Supporting Documentation of RaCoSS (Random Code-based Signature Scheme)

Partha Sarathi Roy[1], Rui Xu[1], Kazuhide Fukushima[1], Shinsaku Kiyomoto[1], Kirill Morozov[2], and Tsuyoshi Takagi[3]

[1]KDDI Research, Inc.
[2]University of North Texas
[3]The University of Tokyo

November 29, 2017

Contents

Chapter 1

Complete Specification of Algorithm

The signature scheme RaCoSS consists of three algorithms, viz., *Key Generation*, *Signature Generation* and *Signature Verification*.

1.1 Notations

We use the following notations throughout the chapter.

- $\lfloor r \rfloor$: Maximum integer that does not exceed r.
- $\lceil r \rceil$: Minimum integer that is not smaller than r.
- S^t: Transposed matrix of S.
- $||$: Concatenation of byte strings.
- $(x_1, x_2, \ldots) \overset{l\text{-bits}}{\leftarrow} x$: Separate a bit string with n bit x to bit strings x_1, x_2, ..., $x_{\lfloor n/l \rfloor}$.
- $\mathsf{0x}$: Prefix for hexadecimal digits.
- $\mathsf{wt}(c)$: Hamming weight of bit string c.
- $\mathsf{int}(x)$: Convert bit string x to integer using base-2 conversion.
- $\mathsf{byte}(x)$: Convert byte string x with n bit to a byte string with $\lceil n/8 \rceil$ bits. It is padded with zero bits if n is not multiple of 8.
- $\mathsf{rand}(x)$: Output random integers from the uniform distribution in $[0, x-1]$.

- $\mathsf{encode}(M)$: Output a byte string for $m \times n$ matrix M. The byte string of the output is the concatenation of byte (n-bit string of the first row of M), byte(n-bits string of the second row of M), ..., and byte(n-bits string of the m-th row of M).

All of the additions and multiplications are modulus two operations. That is,

$$0+0=0, \quad 0+1=1, \quad 1+0=1, \quad 1+1=0,$$
$$0 \cdot 0 = 0 \quad 0 \cdot 1 = 0, \quad 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

1.2 System Parameters

- positive integer n (length of codeword).
- positive integer k such that $k < n$ (dimension of the code).
- positive integer ω such that $0 < \omega < \sqrt{n}$ (minimum distance of the code).
- positive real constant γ such that $\gamma\omega \geq 1$.
- $(n-k) \times n$ binary matrix H whose elements follow Bernoulli distribution $B(1/2)$; thus, it is 1 with probability 1/2 and 0 with probability 1/2.

1.3 Selection of Parameter Sizes and Hash Function for RaCoSS

We use the following parameters and weight restricted hash function to satisfy the security requirements of category 1.

Parameters $n = 2,400$, $k = 2,060$, $\omega = 48$, $\gamma = 0.07$, and H is a $340 \times 2,400$ binary matrix. The matrix H is defined as a two dimentional array in `sign_tab.h` in the reference implementation.

Weight Restricted Hash Function: We show the construction of weight restricted hash function wrhf in RaCoSS. It takes message m as a byte string, integer w and an integer n and output a bit string with n bits and Hamming weight w. Algorithm 1 shows the pseudocode of wrhf. The function calculates the positions of bit-1 using SHA3-512 hash function. It output $\perp$ in the case where it cannot specify w positions of bit-1; however, the probability is less than 2^{-10000} and negligible.

1.4 Key Generation

The key generation algorithm takes security parameter as input and outputs private key S^t and public key T. Algorithm 2 shows the pseudocode of the key generation process.

Algorithm 1 Weight Restricted Hash Function

Input: Message m as a byte string, integer w ($w < 8$), integer n ($2^{10} < n \leq 2^{20}$)
Output: Bit string with n bits and Hamming weight w.

$l \leftarrow \lceil \log_2 n \rceil$.
$I \leftarrow \emptyset$.
$S \leftarrow \{m, \texttt{0x00}||m, \texttt{0x01}||m, \texttt{0x02}||m, \ldots, \texttt{0xFF}||m\}$
for $s \in S$ **do**
  $x \leftarrow$ SHA3-512(s)
  $(x_1, x_2, \ldots, x_{\lfloor 512/l \rfloor}) \xleftarrow{l\text{-bits}} x$
  for $i = 1$ to $\lfloor 512/l \rfloor$ **do**
    $y_i \leftarrow \mathsf{int}(x_i)$.
    if $y_i < n$ **then**
      $I \leftarrow I \cup \{y_i\}$
    end if
    if $|I| = w$ **then**
      break
    end if
  end for
end for
if $|I| < w$ **then**
  Output $\perp$
else
  Output a bit string with n bits where I_i-th bit is 1 ($I_i \in I, 0 \leq i \leq w-1$) and other bits are 0.
end if

- Input: none.
- Output: private key S^t as a binary matrix and public key T as a binary matrix.

1. Generate an $n \times n$ binary matrix S^t. Each element follows Bernoulli distribution $B(\omega/n)$; thus, it is 1 with probability ω/n and 0 with probability $1 - \omega/n$.
2. Calculate an $(n-k) \times n$ binary matrix T as $T = HS^t$.
3. Output private key S^t and public key T.

