Computer Security Symposium 2018
22 - 25 October 2018

Sliding window法からの漏洩情報を用いた秘密鍵復元攻撃の改良

黄　暁萱[1,a)]　大西　健斗[2]　國廣　昇[2]

概要：暗号のアルゴリズムを実装した際に，物理的に秘密鍵に関する情報の一部が漏洩することがある．公開鍵暗号の一種であるCRT-RSA暗号は復号する際にべき乗を行うが，ある実装ではSliding window法を使用する．Sliding window法はwindowサイズ w を決め，2乗算（S）と掛け算（M）を繰り返すことにより行われる．Sliding window法の際，SとMの実行履歴（SM時系列）が漏洩することが知られている．2017年に，Bernstein [1] らはSM時系列から秘密鍵のビットを部分的に復元するルールを提案するとともに，SM時系列に基づいた，CRT-RSA暗号の秘密鍵 d_p, d_q の復元実験を行った．しかし，これ以上ビットの復元ができない最適性証明は与えられていない．本研究では，新たにビット復元ルールを提案し，最適性を証明した．比較するために，改めて実装したBernsteinらのルールと，提案したルールの双方について，実験を行った．その結果，$w = 4$ では，秘密鍵復元の成功率が18%から29%まで向上した．さらに，秘密鍵 d のSM時系列も漏洩した時，$w = 6$ まで鍵全体の復元が可能であることを実験で示した．

キーワード：CRT-RSA暗号，サイドチャネル攻撃，Sliding window法，秘密鍵復元

Improvement of secret key recovery attack by using leakage information from Sliding window method

Xiaoxuan Huang[1,a)]　Kento Oonishi[2]　Noboru Kunihiro[2]

Abstract: When implementing cryptographic algorithms, a part of information on the secret key may be physically leaked. When CRT-RSA, a type of public key cryptography, performs exponentiation in decryption, the sliding window method may be used. Firstly determines the window size w. Then repeats the squaring (S) and the multiplication (M). It is known that the execution history of S and M (SM time series) leaks. In 2017, Bernstein and colleagues [1] proposed a rule to partially recover the secret key bits from SM series, and performed an experiment of recovering the whole key based on SM time series of secret keys d_p, d_q in CRT-RSA. However, no optimality proof that no more bit can be recovered any more. In this research, we propose a new bit recover rule and proved its optimality. For comparison, we implemented both rules. As a result, when $w = 4$, the success rate of recovering the whole key was improved from 18% to 29%. Furthermore, when the SM time series of the secret key d is also known, it has been experimentally demonstrated that it is possible to recover the whole key up to $w = 6$.

Keywords: CRT-RSA, Side-channel attack, Sliding window method, Secret key recovery

1. はじめに

1.1 研究背景

インターネット上で通信する際に，第三者に秘密情報が漏れないように，公開鍵暗号が使われている．AからBに

[1] 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[a)] 7602879282@edu.k.u-tokyo.ac.jp

情報を送る場合，まず，A は公開鍵を用い，情報を暗号化アルゴリズムより暗号文に変換して B に送る．B は，暗号文を受け取ると，秘密鍵を用い，復号アルゴリズムにより暗号文を元の情報に戻す．公開鍵は誰でも入手することができ，秘密鍵は正当の復号者以外は入手できない．

公開鍵暗号に対する典型的な攻撃として，攻撃者が通信路を傍受することにより暗号文を手に入れ，公開鍵と暗号文等の情報から，秘密鍵や平文を求める攻撃が考えられる．暗号に攻撃することは，安全性の根拠となる数学の難問を解くことに帰着できる．

しかし，実装する時に，復号における実行時間 [6] や，消費電力 [5] など物理的な情報により，秘密鍵に関わる情報が漏洩することが知られている．このような手法による攻撃はサイドチャネル攻撃と呼ばれる．典型的な攻撃を考慮するだけでなく，物理的に秘密鍵を得る攻撃に対しても，暗号システムが安全であるかどうかを考慮しなければならない．現在，この漏洩した情報を用いて，秘密鍵全体を復元する研究が多く行われている．

RSA 暗号 [10] は 1978 年に提案された代表的な公開鍵暗号の一つである．現在 SSL, TLS など，多くのプロトコルで活用されるため，様々な脅威を考慮しなければならない．特に，サイドチャネル攻撃では実際に，電力解析 [5]，キャッシュ攻撃 [14] などにより，RSA 秘密鍵の復元に成功した例もある．

1.2 関連研究

Sliding window 法は秘密鍵を変形する上で 2 乗算（S）と掛け算（M）を繰り返すことにより行われる．秘密鍵は，2 乗算と掛け算が行う順番の列で表すことができる．これを SM 時系列と呼ぶ．2017 年，Bernstein ら [1] は秘密鍵のビット復元ルールを提案し，漏洩した秘密鍵 d_p, d_q の SM 時系列を用い，CRT-RSA 暗号における秘密鍵全体の復元に成功した．彼らの手法では，まず，提案したビット復元ルールを適用し，部分的に復元した秘密鍵を得る．その後，部分的に復元した秘密鍵に対して，秘密鍵探索アルゴリズム [3] を実行し，秘密鍵を求めた．彼らは，公開鍵と秘密鍵の制約式を満たす鍵候補が 1,000,000 個以下である時，鍵の復元に成功したとみなした．彼らは 500,000 回の数値実験を行った．彼らの実験では，$w = 4$ における 1024-bit RSA 暗号に対し，32%の確率で秘密鍵半分以上のビットを復元し，28%の確率で鍵全体の復元に成功している．しかし，w が 5 における 2048-bit RSA の鍵全体を一度も復元できなかった．また，それ以上のビット復元ができない最適化の証明は与えられていない．

