Hindawi
Security and Communication Networks
Volume 2018, Article ID 2369507, 9 pages
https://doi.org/10.1155/2018/2369507

Research Article

Under Quantum Computer Attack: Is Rainbow a Replacement of RSA and Elliptic Curves on Hardware?

Haibo Yi

School of Computer Engineering, Shenzhen Polytechnic, Shenzhen 518055, China

Correspondence should be addressed to Haibo Yi; haiboyi@126.com

Received 26 October 2017; Accepted 15 January 2018; Published 11 February 2018

Academic Editor: Umar M. Khokhar

Among cryptographic systems, multivariate signature is one of the most popular candidates since it has the potential to resist quantum computer attacks. Rainbow belongs to the multivariate signature, which can be viewed as a multilayer unbalanced Oil-Vinegar system. In this paper, we present techniques to exploit Rainbow signature on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion based on binary trees. Second, we present an efficient multiplication based on compact construction in composite fields. Third, we present a parallel solving system of linear equations based on Gauss-Jordan elimination. Via further other minor optimizations and by integrating the major improvement above, we implement our design in composite fields on standard cell CMOS Application Specific Integrated Circuits (ASICs). The experimental results show that our implementation takes 4.9 us and 242 clock cycles to generate a Rainbow signature with the frequency of 50 MHz. Comparison results show that our design is more efficient than the RSA and ECC implementations.

1. Introduction

The idea of public key cryptography was introduced by Diffie and Hellman. Their method for key exchange came to be known as Diffie-Hellman key exchange [1]. This was the first published practical method for establishing a shared secret key over an authenticated communications channel without using a prior shared secret. Then a public key cryptographic scheme was invented by Rivest et al. [2]. This scheme came to be known as RSA, from their initials. RSA uses exponentiation modulo, a product of two very large primes, to encrypt and decrypt, performing both public key encryption and public key digital signature. The introduction of elliptic curve cryptography by Koblitz [3] and Miller [4] in the mid-1980s has yielded new public key algorithms based on the discrete logarithm problem. Elliptic curves provide smaller key sizes and faster operations for approximately equivalent estimated security. Since then, various schemes of encryption and signature generation have been developed in the field of public key cryptography.

Efficient implementations of these schemes have played a crucial role in numerous real-world security applications, such as confidentiality, authentication, integrity, and nonrepudiation. Since software implementations even on multicore processors can often not provide the performance level needed, hardware implementations are thus the only option, which appear to be a promising solution to inherent performance issues of public key cryptographic systems and provide greater resistance to tampering. Among hardware implementations of public key cryptographic systems, RSA and elliptic curves systems are the most widely adopted candidates [5–14]. Their security lies in the difficulty of factorizing large integers and the discrete logarithm problem, respectively. Shor algorithm was invented by Shor which could solve the problems of the prime factors of large numbers and elliptic curve discrete logarithm in polynomial time [15]. Such cryptographic schemes have potential weakness under quantum computer attacks.

Multivariate cryptography is one of the most popular postquantum cryptography since it has the potential to

resist quantum computer attacks [16]. The main strength of multivariate cryptography is that its underlying mathematical problem is to solve a set of Multivariate Quadratic (MQ) polynomial equations in a finite field, which is proven to be an NP-hard problem [17]. During the past thirty years, various multivariate cryptographic schemes have been proposed, like Unbalanced Oil-Vinegar Signature (UOV) [18], Rainbow [19, 20], Tame Transformation Signature (TTS) [21, 22], and others [23–25]. Their implementations have been one of the subjects of a lot of researches and continue to be a topic of interest in many areas, for example, efficient multivariate systems on Field Programmable Gate Arrays (FPGAs) [26], small multivariate processors on FPGAs [27], high speed Rainbow on FPGAs [28], and minimized multivariate PKC on Application Specific Integrated Circuits (ASICs) [29].

Among the existing multivariate cryptographic schemes, Rainbow belongs to Oil-Vinegar family, which can be viewed as a multilayer unbalanced Oil-Vinegar system. Compared with RSA and elliptic curves, the security of Rainbow is based on solving a set of MQ polynomial equations, which has the potential to resist quantum computer attacks.

Our Contributions. In this paper, we present techniques to exploit Rainbow signature on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion in $GF((2^4)^2)$ based on binary trees, which is the extension of the work in [30]. Second, we present an efficient multiplication in $GF((2^4)^2)$ based on compact construction, which is the extension of the work in [27]. Third, we present a parallel solving system of linear equations in $GF((2^4)^2)$ based on Gauss-Jordan elimination, which is based on the work in [28]. Via further other minor optimizations and by integrating the major improvement above, our design is implemented on ASICs and provides significant reductions in time-area product. The comparisons with other public key cryptographic systems show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

Moreover, our design can be generalized with minor modifications that also support FPGAs. Besides, Rainbow implementations on hardware must be protected against a wide range of attacks, including side channel attacks. Side channel attack belongs to physical attack, which is any attack based on information gained from the physical implementation of cryptographic systems, rather than brute force or theoretical weaknesses in cryptographic algorithms. Therefore, we discuss defending against a possible differential power analysis for Rainbow and we present countermeasures against fault analysis and differential power analysis attack.