1.5 Signature Generation

Signature generation algorithm takes message m and private key S^t as input and outputs signature (z, c). Algorithm 3 shows the pseudocode of the signature generation process.

Algorithm 2 Key Generation

Input: None
Output: Private key S^t and public key T.

```
for i = 1 to n do
    for j = 1 to n do
        r ← 0
        f ← rand(n)
        if f < ω then
            r ← 1
        end if
        S^t[i][j] ← r
    end for
end for
for i = 1 to n − k do
    for j = 1 to n do
        s ← 0
        for l = 1 to n do
            s ← s + H[i][l]S^t[l][j]
        end for
        T[i][j] ← s
    end for
end for
Output (S^t, T)
```

- Input: message m as a byte string and private key S^t as a binary matrix.
- Output: (z, c) where z and c are binary vectors.

1. Calculate real value ρ as
$$\rho = \left\lfloor \frac{1}{2}\left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right) \cdot 1000 \right\rfloor / 1000.$$
2. Generate an n-column binary vector y. Each element follows Bernoulli distribution $B(\rho)$; thus, it is 1 with probability ρ and 0 with probability $1 - \rho$.
3. Calculate n-column binary vector v as $v = Hy$.
4. Calculate byte string b as $b = \mathsf{byte}(v)||m||\mathsf{encode}(H)$. Vector v shall be considered as bit string v whose i-th bit is $v[i]$.
5. Calculate bit string c as $c = \mathsf{wrhf}(b, \lfloor \gamma\omega \rfloor, n)$.
6. Calculate n-column binary vector z as $z = S^t c + y$. Bit string c shall be considered as vector c where $c[i]$ is the i-th bit of c.
7. Output (z, c) as a signature.

Algorithm 3 Signature Generation

Input: Message m and private key S^t
Output: Signature (z, c).

$\rho \leftarrow \left\lfloor \frac{1}{2} \left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right) \cdot 1000 \right\rfloor /1000$
for $i = 1$ to n **do**
  $r \leftarrow 0$
  $f \leftarrow \mathsf{rand}(1000)$
  if $f < 1000\rho$ **then**
    $r \leftarrow 1$
  end if
  $y[i] \leftarrow r$
end for
for $i = 1$ to $n - k$ **do**
  $s \leftarrow 0$
  for $j = 1$ to n **do**
    $s \leftarrow s + H[i][j]y[j]$
  end for
  $v[i] \leftarrow s$
end for
$b \leftarrow \mathsf{byte}(v)||m||\mathsf{encode}(H)$
$c \leftarrow \mathsf{wrhf}(b, \lfloor \gamma\omega \rfloor, n)$
for $i = 1$ to n **do**
  $s \leftarrow 0$
  for $j = 1$ to n **do**
    $s \leftarrow s + S^t[i][j]c[j]$
  end for
  $z[i] \leftarrow s + y[i]$
end for
Output (z, c)

1.6 Signature Verification

The signature verification algorithm takes message m, signature (z, c) and public key T as input and outputs zero if the signature is valid or outputs one otherwise. Algorithm 4 shows the pseudocode of the signature verification process.

- Input: message m as a byte string, signature (z, c) where where z and c are binary vectors, and public key T where T is a binary matrix.
- Output: integer 0 if the signature is valid or 1 otherwise.

1. Calculate real value ρ as

$$\rho = \left\lfloor \frac{1}{2} \left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right) \cdot 1000 \right\rfloor /1000.$$

2. If $\mathsf{wt}(z) \geq 12n\rho(1-\rho)$ then 1. Otherwise, goto Step 2. Note that z shall be considered as bit string z whose i-th bit is $z[i]$.

3. Calculate an n-column binary vector v_1 as $v_1 = Hz$.

4. Calculate an n-column binary vector v_2 as $v_2 = Tc$.

5. Calculate an n-column binary vector v as $v = v_1 + v_2$.

6. Calculate bit string b as $b = \mathsf{byte}(v)||m||\mathsf{encode}(H)$. Vector v shall be considered as bit string v whose i-th bit is $v[i]$.

7. Calculate a bit string c' as $c' = \mathsf{wrhf}(b, \lfloor \gamma\omega \rfloor, n)$. Output 0 if $c' = c$ and output 1 otherwise. Bit string c' shall be considered as vector c' where $c'[i]$ is the i-th bit of c'.

Algorithm 4 Signature Verification

Input: Message m, signature (z, c), and public key T
Output: 0 if the signature is valid or 1 otherwise.

