Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 475678, 18 pages
http://dx.doi.org/10.1155/2014/475678

Research Article

Preimage Selective Trapdoor Function: How to Repair an Easy Problem

Baocang Wang

The State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

Correspondence should be addressed to Baocang Wang; bcwang79@aliyun.com

Received 23 August 2013; Accepted 9 March 2014; Published 27 April 2014

Academic Editor: Yi-Kuei Lin

Copyright © 2014 Baocang Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Public key cryptosystems are constructed by embedding a trapdoor into a one-way function. So, the one-wayness and the trapdoorness are vital to public key cryptography. In this paper, we propose a novel public key cryptographic primitive called preimage selective trapdoor function. This scenario allows to use exponentially many preimage to hide a plaintext even if the underlying function is not one-way. The compact knapsack problem is used to construct a probabilistic public key cryptosystem, the underlying encryption function of which is proven to be preimage selective trapdoor one-way functions under some linearization attack models. The constructive method can guarantee the noninjectivity of the underlying encryption function and the unique decipherability for ciphertexts simultaneously. It is heuristically argued that the security of the proposal cannot be compromised by a polynomial-time adversary even if the compact knapsack is easy to solve. We failed to provide any provable security results about the proposal; however, heuristic illustrations show that the proposal is secure against some known attacks including brute force attacks, linearization attacks, and key-recovery attacks. The proposal turns out to have acceptable key sizes and performs efficiently and hence is practical.

1. Introduction

1.1. Background. Public key cryptosystem (PKC) is an important cryptographic primitive in the area of network and information security. The basic idea to construct a PKC is to embed a trapdoor into a mathematically intractable problem. The trapdoor helps the trapdoor holder reverse the underlying one-way function. However, without the trapdoor, one should attack a presumed intractable problem in order to reconstruct the plaintext corresponding to a given ciphertext. So the existence of mathematically hard problems is vital in public key cryptography. This explains why we say bad news in computational complexity is good news for cryptography.

In the cryptographic community, a PKC is always considered as synonymous with the so-called trapdoor one-way function. Bellare et al.'s observation [1] provides a better understanding of the two concepts. On one hand, a probabilistic public key encryption does not necessarily imply a trapdoor one-way function. For example, the underlying encryption function $f(m, r) = (c_1, c_2) = (g^r, mh^r)$ of ElGamal [2] is not one-way in that the auxiliary key r randomly chosen by the encrypter cannot be recovered knowing the trapdoor x satisfying $h = g^x$. On the other hand, in some cases, noninjective trapdoor one-way functions cannot be used to construct PKCs because a PKC is required to recover a unique plaintext from any valid ciphertext. So the encryption function underlying a PKC should be injective. Bellare et al. showed that any one-way function implies a highly noninjective trapdoor one-way function. However, the authors derived no PKCs from their universal constructions. They only showed that if the image of a trapdoor one-way function has polynomially bounded number of preimages, the trapdoor one-way function can be used to derive a PKC.

In this paper, we propose a probabilistic public key encryption algorithm from a highly noninjective one-way function. We think that our construction not only enriches the cryptographic basket and deepens our insights into public key cryptographic designs but also provides weaker cryptographic assumptions. Our confidence in the hardness

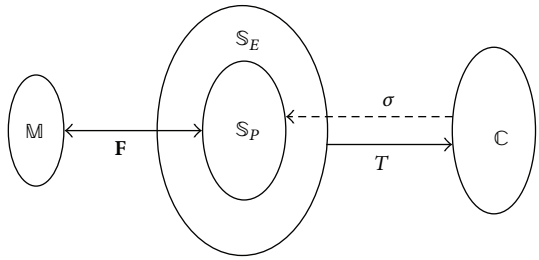


Figure 1: Preimage selective trapdoor one-way function.

of noninjective one-way functions rests on the proven fact that it is not easier to reverse a given one-way function equipped with an additional noninjectivity property than an injective one-way function [3]. More importantly, what we observed in this paper is that it remains possible to develop a secure PKC by fully exploiting the noninjectivity property of functions even if these functions turn out to be not one-way.

1.2. Our Contribution. In this paper, we firstly formalize a novel public key cryptographic scenario called preimage selective trapdoor one-way function (PSTOF, for short) and then use the compact knapsack problem to realize such a scenario.

1.2.1. Preimage Selective Trapdoor One-Way Function (See Figure 1). A PSTOF T is in fact a noninjective public key encryption function. When encrypting a message $\mathbf{m}$ in the message space $\mathbb{M}$, the encrypter firstly encodes the message into a plaintext $\mathbf{x} = \mathbf{F}(\mathbf{m})$ in the plaintext space $\mathbb{S}_P$ and then derives a ciphertext $c = T(\mathbf{x})$ in the ciphertext space $\mathbb{C}$. When decrypting a ciphertext c, the decrypter utilizes the trapdoor σ to decrypt a valid ciphertext into a unique plaintext $\mathbf{x} \in \mathbb{S}_P$ and then uniquely decodes $\mathbf{x}$ into a message $\mathbf{m} \in \mathbb{M}$.

The scenario allows us to base the security of a PKC on the high noninjectivity rather than the one-wayness of a function (an intractable problem). In this scenario, $\mathbb{S}_P$ is a subset of the equivalent plaintext space $\mathbb{S}_E$, and the ciphertext space $\mathbb{C} = T(\mathbb{S}_E) = \{T(\mathbf{x}) \mid \mathbf{x} \in \mathbb{S}_E\}$. For any valid ciphertext $c \in \mathbb{C}$, c will have many preimages in $\mathbb{S}_E$, which are called equivalent ciphertexts in that all of them can be mapped into c under the function T, and a unique preimage (the plaintext) $\mathbf{x} \in \mathbb{S}_P$ with respect to the function T, which is a valid plaintext and hence can be further decoded into a meaningful message. The message encoding function $\mathbf{F}$ simulates the process of randomly selecting a subset as the plaintext space $\mathbb{S}_P$ from the whole preimage domain $\mathbb{S}_E$, from which the name PSTOF comes. In other words, the message encoding function imposes some restrictions on the preimage set $\mathbb{S}_E$ to get a subset with a predesignated special structure as the plaintext space $\mathbb{S}_P$. The equivalent plaintexts in $\mathbb{S}_E$ are used to hide a (valid) plaintext in $\mathbb{S}_P$. If T is preassumed one-way, any polynomial-time adversary cannot obtain a preimage in $\mathbb{S}_E$ given a $c \in \mathbb{C}$; let alone the unique preimage (the plaintext) $\mathbf{x} \in \mathbb{S}_P \subset \mathbb{S}_E$. If the underlying function T is not one-way and when a $c \in \mathbb{C}$ has exponentially many preimages in $\mathbb{S}_E$, the corresponding plaintext in $\mathbb{S}_P$ is hidden by these exponentially many preimages. A polynomial-time adversary only obtains polynomially many preimages, which can be seen as randomly output from all the exponentially many preimages and hence include the target plaintext with a negligible probability.

1.2.2. Probabilistic Compact Knapsack PKC. By using the concept of PSTOF and from the compact knapsack problem, a probabilistic public key encryption scheme is designed. An easy compact knapsack-type problem is defined, and an algorithm called *Onion Algorithm* is developed to peel off the solutions to the easy problem. The easy problem is used to construct a probabilistic PKC. The main results are that the trapdoor one-way function underlying the proposal is PSTOFs under different linear attack models (See Theorems 25, 26, and 30), and the images of the PSTOFs have exponentially many preimages (see Table 3). Some design tricks are adopted to make the PKC immune from some known key-recovery attacks such as GCD attack [4, 5], Diophantine approximation attack [6, 7], and orthogonal lattice attack [8, 9].

1.3. Organization. The rest of the paper is organized as follows. In Section 2, we formalize the definition of PSTOF and use the cryptographic scenario to explain some known PKCs. Section 3 provides a concrete probabilistic public key encryption scheme to realize the PSTOF. Sections 4 and 5 discuss the performance and security related issues, respectively. Section 6 gives some concluding remarks.

2. Preimage Selective Trapdoor One-Way Function

2.1. Symbols and Notations. The notations listed at the end of the paper will be used throughout the paper.

2.2. Message Encoding Function. When encrypting a message using a PKC, we need to firstly encode a binary string message into a plaintext (an element in the algebraic structure underlying the PKC) in order that the PKC can deal with the message. However, sometimes, it is not necessary to explicitly specify the encoding algorithm. For example, it is more convenient to just look a message as a binary integer in $\mathbb{Z}_N$ than to develop an algorithm to transform a message into an element in $\mathbb{Z}_N$ in case of RSA [10]. Formally, we define a message encoding function as follows.

Definition 1. A function $\mathbf{F} : \mathbb{M} \mapsto \mathbb{S}_P$ with $\mathbb{S}_P = \mathbf{F}(\mathbb{M}) = \{\mathbf{F}(\mathbf{m}) \mid \mathbf{m} \in \mathbb{M}\}$ is called a message encoding function if it satisfies the following conditions.

(1) The function $\mathbf{F}$ is injective.

(2) It can be efficiently performed to compute $\mathbf{F}(\mathbf{m})$ for a given $\mathbf{m} \in \mathbb{M}$, and $\mathbf{F}^{-1}(\mathbf{p})$ for a given $\mathbf{p} \in \mathbb{S}_P$.

2.3. Preimage Selective Trapdoor One-Way Function. The PSTOF is formalized as follows.

Definition 2. A function $T : \mathbb{S}_E \mapsto \mathbb{C} = T(\mathbb{S}_E)$ is called a PSTOF if it satisfies the following conditions.

(1) For an arbitrary $\mathbf{x} \in \mathbb{S}_E$, it is easy to compute $c = T(\mathbf{x})$.

(2) T is noninjective.

(3) Given any $c \in \mathbb{C}$, there exists at most an $\mathbf{x} \in \mathbb{S}_P$ such that $T(\mathbf{x}) = c$.

(4) Given any $c \in \mathbb{C}$ and the trapdoor, there is a polynomial-time algorithm to output a unique $\mathbf{x} \in \mathbb{S}_P$ with $T(\mathbf{x}) = c$ or the invalid ciphertext symbol $\perp$.

(5) Given any $c \in \mathbb{C}$ and without knowing the trapdoor, anyone cannot efficiently compute a preimage $\mathbf{x} \in \mathbb{S}_E$ such that $T(\mathbf{x}) = c$.

Remark 3. The third and the forth conditions in Definition 2 guarantee that when viewed as a public key encryption function, the PSTOF is inverted according to the knowledge of the trapdoor σ to derive a unique preimage (the plaintext) in $\mathbb{S}_P$, and both conditions in Definition 1 mean that the plaintext can be uniquely decoded into the original message. If we remove the last requirement in Definition 2, we just call T a preimage selective trapdoor (not non-way) function.

Definition 4. The preimage density P_d of a noninjective function T is defined as the ratio of the cardinality of the preimage set $\mathbb{S}_E$ to that of the image set $\mathbb{C}$:

$$P_d = \frac{|\mathbb{S}_E|}{|\mathbb{C}|}. \tag{1}$$

On average, a ciphertext will have P_d preimages in $\mathbb{S}_E$.

Remark 5. When P_d is exponentially large, we can use the high noninjectivity of the underlying function T to hide a plaintext. On average, for a valid ciphertext $c = T(\mathbf{x})$ for a plaintext $\mathbf{x} \in \mathbb{S}_P$, c will have exponentially many P_d preimages in $\mathbb{S}_E$. So a polynomial-time adversary only can obtain polynomially many p preimages in $\mathbb{S}_E$ of c, which contains the targeted unique valid plaintext $\mathbf{x} \in \mathbb{S}_P$ with a negligible probability p/P_d. One may doubt that the function T is indeed a trapdoor one-way function mapping from $\mathbb{S}_P$ to $\mathbb{C}$. So if one breaks the one-wayness of $T : \mathbb{S}_P \mapsto \mathbb{C}$, he also recovers the unique $\mathbf{x} \in \mathbb{S}_P$ such that $c = T(\mathbf{x})$. However, if the plaintext space $\mathbb{S}_P$ can be seen as randomly chosen from the whole equivalent plaintext space $\mathbb{S}_E$, the adversary cannot use the special structure of $\mathbb{S}_P$ to develop an efficient algorithm for reversing the function $T : \mathbb{S}_P \mapsto \mathbb{C}$. We will explain this point later on by using an example.

2.4. Examples

2.4.1. High-Density Knapsack PKCs. To illustrate the aforementioned definitions, we consider high-density 0-1 knapsack (subset sum) PKCs Chor-Rivest and [11] Okamoto-Tanaka-Uchiyama [12]. In both cryptosystems, a bit-string message is encoded into an n-dimensional 0-1 plaintext vector with a fixed k Hamming weight (the so-called predesignated special plaintext structure) by using the source encoding algorithm (message encoding function) [13]:

$$\mathbb{S}_P = \{\mathbf{x} \mid \mathbf{x} \in \{0,1\}^n, Hw(\mathbf{x}) = k\} \subset \{0,1\}^n = \mathbb{S}_E. \tag{2}$$

Let the public key $\mathbf{a} = (a_1, \ldots, a_n)$. The underlying PSTOF is $T : \mathbb{S}_E \mapsto \mathbb{C} = \{0, 1, \ldots, \sum_{i=1}^{n} a_i\}$; that is, for $\mathbf{x} \in \mathbb{S}_E$,

$$T(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle = \sum_{i=1}^{n} a_i x_i = c. \tag{3}$$

The density of both PKCs defined as $d = n/\log_2 \max \mathbf{a}$ can be made sufficiently high (>1), in which case asymptotically (3) will have more than one (i.e., $P_d > 1$) binary solution [14]:

$$P_d = \frac{|\mathbb{S}_E|}{|\mathbb{C}|} \geq \frac{2^n}{n \max \mathbf{a}} = \frac{2^n}{n 2^{n/d}} = 2^{(1-1/d)n - \log_2 n}. \tag{4}$$

The illustrations say that when $d > 1$, a valid ciphertext will have exponentially many $2^{(1-1/d)n-\log_2 n}$ preimages in $\mathbb{S}_E$ and a unique preimage in $\mathbb{S}_P$. In other words, the plaintext is hidden by exponentially many preimages. Even if a polynomial-time adversary conquers the one-wayness of the underlying subset sum problem and hence the underlying encryption function is only a preimage selective trapdoor (not one-way) function; the adversary just obtains polynomially many preimages to (3) which contain the plaintext with a negligible probability. High-density subset sum PKCs can base their security on the noninjectivity of the underlying function, which is a weaker intractability assumption.