Organization. The rest of this paper is organized as follows: Section 2 introduces Rainbow signature schemes. Section 3 presents building blocks for Rainbow schemes. Section 4 presents efficient implementations of Rainbow on ASICs. Section 5 compares our design with other public key cryptographic systems. Section 6 discusses defending against a possible differential power analysis for Rainbow. Section 7 summarizes our design.

2. Preliminary

Among multivariate signatures, Rainbow belongs to Oil-Vinegar family, which can be viewed as a multilayer unbalanced Oil-Vinegar system. The construction of Rainbow includes affine transformation L_1, central map transformation F, and affine transformation L_2; that is,

$$F \circ L_2 (x_0, x_1, \ldots, x_{n-1}) = L_1^{-1} (y_0, y_1, \ldots, y_{m-1}). \quad (1)$$

The hash value of the message of Rainbow is $y(y_0, y_1, \ldots, y_{m-1})$ and its size is m, where $y_0, y_1, \ldots, y_{m-1}$ are elements in a finite field. We also suppose that the signature is $x(x_0, x_1, \ldots, x_{n-1})$ and its size is n, where $x_0, x_1, \ldots, x_{n-1}$ are elements in a finite field. The private keys of Rainbow are L_1, F, and L_2.

Among the existing Rainbow schemes, Rainbow(17, 13, 13) is commonly believed to provide a security level of 2^{80} [20], which works with 17 first-layer Vinegar variables and 13 first-layer and 13 second-layer Oil variables in $GF(256)$. This scheme is depicted in Table 1 and is introduced as follows.

We suppose that the hash value of the message is $y(y_0, y_1, \ldots, y_{25})$ and its size is 26, where $y_0, y_1, \ldots, y_{25}$ are field elements. We also suppose that the signature is $x(x_0, x_1, \ldots, x_{42})$ and its size is 43, where $x_0, x_1, \ldots, x_{42}$ are field elements.

In order to sign a message, we need to solve the equation

$$F \circ L_2 (x_0, x_1, \ldots, x_{42}) = L_1^{-1} (y_0, y_1, \ldots, y_{25}). \quad (2)$$

To do this, we first solve

$$\overline{y} = L_1^{-1} (y_0, y_1, \ldots, y_{25}). \quad (3)$$

L_1^{-1} is an affine transformation:

$$\overline{y} = Ay + B, \quad (4)$$

where A is a matrix with the size of 26×26 and B is a vector with the size of 26. A and B are parts of private keys.

Second, we solve

$$\overline{x} = F^{-1} (\overline{y}_0, \overline{y}_1, \ldots, \overline{y}_{25}), \quad (5)$$

where the construction depends on a map

$$F (\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}) = (f_0, f_1, \ldots, f_{25}). \quad (6)$$

F is a two-layer construction; namely, $(f_0, f_1, \ldots, f_{25})$ are divided into two layers:

$$\begin{aligned} &0 : f_i \mid i = 0, 1, \ldots, 12 \\ &1 : f_i \mid i = 13, 14, \ldots, 25. \end{aligned} \quad (7)$$

Similarly, $\overline{x}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42})$ are divided into two layers. $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{16}$ and $\overline{x}_{17}, \overline{x}_{18}, \ldots, \overline{x}_{29}$ are Vinegar variables and Oil variables of the first layer, respectively; $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}$ and $\overline{x}_{30}, \overline{x}_{31}, \ldots, \overline{x}_{42}$ are Vinegar variables and Oil variables of the second layer, respectively.

Table 1: Parameters of Rainbow signature schemes.

Finite field	Message size	Signature size	Private key	Public key	Number of layers	LSE size
$GF((2^4)^2)$	26 bytes	43 bytes	L_1, L_2, F	$L_1 \circ F \circ L_2$	2	13×13

MQ polynomials f are defined by

$$f(O_0, O_1, \ldots, O_{o-1}) = \sum \alpha_{ij} O_i V_j + \sum \beta_{ij} V_i V_j + \sum \gamma_i V_i + \sum \delta_i O_i + \eta, \quad (8)$$

where $O_i, V_i/V_j$ are Oil and Vinegar variables on this layer and the coefficients α_{ij}, β_{ij}, γ_i, δ_i, and η are parts of private keys.

We randomly choose $\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{16}$ and evaluate $f_0, f_1, \ldots, f_{12}$. Then we solve the systems of linear equations on $\overline{x}_{17}, \overline{x}_{18}, \ldots, \overline{x}_{29}$. Then we evaluate $f_{13}, f_{14}, \ldots, f_{25}$ and solve the systems of linear equations on $\overline{x}_{30}, \overline{x}_{31}, \ldots, \overline{x}_{42}$.

Last, we solve

$$x = L_2^{-1}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42}). \quad (9)$$

L_2^{-1} is an affine transformation

$$x = C\overline{x} + D, \quad (10)$$

where C is a matrix with the size of 43×43 and D is a vector with the size of 43. C and D are parts of private keys.