$\rho \leftarrow \left\lfloor \frac{1}{2}\left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right) \cdot 1000 \right\rfloor / 1000$
if $\mathsf{wt}(\mathsf{z}) \geq 12\rho(1-\rho)$ **then**
    Output 1
end if
for $i = 1$ to $n - k$ **do**
    $s \leftarrow 0$
    for $j = 1$ to n **do**
        $s \leftarrow s + H[i][j]z[j]$
    end for
    $v_1[i] \leftarrow s$
end for
for $i = 1$ to $n - k$ **do**
    $s \leftarrow 0$
    for $j = 1$ to n **do**
        $s \leftarrow s + T[i][j]c[j]$
    end for
    $v_2[i] \leftarrow s$
end for
for $i = 1$ to $n - k$ **do**
    $v[i] \leftarrow v_1[i] + v_2[i]$
end for
$b \leftarrow \mathsf{byte}(v)||m||\mathsf{encode}(H)$
$c' \leftarrow \mathsf{wrhf}(b, \lfloor \gamma\omega \rfloor, n)$
for $i = 1$ to n **do**
    if $c'[i] \neq c[i]$ **then**
        Output 1
    end if
end for
Output 0

Chapter 2

Performance Analysis

We implemented RaCoSS on a PC with Intel Core i7-4770K CPU using Microsoft Visual C++ 2010 and evaluated its performance. We describe information about evaluation environments in Table 2.1. Two version of implementations (reference and optimized implementations) are written using ANSI C. The reference implementation is to promote understanding of how RaCoSS is implemented. The optimized implementation is intended to demonstrate the performance of RaCoSS.

The execution times are shown in Table 2.2. The reference implementation completes key generation, signature generation and signature verification in 7,090 milliseconds, 17.4 milliseconds, and 9.10 milliseconds, respectively. The optimized implementation completes key generation, signature generation and signature verification in 243 milliseconds, 7.07 milliseconds, and 6.87 milliseconds, respectively.

Table 2.3 shows the size of the public key, private key, and signature. The size of the private key, public key, and signature based on naive data expression are 703 kilobytes, 99.6 kilobytes, and 0.586 kilobytes, respectively. Note that one kilobyte equals 1024 bytes and one byte equals eight bits. At the time of calculation of private key size, we have considered only the secret entity, i.e., S since system parameters are embedded in the implementation.

We can reduce the size of private key and signature using a classical data compression technique. Private key S has n^2 elements but only around $n\omega$ is 1 and other elements are 0. We can specify the position of a bit-1 in each row with $\lceil \log_2 n \rceil$ bits using an index. Thus, we can reduce the size of the secret key to $n\omega\lceil \log_2 n \rceil = \Theta(n\sqrt{n}\log n)$ bits. The signature size can be reduce to $n + \lfloor \gamma\omega \rfloor \lceil \log_2 n \rceil$ bits using the same technique. However, we need to specify the fixed size of the private key, public key, and signature in the symbolic constants `CRYPTO_SECRETKEYBYTES`, `CRYPTO_PUBLICKEYBYTES`, and `CRYPTO_BYTES`, respectively, in our reference and optimized implementation. Thus, we use the naive data expression in these implementations.

Table 2.1: Evaluation Environments for Software Implementations

CPU	Intel Core i7-4770K CPU (3.50GHz)
Memory	32 Gigabyte
OS	Microsoft Windows 8.1 64bit
Compiler	Microsoft Visual C++ 2010

Table 2.2: Execution Time of Software Implementation

Algorithm	Reference implementation	Optimized implementation
Key generation	7,090 ms	240 ms
Signature generation	17.4 ms	6.48 ms
Signature verification	9.10 ms	6.09 ms

Table 2.3: Data Size of Software Implementation

Algorithm	Reference implementation	Optimized implementation
Public key	99.6 KB	99.6 KB
Private key	703 KB	703 KB
Signature	0.586 KB	0.586 KB

Chapter 3

Known Answer Test Values

The known answer test repeats key generation, signature generation, and signature verification 100 times. We assume an implementation is correct if its all outputs are identical to the known answers for given inputs. The given input parameters are written in the REQUEST file (`PQCsignKAT_720000.req`) and all the known answers are written in the RESPONSE file (`RQCsignKAT_720000.rsp`).

The amount of description of REQUEST and RESPONSE files are large. Thus, we provide table 3.1 and 3.2 shows the format of the REQUEST and RESPONSE files, respectively. The source code `PQCgenKAT_sign.c` in our reference and optimized implementations generates the same REQUEST and RESPONSE files.

Table 3.1: Format for REQUEST File

Symbol	Description
`count =`	Test number (0 to 99)
`seed =`	Seed (48-byte hexadecimal digits) for random number generator
`mlen =`	Byte length of input message
`msg =`	Input message (hexadecimal digits)
`pk =`	Blank
`sk =`	Blank
`smlen =`	Blank
`sm =`	Blank

Table 3.2: Format for RESPONSE File

Symbol	Description
`count =`	Test number (0 to 99)
`seed =`	Seed (48-byte hexadecimal digits) for random number generator
`mlen =`	Byte length of the input message
`msg =`	Input message (hexadecimal digits)
`pk =`	Public key (hexadecimal digits)
`sk =`	Secret key (hexadecimal digits)
`smlen =`	Byte length of signature
`sm =`	Signature (hexadecimal digits)

Chapter 4

Expected Security Strength

A *digital signature scheme* Σ = (**Key Generation**, **Signature Generation**, **Signature Verification**) is consists of three algorithms.

Key Generation: Takes a security parameter 1^λ and outputs a pair of keys (*private key*, *public key*).

Signature Generation: On input a message m and the private key, outputs a signature σ on the message m.

