Generalized Powering Functions and their Application to Digital Signatures

Hisayoshi Sato[1], Tsuyoshi Takagi[2], Satoru Tezuka[1], and Kazuo Takaragi[1]

[1] Hitachi, Ltd., Systems Development Laboratory,
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
{hisato,tezuka,takara}@sdl.hitachi.co.jp
[2] Technische Universität Darmstadt, Fachbereich Informatik,
Alexanderstr.10, D-64283 Darmstadt, Germany
takagi@informatik.tu-darmstadt.de

Abstract. This paper investigates some modular powering functions suitable for cryptography. It is well known that the Rabin encryption function is a 4-to-1 mapping and breaking its one-wayness is secure under the factoring assumption. The previously reported encryption schemes using a powering function are variants of either the 4-to-1 mapping or higher n-to-1 mapping, where $n > 4$. In this paper, we propose an optimized powering function that is a 3-to-1 mapping using a p^2q-type modulus. The one-wayness of the proposed powering function is as hard as the infeasibility of the factoring problem. We present an efficient algorithm for computing the decryption for a p^2q-type modulus, which requires neither modular inversion nor division. Moreover, we construct new provably secure digital signatures as an application of the optimized functions. In order to achieve provable security in the random oracle model, we usually randomize a message using random hashing or padding. However, we have to compute the randomization again if the randomized message is a non-cubic residue element — it is inefficient for long messages. We propose an algorithm that can deterministically find the unique cubic residue element for a randomly chosen element.

Keywords: factoring, RSA, modular powering function, digital signature.

1 Introduction

Modular powering functions with composite moduli play an important role in cryptography. The RSA cryptosystem [15] and its variations [2, 3] use one-to-one modular powering functions (permutations) as primitives. The Rabin cryptosystem [14] and its variants such as Williams' scheme [19] and Kurosawa et al.'s schemes [8, 9] are composed of modular squaring functions (4-to-1 mapping). The other encryption schemes using powering functions [16],[10],[20],[21] utilize n-to-1 mappings ($n \geq 4$). Although the types of moduli for these functions are various, the following types are mainly used: pq, pqr and p^2q (*e.g.* [5],[7],[13],[18]), where p, q, r are distinct prime numbers. The pqr-type modulus can efficiently compute its decryption using the Chinese remainder theorem [13], and the p^2q-type modulus can achieve faster decryption through the addition of Hensel lifting [7],[18].

These various kinds of functions have advantages and disadvantages. In cryptographic use, we expect that these functions will be proven to be one-way under some reliable assumptions such as the infeasibility of factoring large composite numbers. In view of this, strictly speaking, computational equivalence between one-wayness and the infeasibility of factoring is not proven for RSA functions. On the other hand, it is proven that Rabin functions are one-way under the factoring assumptions. However, for pq and p^2q-type moduli, these functions are 4-to-1, where four is the cardinality of the kernel (for pqr-type, the functions are 8-to-1), and this causes some inconvenience in cryptography such as non-uniqueness in decryption. In avoiding this, additional treatment is required for decryption or efficiency is decreased, and thus a smaller kernel would better suit for our purposes. Moreover, for a p^2q-type modulus, the conventional methods ([7],[18]) require modular inverses and integer divisions to be calculated. Even though these operations are fast in software, they are relatively expensive in hardware, especially for smartcards that are not equipped with coprocessors to calculate these inverses and divisions.

In this paper, we investigate optimized modular powering functions whose one-wayness can be proven secure under factoring assumptions. We deal with the general powering function $f(x) = x^e \bmod n$ for modulus $n = p^d q$. We show some criteria related to parameters g, d, p, q, which determine the number of pre-images of $f(x)$. We conclude that the optimal encryption of our proposed scheme is a 3-to-1 mapping with the p^2q-type modulus. Moreover, we propose an efficient algorithm to calculate the preimages of a p^2q-type modulus, which needs neither modular inversion nor division of integers. Moreover, as an application of these optimized functions, we construct new provably secure digital signatures using these functions in the random oracle model. In order to achieve the security in the random oracle model, we randomize a message using a random hashing or padding. If the randomized message is a non-cubic element, we have to randomize it again before the primitive computation — it is inefficient for long messages. In this paper, we propose an algorithm, with which the three possible kernel elements can easily be distinguished by a non-cubic residue element. This trick was initially proposed by Kurosawa et al. for the Rabin signature scheme [9]. Finally, we estimate the efficiency of the proposed signature scheme in contrast with other conventional signature schemes. The decryption of the proposed scheme with a p^2q-type modulus is about 1.7 time faster than that of the Multi-Prime RSA with a pqr-type modulus of the same size.

This paper is organized as follows: In Section 2, we discuss the proposed primitives and propose an optimal powering function. Section 3 discuss our construction of digital signature schemes based on the optimal powering function. Section 4 concludes the paper with a few closing remarks.

2 Proposed Primitives

In this section, we first study generalized powering functions with respect to security, efficiency and convenience for cryptography, and we then propose a new type of function that has not been applied to cryptography, that is, the conditions for prime factors of the modulus are asymmetric. In the following, because of efficiency, we concentrate on moduli of the powering functions that have two distinct prime factors (*cf.* Section 3.4).

2.1 Generalized Powering Functions

Let us recall the general properties of powering functions over some finite rings.

We deal with a modulus $N = p^d q$, where p and q are distinct odd prime numbers and $d \geq 1$ a positive integer[1]. Let us denote the g-th power map on the multiplicative group $\mathbb{Z}_N^*$ by

$$f = f_{N,g} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*, \qquad f(x) = x^g \bmod N. \tag{1}$$

Note that f is a $g_p g_q$-to-one map, and the image of f is equal to $(\mathbb{Z}_N^*)^g = \left(\mathbb{Z}_{p^d}^*\right)^{g_p} \times \left(\mathbb{Z}_q^*\right)^{g_q}$, where $g_p = \gcd(g, p-1)$ and $g_q = \gcd(g, q-1)$ for an integer $1 < g < \min(p-1, q-1)$. Let y be an element in the image of f, then the preimage of y by f is the set given by $f^{-1}(y) = \{x' \in \mathbb{Z}_N^* \mid x'^g = y\}$, which consists of $g_p g_q$ elements. We choose $g = 2$ for the Rabin cryptosystem, so that there are four ambiguities for the preimage of map f due to $g_p = g_q = 2$.

We denote the isomorphism by the Chinese remainder theorem by ϕ:

$$\phi = \phi_{p^d,q} : \mathbb{Z}_{p^d} \times \mathbb{Z}_q \xrightarrow{\sim} \mathbb{Z}_N.$$

As is well known, the multiplicative group $\mathbb{Z}_p^*$ is a cyclic group of order $p-1$. For an integer $t \geq 1$, let $Z_{p,t}$ be the subgroup of elements in $\mathbb{Z}_p^*$ whose order divides t: $Z_{p,t} = \{a \in \mathbb{Z}_p^* \mid a^t = 1\}$.