1.3 研究成果

本研究では，Bernstein ら [1] のビット復元ルールを改良し，新たなビット復元ルールを提案した上で，復元可能な上限までビットを復元できることを証明した．また，Bernstein らのルールと提案したルールを実装し，実験を行った．$w = 4$，1024-bit RSA における実験結果により，半分以上のビットが復元できた確率は 25%から 49%まで向上し，鍵全体が復元できた確率は 18%から 29%まで向上した．

さらに，d の漏洩情報も加える実験を行った．秘密鍵 d_p, d_q, d の SM 時系列が知られる時に，同じ w で鍵全体の復元率が大きく向上した．また，鍵全体が復元できた w が 4 から 6 まで拡大した．

2. 準備

2.1 CRT-RSA [11]

最初に標準的な RSA 暗号を説明する．RSA 暗号は公開鍵暗号の一種である．鍵生成は，まず p と q を異なる奇素数とする．次に，N を $N = pq$ とする．さらに，$e, d \in \mathbb{Z}^*_{(p-1)(q-1)}$ を $ed \equiv 1 \bmod (p-1)(q-1)$ を満たすように選ぶ．公開鍵を (N, e)，秘密鍵を (p, q, d) とする．暗号化は，平文 m に対して，暗号文 $c = m^e \bmod N$ を出力する．復号は，平文 $m = c^d \bmod N$ を出力する．

CRT-RSA の暗号化は標準的な RSA と同じである．CRT-RSA では，中国人剰余定理 (CRT) を使い，高速に復号を行う．秘密鍵として，さらに d_p, d_q を使用する．$d_p := d \bmod p - 1$，$d_q := d \bmod q - 1$ とする．公開鍵を (N, e)，秘密鍵を (p, q, d, d_p, d_q) とする．復号する時，$m_1 = c^{d_p} \bmod p$，$m_2 = c^{d_q} \bmod q$ を計算する．$m = m_1 \bmod p$，$m = m_2 \bmod q$ となる m を中国人剰余定理を用いることにより計算する．

本稿では，e の値を，実際に使われている $2^{16} + 1$ とする．

2.2 Sliding window 法

Sliding window 法で用いる Windowed form $(d_n d_{n-1} \cdots d_1)$ はビット列が window サイズ w に依存して変形された形である．$(d_n d_{n-1} \cdots d_1)$ は $d = \sum_{i=1}^{n} d_{i-1} 2^{i-1}$ を満たし，各 d_i は 0 あるいは 1 以上 $2^w - 1$ 以下の奇数である．Sliding window べき乗算は高速に $a = c^d \bmod N$ を計算する計算方法である．手順は以下の通りである．

(1) window size w を選定する．
(2) d を windowed form $(d_n d_{n-1} \cdots d_1)$ に変形する．
(3) c，c^3，c^5，$c^7 \cdots$，c^{2^w-1} $(\bmod N)$ の値をメモリに格納する．
(4) $a = 1$，for $i = n$ to $i = 1$:
  (1) $a^2 (\bmod N)$
  (2) if $d_i \neq 0$ then $a \leftarrow a \cdot c^{d_i} (\bmod N)$
  return a

Windowed form に変形するには，最上位ビットから読み込む left-to-right 変換と最下位ビットから読み込む right-

to-left 変換方法がある．以下，この二つ変換方法を説明する．

Left-to-right 変換ではビット列を左から右に読み込む．w ビットごとで 1 つのブロックとして読み込んでいくが，後ろに 0 が付いている場合，0 をブロックに入らないようにする．また，一つのブロックを取り終わり，次のブロックを取る時に，0 を飛ばし，初めて 1 になるところから window を取る．例えば，$d = 185, w = 3$ の時，まず上位から 3 ビットの 101 を読み取り，005 に変換する．次に，110 を読み取る時，一番後ろのビットが 0 なので，0 がブロックに入らないよう 11 のみを 03 に変換する．最後 0 を読み取らず，00 を飛ばして 1 を読み取る．そして，5 の前の 0 を省略し，$\boxed{101}\boxed{11}00\boxed{1} \rightarrow 503001$ となる．

それに対して，right-to-left 変換ではビット列を右から左に読み込む，ブロックが常に長さが w であり，つまりブロックの左側 0 が入っても良い．例えば，$d = 185, w = 3$ の時，下位から 001 を読み取り，001 になる．次に 111 を読み取り，007 に変換する．最後，0 を飛ばして 1 を読み取る．そして，$\boxed{1}0\boxed{111}\boxed{001} \rightarrow 10007001$ となる．