Signature Verification: Takes as input the public key, a message m and a signature σ, and outputs a bit denoting accept or reject, respectively.

The standard security notion for a signature scheme is *Strong Existential Unforgeability against Chosen Message Attack* (SEUF-CMA): The forger gets a public key from a challenger who generates a key pair (*private key*, *public key*). The forger can query a signing oracle on polynomially many messages m_i hereby obtaining signatures σ_i. The forger can also issue a hash query and obtains its hash value. We say that the forger wins the SEUF-CMA game, if the forger successfully outputs a pair (m^*, σ^*), where σ^* is a valid signature of a message m^* under the private key with the restriction that (m^*, σ^*) has never appeared in the query phase.

4.1 Notations

We use the following notations throughout the chapter 4 and 5.

- $\lfloor r \rfloor$: Maximum integer that does not exceed r.
- $\lceil r \rceil$: Minimum integer that is not smaller than r.
- $\{0,1\}^*$: bit string of arbitrary length.
- $B(p)$: Bernoulli distribution with parameter p.

- $B(n,p)$: Binomial distribution with parameter n and p.
- S^t: Transposed matrix of S.
- $\mathbb{F}_2$: Galois field of two elements.
- $\mathbb{F}_2^n$: Vector with n elements over $\mathbb{F}_2$.
- $\mathbb{F}_2^{m\times n}$: Matrix with m rows and n columns whose elements are in $\mathbb{F}_2$.
- $\mathsf{wt}(c)$: Hamming weight of bit string c.
- $||$: Concatenation of bit strings.
- encode: Output a bit string for the input $m \times n$ matrix M. The bit string of output is the concatenation of the n-bit string of the first row of M, the n-bit string of the second row of M, …, and the bit string of the m-th row of M.

4.2 Security Assumption and Required Definitions

An $[n,k]$ Linear Code $\mathcal{C}$ is a subspace of dimension k of the vector space $\mathbb{F}_2^n$. A linear code can be described by its Parity-Check Matrix H. The parity-check matrix describes the code as follows:

$$\forall x \in \mathbb{F}_2^n, \quad x \in \mathcal{C} \iff Hx^t = 0.$$

The product Hx^t is known as *Syndrome* of the vector x.

Definition 1. ***Gilbert-Varshamov Bound:*** *Let $\mathcal{C}$ be an $[n,k]$ linear code over $\mathbb{F}_2$. The Gilbert-Varshamov (GV) Distance is the largest integer d such that*

$$\sum_{i=0}^{d-1} \binom{n}{i} \leq 2^{n-k}.$$

Definition 2. ***Null Syndrome Decoding Problem (NSDP) [2]:***
input: a matrix $H \in \mathbb{F}_2^{(n-k)\times n}$ and a non negative integer w.
output: a codeword x of weight w', where $0 < w' \leq w$, such that $Hx^t = 0$.

It is also proved that there exists the unique solution to SDP if the weight w is below the GV Bound.

Lemma 1. ***Left-over hash lemma [11, 7]:*** *Let D be a distribution over $\mathbb{F}_2^n$ with* min-entropy *E. For $\epsilon > 0$ and $r \leq E - 2\log(\frac{1}{\epsilon}) - \mathcal{O}(1)$, the statistical distance between the distribution of (H, Hs^t), where H is randomly sampled from the uniform distribution on $\mathbb{F}_2^{r\times n}$ and $s \in \mathbb{F}_2^n$ is drawn from distribution D, and the uniform distribution over $\mathbb{F}_2^{r\times n} \times \mathbb{F}_2^n$ is at most ϵ.*

In particular, if $\omega < n$ is an integer such that $r \leq E - 2\lambda - \mathcal{O}(1)$ and D is the Binomial distribution $B(n, \omega/n)$ over $\mathbb{F}_2^n$ (i.e., D has min-entropy E), then the statistical distance between the distribution of (H, Hs^t) and the uniform distribution over $\mathbb{F}_2^{r \times n} \times \mathbb{F}_2^n$ is at most $2^{-\lambda}$.

Collision Resistant Hash Function To pick a hash function with security parameter n and k, we choose a matrix $H \in \mathbb{F}_2^{(n-k) \times n}$.

Evaluating a Hash Function [2] Given a matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ and a string $x \in \{0,1\}^n$, compute

$$\hat{h}_H(x) = Hx^t.$$

For the sake of completeness, proof of security of the hash function is provided in Theorem 1.

Theorem 1. *Suppose that given a uniformly chosen matrix, $H \in \mathbb{F}_2^{(n-k) \times n}$, as above it is possible to find in (expected) $T(k,n)$ -time $x \neq y \in \{x : x \in \{0,1\}^n, \mathsf{wt}(x) \leq i,\ 2i \leq w\}$, so that $Hx^t = Hy^t$. Then it is possible to solve a given instance (H, w) of NSDP with probability*

$$P_{NSDP} = 2^{-(n-k)} \sum_{i=1}^{w} \binom{n-k}{2i}$$

in (expected) $T(k,n)$ -time.