Remark 6. There exist two ways to break Chor-Rivest [11] and the Okamoto-Tanaka-Uchiyama [12] cryptosystems: recovering the secret key [15] and solving the underlying knapsack problem. Some work still was done to make Chor-Rivest immune from key-recovery attacks [16–18]. However, some known cryptanalytic algorithms [14, 19–21] only can solve the preimages $\mathbf{x} \in \mathbb{S}_E = \{0,1\}^n$ but not necessarily the unique plaintext $\mathbf{x} \in \mathbb{S}_P = \{\mathbf{x} \mid \mathbf{x} \in \{0,1\}^n, Hw(\mathbf{x}) = k\}$. So in the adversary's point of view, the underlying problem remains the knapsack problem. This explains why we say in Remark 5 that the adversary cannot use the special structure of $\mathbb{S}_P$ to efficiently reverse the function $T : \mathbb{S}_P \mapsto \mathbb{C}$.

2.4.2. Rabin Cryptosystem. We also can view Rabin cryptosystem [22] as a PSTOF. For convenience of discussions, we just set Rabin encryption function as $c = x^2 (\mathrm{mod}\ N)$ with $N = pq$ being the product of two primes, and $\mathbb{S}_E = \mathbb{Z}_N^*$. So, the ciphertext space $\mathbb{C}$ consists of the quadratic residues modulo N; that is, $\mathbb{C} = \mathbb{QR}_N$. In order to uniquely recover a plaintext, we need to embed some redundant information in $\mathbb{S}_E$. The redundant information forms the special structure of $\mathbb{S}_P$. The PSTOF underlying Rabin cryptosystem is $T : \mathbb{Z}_N^* \mapsto \mathbb{QR}_N$; for $x \in \mathbb{Z}_N^*$, $T(x) = c = x^2 (\mathrm{mod}\, N)$. We note that each of the $\phi(N)/4$ quadratic residues modulo N has four roots, from which we use the redundant information to pick out the exact plaintext. Hence, the noninjectivity of the PSTOF for Rabin is given as

$$P_d = \frac{|\mathbb{S}_E|}{|\mathbb{C}|} = \frac{|\mathbb{Z}_N^*|}{|\mathbb{QR}_N|} = \frac{\phi(n)}{\phi(n)/4} = 4. \tag{5}$$

3. The Proposed Probabilistic Public Key Encryption Scheme

3.1. Knapsack Problems

Definition 7 (knapsack problem). Given a positive integral vector $\mathbf{a} = (a_1, \ldots, a_n)$ and an integer $s \in \mathbb{Z}$, the knapsack or subset sum problem is to find a binary vector $\mathbf{x} \in \{0,1\}^n$ such that $\langle \mathbf{a}, \mathbf{x} \rangle = s$. A knapsack problem is denoted as $\mathrm{KP}(\mathbf{a}, s)$. The density of the knapsack problem $\mathrm{KP}(\mathbf{a}, s)$ is defined as $d = n/\log_2 \max \mathbf{a}$.

The density of a compact knapsack problem $\mathrm{KP}(\mathbf{a}, s)$ imposes an important effect on the hardness of the problem. It was shown that if the density $d < 0.9408$, the knapsack problem $\mathrm{KP}(\mathbf{a}, s)$ can be solved with an overwhelming probability [23, 24].

Definition 8 (compact knapsack problem). Given a positive integral vector $\mathbf{a} = (a_1, \ldots, a_n)$, an integer $2 \le K \in \mathbb{Z}^+$, and an integer $s \in \mathbb{Z}$, the compact or general knapsack problem is to find a vector $\mathbf{x} = (x_1, \ldots, x_n)$ with $0 \le x_i \le K - 1$ such that $\langle \mathbf{a}, \mathbf{x} \rangle = s$. A compact knapsack problem is denoted as $\mathrm{CKP}_K(\mathbf{a}, s)$. The density of a compact knapsack problem $\mathrm{CKP}_K(\mathbf{a}, s)$ is defined as $d = n \log_2 K / \log_2 \max \mathbf{a}$.

If the density of a compact knapsack problem $\mathrm{CKP}_K(\mathbf{a}, s)$ is $d < 1$, it was shown that the compact knapsack problem can almost always be efficiently solved with lattice reduction algorithms [25]. In this paper, we consider the compact knapsack problem with $K = 192$.

3.2. The Proposed Message Encoding Function. The message space $\mathbb{M}$ consists of n blocks with each block 3 bit long; namely, $\mathbb{M} = \mathbb{Z}_8^n$. For a message $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{M}$, we firstly define a message encoding function that maps the message $\mathbf{m}$ into a vector in $\mathbf{Z}_{192}^n$ under an auxiliary key $\mathbf{r} = (r_1, \ldots, r_n) \in \{0,1\}^n$, which is randomly chosen by the encrypter. We define $F_i : \mathbb{Z}_8 \times \{0,1\} \mapsto \mathbb{Z}_{192}$ for $i = 1, \ldots, n$ as

$$F_i(m_i, r_i) = \left\langle 5^{8(i-1)+m_i} - 1 \right\rangle_{97} + 96 r_i \in \mathbb{Z}_{192}. \tag{6}$$

Remark 9. We note that 97 is a prime and 5 is a primitive root modulo 97, so the function $0 \le \langle 5^x - 1 \rangle_{97} \le 95$ forms a permutation when x runs through $\mathbb{Z}_{96}$ and hence $F_i(x, r)$ permutates $\mathbb{Z}_{192}$ when x and r run through $\mathbb{Z}_{96}$ and $\{0,1\}$, respectively. We denote

$$\mathbf{F}_i = \{F_i(m_i, r_i) \mid m_i \in \mathbb{Z}_8, r_i \in \{0,1\}\}. \tag{7}$$

It is easy to verify that $\bigcup_{i=1}^{12} \mathbf{F}_i$ forms a partition of $\mathbb{Z}_{192}$. So we can deduce that if i, m_i, and r_i are uniformly distributed over $\{1, \ldots, n\}$, $\mathbb{Z}_8$ and $\{0,1\}$, respectively, $F_i(m_i, r_i)$ is uniformly distributed over $\mathbb{Z}_{192}$. This means that the generating of each $F_i(m_i, r_i) \in \mathbf{F}_i$ can simulate the process of randomly choosing an integer from $\mathbb{Z}_{192}$, which allows the predesignated special structure of $\mathbb{S}_P$ defined in (9) as random as possible.

The message encoding function is defined as $\mathbf{F} : \mathbb{M} \times \{0,1\}^n \mapsto \mathbb{S}_P$:

$$\mathbf{F}(\mathbf{m}, \mathbf{r}) = (F_1(m_1, r_1), \ldots, F_n(m_n, r_n)), \tag{8}$$

where the plaintext space is

$$\mathbb{S}_P = \{\mathbf{F}(\mathbf{m}, \mathbf{r}) \mid \mathbf{m} \in \mathbb{M}, \mathbf{r} \in \{0,1\}^n\} = \mathbf{F}_1 \times \cdots \times \mathbf{F}_n. \tag{9}$$

In the rest of the paper, we also define the equivalent plaintext space $\mathbb{S}_E = \mathbb{Z}_{192}^n$ including the plaintext space $\mathbb{S}_P \subset \mathbb{S}_E$.

Remark 10. Another fact about F_i is

$$F_i(m, r) = F_{i+12}(m, r), \tag{10}$$

for $m \in \mathbb{Z}_8$ and $r \in \{0,1\}$ because $5^{96} = 1 \pmod{97}$. Hence, we have that if $i = j \pmod{12}$, $\mathbf{F}_i = \mathbf{F}_j$.

Remark 11. Now we remark on how to encode a message and decode a plaintext with respect to $\mathbf{F}$. We can construct n tables similar to truth-value tables to show the functions F_j. By observing (10), we see that the tables have a periodic structure with period 12. That is to say those j sharing a common residue modulo 12 have a same table. So only 12 tables suffice to list the values of $\mathbf{F}(\mathbf{m}, \mathbf{r})$. See Table 1. Here, we spend some more words on the message encoding function and Table 1. At present, we only consider the preceding three columns of the 12 subtables in Table 1, which list the values of r_j, m_j, and the corresponding $F_j(m_j, r_j)$, respectively. To encode a message $\mathbf{m} \in \mathbb{M}$ and given a randomly chosen auxiliary key $\mathbf{r} \in \{0,1\}^n$, for $i = 1, \ldots, n$, we determine the $F_i(m_i, r_i)$ one by one by looking up the subtable indexed by $i = \langle j \rangle_{12} \pmod{12}$ in Table 1 denoted as Sub-$\mathrm{T}_{\langle j \rangle_{12}}$. When $r_i = 0$ (or 1, resp.), we only search the 8 rows covered by $r_i = 0$ (or 1, resp.) in Sub-$\mathrm{T}_{\langle j \rangle_{12}}$, and output the integer in the column marked by $\mathbf{F}_i$ that lies in the same row with m_i as the value of $F_i(m_i, r_i)$. For example, to encode $\mathbf{m} = (5, 4, 7, 6)$ under the control of $\mathbf{r} = (0, 1, 1, 0)$, by looking up Sub-$\mathrm{T}_1$, we determine $F_1(m_1, r_1) = 20$. Similarly, we get $F_2(m_2, r_2) = 159$, $F_3(m_3, r_3) = 177$, $F_4(m_4, r_4) = 78$. So, we encode $(5, 4, 7, 6)$ as $\mathbf{F}(\mathbf{m}, \mathbf{r}) = (20, 159, 177, 78)$. On the contrary, we can decode a plaintext $\mathbf{x} \in \mathbb{S}_P$ by looking up the table. For example, $\mathbf{x} = (100, 101, 76, 6)$. We search the third column of Sub-T_1 for 100 which lies in the same line with $m_1 = 1$, so $m_1 = 1$. Similarly, we get $m_2 = 0$, $m_3 = 5$, $m_2 = 7$. Hence, $\mathbf{m} = (1, 0, 5, 7)$.

3.3. Onion Algorithm

3.3.1. Indistinguishability and Associated Integer Pairs. Given an integer pair $(a, b) \in (\mathbb{Z}^+)^2$ and $f_i \in \mathbf{F}_i$, we define $f_i \pmod{(a, b)} = (f_i \pmod a, f_i \pmod b)$ and the set $\mathbf{F}_i \pmod{(a, b)} = \{f_i \pmod{(a, b)} \mid f_i \in \mathbf{F}_i\}$. In fact, we can look $\mathbf{F}_i \pmod{(a, b)}$ as a transformation on $\mathbf{F}_i$. If the two cardinalities are identical, $|\mathbf{F}_i \pmod{(a, b)}| = |\mathbf{F}_i|$, the transformation forms a bijection.

Table 1: The values of $F_i(M_i, r_i)$ and their associated integer pairs.

(a)

$i = 1 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 34)	mod (2, 17)
0	0	0	(0, 0)	(0, 0)
	1	4	(0, 4)	(1, 4)
	2	24	(0, 24)	(0, 7)
	3	27	(0, 27)	(1, 10)
	4	42	(0, 8)	(0, 8)
	5	20	(0, 20)	(0, 3)
	6	7	(0, 7)	(1, 7)
	7	39	(0, 5)	(1, 5)
1	0	96	(0, 28)	(0, 11)
	1	100	(0, 32)	(0, 15)
	2	120	(0, 18)	(0, 1)
	3	123	(0, 21)	(1, 4)
	4	138	(0, 2)	(0, 2)
	5	116	(0, 14)	(0, 14)
	6	103	(0, 1)	(1, 1)
	7	135	(0, 33)	(1, 16)

(b)

$i = 2 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 33)	mod (3, 11)
0	0	5	(0, 5)	(2, 5)
	1	29	(0, 29)	(2, 7)
	2	52	(0, 19)	(1, 8)
	3	70	(0, 4)	(1, 4)
	4	63	(0, 30)	(0, 8)
	5	28	(0, 28)	(1, 6)
	6	47	(0, 14)	(2, 3)
	7	45	(0, 12)	(0, 1)
1	0	101	(0, 2)	(2, 2)
	1	125	(0, 26)	(2, 4)
	2	148	(0, 16)	(1, 5)
	3	166	(0, 1)	(1, 1)
	4	159	(0, 27)	(0, 5)
	5	124	(0, 25)	(1, 3)
	6	143	(0, 11)	(2, 0)
	7	141	(0, 9)	(0, 9)

(c)

$i = 3 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 31)	mod (1, 37)
0	0	35	(0, 4)	(0, 35)
	1	82	(0, 20)	(0, 8)
	2	26	(0, 26)	(0, 26)
	3	37	(0, 6)	(0, 0)
	4	92	(0, 30)	(0, 18)
	5	76	(0, 14)	(0, 2)
	6	93	(0, 0)	(0, 19)
	7	81	(0, 19)	(0, 7)

(c) Continued.

$i = 3 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 31)	mod (1, 37)
1	0	131	(0, 7)	(0, 20)
	1	178	(0, 23)	(0, 30)
	2	122	(0, 29)	(0, 11)
	3	133	(0, 9)	(0, 22)
	4	188	(0, 2)	(0, 3)
	5	172	(0, 17)	(0, 24)
	6	189	(0, 3)	(0, 4)
	7	177	(0, 22)	(0, 29)

(d)

$i = 4 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 46)	mod (2, 23)
0	0	21	(0, 21)	(1, 21)
	1	12	(0, 12)	(0, 12)
	2	64	(0, 18)	(0, 18)
	3	33	(0, 33)	(1, 10)
	4	72	(0, 26)	(0, 3)
	5	73	(0, 27)	(1, 4)
	6	78	(0, 32)	(0, 9)
	7	6	(0, 6)	(0, 6)
1	0	117	(0, 25)	(1, 2)
	1	108	(0, 16)	(0, 1)
	2	160	(0, 22)	(0, 22)
	3	129	(0, 37)	(1, 14)
	4	168	(0, 30)	(0, 7)
	5	169	(0, 31)	(1, 8)
	6	174	(0, 36)	(0, 13)
	7	102	(0, 10)	(0, 10)

(e)

$i = 5 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 29)	mod (1, 47)
0	0	34	(0, 5)	(0, 34)
	1	77	(0, 19)	(0, 30)
	2	1	(0, 1)	(0, 1)
	3	9	(0, 9)	(0, 9)
	4	49	(0, 20)	(0, 2)
	5	55	(0, 26)	(0, 8)
	6	85	(0, 27)	(0, 38)
	7	41	(0, 12)	(0, 41)
1	0	130	(0, 14)	(0, 36)
	1	173	(0, 28)	(0, 32)
	2	97	(0, 10)	(0, 3)
	3	105	(0, 18)	(0, 11)
	4	145	(0, 0)	(0, 4)
	5	151	(0, 6)	(0, 10)
	6	181	(0, 7)	(0, 40)
	7	137	(0, 21)	(0, 43)

(f)

$i = 6 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 39)	mod (3, 13)
0	0	15	(0, 15)	(0, 2)
	1	79	(0, 1)	(1, 1)
	2	11	(0, 11)	(2, 11)
	3	59	(0, 20)	(2, 7)
	4	8	(0, 8)	(2, 8)
	5	44	(0, 5)	(2, 5)
	6	30	(0, 30)	(0, 4)
	7	57	(0, 18)	(0, 5)
1	0	111	(0, 33)	(0, 7)
	1	175	(0, 19)	(1, 6)
	2	107	(0, 29)	(2, 3)
	3	155	(0, 38)	(2, 12)
	4	104	(0, 26)	(2, 0)
	5	140	(0, 23)	(2, 10)
	6	126	(0, 9)	(0, 9)
	7	153	(0, 36)	(0, 20)

(g)

$i = 7 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 34)	mod (2, 17)
0	0	95	(0, 27)	(1, 10)
	1	91	(0, 23)	(1, 6)
	2	71	(0, 3)	(1, 3)
	3	68	(0, 0)	(0, 0)
	4	53	(0, 19)	(1, 2)
	5	75	(0, 7)	(1, 7)
	6	88	(0, 20)	(0, 3)
	7	56	(0, 22)	(0, 5)
1	0	191	(0, 21)	(1, 4)
	1	187	(0, 17)	(1, 0)
	2	167	(0, 31)	(1, 14)
	3	164	(0, 28)	(0, 11)
	4	149	(0, 13)	(1, 13)
	5	171	(0, 1)	(1, 1)
	6	184	(0, 14)	(0, 14)
	7	152	(0, 16)	(0, 16)

(h)

$i = 8 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 33)	mod (3, 11)
0	0	90	(0, 24)	(0, 2)
	1	66	(0, 0)	(0, 0)
	2	43	(0, 10)	(1, 10)
	3	25	(0, 25)	(1, 3)
	4	32	(0, 32)	(2, 10)
	5	67	(0, 1)	(1, 1)
	6	48	(0, 15)	(0, 4)
	7	50	(0, 17)	(2, 6)

(h) Continued.