We consider the g-th power map $f_{p,g}$ on $\mathbb{Z}_p^*$. It can easily be seen that the following sequence is exact[2]: $1 \to Z_{p,g} \hookrightarrow \mathbb{Z}_p^* \xrightarrow{f_{p,g}} \left(\mathbb{Z}_p^*\right)^g \to 1$. Moreover, we have $\left(\mathbb{Z}_p^*\right)^g = \left(\left(\mathbb{Z}_p^*\right)^{g_p}\right)^{g/g_p}$, and the order of $\left(\mathbb{Z}_p^*\right)^{g_p}$ is $(p-1)/g_p$, in particular, it is prime to g/g_p, thus it holds that $\left(\left(\mathbb{Z}_p^*\right)^{g_p}\right)^{g/g_p} = \left(\mathbb{Z}_p^*\right)^{g_p}$. Hence, we have $\#Z_{p,g} = (p-1)/\#\left(\mathbb{Z}_p^*\right)^{g_p} = g_p$, and from the uniqueness of the subgroup of order g_p in $\mathbb{Z}_p^*$, letting $\zeta_{p,g}$ be a primitive g_p-th root of unity, we have $Z_{p,g} = Z_{p,g_p} = \langle \zeta_{p,g} \rangle$, that is, the subgroup of g-th roots of unity is equal to that of g_p-th roots of unity. Let $\chi_{p,g} = f_{p,\frac{p-1}{g_p}}$ be the $(p-1)/g_p$-th power map on $\mathbb{Z}_p^*$:

$$\chi_{p,g} : \mathbb{Z}_p^* \to \mathbb{Z}_p^*, \quad \chi_{p,g}(x) = x^{\frac{p-1}{g_p}} \bmod p,$$

[1] Boneh et al. proposed a polynomial time algorithm for factoring $p^d q$ for large d [4]. The exponent d in this paper is very small, so that their algorithm is not effective.

[2] $A \xrightarrow{f} B \xrightarrow{g} C$ is said to be exact if the image of f is equal to the kernel of g.

then, due to the above arguments, the following sequence is exact: $1 \to \left(\mathbb{Z}_p^*\right)^g = \left(\mathbb{Z}_p^*\right)^{g_p} \hookrightarrow \mathbb{Z}_p^* \xrightarrow{\chi_{p,g}} \langle\zeta_{p,g}\rangle \to 1$. In other words, we have the following.

Lemma 1. *For any $x \in \mathbb{Z}_p^*$, we have $\chi_{p,g}(x) \in \langle\zeta_{p,g}\rangle$, and x is a g-th power residue (i.e. $x \in \left(\mathbb{Z}_p^*\right)^g$) if and only if $\chi_{p,g}(x) = 1$.*

For $\left(\mathbb{Z}_{p^d}\right)^*$, there exists a decomposition $\left(\mathbb{Z}_{p^d}\right)^* \cong \left(\mathbb{Z}_p\right)^* \times \mathbb{Z}_{p^{d-1}}$. Since $\gcd(g, p) = 1$, regarding $\mathbb{Z}_p^*$ as a subgroup of $\mathbb{Z}_{p^d}^*$, we have $Z_{p^d,g} = Z_{p,g}$, and $a \in \mathbb{Z}_{p^d}^*$ is a g-th power residue if and only if $a \bmod p$ satisfies the condition in Lemma 1. For the prime q, the situation is similar, let $Z_{q,g} = \langle\zeta_{q,g}\rangle$, then by the Chinese remainder theorem, the set $Z_{N,g}$ of all g-th roots of unity in $\mathbb{Z}_N^*$ can be written as $Z_{N,g} = \phi(Z_{p,g}, Z_{q,g}) = \left\{\phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \mid 0 \le i < g_p,\ 0 \le j < g_q\right\}$. Since $\zeta_{p,g}$ and $\zeta_{q,g}$ are easily calculated (*e.g.* $\zeta_{p,g} = x^{(p-1)/g_p}$ for some x), and by Lemma 2, $Z_{N,g}$ is easily obtained. Especially, we have $\#Z_{N,g} = g_p \cdot g_q$. Putting $g_{p,q} = \mathrm{lcm}(g_p, g_q)$, it can easily be seen that $Z_{N,g} = Z_{N,g_{p,q}}$.

Consequently, for $y \in (\mathbb{Z}_N^*)^g$, let $x \in f^{-1}(y)$ be a preimage of $y : x^g = y$, then the preimage $f^{-1}(y)$ of y by f can easily be calculated by the following:

$$f^{-1}(y) = \left\{x \cdot \phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \mid 0 \le i < g_p,\ 0 \le j < g_q\right\}, \tag{2}$$

2.2 Security of Generalized Powering Functions

We will now consider computational equivalence between the factoring problem on N and the one-wayness of function f, when f is not injective.

For each divisor e of g, define the set of primes which satisfy the conditions in Lemma 1 as follows: $\mathcal{P}_e = \{p : \text{prime} \mid \gcd(g, p-1) = e,\ \gcd(g, (p-1)/e) = 1\}$.

Let $\mathrm{Div}(g)$ be the set of all divisors of g. For a non-empty set $\mathcal{D} \subset \mathrm{Div}(g)$, we put $\mathcal{P}_\mathcal{D} = \bigcup_{e \in \mathcal{D}} \mathcal{P}_e$. We fix integers $d \ge 1$, $g \ge 2$, and non-empty sets $\mathcal{D}_1, \mathcal{D}_2 \subset \mathrm{Div}(g)$, $\mathcal{D}_1 \cup \mathcal{D}_2 \ne \{1\}$ (namely, one of these contains divisors of g besides 1). For these, let a instance generator $\mathcal{G}_0$ be a probabilistic polynomial time algorithm such that $\mathcal{G}_0(1^k) \to N$, where $N = p^d q$, $|N| = k$, $p \in \mathcal{P}_{\mathcal{D}_1}$, $q \in \mathcal{P}_{\mathcal{D}_2}$, $|p| \approx |q|$ (In the case $1 \in \mathcal{D}_1 \cap \mathcal{D}_2$, we also assume that $(p, q) \notin \mathcal{P}_1 \times \mathcal{P}_1$). Using these notations, we define the factoring problem and its infeasibility.

Definition 1. *The integer factoring problem is a problem which for given d, g, $\mathcal{D}_1$, $\mathcal{D}_2$, and $N \xleftarrow{R} \mathcal{G}_0(1^k)$, finds the factors (p, q) of N. The integer factoring problem is said to be infeasible if for any probabilistic polynomial time (PPT) algorithm A, any constant c and all sufficiently large k,*

$$\Pr\left[A(1^k, N, d, g, \mathcal{D}_1, \mathcal{D}_2) = (p, q) \;\middle|\; N \xleftarrow{R} \mathcal{G}_0(1^k)\right] < \frac{1}{k^c}.$$

Definition 2. *A integer factoring PPT algorithm A is said to (t, ϵ)-break $N \xleftarrow{R} \mathcal{G}_0(1^k)$ if for any $k \in \mathbb{N}$, after at most $t(k)$ processing time, it factors N with probability at least $\epsilon(k)$. The set of outputs of $\mathcal{G}_0$ is (t, ϵ)-secure if there exists no integer factoring PPT algorithm which (t, ϵ)-breaks.*

We next define the one-wayness for the functions defined in Section 2.1 as follows:

Definition 3. *The notations d, g, $\mathcal{D}_1$, $\mathcal{D}_2$ and $\mathcal{G}_0$ are the same as in Definition 1. The powering function f is said to be one-way if for any PPT algorithm A, for any constant c and any sufficiently large k,*

$$\mathrm{Adv}(A) = \Pr\left[A(1^k, N, d, g, \mathcal{D}_1, \mathcal{D}_2, y) = x' \in f^{-1}(y) \;\middle|\; \begin{array}{l} N \xleftarrow{R} \mathcal{G}_0(1^k); \\ x \xleftarrow{R} \mathbb{Z}_N^*; \\ y \xleftarrow{R} f(x) \end{array} \right] < \frac{1}{k^c}.$$

Definition 4. *A PPT algorithm A is said to (t, ϵ)-break f if for any $k \in \mathbb{N}$, after at most $t(k)$ processing time, it calculates a preimage of f with probability at least $\epsilon(k)$. f is (t, ϵ)-secure if there exists no PPT algorithm which (t, ϵ)-breaks.*

Let φ be the Euler totient function. Under these definitions, we can prove the following theorem, whose proof can be found in Appendix A.