Proof. If we can find two strings $x \neq y \in \{x : x \in \{0,1\}^n, \mathsf{wt}(x) \leq i\}$ so that $Hx^t = Hy^t$ then we have $H(x+y)^t = 0$. Since $x \neq y \in \{x : x \in \{0,1\}^n, \mathsf{wt}(x) \leq i\}$, we have $(x+y) \in \{0,1\}^n$ with weight $0 < 2i \leq w$ which constitutes a solution to NSDP for the instance (H, w) with probability $2^{-(n-k)} \sum_{i=1}^{w} \binom{n-k}{2i}$ [2]. □

4.3 Signature Scheme

RaCoSS consists of three algorithms, viz., *Key Generation*, *Signature Generation* and *Signature Verification*.

System Parameters

- positive integer n (length of codeword).
- positive integer k such that $k < n$ (dimension of the code).
- positive integer ω such that $0 < \omega < \sqrt{n}$ (minimum distance of the code).
- positive real constant γ such that $\gamma\omega \geq 1$.
- matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ whose elements follow Bernoulli distribution $B(1/2)$.
- random oracle $h : \{0,1\}^* \to \{c \in \mathbb{F}_2^n : \mathsf{wt}(c) = \lfloor \gamma\omega \rfloor\}$.

Key Generation The key generation algorithm outputs the pair of the private key S^t and public key T.

1. Generate an $n \times n$ matrix $S^t \in \mathbb{F}_2^{n \times n}$. Each element follows Bernoulli distribution $B(\omega/n)$.
2. Calculate matrix $T \in \mathbb{F}_2^{(n-k) \times n}$ as $T = HS^t$.
3. Output private key S^t and public key T.

Signature Generation Signature generation algorithm takes message m and private key S^t as input and outputs signature (z, c).

1. Calculate real value ρ as
$$\rho = \frac{1}{2}\left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right).$$
2. Generate a vector $y \in \mathbb{F}_2^n$. Each element follows Bernoulli distribution $B(\rho)$.
3. Calculate a vector $c \in \mathbb{F}_2^n$ as $c = h(Hy||m||\mathsf{encode}(H))$.
4. Calculate a vector $z \in \mathbb{F}_2^n$ as $z = S^t c + y$.
5. Output (z, c) as a signature.

Signature Verification Signature verification algorithms takes message m, public key T, signature (z, c), and predefined parameters as input. It outputs 0 if the signature is valid and outputs 1 otherwise.

1. Calculate real value ρ as
$$\rho = \frac{1}{2}\left(1 - \left(1 - \frac{2\omega}{n}\right)^{\lfloor \gamma\omega \rfloor}\right).$$
2. Output 0 if $\mathsf{wt}(z) < 12n\rho(1-\rho)$ and $c = h(Hz + Tc||m||\mathsf{encode}(H))$. Output 1 otherwise.

4.4 Correctness and Security

In this section, we will prove the correctness and security of RaCoSS.

Theorem 2. *Honestly generated signature will always be accepted by a honest verifier unless with the negligible probability* $2^{-12n\rho(1-\rho)}$.

Proof. S^tc and y are distributed as $B(n,\rho)$. So, z is identically distributed as $B(n, 2\rho(1-\rho))$. So, by using Chernoff bound $\mathsf{wt}(z) > 12n\rho(1-\rho)$ with probability less than $2^{-12n\rho(1-\rho)}$. Now, as $Hz = H(S^tc+y) = HS^tc+Hy = Tc+Y$, the honest verifier always accept the honestly generated signature without the negligible probability $2^{-12n\rho(1-\rho)}$. □

RaCoSS is an SEUF-CMA digital signature scheme under hardness of Null Syndrome Decoding Problem.

Theorem 3. *If there is a polynomial-time forger, who breaks the strong unforgeability of RaCoSS with probability δ, then there is a polynomial-time algorithm who can solve $(H, \omega' = 2n\rho, 0)$- NSDP instance with probability $p = \frac{\delta'\delta P_{NSDP}}{2}$, where $P_{NSDP} = 2^{-(n-k)}\sum_{i=1}^{2n\rho}\binom{n-k}{2i}$ and δ' is the probability to have two signatures from same random oracle query.*

Proof. Let $\mathcal{F}$ be a forger which breaks the SEUF-CMA security of RaCoSS. We construct a decoder $\mathcal{D}$ that solves the NSDP. The decoder is given input $H \in \mathbb{F}_2^{(n-k)\times n}$ and tries to find a vector x of weight w'', where $0 < w'' \leq w'$, such that $Hx^t = 0$. The decoder will challenge the forger $\mathcal{F}$ and simulate the oracles for it. If the forger succeeds with a valid forgery of the signature then with non-negligible probability, the decoder uses the forgery to solve its given instance of the null syndrome decoding problem.