Then x is the signature of y.

3. Building Blocks for Rainbow Schemes

Considering Section 2, we see that, in order to generate a Rainbow signature, the following operations are required:

(1) Computing affine transformations, that is, $\overline{y} = Ay + b$, where A is a matrix and b is a vector

(2) Computing central map transformation, that is, evaluating multivariate polynomials and solving systems of linear equations.

Computing these operations requires multiplications, inversions, and solving systems of linear equations in a finite field, which are presented in the following.

3.1. A Fast Inversion Based on Binary Trees. We suppose that $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$ are the elements in $GF((2^4)^2)$ and $b(x)$ is the inverse of $a(x)$, where the subfield is $GF(2^4)$ and a_h, a_l, b_h, and b_l are elements in $GF(2^4)$. The irreducible polynomials in $GF((2^4)^2)$ are $q(x) = x^2 + x + 9$. Then the inversion is computed as follows:

$$b_l = (a_h + a_l) \times \left(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2\right)^{-1},$$
$$b_h = a_h \times \left(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2\right)^{-1}. \quad (11)$$

We adopt a pipelined architecture in $GF(2^4)$, which is the extension of the work in [30]. We use two binary trees for computing squares and inversions in $GF(2^4)$, which are illustrated as follows:

(1) Each binary tree has four layers; root nodes are on the third layer.

(2) Each node has at most two child nodes, left node represents value of zero, and right node represents value of one.

(3) Each child must either be a leaf or be the root of another tree; each node has a father node when it is not a root node.

(4) Each element in a finite field has a unique traversal from root to leaf.

(5) Each leaf (most) is linked to another leaf.

Figure 1 is the architecture based on binary trees for computing squares and inversions in $GF(2^4)$. We use two architectures in our design, that is, square-trees for squares and inversion-trees for inversions.

Square-trees: we suppose that traversal from root (e_0) to leaf (e_3) includes tree nodes e_0, e_1, e_2, and e_3, which represents the element $a(x)$ in $GF(2^4)$. If traversal from root (e_4) to leaf (e_7) represents the element $b(x)$ in $GF(2^4)$, which is the square of $a(x)$, then e_3 is linked to e_7. When we are required to compute the square of $a(x)$, it is very convenient to find its square via traversing the square-trees.

Inversion-trees: we suppose that traversal from root (e_0) to leaf (e_3) includes tree nodes e_0, e_1, e_2, and e_3, which represents the element $a(x)$ in $GF(2^4)$. If traversal from root (e_4) to leaf (e_7) represents the element $b(x)$ in $GF(2^4)$, which is the inverse of $a(x)$, then e_3 is linked to e_7. When we are required to compute the inverse of $a(x)$, it is very convenient to find its inverse via traversing the inversion-trees.

Since square-trees and inversion-trees have four layers, we can use them to compute squares and inversions with pipelining. The computation of $b(x) = a(x)^{-1}$ is presented as follows:

(1) Via using square-trees, we can compute ${a_h}^2$ and ${a_l}^2$ with pipelining.

(2) Via using a multiplier, we can compute $a_l \times a_h$.

(3) Via using a multiplier and an adder, we compute $9 \times {a_h}^2$ and $a_l \times a_h + {a_l}^2$.

(4) Via using inversion-trees, we can compute $(9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$.

(5) Via using a multiplier and an adder, we compute $(a_h + a_l) \times (9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$ and $a_h \times (9 \times {a_h}^2 + a_l \times a_h + {a_l}^2)^{-1}$.

(6) The inversion has been computed.

3.2. An Efficient Multiplication Based on Compact Construction. We suppose that $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$

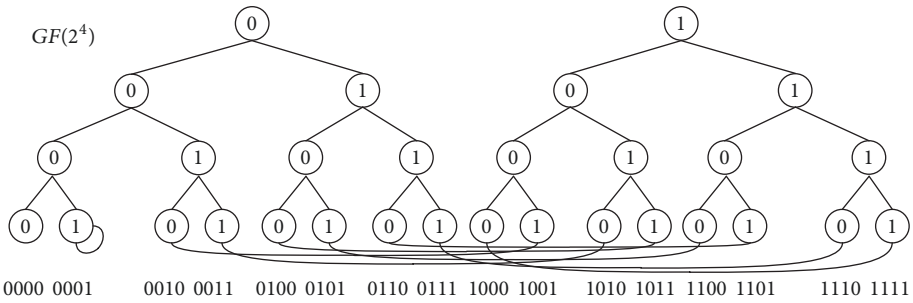

Figure 1: A pipelined architecture based on binary trees for computing squares and inversions in $GF(2^4)$.

are the elements in $GF((2^4)^2)$, where a_h, a_l, b_h, and b_l are elements in $GF(2^4)$. We also suppose that $c(x) = c_h x + c_l$ is the multiplication result of $a(x)$ and $b(x)$, where $a(x)$ is an element in $GF((2^4)^2)$ and c_h, c_l are elements in $GF(2^4)$. The irreducible polynomials in $GF((2^4)^2)$ are $q(x) = x^2 + x + 9$. Then the multiplication is computed as follows:

$$\begin{aligned} a(x) \times b(x) &= (a_h x + a_l)(b_h x + b_l) \\ &= \left(a_h b_h x^2 + (a_h b_l + a_l b_h) x + a_l b_l\right) \bmod q(x). \end{aligned} \tag{12}$$

By substituting $q(x) = x^2 + x + 9$ into (12), we have

$$\begin{aligned} c_h &= (a_h + a_l) \times (b_h + b_l) + a_l \times b_l, \\ c_l &= a_l \times b_l + 9 \times a_h \times b_h. \end{aligned} \tag{13}$$

The computations of c_h and c_l use a compact construction, which is the extension of the work in [27].

