and \(\delta^{-1}\) cannot
be verified in subexponential time, which indicates that MPP can also be resistant to the attack by the
accumulation point of minima.

Additionally, an adversary may divine value of \(A_i\) in about \(|A_i|\) time, where \(i \in [1, 3\beta/2]\), and compute \(\delta\) by \(u_i||W| = v_i||W|\delta\) (% \(\mathcal{M}\)).

However, because of \(|W| | \mathcal{M}\), the equation will have \(|W|\) solutions. Therefore, the time complexity of
finding the original \(\delta\) is at least

\[
F_e = (3\beta/2)|A|L_M[1/3, 1.923] + 2^{10}|A||W|
\]

\[
= 2^{3.9L_M[1/3, 1.923] + 2^{10}2^{|W|}}
\]

\[
\geq 2^{3.9L_M[1/3, 1.923] + 2^{10}2^{|W|}}
\]

It is exponential in \(n\) with \(n = 80, 96, 112\).

### 5.3 Attack by a Certain Single \(C_i\)

Assume that there is only a solitary \(C_i = (A_i, W^{(1)})^\delta \% M \quad i = 1\) for example, and other \(C_i\)'s \((i = 2, ..., 3\beta/2)\) are unknown for adversaries.

Through dividing the values of \(A_i \in \Omega\) and \(\delta\) coprime to \(\mathcal{M}\), the secret parameters \(W \in \Omega\) can be computed. Thus, the number of possible values of \(W\) will be larger than \(|\Omega|\) because \(|\mathcal{M}|\) is at least \(2^\delta\), which manifests that the original \((A_i, \ell(1), W, \delta)\) cannot be determined in subexponential time.

Evidently, if \(g_1 = A_i W^{(1)} \% M\) is a constant, solving \(C_i = g_1^\delta \% M\) for \(\delta\) is equivalent to the DLP.
Factually, \(g_1\) is not a constant, and at present, the time complexity of seeking the original \(g_1\), namely
\(A_i W^{(1)}\) will be \(O(M) > O(2^\delta)\).

In summary, the time complexity of inferring a related private key from a public key is at least \(O(2^\delta)\).

### 6 Analysis of Security of a Plaintext

In this section, we will analyze the security of the new cryptoscheme against recovering a related
plaintext from a ciphertext.

The security of a plaintext depends on the ASPP \(G = \prod_{i=1}^{\beta/2}(C_{(3(3i-1)+B)} \% M)^{\delta} \% M\) with \(C_0 = 1\) and \(r_{1, ..., r_{3\beta/2}}\) a random bit string, but we need also to understand the SPP \(G' = \prod_{i=1}^{\beta/2}(C_{(3(3i-1)+B)} \% M)^{\delta} \% M\) with \(C_0 = 1\).

**Definition 9:** Let \(A\) and \(B\) be two computational problems. \(A\) is said to reduce to \(B\) in polynomial
time, written as \(A \leq^T\_\mathcal{P} B\), if there is an algorithm for solving \(A\) which calls, as a subroutine, a
tactical algorithm for solving \(B\), and runs in polynomial time, excluding the time of the algorithm
for \(B\) \([11][17]\).

The hypothetical algorithm for solving \(B\) is called an oracle. It is easy to understand that no matter
what the time complexity of the oracle is, it does not influence the result of the comparison.

\(A \leq^T\_\mathcal{P} B\) means that the difficulty of \(A\) is not greater than that of \(B\), namely the time complexity of the
fastest algorithm for \(A\) is not greater than that of the fastest algorithm for \(B\) when all polynomial times
are treated as being pairwise equivalent. Concretely speaking, if \(A\) cannot be solved in polynomial or
subexponential time, \(B\) cannot also be solved in corresponding polynomial or subexponential time; and
if \(B\) can be solved in polynomial or subexponential time, \(A\) can also be solved in corresponding
polynomial or subexponential time.

**Definition 10:** Let \(A\) and \(B\) be two computational problems. If \(A \leq^T\_\mathcal{P} B\) and \(B \leq^T\_\mathcal{P} A\), then \(A\) and \(B\) are
said to be computationally equivalent, written as \(A \equiv^T\_\mathcal{P} B\) \([11][17]\).

\(A \equiv^T\_\mathcal{P} B\) means that either if \(A\) is a hardness of a certain complexity on condition that the dominant
variable approaches a large number, \(B\) is also a hardness of the same complexity on the identical
condition; or \(A, B\) both can be solved in linear or polynomial time.