The decoder $\mathcal{D}$ will set up H as system parameter and choose private key S^t. $\mathcal{D}$ will compute $T = HS^t$ and handover H, T to the forger $\mathcal{F}$ as the public key. $\mathcal{F}$ will make $poly(n,k)$ number of hash and signature query to $\mathcal{D}$. $\mathcal{D}$ will respond hash query by choosing a random element from the range of hash function and maintain a list. During the signature query, $\mathcal{D}$ will lookup the hash table first. If the corresponding hash query is already made before, then use it for the response. Otherwise, just choose another random value from the range of hash function and insert it into the hash table. As the secret key is the actual secret key of RaCoSS, the simulation is perfect. After query phase, $\mathcal{F}$ will output $m, \sigma(= z, c)$ which does not appear in the query phase. Suppose, it is a valid signature with probability δ. Using forking lemma [16], we can have *two signatures from a forger that use the same random oracle query* with probability δ'. If δ is non-negligible, then δ' is also non-negligible. Let two signatures be (m, Hy, c, z) and (m, Hy, c', z'), where $c \neq c'$. Our knowledge of the secret key allow us to obtain the equation $\hat{h}_H(z + S^tc) = \hat{h}_H(z' + S^tc')$. Now, $T = HS^t$ has multiple solution and distribution of $(z + S^tc)$ and y are statistically indistinguishable over same sample space. Moreover, due to leftover hash lemma, there is no information about S from T. So, $\mathcal{F}$ will not know the secret key. As $T = HS^t$ has at least two solutions, at least with probability $1/2$, $z + S^tc$ and $z' + S^tc'$ are distinct. So, we have collision for the CRHF with probability $\delta'\delta/2$, which provide a solution for NSDP for the instance (H, w') with probability

$$P = \frac{\delta'\delta P_{NSDP}}{2}.$$

If δ is non-negligible then P is non-negligible.

□

4.5 Security Evaluation

In this section, we will evaluate the security of RaCoSS according to the best known attack - the information set decoding (ISD) algorithm [10]. We have chosen the length of codeword n and calculate the minimum distance as $\lfloor\sqrt{n}\rfloor$. Then we have calculated dimension k follows the Gilbert-Varshamov (GV) bound [14]. The required formula is as follows:

$$\frac{k}{n} \geq 1 + \frac{\omega}{n}\log\left(\frac{\omega}{n}\right) + \left(1 - \frac{\omega}{n}\right)\log\left(1 - \frac{\omega}{n}\right).$$

At the time of calculation of k, we have assumed the equality of the above expression and round down the value of k to the integer.

Security of the construction boils down to the hardness of $(H, \omega' = 2n\rho, 0)$-NSDP instance. Here, $\rho = \frac{1}{2}(1 - (1 - \frac{2\omega}{n})^{\mathsf{wt}(c)})$ where $\mathsf{wt}(c) = \lfloor \gamma w \rfloor$ for some positive constant γ. At the time of calculation, we will truncate the value of ρ at the third decimal position. Existing literature usually assumes that there is a unique solution to the NSDP. This is true in particular when ω' is smaller than the GV-bound. But, in our case, ω' is higher than the GV-bound. So, there are multiple solution of the instance. That is why, the considerable situation is *Single Instance and Multiple Solutions*. The relevant formula to evaluate work factor is as follows [10]:

$$WF_{ISD}(n, k, \omega') \approx \min_{p} \frac{l2^{\frac{n-k}{2}+1}}{\sqrt{\binom{n-k-l}{\omega'-p}}}$$

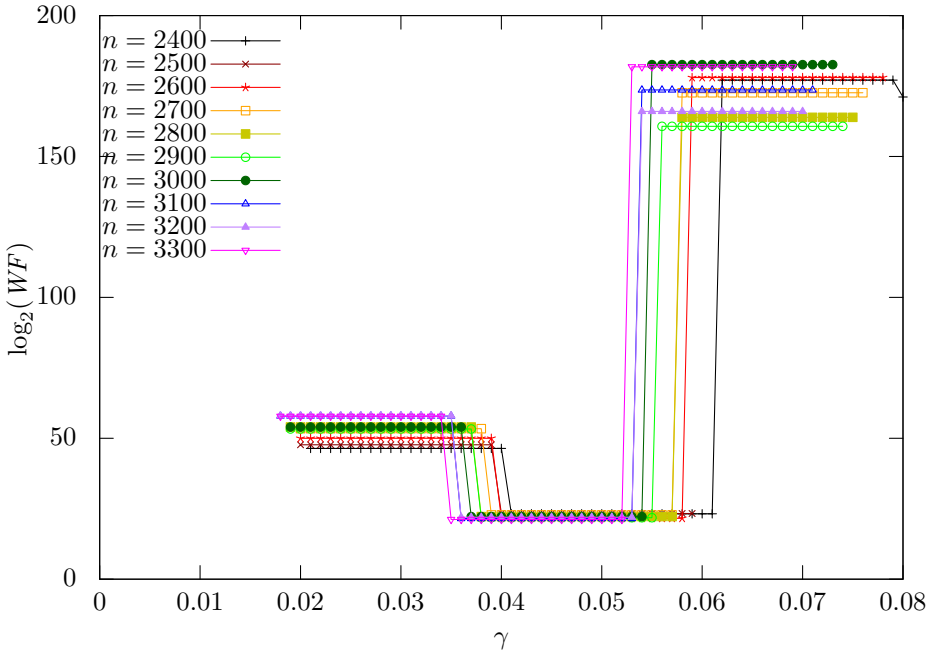
where

$$l = \log_2\left(\frac{k(n-k)2^{\frac{n-k}{2}+1}}{\sqrt{\binom{n-k}{\omega'-p}}}\right).$$