Theorem 1. *Fix integers $d \geq 1$ and $g \geq 2$, and assume that all divisors of g can be efficiently computed. Fix non-empty sets $\mathcal{D}_1, \mathcal{D}_2 \subset \mathrm{Div}(g)$, $\mathcal{D}_1 \cup \mathcal{D}_2 \neq \{1\}$, and for any $e_1 \in \mathcal{D}_1$ and $e_2 \in \mathcal{D}_2$, assume that $\{\sum_{e \mid \gcd(e_1, e_2)} \varphi(e)^2\}/e_1 e_2$ is small. Moreover, we put*

$$\tau = \min_{e_1 \in \mathcal{D}_1, e_2 \in \mathcal{D}_2} \left\{ 1 - \frac{\sum_{e \mid \gcd(e_1, e_2)} \varphi(e)^2}{e_1 e_2} \right\}.$$

(Notice that by the assumptions, τ is close to 1) Let $\mathcal{G}_0$ be the instance generator for the above parameters, and $N \xleftarrow{R} \mathcal{G}_0(1^k)$. If the integer factoring problem for N is infeasible, then the function $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, $f(x) = x^g$ is one-way. More precisely, if the outputs of $\mathcal{G}_0$ are (t_I, ϵ_I)-secure, then f is (t_f, ϵ_f)-secure, where

$$t_I(k) = t_f(k) + O(k^3), \quad \epsilon_I(k) = \tau \epsilon_f(k).$$

According to the argument in Section 2.3, the inverse of f can be calculated using the factors of N, hence, together with the argument in this section, it is proven that the equivalence between the infeasibility of the factoring problem on N and the one-wayness of f.

2.3 Efficient Decryption Algorithms

In this section, we consider an efficient algorithm that can be used to calculate the preimage of the powering function considered in Section 2.1, (1) when the prime factors p and q are known. In the case $d > 1$, the conventional method (*e.g.* [7][18]) needs the modular inverse to be calculated. We now propose an algorithm that does not need the modular inverse to be calculated under some conditions on p, q and g. The proofs for the following are in Appendix B. The notations p, q, d, N and g are the same as in Section 2.1. Let $z = p^{-1} \bmod q$. For $y \in (\mathbb{Z}_N^*)^g$, we put $y_p = y \bmod p$, $y_q = y \bmod q$ and $y_i = y \bmod p^i q$ $(1 \leq i \leq d)$. Moreover, let x_p be a g-th root of y_p in $\mathbb{Z}_p^*$, that is $x_p^g = y_p (\bmod p)$. Similarly, let x_q be a g-th root of y_q: $x_q^g = y_q \bmod q$. Then, by the Chinese remainder theorem, a g-th root of y modulo pq is given by the following lemma.

Lemma 2. *By the isomorphism $\mathbb{Z}_p \times \mathbb{Z}_q \xrightarrow{\sim} \mathbb{Z}_{pq}$, $(x_p, x_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ corresponds to an element $x_1 \in \mathbb{Z}_{pq}$, which is given by $x_1 = x_p + p\,((x_q - x_p)z \bmod q)$, and x_1 is a g-th root of y_1 $(= y \bmod pq)$.*

By using the g-th roots of $y \in (\mathbb{Z}_N^*)^g$ in modulus p, q and pq, we can calculate the g-th root of y in the high-power modulus $(p^i q,\ i = 2, 3, \ldots, d)$ as we will see in the following.

Lemma 3. *The notations are the same as in the above. Let $\eta_y = (g x_p^{g-1})^{-1} \bmod p$ and x_i $(1 \leq i < d)$ be a g-th root of y_i (modulo $p^i q$) such that $x_i \equiv x_p \pmod p$. Then a g-th root x_{i+1} of y_{i+1} modulo $p^{i+1}q$ such that $x_{i+1} \equiv x_p \bmod p$ is given by $x_{i+1} = x_i + \eta_y (y_{i+1} - x_i^g) \bmod p^{i+1} q$.*

Though it needs to calculate a modular inverse modulo p for η_y, under some condition, we can have η_y efficiently.

Lemma 4. *Assume that there exists some integer $0 \leq \alpha < p - 1$ depending only on p and g such that $x_p = y_p^\alpha \bmod p$, then we have $\left(x_p^{g-1}\right)^{-1} \bmod p = y_p^{\alpha - 1} \bmod p$.*

This follows from $y_p^{\alpha-1} \cdot x_p^{g-1} = (x_p \cdot y_p^{-1}) \cdot (y_p \cdot x_p^{-1}) = 1 \pmod p$. Thus, once we obtain $\left(x_p^{g-1}\right)^{-1} \bmod p$, precalculating $g^{-1} \bmod p$, we have $\eta_y = g^{-1} \left(x_p^{g-1}\right)^{-1} \bmod p$ by single modular multiplication.

Let us next consider the conditions in Lemma 4. That is, let us consider the relation between p and g so that there exists an integer α which depends only on p and g, such that for any $a \in \left(\mathbb{Z}_p^*\right)^g$, a^α is a g-th root of a $((a^\alpha)^g = a)$.

Proposition 1. *Let p be a prime, $1 < g < p-1$ be an integer. Then there exists an integer $\alpha = \alpha(p, g)$ $(1 \leq \alpha < p - 1)$ which depends only on p and g, and satisfies $(a^\alpha)^g = a$ for any $a \in \left(\mathbb{Z}_p^*\right)^g$ if and only if $\gcd(g, (p-1)/g_0) = 1$ where $g_0 = \gcd(g, p-1)$. Here, α is given by*

$$\alpha = \alpha(p, g) = \left(1 + u\{(p-1)/g_0\}\right)/g, \quad \text{where } u = (-(p-1)/g_0)^{-1} \bmod g.$$

Using the above discussion, if for p and q, g satisfies the conditions in Lemma 1, we have an efficient algorithm which calculates a g-th root (Note that using a g-th root, all g-th root are given by (2)).

Corollary 1. *Let p, q be prime integers. Let $d > 1$ be an integer, and $N = p^d q$. Let $g > 1$ be an integer which satisfies $g < \min(p-1, q-1)$ and $\gcd(g, (p-1)/\gcd(g, p-1)) = \gcd(g, (q-1)/\gcd(g, q-1)) = 1$. Moreover, let $z = p^{-1} \bmod q$, $\gamma = g^{-1} \bmod p$. For any $y \in (\mathbb{Z}_N^*)^g$, put $y_p = y \bmod p$, $y_q = y \bmod q$, $y_i = y \bmod p^i q$ $(1 \le i < d)$, $y_d = y$. Then by calculating*

$$\begin{aligned} x_0 &= y_p^{\alpha(p,g)-1} \bmod p, \quad & x_p &= y_p x_0 \bmod p, \\ \eta &= w x_0 \bmod p, & x_q &= y_q^{\alpha(q,g)} \bmod q, \end{aligned}$$
$$x_1 = x_p + p\left((x_q - x_p) z \bmod q\right),$$

and for $i = 2, 3, \ldots, d$, $x_i = x_{i-1} + \eta(y_i - x_{i-1}^g) \bmod p^i q$, we have that $x := x_d$ is a g-th root of y ($x^g = y \bmod N$).