$i = 8 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 33)	mod (3, 11)
1	0	186	(0, 21)	(0, 10)
	1	162	(0, 30)	(0, 8)
	2	139	(0, 7)	(1, 7)
	3	121	(0, 22)	(1, 0)
	4	128	(0, 29)	(2, 7)
	5	163	(0, 31)	(1, 9)
	6	144	(0, 12)	(0, 1)
	7	146	(0, 14)	(2, 3)

(i)

$i = 9 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 31)	mod (1, 37)
0	0	60	(0, 29)	(0, 23)
	1	13	(0, 13)	(0, 13)
	2	69	(0, 7)	(0, 32)
	3	58	(0, 27)	(0, 21)
	4	3	(0, 3)	(0, 3)
	5	19	(0, 19)	(0, 19)
	6	2	(0, 2)	(0, 2)
	7	14	(0, 14)	(0, 14)
1	0	156	(0, 1)	(0, 8)
	1	109	(0, 16)	(0, 35)
	2	165	(0, 10)	(0, 17)
	3	154	(0, 30)	(0, 6)
	4	99	(0, 6)	(0, 25)
	5	115	(0, 22)	(0, 4)
	6	98	(0, 5)	(0, 24)
	7	110	(0, 17)	(0, 36)

(j)

$i = 10 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 46)	mod (2, 23)
0	0	74	(0, 28)	(0, 5)
	1	83	(0, 37)	(1, 14)
	2	31	(0, 31)	(1, 8)
	3	62	(0, 16)	(0, 16)
	4	23	(0, 23)	(1, 0)
	5	22	(0, 22)	(0, 22)
	6	17	(0, 17)	(1, 17)
	7	89	(0, 43)	(1, 20)
1	0	170	(0, 32)	(0, 9)
	1	179	(0, 41)	(1, 18)
	2	127	(0, 35)	(1, 12)
	3	158	(0, 20)	(0, 20)
	4	119	(0, 27)	(1, 4)
	5	118	(0, 26)	(0, 3)
	6	113	(0, 21)	(1, 21)
	7	185	(0, 1)	(1, 1)

(k)

$i = 11 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 29)	mod (1, 47)
0	0	61	(0, 3)	(0, 14)
	1	18	(0, 18)	(0, 18)
	2	94	(0, 7)	(0, 0)
	3	86	(0, 28)	(0, 39)
	4	46	(0, 17)	(0, 46)
	5	40	(0, 11)	(0, 40)
	6	10	(0, 10)	(0, 10)
	7	54	(0, 25)	(0, 7)
1	0	157	(0, 12)	(0, 16)
	1	114	(0, 27)	(0, 20)
	2	190	(0, 16)	(0, 2)
	3	182	(0, 8)	(0, 41)
	4	142	(0, 26)	(0, 1)
	5	136	(0, 20)	(0, 42)
	6	106	(0, 19)	(0, 12)
	7	150	(0, 5)	(0, 9)

(l)

$i = 0 \pmod{12}$				
r_i	m_i	$\mathbf{F}_i$	mod (1, 39)	mod (3, 13)
0	0	80	(0, 2)	(2, 2)
	1	16	(0, 16)	(1, 3)
	2	84	(0, 6)	(0, 6)
	3	36	(0, 36)	(0, 10)
	4	87	(0, 9)	(0, 9)
	5	51	(0, 12)	(0, 12)
	6	65	(0, 26)	(2, 0)
	7	38	(0, 38)	(2, 12)
1	0	176	(0, 20)	(2, 7)
	1	112	(0, 34)	(1, 11)
	2	180	(0, 24)	(0, 11)
	3	132	(0, 15)	(0, 2)
	4	183	(0, 27)	(0, 1)
	5	147	(0, 30)	(0, 4)
	6	161	(0, 5)	(2, 5)
	7	134	(0, 17)	(2, 4)

Definition 12. Given an integer pair $(a, b) \in (\mathbb{Z}^+)^2$, if $|\mathbf{F}_i(\mathrm{mod}\,(a, b))| = |\mathbf{F}_i|$, we call $\mathbf{F}_i$ to be distinguishable modulo (a, b), or (a, b) is an associated integer pair with $\mathbf{F}_i$.

Example 13. Decide whether $\mathbf{F}_{61}$ is distinguishable modulo $(1, 34)$ and $(2, 19)$ or not, respectively. We note that $\mathbf{F}_{61} = \mathbf{F}_1$ contains the integers listed in the third column of Sub-T_1. Calculations show that $|\mathbf{F}_{61}| = |\mathbf{F}_{61}(\mathrm{mod}\ (1, 34))| = 16$. The 16 values in $\mathbf{F}_{61}(\mathrm{mod}\ (1, 34))$ are listed in the forth column of Sub-T_1. So, $\mathbf{F}_{61}$ is distinguishable modulo $(1, 34)$. However, $|\mathbf{F}_{61}| = 16 \neq |\mathbf{F}_{61}(\mathrm{mod}\,(2, 19))| = 14$ in that 4 and 42 modulo $(2, 19)$ produce a same integer pair $(0, 4)$, and 24 and 138 modulo $(2, 19)$ produce a same integer pair $(0, 5)$. So $\mathbf{F}_{61}$ is not distinguishable modulo $(2, 19)$.

Table 2: Associated integer pairs with $\mathbf{F}_i$.

Set	Associated integer pairs
AIP_1	$\{(1, 34), (2, 17)\}$
AIP_2	$\{(1, 33), (3, 11)\}$
AIP_3	$\{(1, 31), (1, 37)\}$
AIP_4	$\{(1, 46), (2, 23)\}$
AIP_5	$\{(1, 29), (1, 47)\}$
AIP_6	$\{(1, 39), (3, 13)\}$
AIP_7	$\{(1, 34), (2, 17)\}$
AIP_8	$\{(1, 33), (3, 11)\}$
AIP_9	$\{(1, 31), (1, 37)\}$
AIP_{10}	$\{(1, 46), (2, 23)\}$
AIP_{11}	$\{(1, 29), (1, 47)\}$
AIP_0	$\{(1, 39), (3, 13)\}$

Table 3: Suggested security parameters.

	Security level		
	Moderate	Higher	Highest
n	60	90	120
Public key size	21282	47025	82796
Information rate	0.49	0.50	0.51
Costs for BFA I	2^{240}	2^{360}	2^{480}
Costs for BFA II	$2^{125.9}$	$2^{186.5}$	$2^{246.9}$
P_{d1}	$2^{86.9}$	$2^{146.1}$	$2^{205.7}$
P_{d2}	$2^{587.8}$	$2^{899.4}$	$2^{1211.6}$
P_{d3}	$2^{111.4}$	2^{183}	$2^{255.1}$

Remark 14. When $\mathbf{F}_i$ is distinguishable modulo (a, b), there is a bijection between the integers in $\mathbf{F}_i$ and the integer pairs in $\mathbf{F}_i(\mathrm{mod}\ (a, b))$. For example, the third and the forth columns of Sub-T_1 listed the integers in $\mathbf{F}_1$ and the integer pairs in $\mathbf{F}_1(\mathrm{mod}\ (1, 34))$. For example, if we know an integer $f_1 \in \mathbf{F}_1$ satisfies $f_1(\mathrm{mod}\,(1, 34)) = (0, 32)$, we search the forth column generated by $\mathbf{F}_1(\mathrm{mod}\ (1, 34))$ for $(0, 32)$ in Sub- T_1 and find that $(0, 32)$ lies in the same line with 100, so $f_1 = 100$.

Remark 15. From Remark 10, we have that, for congruent indices j's modulo 12, we get a same set $\mathbf{F}_j(\mathrm{mod}\ (a, b))$. For each $i = \langle j \rangle_{12} (\mathrm{mod}\ 12)$, we give two integer pairs (a_{i1}, b_{i1}) and (a_{i2}, b_{i2}) associated with $\mathbf{F}_i$, and the corresponding integer pairs modulo (a_{i1}, b_{i1}) and (a_{i2}, b_{i2}), respectively, in the forth and fifth column of Sub-$T_{\langle j \rangle_{12}}$. We denote the set consisting of the two integer pairs (a_{j1}, b_{j1}) and (a_{j2}, b_{j2}) as $AIP_{\langle j \rangle_{12}}$. For example, $AIP_{\langle 73 \rangle_{12}} = AIP_1 = \{(1, 34), (2, 17)\}$. In Table 2, we list all the 12 sets of $AIP_{\langle j \rangle_{12}}$.

Remark 16. Note that $\mathbf{F}_i$ is distinguishable modulo (a, b) if and only if $\mathbf{F}_i$ is distinguishable modulo $(b, a) = (a, b)^T$. For example, $AIP_1^T = \{(34, 1), (17, 2)\}$ also contains integer pairs associated with $\mathbf{F}_1$. If we know that an integer $f_1 \in \mathbf{F}_1$ satisfies $f_1(\mathrm{mod}\,(34, 1)) = (32, 0)$, we also can determine the value of f_1 by searching the forth column generated by $\mathbf{F}_1(\mathrm{mod}\,(1, 34))$ in Sub-T_1 for $(0, 32)$, and we find a unique

$f_1 = 100$ with $f_1(\mathrm{mod}\,(1,34)) = (0,32)$. So the unique integer $f_1 \in \mathbf{F}_1$ is $f_1 = 100$ such that $f_1(\mathrm{mod}\,(34,1)) = (32,0)$.

3.3.2. Simultaneous Diophantine Equation. Now, we consider a simultaneous Diophantine equation problem as follows; given integers s_1 and s_2 and positive integer sequences $\mathbf{u} = (u_1,\ldots,u_n)$ and $\mathbf{v} = (v_1,\ldots,v_n)$, find $\mathbf{x} = (f_1,\ldots,f_n) \in \mathbb{S}_P$ defined in (9) such that

$$s_1 = \langle \mathbf{u}, \mathbf{x} \rangle, \qquad s_2 = \langle \mathbf{v}, \mathbf{x} \rangle. \tag{11}$$

It is pointed out that if $\max \mathbf{u} \le 2^{d_{\mathbf{u}} n}$ and $\max \mathbf{v} \le 2^{d_{\mathbf{v}} n}$, for some constants $d_{\mathbf{u}}$ and $d_{\mathbf{v}}$ subject to $d_{\mathbf{u}} + d_{\mathbf{v}} < \log_2 192$, (s_1, s_2) will have exponentially many preimages in $\mathbb{S}_E = \mathbb{Z}_{192}^n$ satisfying (11); namely,

$$P_d \ge \frac{\left|\mathbb{S}_E\right|}{(191 \max \mathbf{u})\,(191 \max \mathbf{v})} \ge \frac{192^n}{191^2 2^{d_{\mathbf{u}}+d_{\mathbf{v}}}}, \tag{12}$$

which is $\mathcal{O}(2^{\log_2 192 - d_{\mathbf{u}} - d_{\mathbf{v}}})$. What we mean is that if we require $\mathbf{x} = (f_1,\ldots,f_n) \in \mathbb{S}_E$ and $d_{\mathbf{u}} + d_{\mathbf{v}} < \log_2 192$, (11) will be a many-to-one function. In the subsequent contents, we will define a condition such that (11) will have at most one solution in $\mathbb{S}_P$ (to guarantee the unique decipherability for ciphertexts) and propose an algorithm to solve (11). Besides, some transformations will be introduced in the construction of the proposed PKC to derive a compact knapsack problem equipped with noninjectivity, a trapdoor, and one-wayness natures.

3.3.3. An Easy Problem and the Algorithm Solving It. For some special $\mathbf{u}$ and $\mathbf{v}$, we can efficiently determine f_n in (11) as stated in the following lemma.

Lemma 17. *Assuming that* (11) *has solutions in* $\mathbb{S}_P$, $gcd_n^{\mathbf{u}} = gcd_n^{\mathbf{v}} = 1$, *and* $(gcd_{n-1}^{\mathbf{u}}, gcd_{n-1}^{\mathbf{v}}) \in AIP_{\langle n \rangle_{12}} \cup AIP^T_{\langle n \rangle_{12}}$, *then* $f_n \in \mathbf{F}_n$ *can be efficiently and uniquely determined.*

Proof. We first note that $\gcd_{n-1}^{\mathbf{u}} \mid u_i$ and $\gcd_{n-1}^{\mathbf{v}} \mid v_i$, for $i = 1,\ldots,n-1$, so the two equations in (11) modulo $\gcd_{n-1}^{\mathbf{u}}$ and $\gcd_{n-1}^{\mathbf{v}}$, respectively, give $s_1 = u_n f_n (\mathrm{mod}\ \gcd_{n-1}^{\mathbf{u}})$ and $s_2 = v_n f_n (\mathrm{mod}\ \gcd_{n-1}^{\mathbf{v}})$. Observing that $\gcd_n^{\mathbf{u}} = \gcd\,(\gcd_{n-1}^{\mathbf{u}}, u_n) = 1$, we can invert u_n and similarly v_n:

$$\begin{aligned}(f_{n1}, f_{n2}) &= f_n\left(\mathrm{mod}\left(\gcd_{n-1}^{\mathbf{u}}, \gcd_{n-1}^{\mathbf{v}}\right)\right)\\ &= \left(u_n^{-1} s_1 \left(\mathrm{mod}\ \gcd_{n-1}^{\mathbf{u}}\right), v_n^{-1} s_2 \left(\mathrm{mod}\ \gcd_{n-1}^{\mathbf{v}}\right)\right).\end{aligned} \tag{13}$$

If $(\gcd_{n-1}^{\mathbf{u}}, \gcd_{n-1}^{\mathbf{v}}) \in \mathrm{AIP}_{\langle n \rangle_{12}}$, we determine the unique corresponding value f_n according to Remark 14. If $(\gcd_{n-1}^{\mathbf{u}}, \gcd_{n-1}^{\mathbf{v}}) \in \mathrm{AIP}^T_{\langle n \rangle_{12}}$, we use the method given in Remark 16 to determine the unique f_n. □

The above lemma says that for some special $\mathbf{u}$ and $\mathbf{v}$, we can determine the solution $f_n,\ldots,f_1$ to (11) one by one as if we peel off an onion, so the name *Onion Algorithm* comes.

