WAIS2020@Tokyo University of Science

Improved CRT-RSA Secret Key Recovery Method from Sliding Window Leakage

Kento Oonishi (The University of Tokyo)

Xiaoxuan Huang (The University of Tokyo)

Noboru Kunihiro (University of Tsukuba)

21st. Feb, 2020

Our Contribution

1. Formularize **the exact bit recovery rate** from Square & Multiply sequence on Sliding Window method

2. **Propose the new method** for recovering CRT-RSA secret keys from Square & Multiply sequences

3. **Experiment** of proposed method

Agenda

1. Background
   Threat of Side-Channel Attacks

2. Previous Result [BBG+17]
   Previous Key Recovery Method

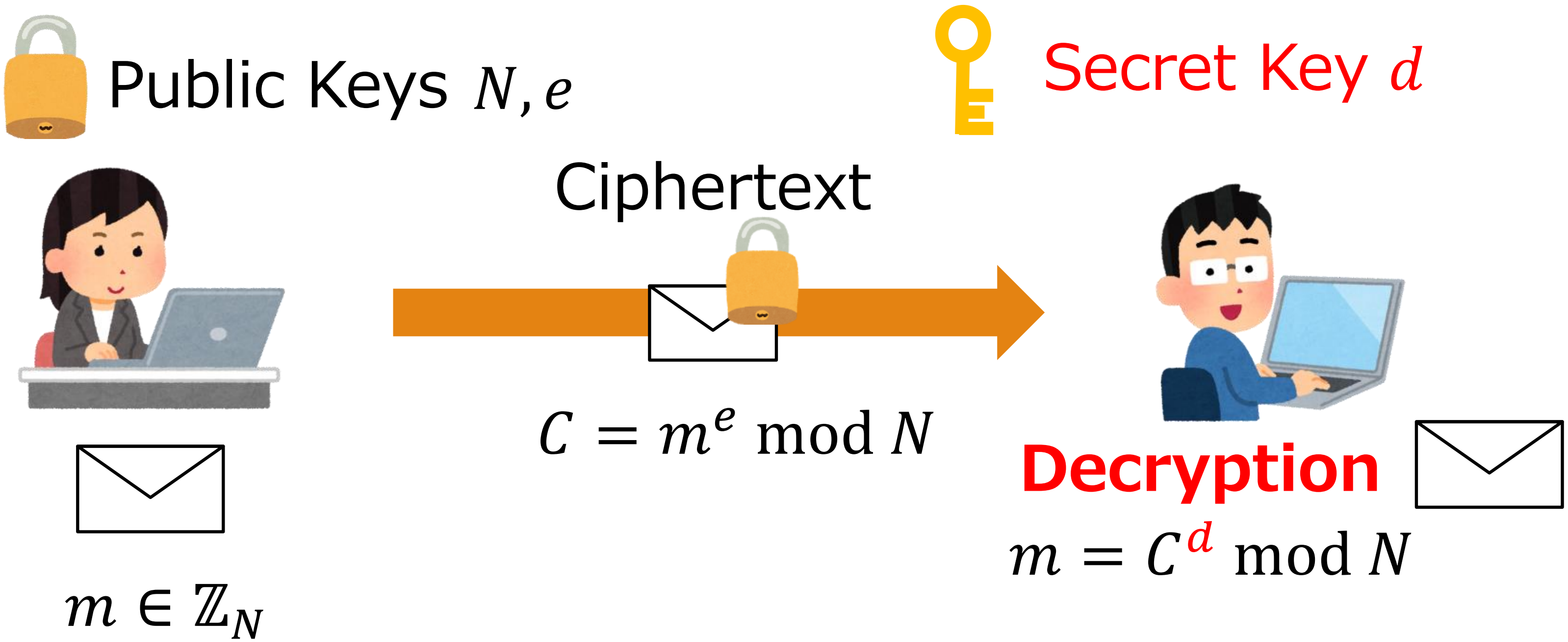
3. Our Result
   New Key Recovery Method

4. Conclusion

RSA Encryption Scheme [RSA78]

p, q : distinct $n/2$-bit prime numbers

$N = pq, ed \equiv 1 \bmod (p-1)(q-1)$.