We design components *SubfieldAdder*, *SubfieldMultiplier*, and *RegshiftA*.

SubfieldAdder. It computes $a_h + a_l$ and $sa1 = b_h + b_l$ in $GF(2^4)$, where a_h, a_l, b_h, and b_l are elements in $GF(2^4)$.

SubfieldMultiplier. It computes $sa0 * sa1$, $a_h * b_h$, and $a_l * b_l$ in $GF(2^4)$, where a_h, a_l, b_h, b_l, $sa0$, and $sa1$ are elements in $GF(2^4)$.

RegshiftA. It performs the computation of right shift and a bit addition.

We adapt four *SubfieldAdder*s, three *SubfieldMultiplier*s, and *RegshiftA*, where *SubfieldAdder* and *SubfieldMultiplier* compute additions and multiplications in $GF(2^4)$, respectively.

*SubfieldAdder*0 and *SubfieldAdder*1 are used to compute

$$\begin{aligned} sa0 &= a_h + a_l, \\ sa1 &= b_h + b_l, \end{aligned} \tag{14}$$

respectively.

*SubfieldMultiplier*0, *SubfieldMultiplier*1, and *SubfieldMultiplier*2 are used to compute

$$\begin{aligned} sm0 &= sa0 * sa1, \\ sm1 &= a_h * b_h, \\ sm2 &= a_l * b_l, \end{aligned} \tag{15}$$

respectively.

RegshiftA is used to compute a right shift and a bit addition:

$$\begin{aligned} &rsa(sm1_3, sm1_2, sm1_1, sm1_0) \longrightarrow \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1), \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1) \longrightarrow \\ &rsa(sm1_0, sm1_3, sm1_2, sm1_1 + sm1_0). \end{aligned} \tag{16}$$

*SubfieldAdder*2 and *SubfieldAdder*3 are used to compute

$$\begin{aligned} c_h &= sm0 + sm2, \\ c_l &= rsa + sm2, \end{aligned} \tag{17}$$

respectively.

The multiplication has been computed.

3.3. A Parallel Solving System of Linear Equations Based on Gauss-Jordan Eliminations. We propose a parallel solving system of linear equations based on Gauss-Jordan eliminations, which is the extension of the work in [28]. We give a straightforward description of the proposed algorithm of the parallel variant of Gauss-Jordan elimination in Algorithm 1, where *operation*(i) stands for operation performed in the ith iteration, and $i = 0, 1, \ldots, 12$. The optimized Gauss-Jordan elimination with 13 iterations consists of pivoting, inversion, normalization, and elimination in each iteration.

We enhance the algorithm in four directions. First, multiplication is computed by invoking efficient multipliers designed in Section 3.2. Second, we adopt fast inverter described in Section 3.1. Third, inversion, normalization, and elimination are designed to perform simultaneously. Fourth, during the elimination in the ith iteration, we simultaneously choose the right pivot for the next iteration; namely, if

```
(1)  var
(2)    i: Integer;
(3)  begin
(4)    i := 0;
(5)    Pivoting(i = 0);
(6)    repeat
(7)      inversion(i), Normalization(i), Elimination(i);
(8)      Pivoting(i + 1);
(9)      i := i + 1;
(10)   until i = 13
(11) end.
```

Algorithm 1: Solving a system of linear equations $Ax = b$ with 13 iterations, where A is a 13×13 matrix.

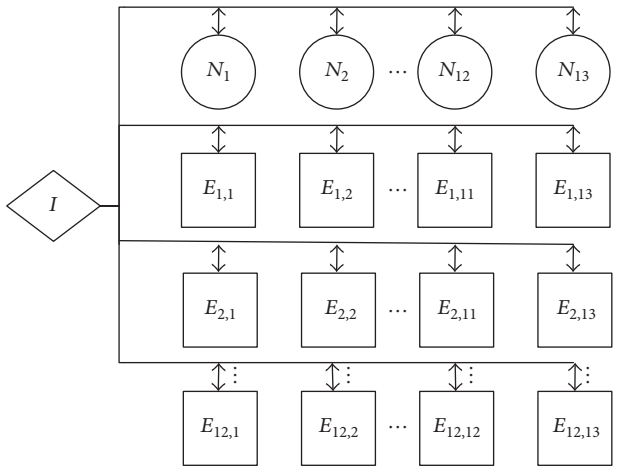

Figure 2: The proposed architecture for parallel solving system of linear equations with matrix size 13×13.

element $a_{i+1,i+1}$ of the next iteration is zero, we swap the $(i+1)$th row with another jth row with the nonzero element a_{ji}, where $i, j = 0, 1, \ldots, 12$. The difference from usual Gauss-Jordan elimination is that the usual Gauss-Jordan elimination chooses the pivot after the elimination, while we perform the pivoting during the elimination. In other words, at the end of each iteration, by judging the computational results in this iteration, we can decide the right pivoting for the next iteration. By integrating these optimizations, it takes only one clock cycle to perform one iteration.