Theorem 18. *Assuming that* (11) *has solutions in* $\mathbb{S}_P$:

$$gcd_n^{\mathbf{u}} = gcd_n^{\mathbf{v}} = 1, \tag{14}$$

and that, for $i = 2,\ldots,n$,

$$\left(\frac{gcd_{i-1}^{\mathbf{u}}}{gcd_i^{\mathbf{u}}}, \frac{gcd_{i-1}^{\mathbf{v}}}{gcd_i^{\mathbf{v}}}\right) \in AIP_{\langle i \rangle_{12}} \cup AIP^T_{\langle i \rangle_{12}}, \tag{15}$$

the solution $\mathbf{x} = (f_1,\ldots,f_n) \in \mathbb{S}_P$ *to* (11) *can be efficiently and uniquely determined.*

Proof. From Lemma 17, f_n can be efficiently and uniquely determined.

For $i = n-1,\ldots,2$, assuming that $f_{i+1},\ldots,f_n$ have been uniquely determined, we know that

$$s_1 - \sum_{j=i+1}^{n} u_j f_j = \sum_{j=1}^{i} u_j f_j. \tag{16}$$

We note that $\gcd_i^{\mathbf{u}} \mid u_j$ for $1 \le j \le i$, so $\gcd_i^{\mathbf{u}}$ divides the right side and hence the left side of (16). We set

$$s_{i1} = \frac{s_1 - \sum_{j=i+1}^{n} u_j f_j}{\gcd_i^{\mathbf{u}}} = \sum_{j=1}^{i} \frac{u_j}{\gcd_i^{\mathbf{u}}} f_j. \tag{17}$$

A similar analysis also applies to the second equation of (11), so we set $s_{i2} = (s_2 - \sum_{j=i+1}^{n} v_j f_j)/\gcd_i^{\mathbf{v}}$ to obtain

$$s_{i1} = \sum_{j=1}^{i} \frac{u_j}{\gcd_i^{\mathbf{u}}} f_j, \qquad s_{i2} = \sum_{j=1}^{i} \frac{v_j}{\gcd_i^{\mathbf{v}}} f_j. \tag{18}$$

If we view (18) as a new simultaneous Diophantine equation problem with integer sequences $(u_1/\gcd_i^{\mathbf{u}},\ldots,u_i/\gcd_i^{\mathbf{u}})$ and $(v_1/\gcd_i^{\mathbf{v}},\ldots,v_i/\gcd_i^{\mathbf{v}})$, what follows shows that the new problem satisfies the conditions in Lemma 17.

Firstly, (11) has solutions, and $f_{i+1},\ldots,f_n$ have been uniquely determined, so (18) must have solutions.

Secondly, recalling the meaning of $\gcd_i^{\mathbf{u}}$, we claim that these entries divided by their gcd must be relatively prime:

$$\gcd\left(\frac{u_1}{\gcd_i^{\mathbf{u}}},\ldots,\frac{u_i}{\gcd_i^{\mathbf{u}}}\right) = \gcd\left(\frac{v_1}{\gcd_i^{\mathbf{v}}},\ldots,\frac{v_i}{\gcd_i^{\mathbf{v}}}\right) = 1. \tag{19}$$

Thirdly,

$$\begin{aligned}\gcd\left(\frac{u_1}{\gcd_i^{\mathbf{u}}},\ldots,\frac{u_{i-1}}{\gcd_i^{\mathbf{u}}}\right) &= \frac{\gcd_{i-1}^{\mathbf{u}}}{\gcd_i^{\mathbf{u}}},\\ \gcd\left(\frac{v_1}{\gcd_i^{\mathbf{v}}},\ldots,\frac{v_{i-1}}{\gcd_i^{\mathbf{v}}}\right) &= \frac{\gcd_{i-1}^{\mathbf{v}}}{\gcd_i^{\mathbf{v}}}.\end{aligned} \tag{20}$$

(1) Compute (f_{n1}, f_{n2}) according to (13).
(2) **if** $(\gcd^{\mathbf{u}}_{n-1}, \gcd^{\mathbf{v}}_{n-1}) \in \mathrm{AIP}_{\langle n\rangle_{12}}$ **then**
(3) Search for (f_{n1}, f_{n2}) in Sub-$T_{\langle n\rangle_{12}}$ generated by $\mathbf{F}_n(\bmod (\gcd^{\mathbf{u}}_{n-1}, \gcd^{\mathbf{v}}_{n-1}))$ to get $f_n \in \mathbf{F}_n$ such that $f_n(\bmod (\gcd^{\mathbf{u}}_{n-1}, \gcd^{\mathbf{v}}_{n-1})) = (f_{n1}, f_{n2})$. Store f_n.
(4) **end if**
(5) **if** $(\gcd^{\mathbf{u}}_{n-1}, \gcd^{\mathbf{v}}_{n-1}) \in \mathrm{AIP}^T_{\langle n\rangle_{12}}$ **then**
(6) Search for (f_{n2}, f_{n1}) in Sub-$T_{\langle n\rangle_{12}}$ generated by $\mathbf{F}_n(\bmod (\gcd^{\mathbf{v}}_{n-1}, \gcd^{\mathbf{u}}_{n-1}))$ to get $f_n \in \mathbf{F}_n$ such that $f_n(\bmod (\gcd^{\mathbf{v}}_{n-1}, \gcd^{\mathbf{u}}_{n-1})) = (f_{n2}, f_{n1})$. Store f_n.
(7) **end if**
(8) **for** $i = n-1, \ldots, 2$ **do**
(9) Compute (18) and

$$(f_{i1}, f_{i2}) = \left(s_{i1}\left(\frac{u_i}{\gcd^{\mathbf{u}}_i}\right)^{-1}\left(\bmod \frac{\gcd^{\mathbf{u}}_{i-1}}{\gcd^{\mathbf{u}}_i}\right), s_{i2}\left(\frac{v_i}{\gcd^{\mathbf{v}}_i}\right)^{-1}\left(\bmod \frac{\gcd^{\mathbf{v}}_{i-1}}{\gcd^{\mathbf{v}}_i}\right)\right). \quad (*)$$

(10) **if** $(\gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i, \gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i) \in \mathrm{AIP}_{\langle i\rangle_{12}}$ **then**
(11) Search the column in Sub-$T_{\langle i\rangle_{12}}$ generated by $\mathbf{F}_i$ modulo $(\gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i, \gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i)$ for (f_{i1}, f_{i2}) such that $(f_{i1}, f_{i2}) = f_i(\bmod (\gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i, \gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i))$. Store f_i.
(12) **end if**
(13) **if** $(\gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i, \gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i) \in \mathrm{AIP}^T_{\langle i\rangle_{12}}$ **then**
(14) Search the column in Sub-$T_{\langle i\rangle_{12}}$ generated by $\mathbf{F}_i$ modulo $(\gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i, \gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i)$ for (f_{i2}, f_{i1}) such that $(f_{i2}, f_{i1}) = f_i(\bmod (\gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i, \gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i))$. Store f_i.
(15) **end if**
(16) **end for**
(17) Compute and store $f_1 = (s_1 - \sum_{i=2}^n u_i f_i)/u_1$.
(18) **return** $\mathbf{x} = (f_1, \ldots, f_n)$.

Algorithm 1: *Onion Algorithm.*

From (15) and the previous two proven things, we know that (18) indeed satisfies the conditions in Lemma 17. So we can efficiently determine a unique $f_i \in \mathbf{F}_i$.

Finally, once $f_2, \ldots, f_n$ have been uniquely determined, it is a trivial thing to determine a unique f_1 just by computing

$$f_1 = \frac{s_1 - \sum_{i=2}^n u_i f_i}{u_1} \quad \text{or} \quad \frac{s_2 - \sum_{i=2}^n v_i f_i}{v_1}. \quad (21)$$

Both values on the right side of the above equation must be not only identical but also in $\mathbf{F}_1$ because (11) has solutions in $\mathbb{S}_P$ and $f_2, \ldots, f_n$ have been uniquely determined. □

Given the problem (11), satisfying the conditions presented in Theorem 18, the *Onion Algorithm* for solving (11) is summarized as shown in Algorithm 1.

Now we use a toy example to illustrate what we discuss in this subsection about the *Onion Algorithm*.

Example 19. Assume that the following equations have solutions in $\mathbb{S}_P$; find the solution $\mathbf{x} = (f_1, f_2, f_3)$ to $11f_1 + 9f_2 + 13f_3 = 2926$, $93f_1 + 62f_2 + 89f_3 = 20783$. We can verify that $(\gcd^{\mathbf{u}}_3, \gcd^{\mathbf{v}}_3) = (1, 1)$, $(\gcd^{\mathbf{u}}_2, \gcd^{\mathbf{v}}_2) = (1, 31)$, $(\gcd^{\mathbf{u}}_1, \gcd^{\mathbf{v}}_1) = (11, 93)$, and $(\gcd^{\mathbf{u}}_2/\gcd^{\mathbf{u}}_3, \gcd^{\mathbf{v}}_2/\gcd^{\mathbf{v}}_3) = (1, 31) \in \mathrm{AIP}_3$, $(\gcd^{\mathbf{u}}_1/\gcd^{\mathbf{u}}_2, \gcd^{\mathbf{v}}_1/\gcd^{\mathbf{v}}_2) = (11, 3) \in \mathrm{AIP}^T_2$, so (14) and (15) are satisfied. Hence, we use the *Onion Algorithm* to compute $(f_{31}, f_{32}) = (2926 \times 13^{-1}(\bmod 1), 20783 \times 89^{-1}(\bmod 31)) = (0, 20)$. We look up Sub-$T_3$ and find that $f_3 = 82(\bmod (1, 31)) = (0, 20)$. So $f_3 = 82$. We compute $s_{21} = (2926 - 13 \times 82)/1 = 1860$, $s_{22} = (20783 - 89 \times 82)/31 = 435$. Then we compute $(f_{21}, f_{22}) = (1860 \times 9^{-1}(\bmod 11), 435 \times (62/31)^{-1}(\bmod 93/31)) = (5, 0)$. We search Sub-$T_2$ for $(0, 5)$ and find that $f_2 = 159 \pmod{(3, 11)} = (0, 5)$. So we determine $f_2 = 159$. Finally, we get $f_1 = (2926 - 9 \times 159 - 13 \times 82)/11 = 39$. Thus, a unique $\mathbf{x} = (39, 159, 82)$ is determined.

3.4. The Proposed Probabilistic Public Key Encryption. In this subsection, we use the results obtained in previous subsections to derive a probabilistic public key encryption.

3.4.1. Key Generation. Randomly choose two sequences $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n) \in (\mathbb{Z}^+)^n$, satisfying conditions (14) and (15) given in Theorem 18, and a two-dimensional invertible square matrix with positive integer entries upper bounded by a constant,

$$\Delta = \begin{pmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{pmatrix}, \quad \delta_{ij} = \mathcal{O}(1), \quad 1 \le i,\ j \le 2. \tag{22}$$

Compute

$$\begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix} = \begin{pmatrix} g_1 & \cdots & g_n \\ h_1 & \cdots & h_n \end{pmatrix} = \Delta \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}. \tag{23}$$

Randomly choose two primes $p \neq q$ such that

$$p > \sum_{i=1}^{n} g_i \max \mathbf{F}_i, \qquad q > \sum_{i=1}^{n} h_i \max \mathbf{F}_i. \tag{24}$$

Let $N = pq$. Compute the vector $\mathbf{b} = (b_1, \ldots, b_n)$ using the Chinese remainder theorem,

$$b_i \equiv g_i \pmod{p}, \qquad b_i \equiv h_i \pmod{q}. \tag{25}$$

Randomly choose an integer $w \in \mathbb{Z}_N^*$ and compute

$$a_i = w b_i \pmod{N}, \quad i = 1, \ldots, n. \tag{26}$$

The public key is $\mathbf{a} = (a_1, \ldots, a_n)$. The secret key consists of p, q, and $w^{-1} \pmod{N}$, Δ^{-1}. When decrypting a ciphertext, the decrypter also needs to store the values of $\gcd_i^{\mathbf{u}}$, $\gcd_i^{\mathbf{v}}$, $i = 1, \ldots, n$, and Table 1.

3.4.2. Encryption. To encrypt a message $\mathbf{m} = (m_1, \ldots, m_n) \in \mathbb{M} = \mathbb{Z}_8^n$, the encrypter randomly chooses an auxiliary key $\mathbf{r} = (r_1, \ldots, r_n) \in \{0, 1\}^n$ and computes the plaintext $\mathbf{x} = (f_1, \ldots, f_n) = \mathbf{F}(\mathbf{m}, \mathbf{r})$ and the ciphertext using the public key $\mathbf{a}$,

$$c = \langle \mathbf{a}, \mathbf{x} \rangle. \tag{27}$$

3.4.3. Decryption. To decipher a ciphertext c, the decrypter firstly computes $c_N = \langle w^{-1} c \rangle_N = w^{-1} c \pmod{N}$, and then $c_p = \langle c_N \rangle_p$, $c_q = \langle c_N \rangle_q$. Secondly, the decrypter computes

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \Delta^{-1} \begin{pmatrix} c_p \\ c_q \end{pmatrix}. \tag{28}$$

Thirdly, the decrypter uses the *Onion Algorithm* to determine a unique solution $\mathbf{x} = (f_1, \ldots, f_n) \in \mathbb{S}_P$ to (11). Finally, the decrypter decodes $\mathbf{x}$ into the corresponding message $\mathbf{m}$ by using the method illustrated in Remark 11.