Right-to-left sliding window 法は，秘密鍵の right-to-left 変換を完了してから，sliding window 法を使うという順番で計算しなければならないが，left-to-right sliding window 法は秘密鍵の left-to-right 変換をしながら，sliding window 法を計算することができる．Algorithm 1 [8] は left-to-right sliding window 法となるアルゴリズムである，様々な実装で使われている．

Algorithm 1 Left-to-right sliding window 法

Input: 整数 p, b と d を表現するビット列 $(d_n d_{n-1} \cdots d_1)$
Output: $a \equiv b^d (\mathrm{mod} p)$.
1 : **procedure** MOD_EXP(b, d, p)
2 : $b_1 = b$, $b_2 = b^2$, $a = 1$, $z = 0$
3 : **for** $i = 1$ **to** $2^{w-1} - 1$ **do**
4 : $b_{2i+1} = b_{2i-1} \cdot b_2 \bmod p$
5 : $i = n$
6 : **while** $i \neq 1$ **do**
7 : $z = z$+COUNT_LEADING_ZEROS$(d_i \cdots d_1)$
8 : $i = i - z$
9 : $l =$min(i, w)
10 : $u = d_i \cdots d_{i-l+1}$
11 : $t =$COUNT_TRAILING_ZEROS(u)
12 : $u =$SHIFT_RIGHT(u, t)
13 : **for** $j = 1$ **to** $z + l - t$ **do**
14 : $a = a^2 \bmod p$
15 : $a = a \cdot b_u \bmod p$
16 : $i = i - l$
17 : $z = t$
18 : **return** a
19 : **end procedure**

また，Algorithm 1 の 14 と 15 行目を行うと，秘密鍵を 2 乗算 (S) と掛け算 (M) の列で表すことができる．これは秘密鍵の SM 時系列と呼ぶ．Windowed form の各桁が 0 の場合，2 乗算のみを行い，s で表される．非 0 の場合，2 乗算を行なった後に，掛け算を行い，sm で表される．例えば，503001 の SM 時系列は smssmsssm である．

2.3 サイドチャネル攻撃

サイドチャネル攻撃は暗号解読手法の一つであり，暗号処理を行う装置が発する電磁波 [2]、電力量 [5] や処理時間 [6] の違いなどを物理的手段で観察することで秘密鍵に関する漏洩情報を得ようとする攻撃である。サイドチャネルとは、正規の入出力経路ではないことを意味しており，暗号本体のアルゴリズムとは異なる副次的情報であることからこのように呼ばれている．

Bernstein らは Flush+Reload [12, 13] により SM 時系列を得る方法を提案した．Flush+Reload とは，サイドチャネル攻撃中のキャッシュ攻撃の一種類である．攻撃者は秘密鍵を持つ正当な復号者が復号する際に，ターゲットとしたメモリ領域にアクセスしたか否かを利用し，どのタイミングでどのメモリ領域にアクセスしたかを調べることにより，秘密鍵の SM 時系列を得ることができる．

2.4 問題設定

Flush+Reload [12, 13] を用いて，CRT-RSA の復号アルゴリズムの実装メモリーを観察することにより，秘密鍵の SM 時系列を得ることができる．つまりこの時，d_p, d_q の SM 時系列と window サイズ w は攻撃者が知っている．攻撃者はこれらの情報を用い，秘密鍵全体の復元を目標とする．もし d の SM 時系列も加われば，漏洩した情報が多くなるため，攻撃範囲を拡大できると考える．

3. Bernstein らの結果

Bernstein らは秘密鍵のビット復元ルールを提案するとともに，鍵復元実験を行った．まずビット復元ルールによって，d_p, d_q の SM 時系列と window サイズ w を用い，d_p, d_q それぞれ一部のビットを復元した．次に，秘密鍵探索アルゴリズム [3] で，秘密鍵 (p, q, d, d_p, d_q) と公開鍵 (N, e) 間の制約式により，秘密鍵を復元する操作を彼らは行った．まず，彼らが提案したビット復元ルールを説明する．その後，実験結果を紹介する．

3.1 SM 時系列からのビット復元ルール

Bernstein らは秘密鍵の SM 時系列から 2 進数展開のビット列を得るため，4 つのルールを提案した．x は 0 か 1 が確定できないビットを表す．$\underline{*}$は掛け算を行った位置を表す．x^i は連続する x の個数が i 個であることを表す．以下，2 乗算のみ行ったビットを 2 乗算ビットと呼び，2 乗算と掛け算両方を行ったビットを掛け算ビットと呼ぶこととする．各ルールは全て上位ビットから下位ビットへ変換していく．