Definition 9 and 10 suggest a reductive proof method called polynomial time Turing reduction (PTR)
Asymptotic security implies that even if a cryptosystem based on intractabilities is proven to be secure, it is practically secure only on condition that the dominant parameter is large enough.

Naturally, we will explore whether \( A \leq_B B \) exists or not. The definition of \( A \leq_B B \) may possibly be given theoretically, but the proof of \( A \leq_B B \) is not easy in practice.

Let \( \hat{H}(y = f(x)) \) represent the complexity or hardness of solving the problem \( y = f(x) \) for \( x \) [15].

### 6.1 Three Properties

According to Definition 7, the SPP has the following property.

**Property 7:** The SPP \( \mathcal{G} = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3 / 3} \) (\( \% M \)) with \( C_0 = 1 \) is computationally at least equivalent to the DLP in the same prime field, where \( B_1 \ldots B_{\beta_2 / 2} \neq 0 \) is a bit-pair string.

**Proof:**

For clear explanations, we extend \( B_1 \ldots B_{\beta_2 / 2} \) to a bit string \( b'_1 \ldots b'_{\beta_2 / 2} \) by the following rule for \( i = 1, \ldots, \beta_2 / 2 \):

1. when \( B_i = 0 \), \( b'_i = 0 \);
2. when \( B_i = 1 \), \( b'_i = b'_i \).

Hence, we have the equivalent

\[
\mathcal{G} = \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} \text{ (\( \% M \))}.
\]

The form of \( \mathcal{G} \) here is similar to that of \( \mathcal{G} \) in [1].

Especially, define \( \mathcal{G} = \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} = \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} \text{ (\( \% M \))} \) when \( C_1 = \ldots = C_{\beta_2 / 2} = C \).

Obviously, \( \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} = L \cdot M = C \). Owing to \( L \in [1, M] \), deriving the non-modular product \( \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} \) from \( \mathcal{G} \) is infeasible, which means inferring \( b'_1 \ldots b'_{\beta_2 / 2} \) from \( \mathcal{G} \) is not a factorization problem.

Assume that \( \hat{O}(\mathcal{G}, C_1, \ldots, C_{\beta_2 / 2}, M) \) is an oracle on solving \( \mathcal{G} = \prod_{i=1}^{\beta_2} C_i^{\beta_3 / 3} \) (\( \% M \)) for \( b'_1 \ldots b'_{\beta_2 / 2} \).

Let \( y = g^x \) (\( \% M \)) be of the DLP, where \( g \) is a generator of \( \mathbb{Z}_{M^*} \), and the binary form of \( x \) is \( b_1 \ldots b_{\beta_2 / 2} \), namely \( y = \prod_{i=1}^{\beta_2} (g^{2^{\beta_2 - i}})^{b_i} \text{ (\( \% M \))} \).

Then, by calling \( \hat{O}(y, g^{2^{\beta_2 - i}}, \ldots, g, M, b_1 \ldots b_{\beta_2 / 2} \) namely \( x \) can be found.

According to Definition 9, there is

\[
\hat{H}(y = g^x \text{ (\( \% M \))}) \leq \hat{H}(\mathcal{G}) = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3 / 3} \text{ (\( \% M \))},
\]

namely the SPP is at least equivalent to the DLP in the same prime field in complexity. \( \square \)

According to Definition 8, and the ASPP has the following property.

**Property 8:** The ASPP \( \mathcal{G} = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3} \text{ (\( \% M \))} \) with \( C_0 = 1 \) is computationally at least equivalent to the DLP in the same prime field, where \( B_1 \ldots B_{\beta_2 / 2} \neq 0 \) is the bit-pair shadow string of \( B_1 \ldots B_{\beta_2 / 2} \).

**Proof:**

For clear explanations, we extend \( B_1 \ldots B_{\beta_2 / 2} \) to a nonrigid shadow string \( \mathcal{G} = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3} \text{ (\( \% M \))} \) by the following rule for \( i = 1, \ldots, \beta_2 / 2 \):

1. when \( B_i = 0 \), \( b'_i = 0 \);
2. when \( B_i = 1 \), \( b'_i = b'_i \).