CRT-RSA Scheme (PKCS#1)

Faster Decryption than the standard RSA

Secret Keys

$d \in \mathbb{Z}^*_{(p-1)(q-1)}$

smaller →

Chinese Remainder Theorem (CRT) decomposition

Secret Keys

$(d_p, d_q) \in \mathbb{Z}^*_{p-1} \times \mathbb{Z}^*_{q-1}$

$d_p \equiv d \bmod p-1$

$d_q \equiv d \bmod q-1$

Decryption

$m = C^d \bmod N$

faster →

$m_1 = C^{d_p} \bmod p$

$m_2 = C^{d_q} \bmod q$

We deal half bits.

↓ **CRT**

m

Sliding Window Method

We calculate c^d by **squaring** and **multiplication**.

Squaring: $X \to X^2$ **M**ultiplication: $X \to aX$

Input: c, d, and **window size w**
Output: c^d

We set c^0 as initial value.

We read bits of d from the most significant bit.
-If bit is "0", **S**quaring
-If bit is "1", we read w-bits.

w-bits ➡ $\begin{cases} w & \textbf{S}\text{quarings} \\ 1 & \textbf{M}\text{ultiplication} \end{cases}$

Threat of Side Channel Attacks

CRT-RSA scheme implemented by SW-method is known to be vulnerable against **side channel attacks**! [BBG+17]

SMSMSSM

⟹Secret keys ($\boldsymbol{d_p}, \boldsymbol{d_q}$)

*Difficult in larger w

Cache Access of decryption

c^{d_p}, c^{d_q}

[BBG+17] Bernstein et al. “Sliding Right into Disaster: Left-to-Right Sliding Windows Leak.” CHES 2017.

Our Motivation from [BBG+17]

	Success Rate
$w = 4$, 1024-bit CRT-RSA	≈100%
$w = 5$, 2048-bit CRT-RSA	8.6%

In $w = 5$, we only recover 8.6% CRT-RSA secret keys because of **too many candidates**.

By decreasing the number of candidates,
we may recover more CRT-RSA secret keys.

Question
How do we **decrease** the number of candidates?

Our New Method

We extract more information as bits.

SSMSSSSMSSM…

↓

SSMSSSSMSSM… **+ x101x1x1…**

↓

11010111…

We recover **more** secret keys when $w = 5$.

Technique in [BBG+17]

Proposed key recovery algorithm from **S&M sequences**

They use **Branch and Bound** Strategy.

-Compute some candidate bits
-Prune branches

Iteration

*If there is no pruning, final candidates **always** include **the correct secret keys**.

How do they compute candidate bits sequentially?

How do they prune branch?

Computing Candidate Bits [HS09]

We can compute candidate bits using **mathematical relationship** in CRT-RSA.