- **前処理： sm→$\underline{\mathbf{x}}$，s→ x**
  sm は掛け算ビットを意味し，s は 2 乗算ビットを意味する．
- **Rule 0：$\underline{\mathbf{x}}\rightarrow \underline{1}$**
  掛け算ビットを$\underline{1}$とする．
- **Rule 1：$\underline{1}\mathbf{x}^{i}\underline{1}\mathbf{x}^{w-i-1}\rightarrow \underline{1}\mathbf{x}^{i}\underline{1}0^{w-i-1}$**
  掛け算ビットの次のビットから，新しい w サイズのブロックを取り始め，w ビット以内に掛け算ビットが現れた場合，掛け算ビットの後ろの $w-i-1$ ビットは全部 0 で埋め込む．
- **Rule 2：$\mathbf{x}\mathbf{x}^{w-2}\underline{11}\rightarrow 1\mathbf{x}^{w-2}\underline{11}$**
  掛け算ビットが二つ連続する時，前の掛け算ビットが所属するブロックの先頭ビットを 1 とする．
- **Rule 3：$\underline{1}\mathbf{x}^{i}\mathbf{x}^{w-1}\underline{1}\rightarrow \underline{1}0^{i}\mathbf{x}^{w-1}\underline{1}$**
  後の掛け算ビットが所属するブロックに含まれないビットを 0 で埋め込む．

$w=4$ の例を挙げる．SMSSSSSSSSMSMSSSSSM の場合は，

前処理：$\underline{\mathrm{x}}$xxxxxx$\underline{\mathrm{xx}}$xxxxx$\underline{\mathrm{x}}$

Rule 0：$\underline{1}$xxxxxx$\underline{11}$xxxx$\underline{1}$

Rule 1：$\underline{1}$xxxxxx$\underline{11}$000x$\underline{1}$

Rule 2：$\underline{1}$xxx1xx$\underline{11}$000x$\underline{1}$

Rule 3：$\underline{1}$0001xx$\underline{11}$000x$\underline{1}$ となる．

3.2 Bernstein らの実験結果

前述のビット復元ルールにより，$w=4$ の時に 1,000,000 回以上実行した結果は以下のようになる．1024-bit RSA の d_p, d_q は left-to-right 変換を行った時，平均 251 ビット (49%) が復元できた．$w=5$ の時に，2048-bit RSA の秘密鍵は，平均 425 ビット (41.5%) が復元できた．right-to-left 変換を行った時に，平均 204 ビット (40%) 復元できた．Left-to-right 変換は right-to-left 変換より，多くの情報が漏洩することを示した．Right-to-left 変換では，秘密鍵の windowed form での二つ非零桁の間に，少なくとも $w-1$ 個 0 がある．対照的に，left-to-right 変換では，windowed form での二つ非零桁の間の 0 は何個でも取りうる．間の 0 が $w-2$ 個以下の非零桁は，right-to-left 変換より多くのビットを漏洩する．しかし，d_p, d_q の 50%以上のビットを知らないと，秘密鍵探索アルゴリズム [3] による秘密鍵の復元が困難であることが知られている [7,9]．

さらに，Bernstein らは $w=4$ と $w=5$ の時，秘密探索アルゴリズム [3] を用い，鍵全体を復元する実験を行った．実験結果は以下のようになった．

$w=4$ の時の実験結果：

彼らの実験では，1024-bit RSA における秘密鍵復元実験を 500,000 回実行した．秘密鍵候補が 1,000,000 個以上になる場合，アルゴリズムは停止し，秘密鍵を復元できなかったと見なす．Rule 0-3 により，平均的に 251 ビット (49%) を復元できた．50%以上のビットが復元できた回数は総回数の 32%であった．さらに，秘密鍵を完全に復元できた回数は総回数の 28%であった．

$w=5$ の時の実験結果：

2048-bit RSA における秘密鍵復元実験を 500,000 回実行した．Rule 0-3 により，平均的に 41%のビットが復元できた．しかし，秘密鍵を完全に復元するには一度も成功しなかった．これは平均値 41%は閾値 50%との距離が大きいからである．

Bernstein らはルールを提案したのみであり，これ以上のビットの復元ができないことを示さなった．

4. SM 時系列に基づく新たなビット復元ルールの提案

本節では，与えられた SM 時系列から，復元可能な上限までビットを復元する手法を提案する．SM 時系列と矛盾のないビット系列の全候補について，ビットがすべて同じ値になる時，ビットの値を確定する．この時，復元可能な上限までビットの復元が行われている．既存研究では，Bernstein らが，前節のビットの復元ルールを数回適用することで，1 回だけの適用よりも多くのビットが求まる場合があると主張している [1]．しかし，彼らは，復元ルールを繰り返し適用した場合の振る舞いを正確に記述することは困難であると主張した．本研究では，SM 時系列が与えられた場合に，一回適用するだけで，復元可能な上限までビットを復元することができるビット復元ルールを提案する．

4.1 ビット復元ルール

はじめに，提案するビット復元ルールについて述べる．記法については，既存研究 [1] に従うものとする．Bernstein らと前処理および Rule 0 は同じである．本節では，ビット列の長さを L とし，上位ビットから $1,2,\ldots,L$ 番目とインデックスを付け，$\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_L$ と表す．さらに，系列を上位ビット側から見ていって，1 番目の $\underline{\mathbf{1}}$ のインデックスを m_1，2 番目の $\underline{\mathbf{1}}$ のインデックスを m_2，... と定義する．さらに，$\underline{\mathbf{1}}$ の数を M とする．この下で，Rule 1-3 によって，ビットの復元を行う．