Hence, we have the equivalent

\[
\mathcal{G} = \prod_{i=1}^{\beta_2} C_i^{\beta_3} \text{ (\( \% M \))}.
\]

The form of \( \mathcal{G} \) here is similar to that of \( \mathcal{G} \) in [1].

Assume that \( \hat{O}(\mathcal{G}, C_1, \ldots, C_{\beta_2 / 2}, M) \) is an oracle on solving \( \mathcal{G} = \prod_{i=1}^{\beta_2} C_i^{\beta_3} \text{ (\( \% M \))} \) for \( b'_1 \ldots b'_{\beta_2 / 2} \).

Let \( y = g^x \) (\( \% M \)) be of the DLP, where \( g \) is a generator of \( \mathbb{Z}_{M^*} \), and the binary form of \( x \) is \( b_1 \ldots b_{\beta_2 / 2} \), namely \( y = \prod_{i=1}^{\beta_2} (g^{2^{\beta_2 - i}})^{b_i} \text{ (\( \% M \))} \).

Then, by calling \( \hat{O}(y, g^{2^{\beta_2 - i}}, \ldots, g, M, b_1 \ldots b_{\beta_2 / 2} \) namely \( x \) can be found.

According to Definition 9, there is

\[
\hat{H}(y = g^x \text{ (\( \% M \))}) \leq \hat{H}(\mathcal{G}) = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3} \text{ (\( \% M \))},
\]

further by transitivity, there is

\[
\hat{H}(y = g^x \text{ (\( \% M \))}) \leq \hat{H}(\mathcal{G}) = \prod_{i=1}^{\beta_2} (C_{3(i-1)+B})^{\beta_3} \text{ (\( \% M \))}.
\]
namely the ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{3(i-1)+j})^6 (\% M)$ is at least equivalent to the DLP in the same prime field in computational complexity.

**Property 9:** The ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{r(3(i-1)+j)} + r_3(i-1) + B_j)^6 (\% M)$ with $C_0 = 1$ and $r_1...r_{\delta^2}$ a random bit string is computationally at least equivalent to the DLP in the same prime field, where $B_1...B_{\delta^2}$ is the bit-pair shadow string of $B_1...B_{\delta^2} \neq 0$.

**Proof:**
Let $r_1...r_{\delta^2} = 1...1$, then the ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{r(3(i-1)+j)} + r_3(i-1) + B_j)^6 (\% M)$ is reduced to the ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{3(i-1)+j})^6 (\% M)$.

By Property 8 and the transitivity, there exists $\hat{H}(v = g^r (\% M)) \leq \hat{H}(\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{3(i-1)+j})^6 (\% M))$, namely the ASPP is at least equivalent to the DLP in the same prime field in complexity.  

**6.2 Resisting LLL Lattice basis Reduction**

We know that after a lattice basis is reduced through the LLL algorithm, the final reduced base will contain the shortest or approximately shortest vectors, but among them does not necessarily exist the original solution to a subset sum problem because only if

1. the solution vector for the SSP is the shortest,
2. the shortest vector is unique in the lattice,

will the original solution vector appear in the reduced base with large probability.

In the new cryptoscheme, there are $\bar{n} = 96, 112, 128$ and $\lceil \text{lg} M \rceil = 464, 544, 640$. Under the circumstances, the DLP and IFP can be solved in tolerable subexponential time, namely the DLP and IFP cannot resist the attack of adversaries.

We first consider the ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{3(i-1)+j})^6 (\% M)$.

For convenience, extend $B_1...B_{\delta^2}$ to $b_1...b_{3\delta^2}$ by the following rule for $i = 1, \ldots, \delta^2/2$:

1. when $B_i = 0$, let $b_{3(i-1)+j} = b_{3(i-1)+j+1} = 0$;
2. when $B_i \neq 0$, let $b_{3(i-1)+j} = b_{3(i-1)+j+1} = b_{3(i-1)+j+2} = B_i$.

For example, suppose that $B_1...B_4 = 00101100$, then $b_1...b_4 = 01001000$.

In this way, the ASPP $\mathcal{G} = \prod_{i=1}^{\delta^2} (C_{3(i-1)+j})^6 (\% M)$ is converted into

$$\mathcal{G} = \prod_{i=1}^{3\delta^2} C_i^6 (\% M).$$

Let $g$ be a generator of the group $(\mathbb{Z}_{\bar{n}})^*$. Let $C_1 = g^{u_1} (\% M), \ldots, C_{3\delta^2} = g^{u_{3\delta^2}} (\% M)$, and $\mathcal{G} = g^v (\% M)$.