$$p[i] + q[i] \equiv c_1 \bmod 2,$$

$$d_p[i + \tau(k_p)] + p[i] \equiv c_2 \bmod 2,$$

$$d_q[i + \tau(k_q)] + q[i] \equiv c_3 \bmod 2.$$

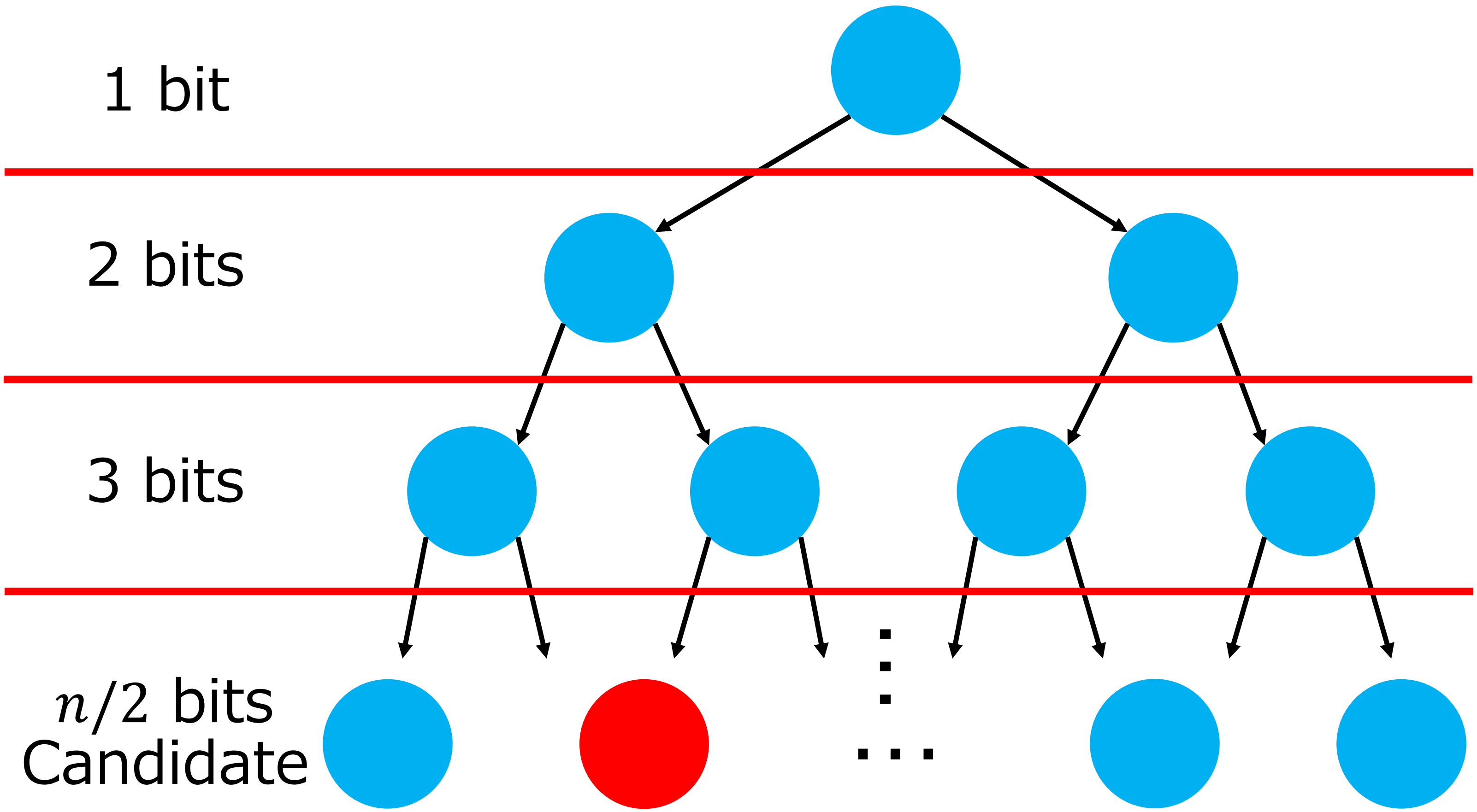
Candidate bits | Known

This simultaneous equations have one degree of freedom.

We obtain **two candidates**.

[HS09] Heninger and Shacham. "Reconstructing RSA Private Keys from Random Key Bits." CRYPTO 2009.

Visualization of Calculation



There are exponential candidates in tree.
How do we prune branch?

Pruning in [BBG+17]

Method A	Method B*
SSMSSSSMSSM⋯	**SSMSSSSMSSM⋯**
↓ We partially recover bits.	↓ [HS09]
x10xx1x1⋯	
↓ [HS09]	
11010111⋯	**11010111⋯**

*They showed that we can recover CRT-RSA secret keys in polynomial time by method B when $w \leq 4$.

Method A

Given sequences

d_p **SMSSSSM**

d_q **SSMSSSS**

Recover bits **partially***

d_p 1 0 0 ? 1

d_q ? 1 0 0 ?

Calculated Bits

d_p 1 0 1 1 0

d_q 0 1 1 0 1

Compare on **bits**
We remain leaves only when there is no mismatch.

*[Vre18] proposed the optimal bit recovery method.

[Vre18] van Vredendaal. "Exploiting Mathematical Structures in Cryptography." Eindhoven University of Technology, 2018.

Optimal Bit Recovery [Vre18]

"Optimal" means we recover **all common bits** when we consider all candidates.

Example **SSMSSM**, $w = 2$

All Candidates	0	**1**	0	**1**	→	x **1** x **1**
	1	**1**	0	**1**		
	1	**1**	1	**1**		

*Recovered bits are always correct.

Method B

Given sequences

d_p **SMSSSSM**

d_q **SSMSSSS**

Calculated Bits

d_p 1 0 1 1 0

d_q 0 1 1 0 1

Convert into S&M sequences

d_p **SSSMSMS**

d_q **SSSMSSM**

Compare on **S&M sequences**

We remain leaves only when there is no mismatch.

Experimental Results of [BBG+17]

- Implement with Depth first search (DFS)
- If we generate 1,000,000 leaves, return failure.
- They measure success rate on 500,000 trials.