The architecture for solving systems of linear equations in $GF((2^4)^2)$ is depicted in Figure 2 with matrix size 13×13. There exist three kinds of cells in the architecture, namely, I, N_l, and E_{kl}, where $k = 1, 2, \ldots, 12$ and $l = 1, 2, \ldots, 13$. The I cell is for fast inversion. As described in Section 3.1, two binary trees are included in the I cell for computed inversion. The N_l cells are for normalization. And the E_{kl} cells are for elimination. The architecture consists of one I cell, 13 N_l cells, and 156 E_{lk} cells.

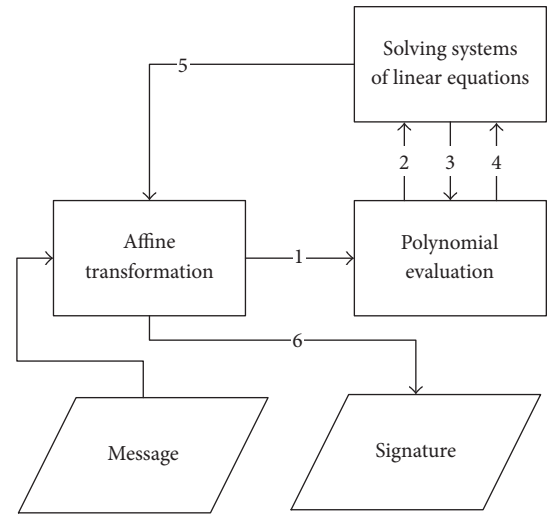

Figure 3: The flowchart of implementations of Rainbow scheme.

4. Efficient Implementation and Performance Evaluation

Rainbow(17, 13, 13) is computed via invoking affine transformation, polynomial evaluation, and solving systems of linear equations in $GF((2^4)^2)$. We depict the flowchart of implementations of Rainbow(17, 13, 13) in Figure 3:

(1) Compute the first affine transformation L_1 via invoking matrix-vector multiplication and vector addition.

(2) Evaluate the first 13 multivariate polynomials $f_0, f_1, \ldots, f_{12}$ on the first layer of central map transformation F.

(3) Solve the first systems of linear equations with matrix size 13×13 of central map transformation F.

(4) Evaluate the second 13 multivariate polynomials $f_{13}, f_{14}, \ldots, f_{25}$ on the second layer of central map transformation F.

(5) Solve the second systems of linear equations with matrix size 13×13 of central map transformation F.

(6) Compute the second affine transformation L_2 via invoking matrix-vector multiplication and vector addition.

In order to prove that the designs of Rainbow(17, 13, 13) are efficient on hardware, Hardware Description Language (Verilog HDL) code for modeling the designs has been implemented on ASICs. We implement our design in $GF((2^4)^2)$ on TSMC-0.18 μm standard cell CMOS ASICs. We use Synopsys Design Vision, which is a GUI for Synopsys Design Compiler tools. The map effort is set to medium. We present the experimental results in Tables 2 and 3, which are extracted after place and route.

Tables 2 and 3 show that Rainbow implementation includes two affine transformations with matrix sizes 26×26 and 43×43, respectively, and 26 MQ polynomial evaluations and solving two systems of linear equations with matrix size

Table 2: Implementation Results of rainbow scheme.

Signature scheme	Message size	Signature size	Time frequency	Clock cycle	Executing time	Gate equivalents
Rainbow(17, 13, 13)	26 bytes	43 bytes	50 MHz	242	4.9 us	30000

Table 3: Executing time of the implementation in clock cycles.

Steps	Components	Clock cycles
(1)	L_1^{-1} transformation	28
(2)	The first round of 13 polynomial evaluations	65
(3)	The first round of solving system of linear equations	13
(4)	The second round of 13 polynomial evaluations	78
(5)	The second round of solving system of linear equations	13
(6)	L_2^{-1} transformation	45
	Total	242

Table 4: Comparison on public key cryptographic systems.

Signature scheme	Clock cycle	Executing time (us)	Gate equivalents	Time-area product*
RSA [5]	813	1790	107000	12.00
ECC [6]	1699	9.5	138000	32.30
UOV [26]	2300	27.7	56700	17.97
amTTS [26]	312	3.9	61600	2.65
enTTS [27]	10267	170	1000	1.42
Rainbow [28]	198	3.96	150000	4.10
SFLASH [29]	3200	1600	25900	11.42
This work	242	4.84	30000	1

*The time-area (clock cycle-gate equivalent) product of our implementations is normalized to 1.