Then, through a conversion in subexponential time, seeking $B_1...B_{\delta^2}$ from $\mathcal{G}$ is equivalent to seeking $b_1...b_{3\delta^2}$ from the congruence

$$u_1 b_1^3 + \ldots + u_{3\delta^2} b_{3\delta^2}^3 = v (\% M),$$

where $v$ may be substituted with $v + kM$ along with $k \in [0, 3\delta^2/2]$.  

Similar to Section 1, $\{u_1, \ldots, u_{3\delta^2}\}$ is called a compact due to every $b_i \in [0, \bar{n}/4 + 1]$ [4], and solving Equation (3) for $b_1...b_{3\delta^2}$ is called an ASSP [1].

May also convert this ASSP into a SSP through splitting $u_i$ into bits, and thus according to $b_i \in [0, \bar{n}/4 + 1]$, the density of the related ASSP knapsack is defined as

$$D = \sum_{i=1}^{3\delta^2} \frac{\lceil \text{lg}(\bar{n}/4 + 1) \rceil}{\lceil \text{lg} M \rceil} = (3\delta^2) \frac{\lceil \text{lg}(\bar{n}/4 + 1) \rceil}{\lceil \text{lg} M \rceil}.$$  

Namely,

$$D = 3\delta^2 \frac{\lceil \text{lg}(\bar{n}/4 + 1) \rceil}{(2 \lceil \text{lg} M \rceil)}.$$  

which is slightly different from Formula (2).

Concretely speaking, in the new cryptoscheme, there are

- $D = 144 \times 5 / 464 \approx 1.5517 > 1$ for $\bar{n} = 96$ and $\lceil \text{lg} M \rceil = 464$;
- $D = 168 \times 5 / 544 \approx 1.5441 > 1$ for $\bar{n} = 112$ and $\lceil \text{lg} M \rceil = 544$;
- $D = 192 \times 6 / 640 \approx 1.8000 > 1$ for $\bar{n} = 128$ and $\lceil \text{lg} M \rceil = 640$.

Therefore, Equation (3) does represent an ASSP of high density, which indicates that many different subsets will have the same sum, and probability that the original solution vector will occur in the final reduced lattice basis is nearly zero. Meanwhile, our experiment demonstrates that the original solution vector does not occur in the final reduced base [18].
Because $\mathcal{G} = \prod_{j=1}^{d_2} (C_{f(j-1)+\beta_j})^{\beta_j} (\% M)$ is only a special case of $\mathcal{G} = \prod_{j=1}^{d_2} (C_{f(j-1)+\beta_j} + C_{f(j-1)-\beta_j} + \beta_j)^{\beta_j} (\% M)$, the latter is also able to resist the LLL lattice basis reduction.

### 6.3 Avoiding Meet-in-the-middle Attack


**INPUT:** a set of positive integers $\{c_1, \ldots, c_n\}$ and a positive integer $s$.

**S1:** Set $t \leftarrow [n / 2]$.

**S2:** Construct a table with entries $(\sum_{i=1}^{t} c_i b_i, (b_1, \ldots, b_t))$ for $(b_1, \ldots, b_t) \in \mathbb{Z}_2^t$.

Sort this table by the first component.

**S3:** For each $(b_{i-1}, b_{i+1}, \ldots, b_k) \in \mathbb{Z}_2^{k-i}$, do the following:

**S3.1:** Compute $r = s - \sum_{i=1}^{t} c_i b_i$ and check, using a binary search, whether $r$ is the first component of some entry in the table;

**S3.2:** If $r = \sum_{i=1}^{t} c_i b_i$, then return (a solution is $(b_1, \ldots, b_t)$).

**S4:** Return (no solution exists).

**OUTPUT:** $b_i \in \{0, 1\}, 1 \leq i \leq n$, such that $\sum_{i=1}^{n} c_i b_i = s$, provided such $b_i$ exist.

It is not difficult to understand that the time complexity of the above algorithm is $O(n2^{n/2})$.