- **Rule 1.** 系列を上位ビット側から見ていき，$\underline{\mathbf{1}}$ 同士の関係性からビットを復元する．具体的には，Algorithm 2 の通りである．
- **Rule 2.** Rule 2 は下位ビットからビット列を読み取り，ビットの復元を行う．具体的には，Algorithm 3 の通りである．
- **Rule 3.** Rule 3 は上位ビットからビット列を読み取り，ビットの復元を行う．具体的には，Algorithm 4 の通りである．

以上の復元ルールを適用すると，以下の定理 1 が成立

Algorithm 2 Rule 1

Input: ビット列 (Rule 0 完了)
Output: ビット列 (Rule 1 完了)
m_1 に着目
if 最上位ビットからビットを見た際に，$\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{x}^{w-1-j_2}$ かつ $1 \leq j_2 \leq w-1$
then $\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{x}^{w-1-j_2} \rightarrow \mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}$
for $i = 1$ **to** $M-1$
  m_i と m_{i+1} に着目
  if $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{x}^{w-1-j_2}$ かつ $1 \leq j_2 \leq w-1$
  then $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{x}^{w-1-j_2} \rightarrow \underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}$
end for

Algorithm 3 Rule 2

Input: ビット列 (Rule 1 完了)
Output: ビット列 (Rule 2 完了)
for $i = L$ **down to** $i = 1$
  if $\mathbf{x}_i = \mathbf{1}$
    if $\mathbf{x}_i(\mathbf{1})$ と，その上位 w ビットが

$$\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\mathbf{1} \text{ かつ } 1 \leq j_2 \leq w-1$$

    then $\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\mathbf{1} \rightarrow \mathbf{1}\mathbf{x}^{j_2-1}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\mathbf{1}$ **and** $i = i-w$
  elif $\mathbf{x}_i = \underline{\mathbf{1}}$
    if $\mathbf{x}_i(\underline{\mathbf{1}})$ と，その上位 w ビットが

$$\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\underline{\mathbf{1}} \text{ かつ } 1 \leq j_2 \leq w-1$$

    then $\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\underline{\mathbf{1}} \rightarrow \mathbf{1}\mathbf{x}^{j_2-1}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\underline{\mathbf{1}}$ **and** $i = i-w$
end for

Algorithm 4 Rule 3

Input: ビット列 (Rule 2 完了)
Output: ビット列 (Rule 3 完了)
for $i = 1$ **to** $M-1$
  m_i に着目
  if $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2}\mathbf{1}$
  then $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2}\mathbf{1} \rightarrow \underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{0}^{j_2}\mathbf{1}$
  elif $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2+w-1}\underline{\mathbf{1}}$
  then $\underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{x}^{j_2+w-1}\underline{\mathbf{1}} \rightarrow \underline{\mathbf{1}}\mathbf{0}^{j_1}\mathbf{0}^{j_2}\mathbf{x}^{w-1}\underline{\mathbf{1}}$
end for

する．

定理 1. 与えられた SM 時系列に，前処理を行った後，Rule 0-3 を Rule 0，1，2，3 の順に一度ずつ適用すると，復元可能な上限までビットを復元した状態となる．

Proof. まず，Sliding Window 法について振り返る．Sliding Window 法では，上位ビットからビット値が読み取られる．ビット値 1 を読み取ったときに，倍算が行われる．倍算は，w ビットをいったん読み取り，その情報に応じて行われる．具体的には，まず，w ビットの中で，最上位ビットから一番下位にある 1 までを読み取る．このとき読み取ったビットの長さを $l+1$ として，上位 $l+1$ ビットの情報を倍算に用い，残りの下位 $w-l-1$ ビットについては二乗算のみを行う．

次に，SM 時系列の全候補について，ビットごとに着目し，値が全て同じビットを復元した状態について，提案したビット復元ルールと対応付けながら考察を行う．ここで，インデックス m_i に着目する．インデックス m_i を含む w ビットについて，インデックス m_{i-1} を含む w ビットやインデックス m_{i+1} を含む w ビットの存在を考慮しなければ，取りうる状態は，

$$\mathbf{x}^{l}\underline{\mathbf{1}}\mathbf{0}^{w-1-l} \text{ under } 0 \leq l \leq w-1$$

となる．しかし，実際には，インデックス m_{i-1} を含む w ビットやインデックス m_{i+1} を含む w ビットとの関係を考慮することが必要であり，l の値の範囲が制限される．

以下，各インデックス m_i 周辺について，l の範囲が定まった下で，とりうるビット列全てで共通し，確定できるビットについて，順番に考察する．

(1) インデックス m_i を含む w ビットが，

$$\mathbf{x}^{l}\underline{\mathbf{1}}\mathbf{0}^{w-1-l} \text{ under } a \leq l \leq b$$

をとりうるとする．このとき，全ての w ビットのとり方に応じて，ビット列の共通部分をとると，

- $\underline{\mathbf{1}}$ よりも上位の b ビットは 0 も 1 もとりうるため，確定できない．
- $\underline{\mathbf{1}}$ よりも下位の $w-1-b$ ビットは 0 で共通．
- さらに下位のビットは，後の window に依存するため，0 も 1 もとりうる．