We compute the work factor, which is minimum for $p = 2$, and have $n = 2,400$, $k = 2,060$, $\omega' = 273$ for 177 bits security. Now, it is evident from the above discussion that when we fix the value of n, $\omega(\lfloor\sqrt{n}\rfloor)$ and k (GV-bound) become fixed. Moreover, ρ is a function of n, ω, γ and $\omega' = 2n\rho$. So, the formula of work factor consists of two independent variables, viz., n, γ. γ is required to choose in such a way that $0 < \omega' - p < n - k$ and $0 < \omega' - p < n - k - l$ should hold. In figure 4.1, we have shown the dependency between work factor and γ for different fixed values of n.

The size of the private key, public key, and signature based on naive data expression are 703 kilobytes, 99.6 kilobytes, and 0.586 kilobytes, respectively. Note that one kilobyte equals 1024 bytes and one byte equals eight bits. At the time of calculation of private key size, we have considered only the secret entity, i.e., S.

Figure 4.1: Graph of Work Factor



We can reduce the size of private key and signature using a classical data compression technique. Private key S^t has n^2 elements but only around $n\omega$ is 1, and other elements are 0. We can specify the position of a bit-1 in each row with $\lceil \log_2 n \rceil$ bits using an index. Thus, we can reduce the size of the secret key to $n\omega\lceil \log_2 n \rceil = \Theta(n\sqrt{n} \log n)$ bits. The signature size can be reduce to $n + \lfloor \gamma\omega \rfloor \lceil \log_2 n \rceil$ bits using the same technique. We can thus achieve 177-bit security using 169 kilobytes private key and 0.297 kilobytes signature.

Chapter 5

Analysis of Algorithm with Respect to Known Attacks

5.1 Adaptive Chosen Message Attacks

We proved that RaCoSS is an SEUF-CMA digital signature scheme in Chapter 4.2, which implies the RaCoSS is secure against adaptive chosen message attacks.

5.2 Key Substitution Attacks

Given a signature scheme Σ = (**Key Generation**, **Signature Generation**, **Signature Verification**), a key substitution attack (with malicious signer) is a probabilistic polynomial time algorithm KSA which on input of valid parameters, outputs two different valid public keys vk and vk' and a message, signature pair (m, σ), where the verification algorithm accepts on input both (m, σ, vk) and (m, σ, vk').

A key substitution attack KSA is called *weak* if the KSA also needs to output private keys corresponding to public keys. Otherwise, the KSA is called *strong*.

Theorem 4. *RaCoSS is secure against strong key substitution attacks.*

Proof. Let $\mathcal{F}_{KSA}$ be a forger who can successfully attack signature algorithm with non-negligible probability. Let $\mathcal{B}_{CRH}$ be an adversary who wants to find a collision of h. Now, as the signer is malicious, the signer can provide private key sk to $\mathcal{F}_{KSA}$ after signing. Still, to succeed, $\mathcal{F}_{KSA}$ has to find some vk' such that $h(m, vk, \sigma) = h(m, vk', \sigma)$. It is required because $\mathcal{F}_{KSA}$ has to pass the verification $c = h(Hz + Tc||m||\mathsf{encode}(H)) = h(H'z + T'c||m||\mathsf{encode}(H'))$. Now, after getting success, $\mathcal{F}_{KSA}$ hands over (m, vk, σ), (m, vk', σ) to $\mathcal{B}_{CRH}$. Definitely it is a collision for h. So, the probability of getting a collision of h is at least as large as the success probability of $\mathcal{F}_{KSA}$. But, h is assumed to

be collision resistant. $\mathcal{F}_{KSA}$ cannot get success with non-negligible probability, and the claim follows. □

Chapter 6

Advantages and Limitations

CFS [4], KKS [12] and their variants are the most celebrated code-based signature schemes. Unfortunately, there are various drawbacks in respect of security, computation time, key and/or signature size. The CFS scheme was proven existentially unforgeable under chosen message attack (EUF-CMA) by Dallot [5] under hardness of the Goppa-Parametrized Bounded Decoding problem and the Goppa-Code Distinguishing (GD) problem. But, the existence of a distinguisher [8] for the Goppa codes of high rate disproves the hardness of GD problem for the parameters relevant to the CFS scheme and hence invalidates Dallot's security argument. Moreover, signing time of CFS signature is somewhat high due to the difficulty of finding decodable syndrome. One recent work is done by Debris-Alazard et al. [6] based on FDH paradigm as CFS. But, the hardness assumption is not a standard one. In case of KKS-like signature schemes, the main drawback is their security. It was shown by Otmani et al. [13] that even the one-time version of KKS-like is not secure. In another line of works, Alaoui et al. [1] studied[1] the construction of code-based signature schemes via Fiat-Shamir transformation on zero-knowledge identification schemes by Stern [15], Veron [17] and Cayrel et al [3]. Their study shows that the signature size is 140 times of the underlying code length in case of the Stern and Veron identification schemes, and 80 times of the underlying code length in case of the Cayrel et al.'s identification scheme, which results in signatures of size about 19 kilobytes and 25 kilobytes, respectively. So, it is required to have a provably secure (based on standard hardness assumptions) code-based signature scheme with reasonable key sizes and execution time. According to our understanding, RaCoSS can serve the purpose mentioned above. It is evident from the Table 6.1 that RaCoSS has a shorter signature with higher security level in respect of [1, 6]. However, our proposal has little higher key sizes.