13×13. Table 3 summarizes the performance of our implementation of Rainbow signature measured in clock cycles, which shows that our design takes only 242 clock cycles to generate a Rainbow signature. In other words, our implementation takes 4840 ns to generate a Rainbow signature with the frequency of 50 MHz. Among all of the operations, MQ polynomial evaluation occupies most of the executing time.

5. Comparisons with Other Implementations

The works in [5, 6, 26–29] are believed to be the latest RSA, ECC, and multivariate public key cryptographic systems on hardware, respectively. We compare our design with these systems, which is depicted in Table 4. Comparison results show that our design is more efficient than the related implementations.

Besides, Rainbow implementation of the work in [28] is believed to be the fastest multivariate implementation, and Rainbow implementation of the work in [27] is believed to be the smallest multivariate implementation. Thus, the implementations of the work in [28], the work in [27], and this work show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

6. Side Channel Attack Considerations

Cryptographic systems must be protected against a wide range of attacks, including side channel attacks. Side channel attack belongs to physical attack, which is any attack based on information gained from the physical implementation of cryptographic systems, rather than brute force or theoretical weaknesses in cryptographic algorithms. The underlying principle of side channel attack is that side channel information such as power consumption, electromagnetic leaks, timing information, or even sound can provide extra sources of information about secrets in cryptographic systems, for example, cryptographic keys, partial state information, full or partial plain texts, which can be exploited to break the cryptographic systems. General classes of side channel attack include timing analysis [31], power analysis [32], electromagnetic analysis [33], fault analysis [34], acoustic cryptanalysis [35], data remanence analysis [36], and row hammer analysis attacks [37].

Fault analysis attacks intend to manipulate the environmental conditions of cryptographic systems, such as voltage, clock, temperature, radiation, light, and eddy current, to generate faults during secret-related computations, for example, multiplications and inversions in a finite field, and

observe the related behavior, which may help a cryptanalyst break the cryptographic systems. Fault analysis attacks can be engineered by simply illuminating a transistor with a laser beam, which causes some bits to assume wrong values. The notion of using a fault induced during a secret-related computation to guess the secret key has been practically observed in implementations of the RSA that use the Chinese remainder theorem [38, 39]. A general fault analysis attack on schemes of MPKC is proposed in [40]. The work in [40] has attacked partial secret keys from affine transformations of the multivariate public key cryptographic schemes.

Power analysis attack can provide detailed information by observing the power consumption of cryptographic systems, which is roughly categorized into Simple Power Analysis (SPA) [41] and Differential Power Analysis (DPA) [32]. In the family of power analysis attacks, DPA is of particular interest and is a statistical test which examines a large number of power consumption signals to retrieve secret keys. A differential power analysis attack on SFLASH is proposed in [42]. The work in [42] has attacked secret keys from SHA-1 module of the SFLASH schemes. A side channel attack to enTTS has been proposed in [43], which uses differential power analysis and fault analysis to attack two affine transformations and central map transformation. The method in [43] shows that it can obtain all secret keys of enTTS.

Since the construction of Rainbow includes two affine transformations and central map transformation, such methods in [40, 42, 43] have the potential to obtain its secret keys. Thus, we discuss defending against a possible side channel attack for Rainbow and the countermeasure is described in the following:

(1) We suppose that $y(y_0, y_1, \ldots, y_{25})$ is the message and each element of y is in $GF((2^4)^2)$.

(2) We take a random vector $y'(y'_0, y'_1, \ldots, y'_{25})$; the elements of y' are in $GF((2^4)^2)$.

(3) We compute $y'' = y' + y$.

(4) We compute $\overline{y}' = Ay' + b$ and $\overline{y}'' = Ay''$, where A is a 26×26 matrix and b is a vector with size 26.

(5) We compute $\overline{y} = \overline{y}' + \overline{y}''$, which is equivalent to $\overline{y} = Ay + b$.

(6) The first affine transformation L_1 has been computed; then we take random bytes for Vinegar variables.

(7) We double check the random bytes to protect against fault analysis attacks.

(8) We compute the multivariate polynomial evaluations and solving systems of linear equations until the central map transformation is completed.

(9) $\overline{x}(\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{42})$ is the result of central map transformation; then we take two random vectors $\overline{x}'$ and $\overline{x}''$, where $\overline{x} = \overline{x}' + \overline{x}''$ and the elements are in $GF((2^4)^2)$.

(10) We compute $x' = C\overline{x}'$ and $x'' = C\overline{x}'' + d$, where C is a 43×43 matrix and b is a vector with size 43.

(11) We compute $x = x' + x''$, which is equivalent to $x = C\overline{x} + d$.

(12) $x(x_0, x_1, \ldots, x_{42})$ is the Rainbow signature of $y(y_0, y_1, \ldots, y_{25})$.

The work in [40] uses fault analysis to attack the random bytes in central map transformations; thus we double check the random bytes to protect against fault analysis attacks. The work in [42] uses differential power analysis to attack SHA-1 module; thus we take a method to protect affine transformations. However, the countermeasure mentioned above is theoretical; we should be able to implement and verify it on hardware.