3.4.4. Why Decryption Works. To illustrate why the decryption works, we observe that $c_N = \langle \mathbf{b}, \mathbf{x} \rangle \pmod{N}$ from (26), so $c_p = \langle \mathbf{g}, \mathbf{x} \rangle \pmod{p}$ and $c_q = \langle \mathbf{h}, \mathbf{x} \rangle \pmod{q}$ according to (25). From (23), we know that $g_i = \delta_{11} u_i + \delta_{12} v_i > 0$, so $0 \le c_p < p$ from (24). Now we conclude that $c_p = \langle \mathbf{g}, \mathbf{x} \rangle$ and similarly $c_q = \langle \mathbf{h}, \mathbf{x} \rangle$. Hence, we have

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \Delta^{-1} \begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}, \tag{29}$$

from which (11) is derived. It is easy to verify that (11) satisfies the conditions listed in Theorem 18, and hence can be efficiently solved with *Onion Algorithm* to give a unique solution $\mathbf{x} \in \mathbb{S}_P$. Then $\mathbf{x}$ is decoded into the original message $\mathbf{m}$.

3.4.5. On Generating $\mathbf{u}$ *and* $\mathbf{v}$. Now, we give an algorithm to generate the n-dimensional positive integer sequences $\mathbf{u}$ and $\mathbf{v}$ satisfying conditions (14) and (15) in Theorem 18.

Theorem 20. *The generated* $\mathbf{u}$ *and* $\mathbf{v}$ *satisfy* (14) *and* (15).

Proof. We note that, for $i = 1, \ldots, n$,

$$\begin{aligned} \gcd_i^{\mathbf{u}} &= \gcd(u_1, \ldots, u_i) \\ &= \gcd\left(\zeta_1 \prod_{j=1}^{n} \alpha_j, \ldots, \zeta_i \prod_{j=i}^{n} \alpha_j\right) \\ &= \gcd\left(\prod_{j=1}^{n} \alpha_j, \ldots, \prod_{j=i}^{n} \alpha_j\right) = \prod_{j=i}^{n} \alpha_j. \end{aligned} \tag{30}$$

Similarly, $\gcd_i^{\mathbf{v}} = \gcd(v_1, \ldots, v_i) = \prod_{j=i}^{n} \beta_j$. We immediately have $\gcd_n^{\mathbf{u}} = \alpha_n = \gcd_n^{\mathbf{v}} = \beta_n = 1$. Thus, (14) is satisfied. For $i = 2, \ldots, n$, we have

$$\begin{aligned} \left(\frac{\gcd_{i-1}^{\mathbf{u}}}{\gcd_i^{\mathbf{u}}}, \frac{\gcd_{i-1}^{\mathbf{v}}}{\gcd_i^{\mathbf{v}}}\right) &= \left(\frac{\prod_{j=i-1}^{n} \alpha_j}{\prod_{j=i}^{n} \alpha_j}, \frac{\prod_{j=i-1}^{n} \beta_j}{\prod_{j=i}^{n} \beta_j}\right) \\ &= (\alpha_{i-1}, \beta_{i-1}) \in \mathrm{AIP}_{\langle i \rangle_{12}} \cup \mathrm{AIP}_{\langle i \rangle_{12}}^{T}, \end{aligned} \tag{31}$$

so (15) is satisfied, as desired. □

Now we use an example to illustrate Algorithm 2.

Example 21. We randomly choose $(\alpha_1, \beta_1) = (11, 3) \in \mathrm{AIP}_2 \cup \mathrm{AIP}_2^T$, $(\alpha_2, \beta_2) = (1, 31) \in \mathrm{AIP}_3 \cup \mathrm{AIP}_3^T$ and $\zeta_2 = 9$, $\zeta_3 = 13$, $\eta_2 = 2$, $\eta_3 = 89$, and set $\zeta_1 = \eta_1 = \alpha_3 = \beta_3 = 1$, so we can compute $u_1 = \zeta_1 \alpha_1 \alpha_2 \alpha_3 = 11$, $u_2 = \zeta_2 \alpha_2 \alpha_3 = 9$, $u_3 = \zeta_3 \alpha_3 = 13$, and $v_1 = \eta_1 \beta_1 \beta_2 \beta_3 = 93$, $v_2 = \eta_2 \beta_2 \beta_3 = 62$, $v_3 = \eta_3 \beta_3 = 89$, which are the coefficients in Example 19.

3.4.6. Remarks and Suggested Security Parameters. The suggested parameters n are listed in Table 3.

We provide some remarks on the proposed PKC as follows.

Remark 22. We can choose a 2-dimensional square matrix Δ with determinant equal to ± 1 because the inverse Δ^{-1} can be easily represented when $|\Delta| = \pm 1$.

(1) For $i = 1, \ldots, n-1$, randomly choose $(\alpha_i, \beta_i) \in \mathrm{AIP}_{\langle i+1\rangle_{12}} \cup \mathrm{AIP}^T_{\langle i+1\rangle_{12}}$ with repetition permitted.
(2) Randomly choose $2(n-1)$ numbers $\zeta_2, \ldots, \zeta_n$ and $\eta_2, \ldots, \eta_n$ and set $\zeta_1 = \eta_1 = \alpha_n = \beta_n = 1$ such that $\gcd(\zeta_i, \alpha_j) = \gcd(\eta_i, \beta_j) = 1$ for $1 \le i, j \le n$, and $\gcd(\zeta_i, \zeta_{i+1}) = \gcd(\eta_i, \eta_{i+1}) = 1$ for $i = 1, \ldots, n-1$.
(3) **for** $i = 1, \ldots, n$ **do**
(4) $u_i = \zeta_i \prod_{j=i}^{n} \alpha_j, \quad v_i = \eta_i \prod_{j=i}^{n} \beta_j. \quad (\star)$
(5) **end for**
(6) **return** $\mathbf{u} = (u_1, \ldots, u_n)$, $\mathbf{v} = (v_1, \ldots, v_n)$.

Algorithm 2: Generating **u** and **v**.

Remark 23. We should choose p and q slightly greater than $\sum_{i=1}^{n} g_i \max \mathbf{F}_i$ and $\sum_{i=1}^{n} h_i \max \mathbf{F}_i$, respectively. See (24). So for convenience of discussions, we always assume that

$$p \approx \sum_{i=1}^{n} g_i \max \mathbf{F}_i, \qquad q \approx \sum_{i=1}^{n} h_i \max \mathbf{F}_i. \tag{32}$$

Remark 24. In Algorithm 2, we suggest choosing ζ_i and η_i with some special lengths in order that the generated u_i's (v_i's, resp.) share an almost same binary length; that is,

$$|\zeta_i|_2 \approx \left|\prod_{j=1}^{n} \alpha_j\right|_2 - \left|\prod_{j=i}^{n} \alpha_j\right|_2, \qquad |\eta_i|_2 \approx \left|\prod_{j=1}^{n} \beta_j\right|_2 - \left|\prod_{j=i}^{n} \beta_j\right|_2, \tag{33}$$

So, for $i = 1, \ldots, n-1$, we have

$$|u_i|_2 \approx \left|\prod_{j=1}^{n} \alpha_j\right|_2 = |u_1|_2, \qquad |v_i|_2 \approx \left|\prod_{j=1}^{n} \beta_j\right|_2 = |v_1|_2. \tag{34}$$

For convenience of discussions, we also assume that u_1 and v_1 have an almost same binary length, so from (34), we have

$$|u_i|_2 \approx |u_1|_2 \approx |v_i|_2 \approx |v_1|_2. \tag{35}$$

4. Performance

This section analyzes the performance related issues, such as the computational complexity of the encryption and decryption algorithms, the public key size and the information rate.

4.1. Estimation of Public Key Size. We first point out two facts. Firstly, when generating **u** and **v** by using Algorithm 2, for $i = 1, \ldots, n-1$, we randomly choose $(\alpha_i, \beta_i) \in \mathrm{AIP}_{\langle i+1\rangle_{12}} \cup \mathrm{AIP}^T_{\langle i+1\rangle_{12}}$ in the first step. We also set $\alpha_n = \beta_n = 1$. So from Table 2, we see that $\alpha_i \beta_i \le 47$, for $i = 1, \ldots, n-1$. Secondly, $\max \mathbf{F}_i \le 191$.

From (23), we rewrite g_i and h_i as $g_i = \delta_{11} u_i + \delta_{12} v_i$ and $h_i = \delta_{21} u_i + \delta_{22} v_i$ with $\delta_{ij} = \mathcal{O}(1)$, and from (35), we have $|g_i|_2 \approx |h_i|_2 \approx |u_i|_2 \approx |v_i|_2 \approx |u_1|_2 \approx |v_1|_2$.

We recall that the public key consists of n integers $\mathbf{a} = (a_1, \ldots, a_n)$, and the length of each of them is upper bounded by that of $N = pq$, so from (32) we have

$$\begin{aligned} |N|_2 &\approx |p|_2 + |q|_2 \le \left|191 \sum_{i=1}^{n} g_i\right|_2 + \left|191 \sum_{i=1}^{n} h_i\right|_2 \\ &\approx \left|191 n g_i\right|_2 + \left|191 n h_i\right|_2 \approx \left|191^2 n^2 u_1 v_1\right|_2 \\ &= \left|191^2 n^2 \prod_{i=1}^{n-1} \alpha_i \beta_i\right|_2 \le \left|191^2 n^2 47^{n-1}\right|_2 \\ &\approx (n-1) \log_2 47 + 2\log_2 n + 2\log_2 191. \end{aligned} \tag{36}$$

So the public key size is estimated via

$$n|N|_2 \le n(n-1) \log_2 47 + 2n \log_2 n + 2n \log_2 191, \tag{37}$$

which is upper bounded by $\mathcal{O}(n^2)$.

4.2. Computational Complexity. We analyze the computational costs for encrypting a message and decrypting a ciphertext. Given **m** and **r**, we only need $\mathcal{O}(n)$ bit operations to compute **x**. To encrypt **x**, the encrypter only needs to perform n multiplications with the multipliers a_i bounded by $\mathcal{O}(n)$ and $f_i \le 191$, and $n-1$ additions. So $\mathcal{O}(n^2)$ bit operations suffice to do the computations in (27). The computational complexity for the encryption algorithm is given as $\mathcal{O}(n^2)$.

The decrypter first performs a modular multiplication $c_N = \langle w^{-1} c \rangle_N$, and two modular reductions $c_p = \langle c_N \rangle_p$, $c_q = \langle c_N \rangle_q$, which cost $\mathcal{O}(n^2)$ bit operations in total. To solve (11) for each f_i and hence m_i, the decrypter computes $(*)$ to determine (f_{i1}, f_{i2}) with the moduli $\gcd^{\mathbf{u}}_{i-1}/\gcd^{\mathbf{u}}_i$, $\gcd^{\mathbf{v}}_{i-1}/\gcd^{\mathbf{v}}_i \le 47$. So it will take $\mathcal{O}(1)$ bit operations to compute each (f_{i1}, f_{i2}). The look-up operation on Sub-$\mathrm{T}_{\langle i\rangle_{12}}$ for f_i and m_i only costs $\mathcal{O}(1)$ bit operations. So the decrypter only needs $\mathcal{O}(n)$ bit operations to use the *Onion Algorithm* to solve (11) for **x** and **m**. The computational complexity of the decryption algorithm is also $\mathcal{O}(n^2)$.

4.3. Information Rate. The information rate ρ of a cryptosystem is defined as the ratio of the binary length of the message to that of the cipher-text. In the proposed cryptosystem, the information rate is

$$\rho = \frac{3n}{\log_2 \max \mathbb{C}}. \tag{38}$$

We need to evaluate the binary length of $\max \mathbb{C}$. Note that

$$\max \mathbb{C} = \sum_{i=1}^{n} a_i \max \mathbf{F}_i < 191 \sum_{i=1}^{n} a_i < 191 nN. \tag{39}$$

If we rewrite (36) as

$$N \le 191^2 n^2 47^{n-1}, \tag{40}$$

we obtain

$$\max \mathbb{C} \le 191 nN < 191^3 n^3 47^{n-1}. \tag{41}$$

So the information rate is evaluated by

$$\rho \approx \frac{3n}{\log_2\left(191^3 n^3 47^{n-1}\right)}, \tag{42}$$

which is asymptotically $3/\log_2 47 \approx 0.54$.

The public key sizes and information rates for the suggested security parameters are listed in Table 3, from which we see that the public key size is about dozens of a 1024-RSA modulus. The public key size is acceptable.

5. Security Analysis

The adversary has two methods to attack the proposed cryptosystem: solve the cracking problem [26], that is, breaking the one-wayness of the underlying encryption function without knowing the trapdoor, and solve the trapdoor problem, that is, reversing the basic mathematical construction of the trapdoor in a PKC. Efficient algorithms for the latter also help to efficiently solve the former. In this section, we first analyze the hardness of the cracking problem by investigating the underlying PSTOFs under some attack models, and then examine the intractability for key-recovery attacks.

5.1. Brute Force Attack

5.1.1. Brute Force Attack I. One straightforward way to attack the system is to solve (27) for $\mathbf{x}$ by exhausting all the $\sum_{i=1}^{n} a_i f_i$ with $\mathbf{x} \in \mathbb{S}_P$. But note that $|\mathbf{F}_i| = 16$, so the brute force attack will take on the order of $\mathcal{O}(16^n)$ steps. The computational costs under brute force attack I (BFA I for short) given in Table 3 are measured in bit operations with respect to the suggested parameters.

5.1.2. Brute Force Attack II. A better method is to compute and sort each of the sets

$$S_1 = \left\{ \sum_{i=1}^{n/2} a_i f_i \mid f_i \in \mathbf{F}_i \right\},$$
$$S_2 = \left\{ c - \sum_{i=n/2+1}^{n} a_i f_i \mid f_i \in \mathbf{F}_i \right\}, \tag{43}$$

and then scan S_1 and S_2, looking for a common element. If a common element $s = \sum_{i=1}^{n/2} a_i f_i = c - \sum_{i=n/2+1}^{n} a_i f_i$ is found, then $c = \sum_{i=1}^{n} a_i f_i$. The entire procedure takes $\mathcal{O}(n16^{n/2})$ steps [27]. See Table 3 for the computational costs under brute force attack II (BFA II for short) with respect to the suggested parameters.

5.2. Linearization Attack.

In this subsection, we assume that the adversary aims at recovering the corresponding plaintext (and hence the message) by reversing the underlying encryption function without learning the trapdoor information. When a concrete attack model is concerned, we firstly formalize and then prove the underlying PSTOF.

5.2.1. Linearization Attack I. Encryption function (27) of the proposed PKC is a linear function mapping an element in $\mathbb{S}_P$ into an integer in the ciphertext space $\mathbb{C}$. Now we define

$$T_1 : \mathbb{S}_E = \mathbb{Z}_{192}^n \longmapsto \mathbb{C} = T_1\left(\mathbb{S}_E\right),$$
$$T_1(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle = c, \quad \mathbf{x} \in \mathbb{S}_E. \tag{44}$$

Theorem 25. *If the compact knapsack $CKP_{192}(\mathbf{a}, c)$ is intractable, T_1 is a PSTOF.*

Proof. It is easy to verify that the five conditions except the second one in Definition 2 are satisfied; (45) below guarantees the noninjectivity of function T_1. So T_1 is indeed a PSTOF underlying the proposed PKC under the compact knapsack intractability assumption. □

The distinction between the two functions (27) and (44) is that the preimage sets are different. Given a ciphertext c, c will have a unique preimage in $\mathbb{S}_P$, namely, the corresponding plaintext, while c will have many preimages in the equivalent plaintext space $\mathbb{S}_E$.