2.4 Choices of Cryptographically Suitable Powering Functions

We will discuss the optimal choice of parameters g, g_p, g_q and d suitable for cryptography in the following.

Efficiency must be considered, when we apply powering functions that can be proven to be one-way under the assumption of infeasibility of the integer factoring problem to cryptosystems. The cost of calculating the image will be lower if the powering index is smaller. Moreover, if the number of preimages is larger, then there will be some inconvenience as previously mentioned.

g	g_p	g_q	$g_p \cdot g_q$	τ
2	2	2	4	.500
3	1	3	**3**	.667
	3	3	9	.444
4	2	4	8	.750
	4	4	16	.625
5	1	5	5	.800
	5	5	25	.320
6	2	6	12	.833
	6	6	36	.722
7	1	7	7	.857
	7	7	49	.245
8	2	8	16	.875
	4	8	32	.813
	8	8	64	.656

Table 1. Parameters for $2 \le g \le 8$

Table 1 shows all possibilities of g_p, g_q for relatively small g's. Note that cases where p and q are replaced have been omitted, and for even g, the parities of g_p and g_q (even or odd) coincide. Note also that the case $(g, g_p, g_q) = (2, 2, 2)$ (and $d = 1$) corresponds to the Rabin function. The value $g_p \cdot g_q$ indicates that the g-th power function f is a $(g_p \cdot g_q)$-to-1 mapping and is desired to be small. τ is the constant coefficient appearing in the reduction probability of Theorem 1 (See also Appendix A for more details). Although a larger value is desired, it is sufficient if it is greater than 0.5. From this table, we can conclude that the case $(g, g_p, g_q) = (3, 1, 3)$ (or $(g, g_p, g_q) = (3, 3, 1)$), that has smallest $g_p \cdot g_q$, is optimized for cryptosystems (ratio is 0.667 and sufficiently large). Moreover, as we will see in Section 3.4, the $p^d q$-type modulus ($d \ge 2$) makes the preimage calculation more efficient using the proposed algorithm in Section 2.3.

Thus letting F $(= f_{N,3})$ be the (3,1,3)-type function with the $p^d q$-type modulus $(d \geq 2)$, we can conclude that F is most suitable for cryptography. Table 2 sums up the positions of F and other typical functions including RSA functions. In the table, IF stands for integer factorization.

type (g, g_p, g_q)	map $g_p g_q : 1$	condition for p, q	$p^d q$-type modulus $d = 1$	$d \geq 2$	assumption for one-wayness
(e,1,1)	1:1	symmetric	RSA	—	RSA
(3,1,3)	3:1	asymmetric	—	F	IF
(2,2,2)	4:1	symmetric	Rabin	HIME	IF

Table 2. (3,1,3)-type and other typical functions

3 Application to Digital Signatures

As cryptographic applications of the arguments in the previous sections, we propose digital signature schemes using the cubic function considered in Section 2.4 and ensure the advantages of the proposed cubic function, especially in terms of efficiency (Section 3.4).

3.1 Basic Notation

We start by recalling the basic notion of digital signature schemes according to [3, 6, 12].

Definition 5. *A signature scheme* $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ *is defined as follows:*

The key generation algorithm $\mathcal{G}$ *is a PPT algorithm which has input* 1^k *and outputs a pair of matching public and secret keys* (pk, sk)*.*

The signature generation algorithm $\mathcal{S}$ *takes a message* M *to be signed and public and secret keys* (pk, sk)*, and outputs a signature* $x = \mathcal{S}_{pk,sk}(M)$*.*

The signature verification algorithm $\mathcal{V}$ *takes a message* M*, a candidate signature* x' *and public key pk, and outputs a bit* $\mathcal{V}_{pk}(M, x')$*, equal to 1 if the signature is accepted and 0 otherwise. We require that if* $x = \mathcal{S}_{pk,sk}$*, then* $\mathcal{V}_{pk}(M, x') = 1$*.*

On the security for signatures, we only deal with existential unforgeability under an adoptive chosen message attack which is the strongest notion([3, 6]). In this scenario, a forger of a signature can dynamically obtain signatures of messages of his choice and attempts to output a valid signature, where a pair of message and signature (M, x) is said to be a valid forgery if $\mathcal{V}_{pk}(M, x) = 1$ and the signature of M was never requested by the forger.

Most signature schemes use hash functions, and the security is proven under random oracle models, that is the models which is appropriately replaced the hash functions with random oracles([1]). In these models, forgers are allowed to access to random oracles. The resistance against these attacks is defined as follows:

Definition 6. *A forger $\mathcal{A}$ (a PPT algorithm) is said to (t, q_h, q_s, ϵ)-breaks the signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ if after at most $q_h(k)$ queries to the hash oracles, $q_s(k)$ signature queries and $t(k)$ processing time, it outputs a valid forgery with probability at least $\epsilon(k)$ (for any $k \in \mathbb{N}$). A signature scheme $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ is (t, q_h, q_s, ϵ)-secure if there exists no forger who (t, q_h, q_s, ϵ)-breaks the scheme.*

3.2 Proposed Signature Scheme: Scheme 1

We now propose new signatures constructed with the (3,1,3)-type cubic residue function F in Section 2.4 (in case $d = 2$). These are proven to be secure under the assumption of integer factoring infeasibility.

First, we will consider the (full domain) hash & sign (F-FDHS) signature which is most fundamental. Fix an integer $a > 1$ (regard a as a system parameter).

Key Generation
Generate randomly same length distinct prime numbers p and q such that $p \equiv 2 \bmod 3$, $q \equiv 4$ or $7 \bmod 9$, and choose a non-cubic residue a modulo q, put $N = p^2 q$. Let $H : \{0,1\}^* \to \mathbb{Z}_N^*$ be a hash function. Then output the public key (N, H) and the secret key (p, q) (a is open to public as a system parameter).

Signature Generation
1. For a message M, calculate $w = H(M)$.
2. Let y be one of w, aw, $a^2 w$ which is a cubic residue.
3. Calculate a cubic root x of y ($x \in F^{-1}(y)$).
4. Output x and end.

Signature Verification
1. For the message M, calculate $w' = H(M)$.
2. Calculate $y' = x^3 \bmod N$.
3. If y' coincides one of w', aw', $a^2 w'$, then output 1, else output 0 and end.

Remark 1. Note that from Lemma 1, we can easily seen that one of w, aw or $a^2 w$ is a cubic residue, and we can determine this by calculating $\chi_{p,3}$ (this function is also a powering function). Therefore, we do not have to recompute the hash value $H(M)$, and for a given message m we can uniquely generate the signature x of m. Kurosawa et al. proposed a similar technique for the Rabin signature [9].

We can prove that F-FDHS is secure over the random oracle model (the hash function H is replaced to the random oracles) under the assumption of integer factoring infeasibility. The proof is basically similar to that of [9].

For the fixed a and a positive integer k, let

$$\mathcal{N}_k = \left\{ N = p^2 q \;\middle|\; \begin{array}{l} p, q : \text{primes},\ |p| = |q| = k,\ p \bmod 3 = 2, \\ q \bmod 9 = 4 \text{ or } 7,\ a \in \mathbb{Z}_q^* \setminus \left(\mathbb{Z}_q^*\right)^3 \end{array} \right\},$$

and put $\mathcal{N} = \bigcup_k \mathcal{N}_k$. Then we can prove the following theorem, whose proof can be found in Appendix C.