7. Conclusions

In this paper, we present techniques to exploit Rainbow signature cryptographic systems on hardware meeting the requirements of efficient high-performance applications. We propose a general architecture for efficient hardware implementations of Rainbow and enhance our design in three directions. First, we present a fast inversion in $GF((2^4)^2)$ based on binary trees. Second, we present an efficient multiplication in $GF((2^4)^2)$ based on compact construction. Third, we present a parallel solving system of linear equations in $GF((2^4)^2)$ based on Gauss-Jordan elimination. Via further other minor optimizations and by integrating the major improvement above, we implement our design in $GF((2^4)^2)$ on TSMC-0.18 μm standard cell CMOS ASICs. We use Synopsys Design Vision and the map effort is set to medium. Our design can be generalized with minor modifications that also support FPGAs.

The experimental results show that Rainbow implementation includes two affine transformations with matrix sizes 26×26 and 43×43, respectively, and 26 MQ polynomial evaluations and solving two systems of linear equations with matrix size 13×13. Our implementation takes 4840 ns and 242 clock cycles to generate a Rainbow signature with the frequency of 50 MHz. Among all of the operations, MQ polynomial evaluation occupies most of the executing time. Comparison results show that our design is more efficient than the related implementations.

Moreover, the implementations of a fast Rainbow, a small Rainbow, and this work show that Rainbow has a good performance on hardware and is a better candidate than RSA and elliptic curves under quantum computer attacks.

Besides, Rainbow implementations must be protected against a wide range of attacks, including side channel attacks. We discuss defending against a possible side channel attack for Rainbow and we present countermeasures against fault analysis and differential power analysis attack.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author acknowledges Shenzhen Science and Technology Program under Grants no. JCYJ20170306144219159 and no.

JCYJ20160428092427867; Science and Technology Program of Shenzhen Polytechnic (no. 601722K20018); and Special Funds for Shenzhen Strategic Emerging Industries and Future Industrial Development (no. 20170502142224600).

References

[1] W. Diffie, W. Diffie, and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 26, no. 1, pp. 96–99, 1983.

[3] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[4] V. S. Miller, "Use of elliptic curves in cryptography," in *Proceedings of the International Cryptology Conference (CRYPTO 85)*, 426, 417 pages, Springer-Verlag, Berlin, Germany, 1985.

[5] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 7, pp. 3101–3109, 2011.

[6] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 217–225, 2013.

[7] A. Cilardo, A. Mazzeo, L. Romano, and G. P. Saggese, "Exploring the design-space for FPGA-based implementation of RSA," *Microprocessors and Microsystems*, vol. 28, no. 4, pp. 183–191, 2004.

[8] O. Nibouche, M. Nibouche, A. Bouridane, and A. Belatreche, "Fast architectures for FPGA-based implementation of RSA encryption algorithm," in *Proceedings of the IEEE International Conference on Field-Programmable Technology (FPT 2005)*, pp. 271–278, Washington, DC, USA, December 2004.

[9] Pund S. M., "Implementation of RSA algorithm using mersenne prime," *International Journal of Networking and Parallel Computing*, vol. 1, pp. 33–41, 2014.

[10] Q. A. Al-Haija, M. Smadi, M. Al-Ja'fari, and A. Al-Shua'ibi, "Efficient FPGA implementation of RSA coprocessor using scalable modules," in *Proceedings of the International Symposium on Emerging Inter-networks, Communication and Mobility (EICM 2014)*, pp. 647–654, Elsevier, Amsterdam, Netherlands, 2014.

[11] K. C. C. Loi and S.-B. Ko, "High performance scalable elliptic curve cryptosystem processor for Koblitz curves," *Microprocessors and Microsystems*, vol. 37, no. 4-5, pp. 394–406, 2013.

[12] K. Sakiyama, N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p)," *International Journal of Electronics*, vol. 94, no. 5, pp. 501–514, 2007.

[13] M. N. Hassan and M. Benaissa, "Small footprint implementations of scalable ECC point multiplication on FPGA," in *Proceedings of the 2010 IEEE International Conference on Communications, ICC 2010*, pp. 1–4, Washington, DC, USA, May 2010.

[14] M. Varchola, T. Güneysu, and O. Mischke, "MicroECC: A lightweight reconfigurable elliptic curve crypto-processor," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (Reconfig 2011)*, pp. 204–210, Washington, DC, USA, 2013.

[15] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[16] J. Ding, J. E. Gower, and D. S. Schmidt, *Multivariate Public Key Cryptosystems*, Springer, Berlin, Germany, 2006.

[17] D. S. Johnson, "The NP-completeness column: an ongoing guide," *Journal of Algorithms*, vol. 4, no. 1, pp. 87–100, 1983.

[18] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced oil and vinegar signature schemes," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 99)*, vol. 1999, pp. 206–222, Springer, Berlin, Germany.

[19] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS 2005)*, pp. 164–175, Springer, Berlin, Germany, 2005.

[20] A. Petzoldt, S. Bulygin, and J. Buchmann, "Selecting parameters for the rainbow signature scheme," in *Post-quantum cryptography*, vol. 6061 of *Lecture Notes in Comput. Sci.*, pp. 218–240, Springer, Berlin, 2010.