We have pointed out in Remark 9 that the message encoding function $\mathbf{F}$ can simulate the process of randomly choosing an element from $\mathbb{S}_E = \mathbb{Z}_{192}^n$. That is to say, if one can obtain a preimage of c with respect to T_1, the preimage is a randomly output preimage from all the preimages in $T_1^{-1}(c) = \{\mathbf{x} \in \mathbb{S}_E \mid T(\mathbf{x}) = c\}$ and not necessarily the plaintext $\mathbf{x} \in \mathbb{S}_P$.

Now, we explain why we say we can further weaken the underlying cryptographic hardness assumption. If we assume that the compact knapsack problem $\mathrm{CKP}_{192}(\mathbf{a}, c)$ is easy; that is, T_1 is not one-way, the proposed cryptosystem seems still secure. Under the assumption, there exists an oracle $\mathcal{A}$ for the compact knapsack problem $\mathrm{CKP}_{192}(\mathbf{a}, c)$. We note that each time the adversary can obtain a (at most polynomially many) preimage of c by accessing the oracle $\mathcal{A}$. However, we will show that c has exponentially many (denoted as $2^{\lambda_1(n)}$) preimages, over which the plaintext $\mathbf{x} \in \mathbb{S}_P$ seems uniformly distributed. So we can heuristically argue that the adversary only obtains polynomially many $p(n)$ preimages by performing polynomially many $p(n)$ accesses to the oracle $\mathcal{A}$, and hence those $p(n)$ preimages contain the target plaintext $\mathbf{x} \in \mathbb{S}_P$ with a negligible probability $p(n)/2^{\lambda_1(n)}$. Thus, we can conceive T_1 as a preimage selective trapdoor (not one-way) function assuming that the underlying compact knapsack problem $\mathrm{CKP}_{192}(\mathbf{a}, c)$ is easy. In fact, under the assumption, the security of the proposed PKC is based on the assumption that it is computationally infeasible to recover a solution to a highly noninjective compact knapsack problem $\mathrm{CKP}_{192}(\mathbf{a}, c)$ with a predesignated special structure (i.e., the structure of $\mathbb{S}_P$).

To illustrate, we need to show that a ciphertext has exponentially many preimages with respect to T_1. We use the preimage density given in Definition 4 to roughly evaluate the number of preimages for a ciphertext. To begin with, we need to estimate the cardinalities for the image set (ciphertext space $\mathbb{C}$) and the preimage set (i.e., the equivalent space $\mathbb{S}_E = \mathbb{Z}_{192}^n$). It is obvious that $|\mathbb{S}_E| = 192^n$. We note (41) and $|\mathbb{C}| \leq$

$\max \mathbb{C} + 1 \approx \max \mathbb{C}$, so we have $|\mathbb{C}| \leq 191^3 n^3 47^{n-1}$ and hence the preimage density is

$$P_{d1} = \frac{\mathbb{S}_E}{|\mathbb{C}|} \geq \frac{192^n}{191^3 n^3 47^{n-1}} = 2^{\lambda_1(n)} = 2^{\mathcal{O}(n)}, \quad (45)$$

where $\lambda_1(n) = n\log_2 192 - (n-1)\log_2 47 - 3\log_2 n - 3\log_2 191$. See Table 3 for the manipulations of P_{d1} for the suggested security parameters.

5.2.2. Linearization Attack II. Now, we consider an attack transforming (27) into a standard 0-1 knapsack problem. The adversary can combine all of these 16 values in $\mathbf{F}_i$ denoted as $\mathbf{F}_i = \{f_i^{(1)}, \ldots, f_i^{(16)}\}$ with each entry a_i in $\mathbf{a}$ to derive a new integer sequence $\mathbf{w} = (w_1^{(1)}, \ldots, w_1^{(16)}, \ldots, w_n^{(1)}, \ldots, w_n^{(16)})$ such that $w_i^{(j)} = a_i f_i^{(j)}$ with $i = 1, \ldots, n$, $j = 1, \ldots, 16$. Under the linearization attack model, we define the following function:

$$T_2 : \mathbb{S}_{E2} = \{0,1\}^{16n} \longmapsto \mathbb{C}_2 = T_2(\mathbb{S}_{E2}),$$
$$T_2(\mathbf{y}) = \langle \mathbf{w}, \mathbf{y} \rangle = \sum_{i=1}^{n}\sum_{j=1}^{16} w_i^{(j)} y_i^{(j)} = c, \quad (46)$$

where $\mathbf{y} = (y_1^{(1)}, \ldots, y_1^{(16)}, \ldots, y_n^{(1)}, \ldots, y_n^{(16)}) \in \mathbb{S}_{E2}$. We set

$$\mathbb{S}_{P2} = \left\{ \mathbf{y} \in \{0,1\}^{16n} \mid \sum_{j=1}^{16} y_i^{(j)} = 1, 1 \leq i \leq n \right\}. \quad (47)$$

Theorem 26. *If the knapsack problem KP$(\mathbf{w}, c)$ is intractable, T_2 is a PSTOF.*

Proof. To verify (3) and (4) in Definition 2, we consider a mapping $\tau : \mathbb{S}_P \mapsto \mathbb{S}_{P2}$ as follows. Given an arbitrary $\mathbf{x} = (f_1, \ldots, f_n) \in \mathbb{S}_P$, we must have that for $i = 1, \ldots, n$, there should exist a unique $f_i^{(j)} \in \mathbf{F}_i$ such that $f_i = f_i^{(j)}$ with $1 \leq j \leq 16$. Now we define $\mathbf{y} = (y_1^{(1)}, \ldots, y_1^{(16)}, \ldots, y_n^{(1)}, \ldots, y_n^{(16)}) = \tau(\mathbf{x})$ with $y_i^{(j)} = 1$ when $f_i = f_i^{(j)}$ and $y_i^{(j)} = 0$ otherwise. So we must have for $1 \leq i \leq n$, $\sum_{j=1}^{16} y_i^{(j)} = 1$, and $\sum_{j=1}^{16} f_i^{(j)} y_i^{(j)} = f_i$. So $\mathbf{y} \in \mathbb{S}_{P2}$. Furthermore, any $\mathbf{y} \in \mathbb{S}_{P2}$ must have a preimage $\mathbf{x} = \tau^{-1}(\mathbf{y})$ with $f_i = \sum_{j=1}^{16} f_i^{(j)} y_i^{(j)}$. We further note that $|\mathbb{S}_P| = |\mathbb{S}_{P2}| = 16^n$ and can argue that τ is a bijection from $\mathbb{S}_P$ to $\mathbb{S}_{P2}$. By observing

$$\langle \mathbf{w}, \mathbf{y} \rangle = \sum_{i=1}^{n}\sum_{j=1}^{16} W_i^{(j)} y_i^{(j)} = \sum_{i=1}^{n} a_i \sum_{j=1}^{16} f_i^{(j)} y_i^{(j)}$$
$$= \sum_{i=1}^{n} a_i f_i = \langle \mathbf{a}, \mathbf{x} \rangle, \quad (48)$$

we know that $\mathbf{x} \in \mathbb{S}_P$ is a solution to (27) if and only if $\mathbf{y} = \tau(\mathbf{x})$ is a solution to (46). So (3) and (4) in Definition 2 are satisfied.

To show the noninjectivity of (46), we first note that the image set is $\mathbb{C}_2 = T_2(\mathbb{S}_{E2})$ and that

$$\max \mathbb{C}_2 = \sum_{i=1}^{n}\sum_{j=1}^{16} w_i^{(j)} = \sum_{i=1}^{n} a_i \sum_{j=1}^{16} f_i^{(j)}$$
$$< 191 \times 16 \sum_{i=1}^{n} a_i < 16 \times 191\, nN$$
$$\leq 16 \times 191^3 n^3 47^{n-1}. \quad (49)$$

The last symbol $\leq$ in (49) is due to (40). Hence, we have $|\mathbb{C}_2| < 16 \times 191^3 n^3 47^{n-1} + 1$. So we compute the preimage density via

$$P_{d2} = \frac{\mathbb{S}_{E2}}{|\mathbb{C}_2|} \geq \frac{2^{16n}}{16 \times 191^3 n^3 47^{n-1}} = 2^{\lambda_2(n)} = 2^{\mathcal{O}(n)}, \quad (50)$$

where $\lambda_2(n) = 16n - (n-1)\log_2 47 - 3\log_2 n - 3\log_2 191 - 4$. See Table 3 for the manipulations of P_{d2} for the suggested security parameters. Thus, T_2 is indeed a PSTOF. □

The computational infeasibility for the linearization attack is supported by either of the following arguments.

Remark 27 (noninjectivity-based security argument). If we assume that the adversary can solve the knapsack problem KP$(\mathbf{w}, c)$, then T_2 is a preimage selective trapdoor function. A given ciphertext c will have exponentially many $2^{\lambda_2(n)}$ binary solutions. So a polynomial-time adversary only obtains polynomially many $p(n)$ preimages including the target solution in $\mathbb{S}_{E2}$ with a negligible probability $p(n)/2^{\lambda_2(n)}$.

Remark 28 (density-based security argument). It is known that knapsack problems achieving a density smaller than 0.9408 are solvable with an overwhelming probability by accessing an oracle for the shortest vector problem over lattices [23, 24]. We show that the underlying knapsack problem KP$(\mathbf{w}, c)$ (46) obtain a sufficiently large density. Note that

$$d_2 = \frac{16n}{\log_2 \max \mathbf{w}} \geq \frac{16n}{\log_2 (\max \mathbf{a} \max \mathbf{F}_i)} \geq \frac{16n}{\log_2 (191N)}. \quad (51)$$

Observing (40), we have that d_2 approaches $16/\log_2 47 \approx 2.88$ when n approaches infinity.

Remark 29 (Dimension-based security argument). In real life practice, we use known efficient lattice reduction algorithms to simulate a shortest vector oracle in low-dimensional lattice. However, when the dimension is sufficiently large, say larger than 500, known lattice reduction algorithms fail to output a relatively short lattice vector. Under the suggested parameters, knapsack problem KP$(\mathbf{w}, c)$ (46) achieves sufficiently large dimensions 960, 1440, 1920, respectively.

5.2.3. Linearization Attack III. Now we consider another linear attack III, which also transforms the problem (27) into

a knapsack problem. We note that $0 \le f_i \le 191$, so 8 bits are needed to represent each f_i. For $i = 1, \ldots, n$, we define $d_i^{(j)} = 2^{j-1} a_i$ with $j = 1, \ldots, 8$ and obtain an $8n$-dimensional integer sequence $\mathbf{d} = (d_1^{(1)}, \ldots, d_1^{(8)}, \ldots, d_n^{(1)}, \ldots, d_n^{(8)})$. Now we provide the following function:

$$T_3 : \mathbb{S}_{E3} = \{0,1\}^{8n} \longmapsto \mathbb{C}_3 = T_3\left(\mathbb{S}_{E3}\right),$$
$$T_3(\mathbf{z}) = \langle \mathbf{d}, \mathbf{z} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{8} d_i^{(j)} z_i^{(j)} = c, \tag{52}$$

where $\mathbf{z} = (z_1^{(1)}, \ldots, z_1^{(8)}, \ldots, z_n^{(1)}, \ldots, z_n^{(8)}) \in \mathbb{S}_{E3}$. We set

$$\mathbb{S}_{P3} = \left\{ \mathbf{z} \in \{0,1\}^{8n} \mid \sum_{j=1}^{8} 2^{j-1} z_i^{(j)} \in \mathbf{F}_i, 1 \le i \le n \right\}. \tag{53}$$

Theorem 30. *If the knapsack problem KP*$(\mathbf{d}, c)$ *is intractable, T_3 is a PSTOF.*

Proof. It is easy to verify that (1) and (5) in Definition 2 are satisfied. In fact, any element $\mathbf{z} \in \mathbb{S}_{P3}$ stands for a binary representation of an element $\mathbf{x} \in \mathbb{S}_P$, and in turn, the binary representation of any element $\mathbf{x} \in \mathbb{S}_P$ falls into the set $\mathbb{S}_{P3}$. So if we define a mapping τ from $\mathbb{S}_{P3}$ to $\mathbb{S}_P$, $\tau(z_1^{(1)}, \ldots, z_1^{(8)}, \ldots, z_n^{(1)}, \ldots, z_n^{(8)}) = (\sum_{j=1}^{8} 2^{j-1} z_1^{(j)}, \ldots, \sum_{j=1}^{8} 2^{j-1} z_n^{(j)})$, τ is a bijection. Furthermore, we note that

$$\langle \mathbf{d}, \mathbf{z} \rangle = \sum_{i=1}^{n} a_i \sum_{j=1}^{8} 2^{j-1} z_i^{(j)} = \langle \mathbf{a}, \mathbf{x} \rangle. \tag{54}$$

So $\mathbf{z} = (z_1^{(1)}, \ldots, z_1^{(8)}, \ldots, z_n^{(1)}, \ldots, z_n^{(8)})$ is a solution to (52) if and only if $\tau(\mathbf{z}) = \mathbf{x} \in \mathbb{S}_P$ is a solution to (27). So conditions (3) and (4) in Definition 2 are satisfied. To show the noninjectivity of T_3, we first evaluate the cardinalities of $\mathbb{S}_{E3}$ and $\mathbb{C}_3$. It is obvious that $|\mathbb{S}_{E3}| = 2^{8n}$. To estimate $|\mathbb{C}_3|$, we first note that

$$\begin{aligned} \max \mathbb{C}_3 &= \sum_{i=1}^{n} \sum_{j=1}^{8} d_i^{(j)} = \sum_{i=1}^{n} a_i \sum_{j=1}^{8} 2^{j-1} \\ &= 255 \sum_{i=1}^{n} a_i < 255 \times nN \\ &\le 255 \times 191^2 n^3 47^{n-1}. \end{aligned} \tag{55}$$

So the preimage density of T_3 can be estimated via

$$P_{d3} = \frac{\mathbb{S}_{E3}}{|\mathbb{C}_3|} \ge \frac{2^{8n}}{255 \times 191^2 n^3 47^{n-1}} = 2^{\lambda_3(n)} = 2^{\mathcal{O}(n)}, \tag{56}$$

where $\lambda_3(n) = 8n - (n-1)\log_2 47 - 3\log_2 n - 3\log_2 191 - \log_2 255$. See Table 3 for the values of P_{d3} under the suggested security parameters. Thus, condition (2) in Definition 2 is satisfied, so T_3 is indeed a PSTOF. □

Remark 31 (noninjectivity-based security argument). If one conquers the knapsack problem KP$(\mathbf{d}, c)$, then T_3 is a preimage selective trapdoor function. A ciphertext will have exponentially many $2^{\lambda_3(n)}$ binary solutions. So a polynomial-time adversary only obtains polynomially many $p(n)$ preimages which include the solution in $\mathbb{S}_{E3}$ with a negligible probability $p(n)/2^{\lambda_3(n)}$.