Theorem 2. *If $\mathcal{N}$ is (t_I, ϵ_I)-secure, then F-FDHS is (t, q_H, q_s, ϵ)-secure, where,*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

In the following, combining the idea in Lemma 1 and Corollary 1, we propose an efficient algorithm, denoted by Φ, which for given $w \in \mathbb{Z}_N^*$, determines which of w, aw or a^2w is a cubic residue, and then calculates its cubic root. The validity of the algorithm can be found in Appendix D.

Let $\gamma = (p+1)/3$ and $z = p^{-1} \bmod q$. Let $\beta_p = (2p-4)/3$ and $\beta_q = (2q-8)/9$, $\zeta = a^{(q-1)/3} \bmod q$ if $q \equiv 4 \bmod 9$, $\beta_q = (q-7)/9$, $\zeta = a^{(2(q-1))/3} \bmod q$ if $q \equiv 7 \bmod 9$. Finally, let $b = a^{\beta_q+1} \bmod q$.

Algorithm Φ
Input: N, a, p, q, β_p, β_q, b, ζ, z, γ and $w \in \mathbb{Z}_N^*$.
Output: $x \in \mathbb{Z}_N^*$ s.t. $x^3 \bmod N \in \{w, aw \bmod N, a^2w \bmod N\}$.

Step 1. Check the cubic residuosity modulo q and calculate a cubic root

- **Step 1.1.** $w_q = w \bmod q$.
- **Step 1.2.** $w_1 = w_q^{\beta_q} \bmod q$.
- **Step 1.3.** $x_q = w_1 w_q \bmod q$.
- **Step 1.4.** $w_3 = w_1 x_q^2 \bmod q$.
- **Step 1.5.** if $w_3 \neq 1$ then
  - **Step 1.5.1.** Set $x_q \leftarrow b_q x_q \bmod q$, $w \leftarrow aw \bmod N$.
  - **Step 1.5.2.** If $w_3 \neq \zeta$ then set $x_q \leftarrow b_q x_q \bmod q$, $w \leftarrow aw \bmod N$.

Step 2. Calculate a cubic root modulo p

- **Step 2.1.** $w_p = w \bmod p$.
- **Step 2.2.** $x_0 = w_p^{\beta_p} \bmod p$.
- **Step 2.3.** $x_p = w_p x_0 \bmod p$.
- **Step 2.4.** $\eta = \gamma x_0 \bmod p$.

Step 3. $x_1 = x_p + p((x_q - x_p)z \bmod q)$.
Step 4. $x = x_1 + \eta(w - x_1^3) \bmod N$.

3.3 Other Constructions: Schemes 2 and 3

In the following, we present two additional constructions of digital signatures based on the generalized powering function.

Scheme 2. Let us consider a scheme F-2HS (2-hash and sign) that has been slightly changed from scheme 1 : F-FDHS. Similarly, fix an integer $a > 1$ and let k_1, k_2 be positive integers such that $k_1 + k_2 < |N|$ (modulus length), and regard these as system parameters in addition to a in scheme 1. The key generation is the same as for scheme 1 except for letting $H : \{0,1\}^* \to \{0,1\}^{k_1}$ and $G : \{0,1\}^{k_1} \to \{0,1\}^{k_2}$ be hash functions (H: compressor, G: generator). The public key is (N, H, G), and the secret key is (p, q).

Signature Generation
1. For a message M, calculate $w_1 = H(M)$, $w_2 = G(w_1)$ and let $w = w_1||w_2$.
2. Let y be one of w, aw, a^2w which is a cubic residue.
3. Calculate a cubic root x of y ($x \in F^{-1}(y)$).
4. Output x and end.

Signature Verification
1. For M, calculate $w'_1 = H(M)$, $w'_2 = G(w'_1)$ and let $w' = w'_1||w'_2$.
2. Calculate $y' = x^3 \bmod N$.
3. If y' coincides one of w', aw', a^2w', then output 1, else output 0 and end.

This scheme can also be proven to be secure over the random oracle model (the hash functions H and G are replaced with random oracles) under the factoring assumption. Let $\mathcal{N}$ be the same as in scheme 1. We then have the following:

Theorem 3. *If $\mathcal{N}$ is (t_I, ϵ_I)-secure, then F-2HS is (t,q_H,q_G,q_s,ϵ)-secure, where*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

This scheme is nothing more than a version of PSS([3]) without the random numbers part, and is not essentially different from scheme 1. However, with respect to implementation, we bother with the construction of hash functions with long output using some short output functions (*e.g.* [17]), and in most cases, it is inefficient when the hash function deals with very long messages.

Scheme 3. Finally, we will consider a message recovery signature scheme F-MR (message recovery) based on scheme 2 : F-2HS. For this scheme, the message length is restricted to $|M| = k_2$. The key generation and signature generation are the same as in scheme 2, except that we set $w_2 = G(w_1) \oplus M$ in Step 1 of the signature generation (Fig. 1). The signature verification and message recovery are as follows:

Signature Verification
1. Calculate $y' = x^3 \bmod N$.
2. For $i = 0, 1, 2$,
   2.1. Calculate $y_i = w_{1,i}||w_{2,i} = a_2^i y' \bmod N$ ($|w_{1,i}| = k_1$, $|w_{2,i}| = k_2$).
   2.2. Calculate $M_i = w_{2,i} \oplus G(w_{1,i})$.
3. If for some i, $w_{1,i} = H(M_i)$, then output 1 and M_i, else output 0 and end.

Similarly for this scheme, we can prove following:

Theorem 4. *If $\mathcal{N}$ is (t_I, ϵ_I)-secure, then F-MR is $(t, q_H, q_G, q_s, \epsilon)$-secure, where*

$$t_I(k) = t(k) + (q_H + q_s + 2)O(k^3), \quad \epsilon_I(k) = (2/3)\epsilon(k).$$

Similar to scheme 2, this scheme is nothing more than a version of PSS-R([3]) without the random number part, and this makes the message embedded in the signature longer than that of PSS-R.

As we have seen, the proposed schemes need no trial and error in hashing messages and in finding the cubic residue. This has a good effect on efficiency especially with huge messages. In what follows, we discuss the advantages in efficiency of the proposed schemes in detail.

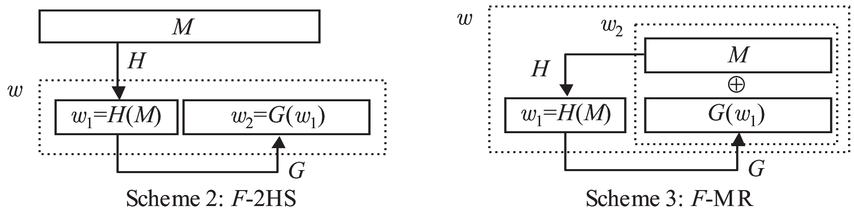

Fig. 1. Paddings for Schemes 2 and 3

3.4 Efficiency Consideration

In this section, we estimate the efficiency of signature schemes 1,2 and 3 that are introduced in the previous sections.

The proposed schemes deal with the modulus of $N = p^2q$ ($|p| \approx |q|$). In order to fairly compare the proposed schemes with the RSA signature, we estimate the efficiency of a fast variant of RSA signature, namely Multi-Prime RSA with $N = pqr$ ($|p| \approx |q| \approx |r|$) [13]. We consider the efficiency of signature generation which has higher costs in comparison with signature verification. Note that Multi-prime (pqr-type) Rabin's scheme has the same efficiency as RSA signature in signature verification.