Likewise, the versatile meet-in-the-middle dichotomy may be used to assault the ASSP $\mathcal{G} = \prod_{j=1}^{d_1} (C_{f(j-1)+\beta_j})^{\beta_j} (\% M)$ with entries $(\prod_{j=1}^{d_1} (C_{f(j-1)+\beta_j} + C_{f(j-1)-\beta_j} + \beta_j)^{\beta_j}, (r_1, \ldots, r_{d_1/2}))$ for $(r_1, \ldots, r_{d_1/2}) \in \{0, 1\}^{d_1/2}$ and $(B_1, \ldots, B_{d_1/2}) \in \{00, 01, 10, 11\}^{d_1/2}$ when $B_{d_1/2} \neq 00$ and $B_{d_1/2} = 00$ which occurs with probability $9 / 16 = 0.5625$. Obviously, the random bit string $r_1\ldots r_{d_1/2}$ extends the scope of exhaustive search. Further, it is easy to see that the running time of this attack task is $O(2^{d_1/2} \cdot 2^{d_1/4} \lg M) = O(\tilde{O}2^{d_1/4} \cdot \lg^2 M)$ bit operations.

Concretely speaking, when $\tilde{n} = 96$ namely $n = 80$ with $\lceil \lg M \rceil = 464$, $T_m = 2^{2^7 \cdot 96/4} \approx 2^{97} > 2^{80}$ bos;

when $\tilde{n} = 112$ namely $n = 96$ with $\lceil \lg M \rceil = 544$, $T_m = 2^{2^7 \cdot 112/4} \approx 2^{111} > 2^{96}$ bos;

when $\tilde{n} = 128$ namely $n = 112$ with $\lceil \lg M \rceil = 640$, $T_m = 2^{2^7 \cdot 128/4} \approx 2^{124} > 2^{112}$ bos.

Therefore, the new cryptoscheme can resist the meet-in-the-middle attack.

### 6.4 Avoiding Adaptive-chosen-ciphertext Attack

Most of public key cryptoschemes may probably be faced with adaptive-chosen-ciphertext attack [20]. However, it is lucky the Cramer-Shoup asymmetric encryption scheme is very indistinguishable and nonmalleable [21], and proven to be secure against the adaptive-chosen-ciphertext attack under the cryptographic assumptions [22]. So is the OAEP+ scheme [23].

### 6.4.1 Indistinguishability of Ciphertexts

In the encryption process of a JUNA plaintext, a random padding string of $\tilde{n} - n$ bits is appended to the terminal of the JUNA plaintext, which changes the original plaintext to an extended plaintext, and a random permutation string of $\tilde{n}/2$ bits is introduced into the arrangement of bit-pairs of the extended plaintext, which is equivalent to the thing that the order of triple items of a public key is varied along with every encryption.

Due to the interlacement of 00-pairs and non-00-pairs and the randomicity of bit string generation, the padding string and the permutation string make an identical original plaintext be able to correspond to at most $2^{\tilde{n}/2} \cdot 2^{\tilde{n}/2}$, which is exponential in $n$, different ciphertexts. It will take the running time of $O(2^{\tilde{n}/2} \cdot 2^{\tilde{n}/4} \lg M)$ bit operations exhaustively to search all the possible ciphertexts of an original plaintext. Therefore, the correspondence between any arbitrary ciphertext and a related original ciphertext is indistinguishable in subexponential time.

Concretely speaking, the running time of searching all the ciphertexts of an original plaintext is $T_c = (96)2^{96/2 \cdot 96/4 - 464/2} \approx 2^{88} > 2^{80}$ for $n = 80$, $\tilde{n} = 96$, and $\lceil \lg M \rceil = 464$;

$T_c = (112)2^{112/2 \cdot 112/4 - 544/2} \approx 2^{98} > 2^{96}$ for $n = 96$, $\tilde{n} = 112$, and $\lceil \lg M \rceil = 544$;

$T_c = (128)2^{128/2 \cdot 128/4 - 644/2} \approx 2^{108} \approx 2^{112}$ for $n = 112$, $\tilde{n} = 128$, and $\lceil \lg M \rceil = 640$. 

6.4.2 Nonmalleability of Ciphertexts

An encryption scheme is said to be malleable if it is possible for an adversary to transform a ciphertext into another ciphertext which reverts to a related plaintext. That is, given an ciphertext of a plaintext $m$, it is possible to generate another ciphertext which reverts to $f(m)$ without necessarily knowing or learning $m$, where $f$ is a known function [21].