Remark 32 (density and dimension based arguments). The density to (52) is estimated via

$$d_3 = \frac{8n}{\log_2 \max \mathbf{d}} \ge \frac{8n}{\log_2 \left(2^7 \max \mathbf{a}\right)} \ge \frac{8n}{\log_2 \left(2^7 N\right)}, \tag{57}$$

which is $8/\log_2 47 \approx 1.44 > 0.9408$ when n approaches infinity. The knapsack problem KP$(\mathbf{d}, c)$ (52) has sufficiently large dimensions 480, 540, and 960 for the suggested parameters.

5.3. Key-Recovery Attacks. When we discuss the cracking problem, we only study the computational infeasibilities for the adversary to solve (27) without considering the structure of the public integer sequence $\mathbf{a}$. To claim the security of the proposed PKC, we need to examine whether the trapdoor is easy to discover or not. So we need to examine the security of the proposed PKC against known attacks.

The public integer sequence $\mathbf{a}$ distinguishes with a randomly chosen positive integer sequence in two respects, that is, the special structures defined in (14) and (15) and the size conditions given in (24). We discuss the effects of these conditions on the security.

5.3.1. GCD Attack. The GCD attack in [4] was used to successfully attack the knapsack-based probabilistic public key encryption [28]. The attack illustrates that if the entries of $\mathbf{b} = (b_1, \ldots, b_n)$ assume a linear combinatorial relation with small integer combinatorial coefficients, we can recover the secret modulus N by computing the greatest common divisor for a set of numbers. This point will be explored later on, or the readers can refer to [4] for more details. We first note that if the modulus $N = pq$ is known, under the suggested parameters, $n = 60$, 90, and 120, the binary length of N is upper bounded by 355, 523, 690, respectively. See (36). It is not intractable to factor the moduli by using the fastest factorization algorithm [29]. Hence, one can factor N into its prime products. Once p and q have been obtained, the adversary can do the modular reductions, $a_{ip} = a_i (\mathrm{mod}\ p)$ and $a_{iq} = a_i (\mathrm{mod}\ p)$. If we denote the modular reductions of w^{-1} modulo p and q as w_p^{-1} and w_q^{-1}, from (25) and (26), we know $g_i = a_{ip} w_p^{-1} = a_i w^{-1} = b_i (\mathrm{mod}\ p)$ and $h_i = a_{iq} w_q^{-1} = a_i w^{-1} = b_i (\mathrm{mod}\ q)$ and then exhaust the matrix $\mathbf{\Delta}$ to reverse the process of (23) as

$$\begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} = \mathbf{\Delta}^{-1} \begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix} = \begin{pmatrix} \delta_{22} & -\delta_{12} \\ -\delta_{21} & \delta_{11} \end{pmatrix} \begin{pmatrix} \mathbf{g} \\ \mathbf{h} \end{pmatrix}, \tag{58}$$

where for convenience, we just set $|\Delta| = 1$. So we have

$$u_i = \delta_{22}\left\langle a_{ip}w_p^{-1}\right\rangle_p - \delta_{12}\left\langle a_{iq}w_q^{-1}\right\rangle_q,$$
$$v_i = \delta_{11}\left\langle a_{iq}w_q^{-1}\right\rangle_q - \delta_{21}\left\langle a_{ip}w_p^{-1}\right\rangle_p. \tag{59}$$

We note that $\gcd^{\mathbf{u}}_{n-1} \mid u_i$, $\gcd^{\mathbf{v}}_{n-1} \mid v_i$ for $i = 1, \ldots, n-1$, a_{ip} and a_{iq} are publicly computable once p and q have been obtained, and $(\gcd^{\mathbf{u}}_{n-1}, \gcd^{\mathbf{v}}_{n-1}) \in \mathrm{AIP}_{\langle n\rangle_{12}} \cup \mathrm{AIP}^{T}_{\langle n\rangle_{12}}$ and δ_{ij} can be efficiently exhausted (and hence we can assume that they are all known). So, for $i = 1, \ldots, n-1$, we have

$$\delta_{22}a_{ip}x_0 - \delta_{22}x_{2i}p - \delta_{12}a_{iq}x_1 - \delta_{12}x_{2i+1}q - k_i\gcd^{\mathbf{u}}_{n-1} = 0,$$
$$\delta_{11}a_{iq}x_1 - \delta_{11}x_{2i+1}q - \delta_{21}a_{ip}x_0 - \delta_{21}x_{2i}p - l_i\gcd^{\mathbf{v}}_{n-1} = 0 \tag{60}$$

with integer unknowns $0 < x_0 = w_p^{-1} < p$, $0 < x_1 = w_q^{-1} < q$, x_{2i} (x_{2i+1}, resp.) being the incomplete quotient of $a_{ip}x_0$ ($a_{iq}x_1$, resp.) divided by p (q, resp.), and k_i and l_i are the quotients of u_i and v_i divided by $\gcd^{\mathbf{u}}_{n-1}$ and $\gcd^{\mathbf{v}}_{n-1}$, respectively. So we have $0 \le x_{2i} < a_{ip}$, $0 \le x_{2i+1} < a_{iq}$, $0 < k_i < p/\gcd^{\mathbf{u}}_{n-1}$, and $0 < l_i < q/\gcd^{\mathbf{v}}_{n-1}$. We can use an efficient heuristic algorithm for solving a system of linear Diophantine equations with lower and upper bounds on the variables given in [30] (see also Section 3 of [4] for a simplified description) to determine these unknowns, and eventually the full secret keys. So we caution that in implementation, we must keep the modulus N secret. However, we will show that the GCD attack is useless to derive modulus N.

The condition for the GCD attack to succeed in finding modulus N is that we can construct some linear relations using the entries of the public sequence $\mathbf{b}$. To illustrate, we recall (26) and argue that there must exist integers s_i such that $b_i w - s_i N = a_i$. We assume that the entries of $\mathbf{b}$ share some linear combinatorial relations with small coefficients, say $t_1b_1 - t_2b_2 = 0$, $t_3b_3 - t_4b_4 = 0$, where by saying small coefficients we mean we can efficiently do exhaustive search for these t_i's. So we conclude that $N \mid t_1a_1 - t_2a_2 = N(s_1t_1 - s_2t_2)$ and $N \mid t_3a_3 - t_4a_4$ and hence can expect $N = \gcd(t_1a_1 - t_2a_2, t_3a_3 - t_4a_4)$. We must admit that the entries of $\mathbf{u}$ (and $\mathbf{v}$) do share a linear combinatorial relation; for example, from $(\star)$ we derive $\zeta_2u_1 - \alpha_1u_2 = 0$. However, we should bear in mind that $\mathbf{b}$ is fully scrambled by (23) and (25) especially through the confusion role of the Chinese remainder theorem in (25). So it is unlikely for the adversary to derive modulus N by launching a GCD-like attack on the proposal.

5.3.2. Simultaneous Diophantine Approximation Attack. After Shamir broke the basic Merkle-Hellman knapsack cryptosystem [31, 32], cryptanalysts began to investigate the security of the multiply iterated Merkle-Hellman cryptosystem. Lagarias observed that the size conditions and the modular transformation in the multiply iterated Merkle-Hellman cryptosystem help him construct a simultaneous Diophantine approximation problem [7]. Lagarias showed that he can successfully launch a key-recovery attack on the doubly iterated Merkle-Hellman cryptosystem through his simultaneous Diophantine approximation approach.

Many knapsack-type cryptosystems use size conditions to disguise an easy knapsack problem. In such a cryptosystem, the designer randomly generates an easy knapsack problem, $y = \sum_{i=1}^{n} a_ix_i, x_i \in [0, 2^b - 1]$, and chooses a modulus m and a multiplier w, $\gcd(m, w) = 1$. He uses the size condition $m > (2^b - 1)\sum_{i=1}^{n} a_i$ to disguise the easy knapsack sequence $\mathbf{a} = (a_1, \ldots, a_n)$ as a seemingly hard knapsack sequence $\mathbf{b} = (b_1, \ldots, b_n)$ by a modular multiplication, $b_i = \langle wa_i\rangle_m$. The size condition and the public sequence can be utilized to derive a simultaneous Diophantine approximation problem, namely, the problem to find a set of rational numbers sharing a common and relatively small denominator to simultaneously approximate a given set of real numbers. By launching the simultaneous Diophantine approximation attack, the adversary can obtain some useful information about (w, m). See [6, 7] for more details about the relation between the simultaneous Diophantine approximation problem and knapsack public key cryptanalytics.

The trapdoor of the proposed system is disguised using the Chinese remainder theorem, which involves no size conditions. Hence, the adversary cannot expect finding some information about the trapdoor by launching a simultaneous Diophantine approximation attack. However, the reader may doubt that the size condition has been used in (24). We should observe that if the adversary wants to launch a simultaneous Diophantine approximation attack, he must peel off the outmost shuffle in (25) and (26). This seems intractable in that the adversary does not know the primes p and q.

5.3.3. Orthogonal Lattice Attack. To examine the security, we must consider the orthogonal lattice attacks introduced by Nguyen and Stern [8, 9], which were used to successfully though heuristically recover the secret keys of two PKCs [33, 34]. We first illustrate why the orthogonal lattice attack can be used to reconstruct the secret key of the PKC in [34], and then show why our proposal is immune from the orthogonal lattice attack.

In the PKC [34], the security parameter $s = 1024$ is suggested, a modulus $N = pq$ with p and q being primes of lengths, respectively, $|p|_2 = 3s/4$ and $|q|_2 = s/4$ is chosen, and the secret sequence $\mathbf{g} = (g_1, \ldots, g_n)$ (typically $n = 180$) with each entry $g_i < p^{1/l}$ (typically $l = 17$) is generated so that the public sequence $\mathbf{a} = (a_1, \ldots, a_n)$ can be computed via a modular multiplication $a_i = wg_i \pmod N$ for a randomly chosen $w \in \mathbb{Z}_N^*$. The core for the orthogonal lattice attack to recover the secret key of [34] is to choose $0 < m \le n$ entries $\mathbf{a}_m = (a_1, \ldots, a_m)$ from $\mathbf{a}$ and then obtain sufficiently many small and linearly independent solutions $\mathbf{x}_m = (x_1, \ldots, x_m)$ to $\langle \mathbf{a}_m, \mathbf{x}_m\rangle = 0$. Given a solution $\mathbf{x}_m$, we must have $\langle \mathbf{a}_m, \mathbf{x}_m\rangle = \langle w\mathbf{g}_m, \mathbf{x}_m\rangle = w\langle \mathbf{g}_m, \mathbf{x}_m\rangle = 0 \pmod p$, where $\mathbf{g}_m = (g_1, \ldots, g_m)$. So $\langle \mathbf{g}_m, \mathbf{x}_m\rangle = 0 \pmod p$, which means that $\mathbf{x}_m$ is orthogonal to $\mathbf{g}_m$, or $\langle \mathbf{g}_m, \mathbf{x}_m\rangle = kp$ with $k \neq 0$. In the second case, from Cauchy-Schwarz inequality, we have

$$\|\mathbf{x}_m\| \ge \frac{|\langle \mathbf{g}_m, \mathbf{x}_m\rangle|}{\|\mathbf{g}_m\|} \ge \frac{p}{\sqrt{m}p^{1/l}} \approx \frac{N^{12/17}}{\sqrt{m}}, \tag{61}$$

when concrete parameters are concerned, which implies that either $\mathbf{x}_m$ is orthogonal to $\mathbf{g}_m$ or the norm of $\mathbf{x}_m$ is quite large. From the observations, the authors of [9] conjectured that among the reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{m-2}, \mathbf{b}_{m-1})$ of the $(m-1)$-dimensional lattice consisting of the integer solutions to $\langle \mathbf{a}_m, \mathbf{x}_m \rangle = 0$, the first $m-2$ basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{m-2}$ are orthogonal to $\mathbf{g}_m$, namely, Assumption 4 of [9]. By noting that $\mathbf{g}_m$ belongs to the two-dimensional orthogonal lattice $\mathscr{L}^{\perp}$ consisting of all m-dimensional integer vectors orthogonal to the lattice vectors in $\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{m-2})$ and that $\|\mathbf{g}_m\| < \sqrt{m}p^{1/l} = \sqrt{m}p^{1/17}$ is a very small value, they provide another assumption conjecturing that $\mathbf{g}_m$ is a shortest vector in $\mathscr{L}^{\perp}$, namely, Assumption 5 of [9]. So given $\mathbf{a}_m$, we can expect to find the reduced basis $\mathbf{b}_1, \ldots, \mathbf{b}_{m-2}$ and the shortest vector $\mathbf{g}_m$ by using lattice reduction algorithms.

In the proposed cryptosystem, we generate p according to (24), so we can assume that $p > 191ng_i$. Equation (61) turns out to be

$$\|\mathbf{x}_m\| \geq \frac{|\langle \mathbf{g}_m, \mathbf{x}_m \rangle|}{\|\mathbf{g}_m\|} \geq \frac{p}{p/(191n)\sqrt{m}} = \frac{191n}{\sqrt{m}}. \tag{62}$$

Under the suggested parameters $n = 60, 90, 120, 191n$ is at most 15 bits long, so the assumption will be too strong to be satisfied that the first $m-2$ basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{m-2}$ have norms less than $191n/\sqrt{m}$ and hence are orthogonal to $\mathbf{g}_m$. Furthermore, in [34], we can argue that $\|\mathbf{g}_m\| < \sqrt{m}p^{1/l} = \sqrt{m}p^{1/17}$ is a very small value; however, we just obtain $\|\mathbf{g}_m\| < \sqrt{m}p/(191n)$ in the proposed cryptosystem and hence cannot assume that $\mathbf{g}_m$ is a shortest vector in $\mathscr{L}^{\perp}$.

To summarize, we can conjecture that the requirement of $g_i^l < p$ in [34] makes both to find $m-2$ linearly independent short vectors $\mathbf{b}_1, \ldots, \mathbf{b}_{m-2}$ and to accept the shortest vector in $\mathscr{L}^{\perp}$ as the target vector $\mathbf{g}_m$ possible, while in the proposed cryptosystem $p > 191ng_i$ is only somewhat larger than those g_i, from which we cannot formalize Assumptions 4 and 5 provided in [9]. Hence, the proposed cryptosystem is invulnerable to the orthogonal lattice attack.