The efficiency of public-key cryptosystems and digital signatures is frequently estimated by the number of modular multiplications. Let us introduce the following notations to represent the amount of calculation. Let $\mathrm{Mul}(t)$ denote the amount of calculation for an integer multiplication of t-bit integers. Similarly, let $\mathrm{RMul}(t)$ be that for a modular multiplication with a t-bit modulus, and $\mathrm{Red}(s, t)$ that for a reduction s-bit integer with a t-bit modulus. Also, let $\mathrm{RP}(t)$ be the number of t-bit modulus modular multiplications for powering with a t-bit exponent.

In Schemes 1,2 and 3, the steps for checking the cubic residuosity and calculating a cubic root (function Φ in Section 3.2), comprise a large percentage of signature generation. Thus we consider the efficiency of Φ. Let ℓ be the bit-length of modulus $N = p^2q$ ($|p| \approx |q|$). Signature generation needs the following amount of calculation:

$$\big(8 + 2 \cdot \mathrm{RP}(\ell/3)\big) \cdot \mathrm{RMul}(\ell/3) + 3 \cdot \mathrm{RMul}(\ell) + 2 \cdot \mathrm{Red}(\ell, \ell/3) + \mathrm{Mul}(\ell/3).$$

On the other hand, let ℓ be the bit-length of Multi-Prime RSA modulus $N = pqr$ ($|p| \approx |q| \approx |r|$), then for the generation of Multi-Prime RSA signature [13], it needs

$$\big(1 + 3 \cdot \mathrm{RP}(\ell/3)\big) \cdot \mathrm{RMul}(\ell/3) + 3 \cdot \mathrm{Red}(\ell, \ell/3) + \mathrm{Mul}(\ell/3).$$

We have approximately $\mathrm{RMul}(t) \approx n^2 \cdot \mathrm{RMul}(t/n)$, $\mathrm{Mul}(t) \approx (1/2) \cdot \mathrm{RMul}(t)$. Moreover, if we use the Montgomery method [11] for modular reduction, then we have $\mathrm{Red}(t, t/n) \approx \big((n+1)/n^2\big) \cdot \mathrm{RMul}(t)$. We set a standard to the number of modular multiplication on 1024-bit modulus: $1 = \mathrm{RMul}(1024)$. We also assume that a t-bit modular multiplication costs $\xi(t) := (t/1024)^2$. Then the amount of calculation for the signature generation of the proposed and RSA schemes is

$$\begin{aligned} &\text{Proposed schemes} : \{29/6 + (2/9)\mathrm{RP}(\ell/3)\}\, \xi(\ell), \\ &\text{Multi-Prime RSA} : \{3/2 + (1/3)\mathrm{RP}(\ell/3)\}\, \xi(\ell). \end{aligned}$$

For modular powering, we adopt the basic binary method. If we assume that half the bits in the exponent are non-zero, then this method needs $3t/2$ modular multiplications with a t-bit modulus (where we also assume that modular squaring and modular multiplication have the same amount of calculation). Taking all this into account, the number of modular multiplications in the proposed schemes and the RSA signature and their ratio are as follows:

$$\text{Proposed Schemes} : (29/6 + \ell/81)\, \xi(\ell), \quad \text{Multi-Prime RSA} : (3/2 + \ell/36)\, \xi(\ell),$$

$$\text{Multi-Prime RSA/Proposed scheme} \approx 1.71.$$

Thus, we can say that the proposed schemes are considerably more efficient in signature generation than Multi-Prime RSA signature. Similarly, we can see that the proposed schemes are three or more times more efficient than the pq-type RSA-CRT signature.

4 Summary

We studied modular powering functions suitable for cryptography. In particular, we proposed a 3-to-1 functions, which can be proven to be one-way under the factoring assumption. The three ambiguities of the kernel can easily be distinguished by a non-cubic residue element. For the $p^d q$-type modulus ($d \geq 2$), we proposed a more efficient method of calculating preimages for these functions, which requires no modular inversion algorithm for Hensel lifting. Thus we can say that the proposed functions are optimized in terms of security and efficiency.

As cryptographic applications, we also proposed new digital signature schemes which utilize the new functions with $d = 2$. Finally, we showed that the proposed schemes are about 1.71 times more efficient than Multi-Prime RSA with the same length modulus (more than three times faster than the pq-type RSA-CRT signature).

References

1. M.Bellare and P.Rogaway, Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ACM Press (1993), 62-73. Available at `http://www.cs.ucdavis.edu/ rogaway/papers/index.html`
2. M.Bellare and P.Rogaway, Optimal asymmetric encryption – How to encrypt with RSA, *Advances in Cryptology – Eurocrypt'94*, LNCS 950, Springer-Verlag (1994), 92-111.
3. M.Bellare and P.Rogaway, The exact security of digital signatures – How to sign with RSA and Rabin, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, Springer-Verlag (1996), 399-416.
4. D.Boneh, G.Durfee and N.Howgrave-Graham, Factoring $N = p^r q$ for large r, *Advances in Cryptology – Crypto'99*, LNCS 1666, Springer-Verlag (1999), 326-337.
5. D.Boneh and H.Shacham, Fast Variants of RSA, *CRYPTOBYTES*, Vol.5, No.1, 2002, 1-9.
6. J.S.Coron, Optimal security proofs for PSS and other signature schemes, *Advances in Cryptology - EUROCRYPT 2002 Proceedings*, Springer-Verlag (2002), 272-287.
7. HIME(R) : High Performance Modular Squaring Based Public-Key Encryption (Revised Edition), Hitachi, Ltd., 2001.
8. K.Kurokawa, T.Ito and M.Takeuchi, Public Key cryptosystem using a reciprocal number with the same intractability as factoring a large number, Cryptologia, 12 (1988), 225-233.
9. K.Kurokawa and W.OGATA, Efficient Rabin-type Digital Signature Scheme, Designs, Codes and Cryptography 16 (1999), 53-64.
10. J.H.Loxton, and D.S.P.Khoo, G.J.Bird and J.Seberry, A cubic RSA code equivalent to factorization, Journal of Cryptology, 5, (1992), 139-150.
11. P.L.Montgomery, Modular mutiplication without trial division, Math. Comp., 44 (1985), 519-521.
12. D.Pointcheval and J.Stern, Security proofs for signature schemes, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, Springer-Verlag (1996), 399-416.
13. Public-Key Cryptography Standards, PKCS # 1, Amendment 1: Multi-Prime RSA, RSA Laboratories.
14. M.O.Rabin, Digitalized signatures and public key cryptosystems as intractable as factorization, Technical Report, MIT/LCS/TR-212, MIT, Cambridge, MA (1979).
15. R.L.Rivest, A.Shamir and L.Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol.21, No.2 (1978), 120-126.
16. R.Scheidler, A Public-Key Cryptosystem Using Purely Cubic Fields, *J. Cryptology*, vol. 11 (1998), 109-124.
17. V.Shoup, A Proposal for an ISO Standard for Public Key Encryption (v. 2.1), ISO/IEC JTC1/SC27, N2563, 2001. Available at `http://shoup.net/papers/` or `http://eprint.iacr.org/`

18. T.Takagi, Fast RSA-type cryptosystem modulo $p^k q$, *Advances in Cryptology — CRYPTO '98*, LNCS 1462, 1998, 318-326.
19. H.C.Williams, A modification of the RSA public key encryption procedure, IEEE Trans. on Information Theory, IT-26, 6 (1980), 726-729.
20. H.C.Williams, Some public-key crypto-functions as intractable as factorization, *Cryptologia*, vol. 9, no.3 (1985), 223-237.
21. H.C.Williams, An M^3 public-key encryption scheme, *Advances in Cryptology – CRYPTO'85 Proceedings*, Springer-Verlag (1986), 358-368.

A Proof of Theorem 1

We begin with the next lemma.