となる．したがって，$\underline{\mathbf{1}}$ と，下位の $w-1-b$ ビットが共通し，確定できる．

(2) 特に，もし，$a = b$ かつ $b > 0$ のとき，インデックス m_i を含む w ビットの位置は一意に定まるため，インデックス m_i を含む w ビットは

$$\mathbf{1}\mathbf{x}^{b-1}\underline{\mathbf{1}}\mathbf{0}^{w-1-b}$$

となり，w ビットの最上位ビットは $\mathbf{1}$ で共通し，確定できる．

(3) 最後に，

- 各インデックス m_i を含む w ビットを全て考察したとき，$\underline{\mathbf{1}}$ よりも上位となりうるビット
- （1）および（2）で値が確定しているビット

以外のビットは，

- $\underline{\mathbf{1}}$ よりも下位．
- $\underline{\mathbf{1}}$ を含む w ビットに含まれない．

のいずれかであり，全て $\mathbf{0}$ で共通し，確定できる．

以上をまとめると，各インデックス m_i における $\underline{\mathbf{1}}$ よりも上位の b ビットは，(2) の条件が無い限り復元不可能であり，それ以外のビットは全て $\mathbf{0}$ で確定することができる．これが，SM 時系列の全候補について，ビットごとに着目した場合，値が全て同じビットを復元した状態となる．ここから，提案した復元ルールで，以上を行えることを示す．

はじめに，Rule 1 では，(1) で確定できるビットを復元していることを示す．まず，インデックス m_1 に着目する．

このとき，インデックス m_1 を含む w ビットのとりうる状態は，b_1 を $w-1$ 以下のある整数として，

$$\mathbf{x}^{l_1}\underline{\mathbf{1}}\mathbf{0}^{w-1-l_1} \text{ under } 0 \le l_1 \le b_1$$

であり，Rule 1 では，$\underline{\mathbf{1}}$ と，その下位 $w-1-b_1$ ビットを確定する．これは，(1) で確定できるビットと一致する．ゆえに，Rule 1 によって，(1) で確定できるビットを復元できる．

次に，インデックス m_2 に着目する．ここで，インデックス m_1 と m_2 の間のビット数を l_{12} とする．このとき，

$$\underline{\mathbf{1}}\mathbf{0}^{w-1-b_1}\mathbf{x}^{l_{12}-(w-1-b_1)}\underline{\mathbf{1}}\mathbf{0}^{\max(2w-2-b_1-l_{12},0)}$$

であり，Rule 1 によって復元するビットは，$\underline{\mathbf{1}}$ と，その下位 $\max(2w-2-b_1-l_{12},0)$ ビットである．ここで，(1) で確定できるビットについて考察する．インデックス m_2 を含む w ビットは，l_1 に応じて，

$$\mathbf{x}^{l_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-l_2} \text{ under } 0 \le l_2 \le \min(l_1+l_{12}-w+1, w-1)$$

と書き表すことができ，$l_1+l_{12}-w+1$ の上限は $b_1+l_{12}-w+1$ なので，$\underline{\mathbf{1}}$ と，その下位

$$w-1-\min(b_1+l_{12}-w+1, w-1) = \max(2w-2-b_1-l_{12},0)$$

ビットが，(1) で確定できるビットであり，これは Rule 1 で復元したビットと一致する．以下，同様に m_3, m_4 を考察すると，Rule 1 によって，(1) と同様にビットを確定できる．

次に，Rule 2 では，(2) で確定できるビットを復元していることを示す．ここで，インデックス m_i に着目したとき，

$$\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\mathbf{1} \text{ かつ } 1 \le j_2 \le w-1$$

または

$$\mathbf{x}^{j_2}\underline{\mathbf{1}}\mathbf{0}^{w-1-j_2}\underline{\mathbf{1}} \text{ かつ } 1 \le j_2 \le w-1$$

であるときに，$\mathbf{x}^{j_2}$ の先頭の $\mathbf{1}$ を復元している．以上の状態は，インデックス m_i を含む w ビットが，

$$\mathbf{x}^{l}\underline{\mathbf{1}}\mathbf{0}^{w-1-l} \text{ under } a \le l \le b$$

をとりうるとしたとき，$a \ge j_2$ であり，かつ，Rule 1 の適用結果より，$b = j_2$ なので，$a = b = j_2$ である状態の下で，(2) によって確定できるビットを確定したことに該当する．ゆえに，Rule 2 によって，(2) によって確定できるビットを確定している．

最後に，Rule 3 では，(3) で確定できるビットを復元していることを示す．Rule 3 では，全ビットのうち，各インデックス m_i を含む w ビットを全て考察したときに $\underline{\mathbf{1}}$ よりも上位となりうるビットおよび 1. で値が確定したビットを除いたビットを，全て $\mathbf{0}$ としている．ゆえに，Rule 3 によって，(3) によって確定できるビットを確定している．

したがって，Rule 1,2,3 で復元可能なビットをすべて復元した状態となる．以上より，示せた． □

例を挙げると，$w = 4$ における SM 時系列，SMSSSSSSSSSMSSSMSSSSSMSMSSSSSSSSM は，Beinstein らのルールを一回だけ適用すると，