Remark 33. Now we explain why we introduce the matrix $\boldsymbol{\Delta}$ when generating the public keys. If $\boldsymbol{\Delta}$ is the two-dimensional identity matrix $\mathbf{I}_2$, $\mathbf{g} = \mathbf{u}$, we can distil a relatively large common divisor $t = \gcd_m^{\mathbf{g}} = \gcd_m^{\mathbf{u}} = \prod_{i=m}^{n}\alpha_i$ (see $(\star)$) from the preceding m entries of $\mathbf{g}$. Thus, if we define $\mathbf{g}'_m = (g'_1, \ldots, g'_m)$ with $g'_i = g_i/t$ being sharply reduced, then we can recover $\mathbf{g}'_m$ by the orthogonal lattice attack and eventually $\mathbf{g}_m$ by firstly exhausting the value of $t = \prod_{i=m}^{n}\alpha_i$ and then combining g'_i and t to derive $g_i = tg'_i$. However, after the transformation of (23), we cannot expect to extract a large greatest common divisor for the entries of $\mathbf{g}_m$. We suggest choosing $\boldsymbol{\Delta}$ with determinant being $\delta_{11}\delta_{22} - \delta_{21}\delta_{12} = \pm 1$ in Remark 22, in which case we can make sure that $\gcd(\delta_{11}, \delta_{12}) = \gcd(\delta_{21}, \delta_{22}) = 1$. If the two gcd's are greater than 1, say $\delta = \gcd(\delta_{11}, \delta_{12}) > 1$, then $\delta \mid g_i = \delta_{11}u_i + \delta_{12}v_i$ and hence $t = \gcd_m^{\mathbf{g}} \geq \delta$, which may compromise the security.

6. Conclusions

In this paper, we defined a public key cryptographic scenario PSTOF and used the knapsack problem to exemplify how to further weaken the cryptographic assumptions used in public key cryptography. Extensive security scrutiny is made on the proposed probabilistic encryption scheme; however, some security drawbacks may still exist in the proposal. If some cryptanalysts announced that the proposed cryptosystem is breakable, we would not be surprised in that some cleverer and efficient cryptanalytic method may be developed. But we are confident that the proposal can be broken only by successfully launching a key-recovery attack. We think that if someone invents a PSTOF with a hard-to-discover trapdoor by fully exploring the noninjectivity property of the underlying one-way function, it can be used to provide a higher level of security. We hope that some more elegant methods can be developed in the cryptographic literature to provide more practical and secure realizations of the concept of PSTOF. Some more cryptanalytic discussions are needed to obtain all-sided and in-depth security analysis of the proposal, which belongs to our future work.

Notations

$\mathbb{Z}$:	Ring of integers
$\mathbb{Z}^+$:	Set of positive integers
$\mathbb{Z}_N$:	The least nonnegative complete residue system modulo N, $\{0, 1, \ldots, N-1\}$
$\phi(N)$:	Euler's totient function
$\mathbb{Z}_N^*$:	Set of invertible elements in $\mathbb{Z}_N$
$\lvert\mathbf{A}\rvert$:	The determinant when $\mathbf{A}$ denotes a square matrix and the cardinality when $\mathbf{A}$ represents a set
$\max \mathbf{a}$:	The maximum value in a set $\mathbf{a}$
$\min \mathbf{a}$:	The minimum value in a set $\mathbf{a}$
$\langle a \rangle_p$, $a \pmod p$:	The least nonnegative residue of a modulo p
$f_i \pmod{(a,b)}$:	$(f_i \pmod a, f_i \pmod b)$
$\mathbf{F}_i \pmod{(a,b)}$:	$\{f_i \pmod{(a,b)} \mid f_i \in \mathbf{F}_i\}$
$\gcd_i^{\mathbf{a}}$:	$\gcd(a_1, \ldots, a_i)$, the greatest common divisor of the preceding i components of the integer sequence $\mathbf{a} = (a_1, \cdots, a_n)$
$Hw(\mathbf{x})$:	$\sum_{i=1}^{n} x_i$, the Hamming weight of a binary vector $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$
$a^{-1} \pmod p$:	The modular inverse of a modulo p
$a \mid b$:	a divides b
$(a,b)^T$:	(b,a)
$\mathbf{A}^T$:	$\{(a,b)^T \mid (a,b) \in \mathbf{A} \subset (\mathbb{Z}^+)^2\}$
$\lvert a\rvert_2$:	The binary length of a positive integer a
$\mathbf{A}^{-1}$:	The inverse of an invertible square matrix $\mathbf{A}$
$\langle \mathbf{u}, \mathbf{v} \rangle$:	The inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$
$\|\mathbf{u}\|$:	The Euclidean norm of a vector $\mathbf{u}$.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61173152), the 111 Project (no. B08038), the ISN Foundation (no. ISN1103007), the Fundamental Research Funds for the Central Universities (no. JY10000901009), and the Natural Science Basic Research Plan in Shaanxi Province of China (Program no. 2012JM8005).

References

[1] M. Bellare, S. Halevi, A. Sahai, and S. Vadhan, "Many-to-one trapdoor functions and their relation to public-key cryptosystems," in *Advances in Cryptology—Crypto 1998*, vol. 1462 of *Lecture Notes in Computer Science*, pp. 283–298, Springer, Santa Barbara, Calif, USA, 1998.

[2] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[3] T. Q. Khoat, "Relation between the hardness of a problem and the number of its solutions," *Acta Mathematica Vietnamica*, vol. 36, no. 1, pp. 55–60, 2011.

[4] A. M. Youssef, "Cryptanalysis of a knapsack-based probabilistic encryption scheme," *Information Sciences*, vol. 179, no. 18, pp. 3116–3121, 2009.

[5] E. F. Brickell and A. M. Odlyzko, "Cryptanalysis: a survey of recent results," in *Contemporary Cryptology*, The Science of Information Integrity, pp. 501–540, IEEE Press, New York, NY, USA, 1992.

[6] J. C. Lagarias, "The computational complexity of simultaneous Diophantine approximation problems," *SIAM Journal on Computing*, vol. 14, no. 1, pp. 196–209, 1985.

[7] J. C. Lagarias, "Knapsack public key cryptosystems and diophantine approximation," in *Advances in Cryptology—Crypto 1983*, pp. 3–23, Plenum, New York, NY, USA, 1984.

[8] P. Nguyen and J. Stern, "Merkle-Hellman revisited: a cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations," in *Advances in cryptology—CRYPTO 1997*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 198–212, Springer, Santa Barbara, Calif, USA, 1997.

[9] P. Nguyen and J. Stern, "Cryptanalysis of a fast public key cryptosystem presented at SAC '97," in *Selected Areas in Cryptography*, vol. 1556 of *Lecture Notes in Computer Science*, pp. 213–218, Springer, Ontario, Canada, 1998.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the Association for Computing Machinery*, vol. 21, no. 2, pp. 120–126, 1978.

[11] B. Chor and R. L. Rivest, "A knapsack-type public key cryptosystem based on arithmetic in finite fields," *IEEE Transactions on Information Theory*, vol. 34, no. 5, part 1, pp. 901–909, 1988.

[12] T. Okamoto, K. Tanaka, and S. Uchiyama, "Quantum public-key cryptosystems," in *Advances in Cryptology—CRYPTO 2000*, vol. 1880 of *Lecture Notes in Computer Science*, pp. 147–165, Springer, Santa Barbara, Calif, USA, 2000.

[13] T. M. Cover, "Enumerative source encoding," *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 73–77, 1973.

[14] P. Q. Nguyen and J. Stern, "Adapting density attacks to low-weight knapsacks," in *Advances in cryptology—ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 41–58, Springer, Chennai, India, 2005.

[15] S. Vaudenay, "Cryptanalysis of the Chor-Rivest cryptosystem," *Journal of Cryptology*, vol. 14, no. 2, pp. 87–100, 2001.

[16] L. H. Encinas, J. M. Masqué, and A. Q. Dios, "Safer parameters for the Chor-Rivest cryptosystem," *Computers & Mathematics with Applications*, vol. 56, no. 11, pp. 2883–2886, 2008.

[17] L. H. Encinas, J. M. Masqué, and A. Q. Dios, "Analysis of the efficiency of the Chor-Rivest cryptosystem implementation in a safe-parameter range," *Information Sciences*, vol. 179, no. 24, pp. 4219–4226, 2009.

[18] A. Kate and I. Goldberg, "Generalizing cryptosystems based on the subset sum problem," *International Journal of Information Security*, vol. 10, no. 3, pp. 189–199, 2011.

[19] K. Omura and K. Tanaka, "Density attack to the Knapsack cryptosystems with enumerative source encoding," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. 84, no. 1, pp. 1564–1569, 2001.

[20] T. Izu, J. Kogure, T. Koshiba, and T. Shimoyama, "Low-density attack revisited," *Designs, Codes and Cryptography*, vol. 43, no. 1, pp. 47–59, 2007.

[21] N. Kunihiro, "New definition of density on knapsack cryptosystems," in *Progress in Cryptology—AFRICACRYPT 2008*, vol. 5023, pp. 156–173, Springer, Berlin, Germany, 2008.

[22] M. Rabin, "Digital signatures and public-key encryptions as intractable as factorization," MIT Technical Report 212, 1979.

[23] M. J. Coster, B. A. LaMacchia, A. M. Odlyzko, and C.-P. Schnorr, "An improved low-density subset sum algorithm," in *Advances in Cryptology—Eurocrypt 1991*, vol. 547 of *Lecture Notes in Computer Science*, pp. 54–67, Springer, Brighton, UK, 1991.

[24] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *Computational Complexity*, vol. 2, no. 2, pp. 111–128, 1992.

[25] M. K. Lee and K. Park, "Low-density attack of public-key cryptosystems based on compact knapsacks," *Journal of Electrical Engineering and Information Science*, vol. 4, no. 2, Article ID 197204, 1999.

[26] N. Koblitz, *Algebraic Aspects of Cryptography*, vol. 3, Springer, Berlin, Germany, 1998.

[27] A. M. Odlyzko, "The rise and fall of knapsack cryptosystems," in *Cryptology and Computational Number Theory*, vol. 42 of *Proceedings of Symposia in Applied Mathematics*, pp. 75–88, 1990.

[28] B. Wang, Q. Wu, and Y. Hu, "A knapsack-based probabilistic encryption scheme," *Information Sciences*, vol. 177, no. 19, pp. 3981–3994, 2007.

[29] T. Kleinjung, K. Aoki, J. Franke et al., "Factorization of a 768-bit RSA modulus," http://eprint.iacr.org/2010/006 .

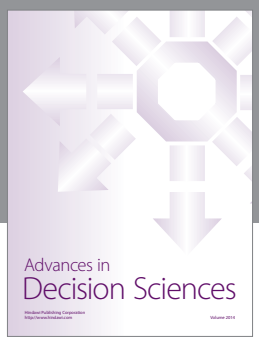
[30] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra, "Solving a system of linear Diophantine equations with lower and upper bounds on the variables," *Mathematics of Operations Research*, vol. 25, no. 3, pp. 427–442, 2000.

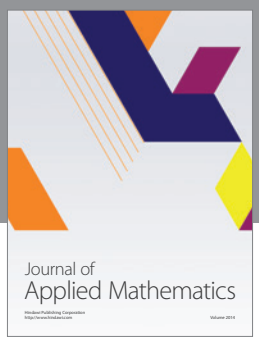
[31] R. C. Merkle and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525–530, 1978.

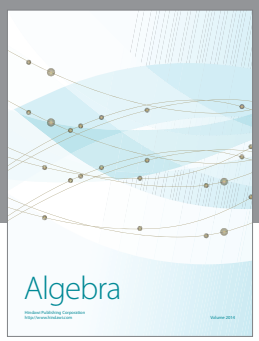
[32] A. Shamir, "A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 699–704, 1984.

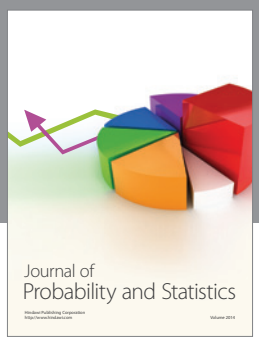
[33] M. H. Qu and S. A. Vanstone, "The knapsack problem in cryptography," in *Finite Fields: Theory, Applications, and Algorithms*, vol. 168 of *Contemporary Mathematics*, pp. 291–308, 1994.

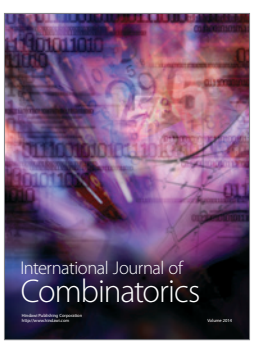
[34] K. Itoh, E. Okamoto, and M. Mambo, "Proposal of a fast public key cryptosystem," in *Proceedings of the Selected Areas in Cryptography (SAC '97)*, Ottawa, Canada, 1997.

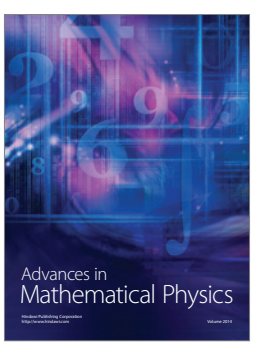


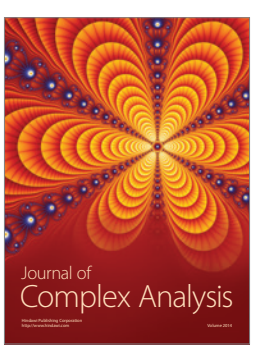


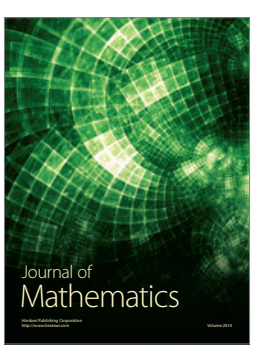


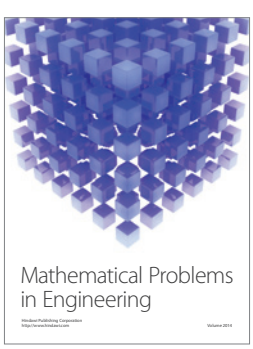


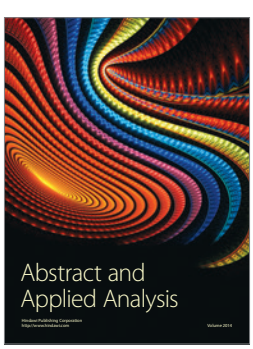


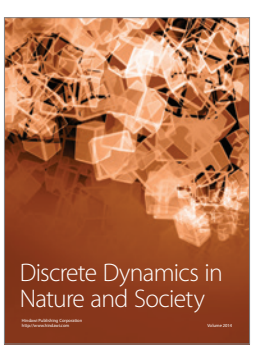


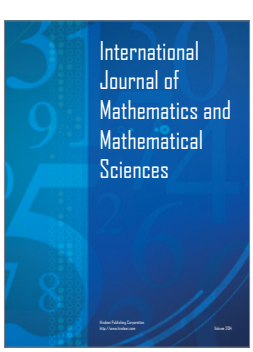












Hindawi

Submit your manuscripts at
http://www.hindawi.com