Lemma 5. *If we identify $\mathbb{Z}_N$ with integers between 0 and $N-1$, then as integers, for $0 < i < g_p$ and $0 < j < g_q$, the followings hold.*

$$\gcd(\phi(\zeta_{p,g}^i, 1) - 1, N) = q, \quad \gcd(\phi(1, \zeta_{q,g}^j) - 1, N) = p^d.$$

Proof. Since $\phi(\zeta_{p,g}^i, 1) - 1 \bmod p^d = \zeta_{p,g}^i - 1 \neq 0$, and $\phi(\zeta_{p,g}^i, 1) - 1 \bmod q = 1 - 1 = 0$, we can see that the greatest common multiple of this and N is equal to q. Similarly we can prove the second equation.

Let us denote $H_{N,g}$ the set of roots of unity in Lemma 5:

$$H_{N,g} = \left\{\phi(\zeta_{p,g}^i, 1) \mid 0 < i < g_p\right\} \cup \left\{\phi(1, \zeta_{q,g}^j) \mid 0 < j < g_q\right\} \subset Z_{N,g}.$$

Moreover, let us define $G_{N,g}$ as follows:

$$G_{N,g} = \{x \in Z_{N,g} \mid x^e \in H_{N,g} \text{ for some divisor } e \text{ of } g\}.$$

From the definition, $H_{N,g} \subset G_{N,g}$. Then the number of elements in $G_{N,g}$, denoted by $g_{N,g}$, is given by following.

Lemma 6. $g_{N,g} = g_p g_q - \sum_{e | \gcd(g_p, g_q)} \varphi(e)^2$.

Proof. $\phi(\zeta_{p,g}^i, \zeta_{q,g}^j) \in Z_{N,g}$ is in $G_{N,g}$ if and only if the order of p-part of $\phi(\zeta_{p,g}^i, \zeta_{q,g}^j)$ is different from that of q-part of it. Hence, elements in $Z_{N,g} \setminus G_{N,g}$ are which have same order in p-part and q-part, in this case, the orders are divisors of $\gcd(g_p, g_q)$. For each divisor $e | \gcd(g_p, g_q)$, the number of elements which have the order of p and q-part e is equal to $\varphi(e)^2$, which gives the desired result.

Proof of Theorem 1: We now give the proof of Theorem 1. Under the assumptions, let us put $g_p = \gcd(g, p-1)$, $g_q = \gcd(g, q-1)$, then f is $g_p g_q : 1$ ($g_p g_q > 1$) function (Of course, the adversary does not know p, q, but she knows that f is not injective). We assume that there exists a PPT algorithm A which computes a preimage of f. That is, A has input k, d, g, $\mathcal{D}_1$, $\mathcal{D}_2$, N, y, and it outputs x' in $f^{-1}(y) = \{x \in \mathbb{Z}_N^* \mid x^g = y\}$ with non-negligible probability. Using A, we construct an algorithm M which factors N as follows:

Input of M : k, d, g, $\mathcal{D}_1$, $\mathcal{D}_2$, N
Output of M : a prime factor of N
1. Choose randomly $x \in \mathbb{Z}_N^*$ $(x \neq 1)$.
2. Calculate $y = x^g \bmod N$.
3. Input $(k, d, g, \mathcal{D}_1, \mathcal{D}_2, N, y)$ to A.
4. For an output x' of A, if $y \neq x'^g \bmod N$, then **Fail**.
5. Calculate $z = x'/x \bmod N$.
6. For each divisor e of g, calculate $w = \gcd((z^e - 1 \bmod N), N)$, if w is non-trivial divisor of N, then output w and end, otherwise **Fail**.

In Step 6, it outputs non-trivial divisor w if and only if $z \in G_{N,g}$, hence the success probability in Step 6 is equal to $g_{N,g}/g_p g_q$. If we put the success probability of A (that is, the probability such that it does not **Fail** in Step 4) to $\mathrm{Adv}(A) = \epsilon$, then, by Lemma 6, the final success probability of M, namely, the probability such that M factors N, is equal to

$$\frac{g_{N,g}}{g_p g_q} \cdot \epsilon = \frac{g_p g_q - \sum_{e \mid \gcd(g_p, g_q)} \varphi(e)^2}{g_p g_q} \cdot \epsilon,$$

which is non-negligible by the assumptions.

B Proofs of lemmas in Section 2.3

Proof of Lemma 3: By the assumptions, we have $y_{i+1} - x_i^g \bmod p^i q = y_i - x_i^g \bmod p^i q = 0$. Hence $x_{i+1} \bmod p = x_i \bmod p = x_p$. Moreover, $x_{i+1}^g = x_i^g + g x_i^{g-1} \eta_y (y_{i+1} - x_i^g) \bmod p^{i+1} q$, and by the assumption on x_i and the definition of η_y, we have $g x_i^{g-1} \eta_y \bmod p = g x_p^{g-1} \eta_y \bmod p = 1 \bmod p$. Therefore, we have $x_{i+1}^g \bmod p^{i+1} q = x_i^g + y_{i+1} - x_i^g \bmod p^{i+1} q = y_{i+1}$.

Proof of Proposition 1: Let a be a generator of the cyclic group $\left(\mathbb{Z}_p^*\right)^g$. The order of a is equal to $(p-1)/g_0$. If there exists an integer α which satisfies the condition, we have $(a^\alpha)^g = a$, thus it must be $\alpha \cdot g \equiv 1 \pmod{(p-1)/g_0}$. That is , g must be prime to $(p-1)/g_0$. Conversely, if $\gcd(g, (p-1)/g_0) = 1$, then let α be as above, it is directly checked that it satisfies the condition. As the order of a is $(p-1)/g_0$, we have $a^{\alpha g} = a^{1+u\{(p-1)/g_0\}} = a \pmod{p}$. Moreover, $u = \{-(p-1)/g_0\}^{-1} \bmod g$, hence the numerator of α is divided by g, thus α is an integer. Since $u \leq g-1$, $g_0 \leq g < p-1$, we have $1 + u\{(p-1)/g_0\} \leq 1 + (g-1)\{(p-1)/g_0\} < g(p-1)/g_0 \leq g(p-1)$. Thus α satisfies $1 \leq \alpha < p-1$.

C Proofs of the Security of Proposed Schemes

Proof of Theorem 2: Let $\mathcal{A}$ be a forger which (t, q_H, q_s, ϵ)-breaks the signature scheme F-FDH. The input of $\mathcal{A}$ is a public key (N, a). $\mathcal{A}$ has oracle access to random oracle H. Then we construct the factoring algorithm I which can (t_I, ϵ_I)-break by using $\mathcal{A}$. The input of I is $N \in \mathcal{N}$. I gives $\mathcal{A}$ the public key N (we assume that I and $\mathcal{A}$ know a as a system parameter). After this, $\mathcal{A}$ begins to make sign queries and hash queries. For these queries, I behaves as follows.

If $\mathcal{A}$ makes a sign query without having made the corresponding hash query, I at once goes ahead and makes the hash query itself, and then corresponds for sign query as described below. Similarly for the output forgery, thus we may assume that if $\mathcal{A}$ makes a sign query or outputs a forgery, then it has already made corresponding hash query. Hence, effective number of hash queries is at most $q(k) = q_H(k) + q_s(k) + 1$.

To answer queries, I makes the query-mapping table (Q, A) as follows: Start with $Q = A = \phi$ (empty set). Suppose $\mathcal{A}$ makes a hash query m.