By way of example, let a RSA ciphertext $C = m^e \mod N$, then $zC = (zm)^e \mod N$ is a malleation of $C$ which reverts to $f(m) = zm \mod N$. Again let an ElGamal ciphertext $C = (g', m\rho^e \mod \rho)$, then $(g', zm\rho^e \mod \rho)$ is a malleation of $(g', m\rho^e \mod \rho)$ which reverts to $f(m) = zm \mod \rho$.

In the new cryptoscheme, there is the ciphertext $\mathcal{C} = f'(\mathcal{B}) = \prod_{i=1}^{\ell_{2}} (C_{r_{i}(3(\ell_{i}-1)+B_{i})+s_{r_{i}(3(\ell_{i}-1)+B_{i})}})^{\%} \% M$ which takes a bit-pair as an operation unit, where $\mathcal{B} = B_{1}, \ldots, B_{\ell_{2}}$ is a related extended plaintext, and thus, evidently the plaintext function $f'(\mathcal{B}) = zm \% M$ is not suitable for $\mathcal{G}$. Again considering that $\mathcal{B} = B_{1}, \ldots, B_{\ell_{2}}$ occurs in the subscript of $\{C_{1}, \ldots, C_{\ell_{2}}\}$, and moreover is relevant to the random bit string $r_{1}, \ldots, r_{\ell_{2}}$ that can be guessed only in exponential time, it is impossible to exist other $f'(\mathcal{B})$ which corresponds to a malleation of $f'(\mathcal{B})$ namely $\mathcal{G}$, where $f$ is such a function of which the inverse can be computed in subexponential time.

6.4.3 Proof of the Semantical Security

If the security requirement of a cryptoscheme can be stated formally in an antagonistic model, as opposed to heuristically, with clear assumptions that certain computational problems are intractable, and an adversary has access to his algorithms of the cryptoscheme as well as enough computational resources, the cryptoscheme possesses provable security [24][25].

**Definition 11:** A cryptoscheme is said to be semantically secure if an adversary who knows the encryption algorithm of the cryptoscheme and is in possession of a ciphertext is unable to determine any information about the related plaintext [24].

It is subsequently demonstrated that semantic security is equivalent to another definition of security called ciphertext indistinguishability [26]. If a cryptoscheme has the property of indistinguishability, then an adversary will be unable to distinguish a pair of ciphertexts based on the two plaintexts encrypted by a challenger.

A chosen plaintext attack (CPA) is an attack model for cryptanalysis which presumes that the attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts that are expected to decrease the security of an encryption scheme [25].

**Definition 12:** A cryptoscheme is said to be IND-CPA (indistinguishable under chosen plaintext attack), namely semantically secure against chosen plaintext attack, if the adversary cannot determine which of the two plaintexts was chosen by a challenger, with probability significantly greater than 1/2, where 1/2 means the success rate of random guessing [25][27].

For a probabilistic asymmetric cryptoscheme based on computational security, indistinguishability under chosen plaintext attack is illuminated by a game between an adversary and a challenger, where the adversary is regarded as a probabilistic polynomial time Turing machine, which means that it must complete the game and output a guess within a polynomial number of operation steps.

Notice that for the JUNA cryptoscheme, the adversary may be also regarded as a probabilistic subexponential time Turing machine since no subexponential time solution to the MPP or ASPP is found so far.

**Theorem 1:** The JUNA cryptoscheme is semantically secure against chosen plaintext attack on the assumption that the MPP and ASPP cannot be solved in subexponential time.

**Proof:**

Let $E(k_{p}, m)$ represents the encryption of a message (plaintext) $m$ under the public key $k_{p}$.

A game between an adversary and a challenger is given as follows.

1. The challenger calls the key generation algorithm with the parameters $n$, $l$, and $\mathcal{D}$, obtains a key pair $(k_{p}, k_{s})$, publishes $k_{p} = (\{C_{1}, \ldots, C_{\ell_{2}}\}, M)$ to the adversary, and retains $k_{s}$ for himself.

2. The adversary may perform any number of encryptions or other compatible operations.

3. Eventually, the adversary chooses any two distinct $n$-bit plaintexts $(m_{1}, m_{2})$, and submits them to the challenger.

4. The challenger selects a bit $x \in \{0, 1\}$ uniformly at random, and sends the challenge ciphertext $\mathcal{G} = E(k_{p}, m_{x}) = \prod_{i=1}^{\ell_{2}} (C_{r_{i}(3(\ell_{i}-1)+B_{i})+s_{r_{i}(3(\ell_{i}-1)+B_{i})}})^{\%} \% M$ back to the adversary, where $m_{x} = b_{1}, \ldots, b_{n}$.