[1]We note that they presented security analysis and implementation, without providing explicit security proofs, although we do not doubt that such the proofs can be constructed.

[2]System parameters

Table 6.1: Comparison with Code-based Signatures Schemes

Scheme	Security	Signature	Private Key	Public Key	Proof of Security
Scheme 1 [1]	80 bits	19 KB	768 bits	36 KB	Not Given
Scheme 2 [1]	80 bits	19 KB	1152 bits	36 KB	Not Given
Scheme 3 [1]	80 bits	25 KB	4.5 KB	326 KB	Not Given
CFS [9, 4]	81 bits	98 bits	3.6 MB	20 MB	Given [5]
SURF [6]	128 bits	0.96 KB	11.76 KB	1.75 MB	Given
RaCoSS	177 bits (Category 1)	0.297KB	169 KB	99.6 KB (+99.6 KB^2)	Given

In a nutshell, advantages and limitations are as follows:

Advantages:

- RaCoSS is proved to be strong existentially unforgeable under chosen message attack (SEUF-CMA).
- RaCoSS has the security proof reduced to standard hardness assumption.
- The signature size is small in respect of other code-based signature schemes apart from CFS signature scheme with 81 bits security. But, the key sizes of CFS signature scheme much higher than RaCoss.
- The signature generation process completes in 7.07 Milliseconds and the signature verification process completes in 6.87 Milliseconds.
- The key generation process, signature generation process and signature verification process can easily be speeded up by parallel computation.

Limitations:

- Range of signature is required to be chosen according to the state of art.

Bibliography

[1] S. M. E. Y. Alaoui, P.-L. Cayrel, R. El Bansarkhani, and G. Hoffmann: Code-Based Identification and Signature Schemes in Software, CD-ARES Workshops 2013, volume 8128 of LNCS: 122–136.

[2] D. Augot, M. Finiasz, and N. Sendrier: A Fast Provably Secure Cryptographic Hash Function, `https://eprint.iacr.org/2003/230`.

[3] P. L. Cayrel, P. Veron, and S. M. El Yousfi Alaoui: A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem, Selected Areas in Cryptography (SAC 2010), volume 6544 of LNCS: 171–186.

[4] N. Courtois, M. Finiasz, and N. Sendrier: How to Achieve a McEliece-Based Digital Signature Scheme, ASIACRYPT 2001, volume 2248 of LNCS: 157–174.

[5] L. Dallot: Towards a Concrete Security Proof of Courtois, Finiasz and Sendrier Signature Scheme, Research in Cryptology, Second Western European Workshop (WEWoRC 2007), volume 4945 of LNCS: 65–77.

[6] T. Debris-Alazard, N. Sendrier, J. Tillich: SURF: A new code-based signature scheme, `https://arxiv.org/abs/1706.08065`.

[7] M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang: A Provably Secure Group Signature Scheme from Code-Based Assumptions, ASIACRYPT 2015, volume 9452 of LNCS: 260–285.

[8] J.-C. Faugére, V. Gauthier-Umana, A. Otmani, L. Perret, and J.-P. Tillich: A Distinguisher for High Rate McEliece Cryptosystems, Information Theory Workshop (ITW 2011): 282–286.

[9] M. Finiasz: Parallel-CFS, Selected Areas in Cryptography (SAC 2010), volume 6544 of LNCS:161-172.

[10] M. Finiasz and N. Sendrier: Security Bounds for the Design of Code-Based Cryptosystems, ASIACRYPT 2009, volume 5912 of LNCS: 88–105.

[11] C. Gentry, C. Peikert, and V. Vaikuntanathan: Trapdoors for hard lattices and new cryptographic constructions, Symposium on Theory of Computing (STOC 2008), 197–206.

[12] G. Kabatianskii, E. Krouk, and B. J. M. Smeets: A digital signature scheme based on random error-correcting codes, Cryptography and Coding 1997, volume 1355 of LNCS: 161–167.

[13] A. Otmani and J. P. Tillich: An Efficient Attack on All Concrete KKS Proposals, PQCrypto 2011, volume 7071 of LNCS: 98–116.

[14] R. Roth: Introduction to Coding Theory, Cambridge University Press, 2006.

[15] J. Stern: A new identification scheme based on syndrome decoding, CRYPTO 1993, volume 773 of LNCS: 13–21.

[16] D. Pointcheval and J. Stern: Security arguments for digital signatures and blind signatures, Journal of Cryptology, 13(3):361–396, 2000.

[17] P. Veron: Improved Identification Schemes Based on Error-Correcting Codes, Applicable Algebra in Engineering, Communication and Computing, 8(1):57–69, 1997.