[21] B. Y. Yang and J. M. Chen, "Building secure tame-like multivariate public-key cryptosystems: the new TTS," in *Proceedings of the Australasian Conference on Information Security and Privacy (ACISP 2005)*, pp. 518–531, Springer, Berlin, Germany, 2005.

[22] E. Thomae and C. Wolf, "Cryptanalysis of enhanced TTS, STS and all its variants, or: why cross-terms are important," in *Proceedings of the International Conference on Cryptology in Africa (Africacrypt 2012)*, pp. 188–202, Springer, Berlin, Germany, 2012.

[23] T. Matsumoto and H. Imai, "Public quadratic polynomial-tuples for efficient signature-verification and message-encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 88)*, pp. 419–453, Springer, Berlin, Germany, 1988.

[24] E. Thomae and C. Wolf, "Solving underdetermined systems of multivariate quadratic equations revisited," in *Proceedings of the International Conference on Practice and Theory of Public-Key Cryptography (PKC 2012)*, 171, pp. Berlin, Germany–156, Springer, 2012.

[25] T. Moh, "A public key system with signature and master key functions," *Communications in Algebra*, vol. 27, no. 5, pp. 2207–2222, 1999.

[26] A. Bogdanov, T. Eisenbarth, A. Rupp et al., "Time-area optimized public-key engines: MQ-cryptosystems as replacement for elliptic curves?" in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2008)*, pp. 45–61, Springer, Berlin, Germany, 2008.

[27] H. Yi and S. Tang, "Very small FPGA processor for multivariate signatures," *The Computer Journal*, vol. 59, no. 7, pp. 1091–1101, 2016.

[28] S. Tang, H. Yi, J. Ding, H. Chen, and G. Chen, "High-speed hardware implementation of rainbow signature on FPGAs," in *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto 2011)*, pp. 228–243, Springer, Berlin, Germany, 2011.

[29] B. Yang, C. Cheng, B. Chen, and J. Chen, "Implementing minimized multivariate PKC on low-resource embedded systems," in *Proceedings of the Security in Pervasive Computing, 3rd International Conference (SPC 2006)*, vol. 3934, pp. 73–88, Springer, Berlin, Germany, 2006.

[30] H. Yi, S. Tang, and R. Vemuri, "Fast inversions in small finite fields by using binary trees," *The Computer Journal*, vol. 59, no. 7, pp. 1102–1112, 2016.

[31] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *Proceedings of the International Cryptology Conference on Advances in Cryptology (CRYPTO 96)*, pp. 104–113, Springer, Berlin, Germany, 1996.

[32] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the International Cryptology Conference on Advances in Cryptology (CRYPTO 99)*, pp. 388–397, Springer, Berlin , Germany, 1999.

[33] J. J. Quisquater and D. Samyde, "ElectroMagnetic analysis: measures and countermeasures for smart cards," in *Proceedings of the International Conference on Research in Smart Cards (E-Smart 2001)*, pp. 200–210, Springer, Berlin, Germany, 2001.

[34] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2002)*, pp. 2–12, Springer, Berlin, Germany, 2003.

[35] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Proceedings of the International Cryptology Conference (CRYPTO 2014)*, pp. 17–21, Springer, Berlin, Germany, 2014.

[36] S. Skorobogatov, "Data remanence in flash memory devices," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2005)*, pp. 339–353, Springer, Berlin, Germany, 2005.

[37] D.-H. Kim, P. J. Nair, and M. K. Qureshi, "Architectural support for mitigating row hammering in DRAM memories," *Computer Architecture Letters*, vol. 14, pp. 9–12, 2015.

[38] M. Joye, A. K. Lenstra, and J.-J. Quisquater, "Chinese remaindering based cryptosystems in the presence of faults," *Journal of Cryptology*, vol. 12, no. 4, pp. 241–245, 1999.

[39] D. M. D. Boneh and R. J. Lipton, "On the importance of eliminating errors in cryptographic computations," *Journal of Cryptology*, vol. 14, pp. 101–119, 1999.

[40] Y. Hashimoto, T. Takagi, and K. Sakurai, "General fault attacks on multivariate public key cryptosystems," in *Post-quantum cryptography*, vol. 7071 of *Lecture Notes in Comput. Sci.*, pp. 1–18, Springer, Heidelberg, 2011.

[41] R. Mayer-Sommer, "Smartly analyzing the simplicity and the power of simple power analysis on smartcards," in *Proceedings of the Conference on Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 78–92, pringer, Berlin, Germany, 2000.

[42] K. Okeya, T. Takagi, and C. Vuillaume, "On the importance of protecting δ in SFLASH against side channel attacks," in *Proceedings of the International Conference on Coding and Computing (ITCC 2004)*, pp. 560–568, Washington, DC, USA, 2004.

[43] H. Yi and W. Li, "On the Importance of Checking Multivariate Public Key Cryptography for Side-Channel Attacks: The Case of enTTS Scheme," *The Computer Journal*, pp. 1–13, 2017.