1̲00000xxx1̲xx1̲01xx1̲1̲0000xxx1̲

まで復元できる．

新しい提案したルールを一回だけ適用すると，

1̲000001xx1̲1x1̲01xx1̲1̲0000xxx1̲

まで復元できる．Beinstein らより 2 ビット多く復元できた．

4.2 実験結果

w が 3 から 7 まで時の実験を行った．各 w に対して，2048-bit RSA における 100 回の実験を行った．Bernstein らのルールによるビット復元割合の平均値と新しい提案したルールによるビット復元の平均値は表 1 でまとめた．

表 1 復元できたビットの割合平均値の比較

w	Berstein らのルール	提案したルール	差
3	60.00%	60.80%	0.80%
4	49.44%	49.96%	0.51%
5	41.60%	41.84%	0.24%
6	36.07%	36.19%	0.12%
7	31.70%	31.76%	0.06%

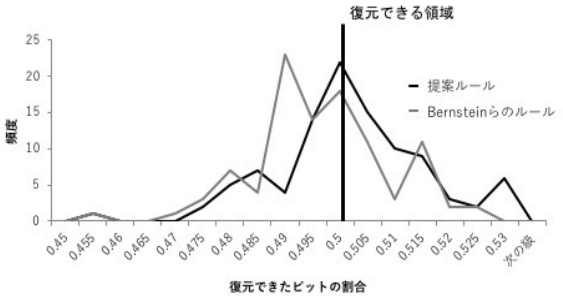

図 1 1024-bit RSA，$w = 4$，復元できたビット割合の分布

表 1 により，確かに我々の提案ルールにより，多くのビットが復元できていることがわかる．提案したルールは最適性を持つので，必ず彼らより多くのビットを復元する．Bernstein らのルールと比較して，新たに復元されるビットは主に Rule 2 によるものである．w の増加により，Bernstein らのルールとの差が小さくなる．これは提案した Rule 2 になる状況が少ないからである．$w = 4$ の時に，提案したルールで秘密鍵のビットを 49.96%まで復元できた．図 1 により，復元できる領域は縦線の右側になる．

また，d_p, d_q の SM 時系列が知られた時，提案したルー

ルを用い，1024-bit CRT-RSA における鍵全体復元の実験を 100 回実行した．Bernstein らと同じ基準で，秘密鍵候補が 1,000,000 以上になる場合，アルゴリズムは停止し，秘密鍵を復元できなかったとみなす．実験結果は，d_p, d_q を復元できたビットの割合の平均値が 50%以上になる確率は 49%，鍵全体が復元できた確率は 29%であった．比較する為に，Bernstein らのルールも同じ設定で 100 回の実験を実行した．実験結果は，d_p, d_q を復元できたビットの割合の平均値が 50%以上になる確率は 25%，鍵全体が復元できた確率は 18%であった．したがって，提案したルールによって，鍵全体の復元率が 18%から 29%まで向上した．これは，提案したルールによる復元割合の平均値は閾値 50%に非常に近くなったことによる．

表 2 $w = 4$，1024-bit RSA における復元率の比較

	50%以上のビットが復元できた確率	鍵全体が復元できた確率
Bernstein らのルール	25%	18%
提案したルール	49%	29%

4.3 考察

鍵全体の復元率の向上の原因は，提案したルールが Bernstein らのルールより多くのビットを復元できたからである．主に提案したルールの Rule 2 で復元できたビットが多くなった．Bernstein らは 2 つの連続した掛け算ビットしか考えなかった．それに対し，提案したルールでは掛け算ビットが所属する window の位置が確定する全てのパターンを考えた．Window の位置が確定する場合，window の先頭が必ず 1 になる．各 window に適用することにより，多くのビットが復元できた．

5. d の情報が加わった時の攻撃性能評価

5.1 モチベーション

Bernstein らの実験では，w が 4 以下である場合しか復元できなかった．しかし，現実では実際に $w = 5$ を使っているシステムも存在しており，w が 5 以上の時に本当に安全であるかを検証する必要がある．ここでは，秘密鍵 d の SM 時系列の情報も含めたら，多くの情報が漏洩し，攻撃可能な範囲が広がる可能性がある．そこで，本研究では，d, d_p, d_q の SM 時系列が漏洩した時の実験を行った．

5.2 実験結果

提案したルールを用い，各 w に対し，2048-bit RSA（$w = 4$ の時のみは 1024-bit RSA）における 100 回の実験を行った．ここで，RSA の公開鍵 N を n ビットとすると，p, q は $n/2$ ビットであり，秘密鍵探索アルゴリズム [3] では $n/2$ ビットを探索すれば十分である．ゆえに，d も下位 $n/2$ ビットだけを考えればよい．我々の実験結果を表 3 でまとめた．d_p, d_q のみの場合，$w = 5$ の時，鍵全体を復元できなかったので，$w = 6$，$w = 7$ の実験は行わなかった．