5. The adversary is free to perform any number of additional computations or encryptions. Finally, it outputs a guess for the value of $x$. Therefore need to analyze the probability of hitting $x$. 

14
Because the MPP and ASPP have no subexponential time solutions, neither can the adversary decrypt \( \Psi \) with a private key, nor can directly solve \( \mathcal{A} = \prod_{i=1}^{\beta_{\tilde{M}}}(C_{r_i(3i-1)+B_3}+\epsilon_{3i(3i-3)+B_3})^{\beta_i} (\% M) \) for \( \Psi = (b_1...b_n = B_1...B_n) \).

It is known from the encryption algorithm that an identical plaintext may correspond to at most \( 2^{\beta_{\tilde{M}}-2} \) different ciphertexts, where \( \tilde{M} = n + 16 \), and it will need the running time of \( \mathcal{O}(2^{\beta_{\tilde{M}}-2} \cdot \log^2 M) \) bit operations to verify all the possible ciphertexts of a plaintext. Thus the probability that the adversary hits \( x \) with guessing is only \( 1/2 + 1/2^{\beta_{\tilde{M}}-2} \), where \( 2^{\beta_{\tilde{M}}-2} \) is exponential in \( n \), which means that \( 1/2^{\beta_{\tilde{M}}-2} \) is a negligible function of \( n \), and for every (nonzero) polynomial function \( \text{poly}(n) \) (notice that in the JUNA cryptoscheme, it may be also a subexponential function), there exists \( n_0 \) such that \( 1/2^{\beta_{\tilde{M}}-2} < 1 / \text{poly}(n) \) for all \( n > n_0 \).

In summary, the JUNA cryptoscheme is semantically secure, namely IND-CPA.

7 Conclusion

In the paper, a new public key cryptoscheme which includes the key generator, encryption algorithm, and decryption algorithm is proposed.

The new cryptoscheme builds its security firmly on two intractabilities: the MPP \( C_i = (A_i W(i))^{\delta} \% M \) with \( A_i \in A \) and \( \ell(i) \) from \( \mathcal{D} \) and the ASPP \( \mathcal{A} = \prod_{i=1}^{\beta_{\tilde{M}}} (C_{r_i(3i-1)+B_3}+\epsilon_{3i(3i-3)+B_3})^{\beta_i} (\% M) \) to which no subexponential time solutions are found, and there exist only exponential time solutions so far \[28\], utilizes a bit-pair string to decrease the bit-length of the modulus \( M \), exploits a bit-pair shadow string to guard against the LLL lattice basis reduction attack, and adopts the approaches of introducing a random bit string into an encryption and appending a random bit string to a plaintext to avoiding the adaptive-chosen-ciphertext attack and the meet-in-the-middle dichotomy.

As \( \tilde{M} = 96, 112, \) or \( 128 \), there exists \( \lceil \log M \rceil = 464, 544, \) or \( 640 \), which assures that when a JUNA ciphertext \( G \) with \( r_1...r_{\beta_{\tilde{M}}} = 1...1 \) is converted into an ASPP through a discrete logarithm, the density of a related ASPP knapsack is pretty high, and larger than 1.

There always exists contradiction between time and security, so does between space and security, and so does between time and space. We attempt to find a balance which is none other than a delicate thing among time, space, and security.

Acknowledgment

The authors would like to thank the Academicians Jiren Cai, Zhongyi Zhou, Jianhua Zheng, Changxiang Shen, Zhengyao Wei, Binxing Fang, Guangnan Ni, Andrew C. Yao, and Xicheng Lu for their important guidance, advice, and suggestions.

The authors also would like to thank the Professors Dingyi Pei, Je Wang, Ronald L. Rivest, Moti Yung, Adi Shamir, Dingzhu Du, Mulan Liu, Huanguo Zhang, Dengguo Feng, Xixian Yang, Hanliang Xu, Yongfei Han, Yupu Hu, Dongdai Lin, Chuanjun Wu, Rongguan Feng, Ping Luo, Jianfeng Ma, Lusheng Chen, Tao Xie, Wenbao Han, Bogang Lin, Lequan Min, Qibin Zhai, Hong Zhu, Renji Tao, Zhiying Wang, Quanyuan Wu, and Zhichang Qi for their important counsel, suggestions, and corrections.

References