If $m \notin Q$, then I chooses randomly $r \in_R \mathbb{Z}_N^*$ and $i \in \{0, 1, 2\}$, returns $H(m) = r^3/a^i \bmod N$, and sets $Q = Q \cup \{m\}$, $A = A \cup \{(m, r, i, H(m))\}$.

If $m \in Q$, then I finds corresponding $(m, r, i, H(m)) \in A$ and returns $H(m)(= r^3/a^i \bmod N)$.

Next, suppose that $\mathcal{A}$ makes a sign query m. As mentioned above, we can assume that there was already a hash query m, hence there exists corresponding $(m, r, i, H(m)) \in A$. I finds this and returns r as the signature for m.

Finally, suppose that $\mathcal{A}$ outputs a forgery $(\hat{m}, \hat{s})$. If $\hat{s}$ is valid, then for some i, $a^i H(\hat{m}) = \hat{s}^3 \bmod N$. N is chosen randomly and H is random from our construction of I. Hence $\mathcal{A}$ can not distinguish the behavior of I from the original game, thus $\mathcal{A}$ succeeds this simulation with original success probability ϵ.

On the other hand, by the assumption, $\hat{m} \in Q$, thus there exists the corresponding $(\hat{m}, \hat{r}, i, H(m)) \in A$ and it holds $\hat{r}^3 = \hat{s}^3 \bmod N$. Suppose that $\hat{s}$ is valid. From the argument in Theorem 1, using the above equations (if $\hat{r} \neq \hat{s}$), a non-trivial factor of N can be calculated with success probability $2/3$. Thus I succeeds in factoring N with probability $\epsilon_I = (2/3)\epsilon$.

Let $t_0(k)$ be processing time for a modular multiplication with k-bit modulus. I carries out 3-modular multiplications for each hash query, hence also from Theorem 1, the processing time t' of I is given by

$$\begin{aligned} t' &\leq t + O(k^3) + 3(q_H + q_s + 1)t_0(k) \\ &= t + (q_H + q_s + 2)O(k^3). \end{aligned}$$

Proof of Theorem 3, 4: Theorem 3 can be proven just like Theorem 2 except for the behavior of simulator I.

First, I makes the query mapping table (Q_H, A_H), (Q_G, A_G) starting with empty sets. Suppose the forger $\mathcal{A}$ makes a H-query m. If $m \notin Q_H$, then I chooses $r \in_R \mathbb{Z}_N^*$ and $i \in \{0, 1, 2\}$ calculate $y = w_1 || w_2 = r^3/a^i \bmod N$ ($|w_1| = k_1$, $|w_2| = k_2$) and sets $Q_H = Q_H \cup \{m\}$, $A_H = A_H \cup \{(m, r, i, w_1, w_2)\}$, $Q_G = Q_G \cup \{w_1\}$, $A_G = A_G \cup \{(w_1, w_2)\}$. Finally, I returns $H(m) = w_1$. If $m \in Q_H$, then I finds corresponding $(m, r, i, w_1, w_2) \in A_H$ and returns $H(m) = w_1$.

When $\mathcal{A}$ makes a G-query w_1, if $w_1 \in Q_G$, then I finds corresponding $(w_1, w_2) \in A_G$ and returns $G(w_1) = w_2$, else I generates randomly r_2, $|r_2| = k_2$, sets $Q_G = Q_G \cup \{w_1\}$, $A_G = A_G \cup \{(w_1, r_2)\}$ and returns $G(w_1) = r_2$.

Finally, suppose that $\mathcal{A}$ makes a sign query m. We can assume that $m \in Q_H$, hence I can finds corresponding $(m, r, i, w_1, w_2) \in A_H$ and returns r as a signature for m.

Then, as in Theorem 2, we can see that I can factor the modulus using the forgery outputted by $\mathcal{A}$ with the indicated probability and processing time.

The proof for Theorem 4 is similar as in Theorem 3, so we omit the detail.

D Validity of the algorithm Φ

We can easily obtain the algorithm Φ from Corollary 1 and the following lemma.

Let $a \in \mathbb{Z}_q^*$ be a non-cubic residue and fix. In each case $q \bmod 9 = 4$ or 7, define the followings: in case of $q \bmod 9 = 4$, $\alpha = (2q+1)/9$, $\zeta = a^{(q-1)/3} \bmod q$, and in case of $q \bmod 9 = 7$, $\alpha = (q+2)/9$, $\zeta = a^{(2(q-1))/3} \bmod q$. Moreover, put $\beta = \alpha - 1$ and $b = a^\alpha \bmod q$.

Lemma 7. *For $w \in \mathbb{Z}_q^*$, put $w_1 = w^\beta \bmod q$, $w_2 = w_1 \cdot w \bmod q$ and $w_3 = w_1 \cdot w_2^2 \bmod q$. Then $w_3 \in \{1, \zeta, \zeta^2\}$, and we have followings: $w_2^3 \equiv w \bmod q$ if $w_3 = 1$, $(bw_2)^3 \equiv aw \bmod q$ if $w_3 = \zeta$ and $(b^2w_2)^3 \equiv a^2w \bmod q$ otherwise ($w_3 = \zeta^2$).*

Proof. By the assumptions and Lemma 1, ζ is a non-trivial cubic root of unity. Note that $\chi_{q,3}(a) = \zeta$ ($q \bmod 9 = 4$), $= \zeta^2$ ($q \bmod 9 = 7$). Moreover, in case of $q \bmod 9 = 4$, $w_3 = w^{2(q-4)/9+2(2q+1)/9} = w^{2(q-1)/3} = \chi_{q,3}(w)^2$, in case of $q \bmod 9 = 7$, $w_3 = w^{(q-7)/9+2(q+2)/9} = w^{(q-1)/3} = \chi_{q,3}(w)$. Hence, by Lemma 1, we have $w_3 \in \{1, \zeta, \zeta^2\}$, and $w_3 = 1$ means that w is a cubic residue. In this case, by Lemma 1, $w_2 = w^\alpha$ is a cubic root of w. In case of $w_3 = \zeta$, if $q \bmod 9 = 4$, then, by the above, we have $\chi_{q,3}(w) = \zeta^2$, and $\chi_{q,3}(aw) = \zeta \cdot \zeta^2 = 1$, moreover since $bw_2 = (aw)^\alpha \bmod q$, by Lemma 1, we have the result. In case of $q \bmod 9 = 7$ or $w_3 = \zeta^2$, it can be shown similarly.

E Decryption of RSA function

We briefly recall that the decryption algorithm for Multi-Prime RSA [13].

Let p, q, r be distinct prime numbers, $N = pqr$, $z_q = p^{-1} \bmod q$, and $z_r = (pq)^{-1} \bmod r$. Let $1 < d < N$ be an integer such that $\gcd(d, (p-1)(q-1)(r-1)) = 1$ and $d_p = d \bmod p$, $d_q = d \bmod q$, $d_r = d \bmod r$. In the case of p, q, r are known, for any $C \in \mathbb{Z}_N^*$, we can calculate $M = C^d \bmod M$ (this is the preimage of C by the RSA function $x^e \bmod N$, where $e = d^{-1} \bmod (p-1)(q-1)(r-1)$) as follows:

Step 1. $C_p = C \bmod p$, $C_q = C \bmod q$, $C_r = C \bmod r$.
Step 2. $M_p = C_p^{d_p} \bmod p$, $M_q = C_q^{d_q} \bmod q$, $M_r = C_r^{d_r} \bmod r$.
Step 3. $M_{pq} = M_p + p((M_q - M_p)z_q \bmod q)$.
Step 4. $M = M_{pq} + (pq)((M_r - M_{pq})z_r \bmod r)$.
Step 5. Output M and end.