表 3 鍵全体が復元できた確率

w	d_p, d_q	d, d_p, d_q
3	100%	100%
4	29%(1024bit)	100%(1024bit)
5	0%	100%
6	—	78%
7	—	0%

d の SM 時系列が漏洩した時に，$w = 6$ までは高い確率で鍵全体の復元に成功した．

5.3 考察

秘密鍵のビットが半分以上知れば，鍵全体の復元が可能になる．d, d_p, d_q 合計ビット数の 1/3 は $n/2$ ビットである．つまり，d の SM 時系列の情報も含めたら，d, d_p, d_q 合計ビット数の 1/3（約 33.3%）以上が知られば，鍵全体の復元が可能になる．Bernstein らの実験結果により，$w = 5$ の時に，平均的に 41%のビットが復元できた．33.3%より十分に大きいので，鍵全体の復元は簡単にできる．

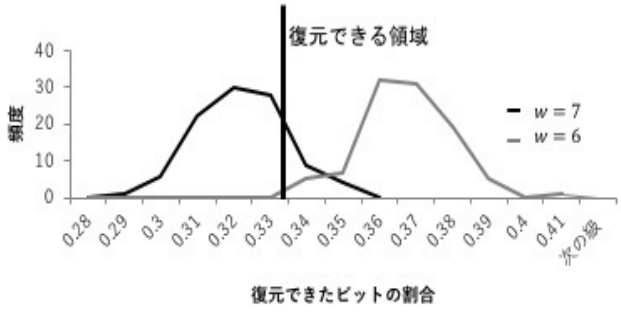

図 2 2048-bit RSA，$w = 6, w = 7$，復元できたビット割合の分布

表 3 から $w = 6$ の時を着目すると，提案したルールにより，平均的に復元できたビットの割合は 33.3%以上になる．100 回実験の中，74 回完全に復元できたが，復元できる領域内では必ず復元にならない．表 3 から $w = 7$ の時を着目すると，図 2 より，平均的に復元できたビットの割合は 31.55%であり，33.3%以下である．33.3%以上になる回数はかなり少ないため，鍵全体の復元ができなかった．

6. まとめ

本研究では，CRT-RSA 暗号が復号する時に，left-to-right sliding window 法を使用する場合を考える．サイドチャネル攻撃により，秘密鍵の SM 時系列がを知ることができる．まず，Bernstein ら [1] の SM 時系列から秘密鍵のビットを復元するルールを改良し，最適性を証明した．実験結果

により，$w = 4$ の時に，鍵全体を復元する確率が 18%から 29%まで大幅に上がった．次に，d_p, d_q だけでなく，d の SM 時系列も知られた時に，鍵全体の復元ができる w を 4 から 6 まで拡大し，より広い範囲の攻撃が可能になった．

参考文献

[1] Bernstein, D., Breitner, J., Genkin, D., Bruinderink, L.G., Heninger, N., Lange, T., Vredendaal, C., Yarom, Y.: Sliding Right into Disaster: Left-to-Right Sliding Windows Leak. In International Conference on Cryptographic Hardware and Embedded Systems, pp. 555-576. Springer, CHAM, 2017.

[2] Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Attacks: Concrete Results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp.252-261. Springer, Heidelberg, 2001

[3] Heninger, N., Shacham, H.: Reconstructing RSA Private Keys From Random Key Bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol.5677, pp. 1-17. Springer, Heidelberg, 2009.

[4] Joye, M., Yen, S.-M.: Optimal Left-to-Right Binary Signed-Digit Recoding. IEEE Trans. Comput. 49(7), pp. 740-748. 2000.

[5] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388-397. Springer, Heidelberg, 1999

[6] Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104-113. Springer, Heidelberg, 1996

[7] Kunihiro, N., Shinohara, N., Izu, T.: Recovering RSA Secret Keys From Noisy Key Bits With Erasures And Errors. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 180-197, Springer, Heidelberg, 2013

[8] Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton, 1996.

[9] Paterson, K. G., Polychroniadou, A., Sibborn, D. L.: A Coding-Theoretic Approach to Recovering Noisy RSA Keys. In: Wang, X., Sako, K. (eds.) ASIACRYPT. LNCS, vol. 7658, pp. 386-403. Springer, 2012

[10] Rivest, R. L., Shamir, A. and Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM, vol. 21, No. 2, pp. 120-126. 1978.

[11] RSA Laboratories: PKCS#1 v2.1: RSA Cryptography Standard., 2002 ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf

[12] Yarom, Y., Benger, N.: Recovering OpenSSL ECDSA nonces Using the Flush+Reload Cache Side-Channel Attack. Cryptology ePrint Archive, Report 2014/140, February 2014.

[13] Yarom, Y., Falkner, F.: FLUSH+RELOAD: A High Resolution Low Noise L3 Cache Side-Channel Attack. In: USENIX Security, San Diego, CA, US, pp. 719-732. 2014.

[14] Yarom, Y., Genkin,D., Heninger, N.: Cache Bleed: A Timing Attack on Open SSL Constant Time RSA. In: Gierlichs, B., Poschmann, A.Y. (eds.). CHES 2016. LNCS, vol. 9813, pp. 346-367, Springer, Heidelberg, 2016.