	Method A	Method B
$w = 4$, 1024-bit CRT-RSA	28%	≈100%
$w = 5$, 2048-bit CRT-RSA	0%	8.6%

Method B recovers more CRT-RSA secret keys

Our Key Idea

Method A and B can be combined!

Method A

SSMSSSSMSSM···

↓ [Vre18]

x10xx1x1···

↓ [HS09]

11010111···

Combine Here!!!

Method B

SSMSSSSMSSM···

↓ [HS09]

11010111···

Our New Method

We extract more information on bits.
Then, we decrease candidates!

SSMSSSSMSSM…

↓ We partially recover bits by [Vre18].
We determine more bits with high accuracy.
*These bits are not always correct.

SSMSSSSMSSM… + **x101x1x1…**

↓ [KSI14]

11010111…

[KSI14] Kunihiro et al. “Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors. ” IEICE Trans. Fundamentals. E97-A, 1273—1284, 2014.

Explicit Form of the Bit Recovery Rate

When $w \geq 2$,

$$\frac{2}{w+1} + \frac{\sum_{k=0}^{w-1} f_w(k) g(k)}{2(w+1)} + \frac{2^w - 1}{2^{w-1}(2^{w-1}+1)} \frac{1}{3(w+1)}$$

$$f_w(k) = \frac{2}{3 \cdot 2^k}\left(1 - \frac{1}{2^{w-k}}\right)\left(1 - \frac{2}{2^{w-k}}\right)$$

$$g(k) = 2\left(1 - \frac{2^k}{2^{k+2} - 1}\right)\prod_{j=1}^{k} \frac{2^{j-1}}{2^{j+1} - 1}$$

w	Theoretical
3	60.95%
4	49.81%
5	41.92%

Experimental Results

w	Theoretical (%)	2048-bit CRT-RSA 100 times (%)
3	60.95	60.80
4	49.81	49.96
5	41.92	41.84
6	36.09	36.19
7	31.65	31.76

Our analysis matches with the experiment.

Hidden Information

w	Theoretical (%)	2048-bit CRT-RSA 100 times (%)
4	49.81	49.96

We **cannot** recover secret keys when $w = 4$, because less than 50% bits are recovered [PPS12].

However, we **can** recover secret keys when $w = 4$, by method B of [BBG+17].

*Using S&M sequences directly

There are more information in unrecovered bits!

How do we extract more information?

[PPS12] Paterson et al. "A Coding-Theoretic Approach to Recovering Noisy RSA Keys." Asiacrypt 2012.

Extracting Hidden Information

We extract information as **the proportion of "1"**.

Example SSMSSM, $w = 2$

All Candidates	Recovered bits
0 1 0 1	
1 1 0 1	x 1 x 1
1 1 1 1	
# of candidates: 3	# of candidates: 4

# of candidates: 3 < # of candidates: 4

There are some bits as almost "1".

*We check this heuristically by Monte-Calro approach.

Thus, we determine more bits with high accuracy!!!

Our New Method **(Again)**

We extract more information on bits.
Then, we decrease candidates!

SSMSSSSMSSM···

↓ We partially recover bits by [Vre18].
We determine more bits with high accuracy.
*These bits are not always correct.

SSMSSSSMSSM··· **+** **x101x1x1···**

↓ [KSI14]

11010111···

[KSI14] Kunihiro et al. “Recovering RSA Secret Keys from Noisy Key Bits with Erasures and Errors. ” IEICE Trans. Fundamentals. E97-A, 1273—1284, 2014.

Our New Method: Step 1

SSMSSSSMSSM⋯

We partially recover bits by [Vre18].

*Recovered bits are <u>always</u> correct.

SSMSSSSMSSM⋯ + x10xx1x1⋯

We choose sufficient candidates randomly, from <u>all</u> candidates.

We determined bits as “1”, when the proportion of “1” is more than $1-\varepsilon$.

*These bits are <u>not always</u> correct.

SSMSSSSMSSM⋯ + x101x1x1⋯

Our New Method: Step 2

SSMSSSSMSSM··· + x101x1x1···

Pruning in Method A, B of [BBG+17]

- Method A: Recovered Bits
- Method B: S&M Sequences

+

[KSI14] on additional determined bits

- Calculate $\lfloor 1/\varepsilon \rfloor$-bits
- If there are more than one mismatches, we discard a leaf.

11010111···

Experiment

We perform experiment on 2048-bit CRT-RSA, $w = 5$.

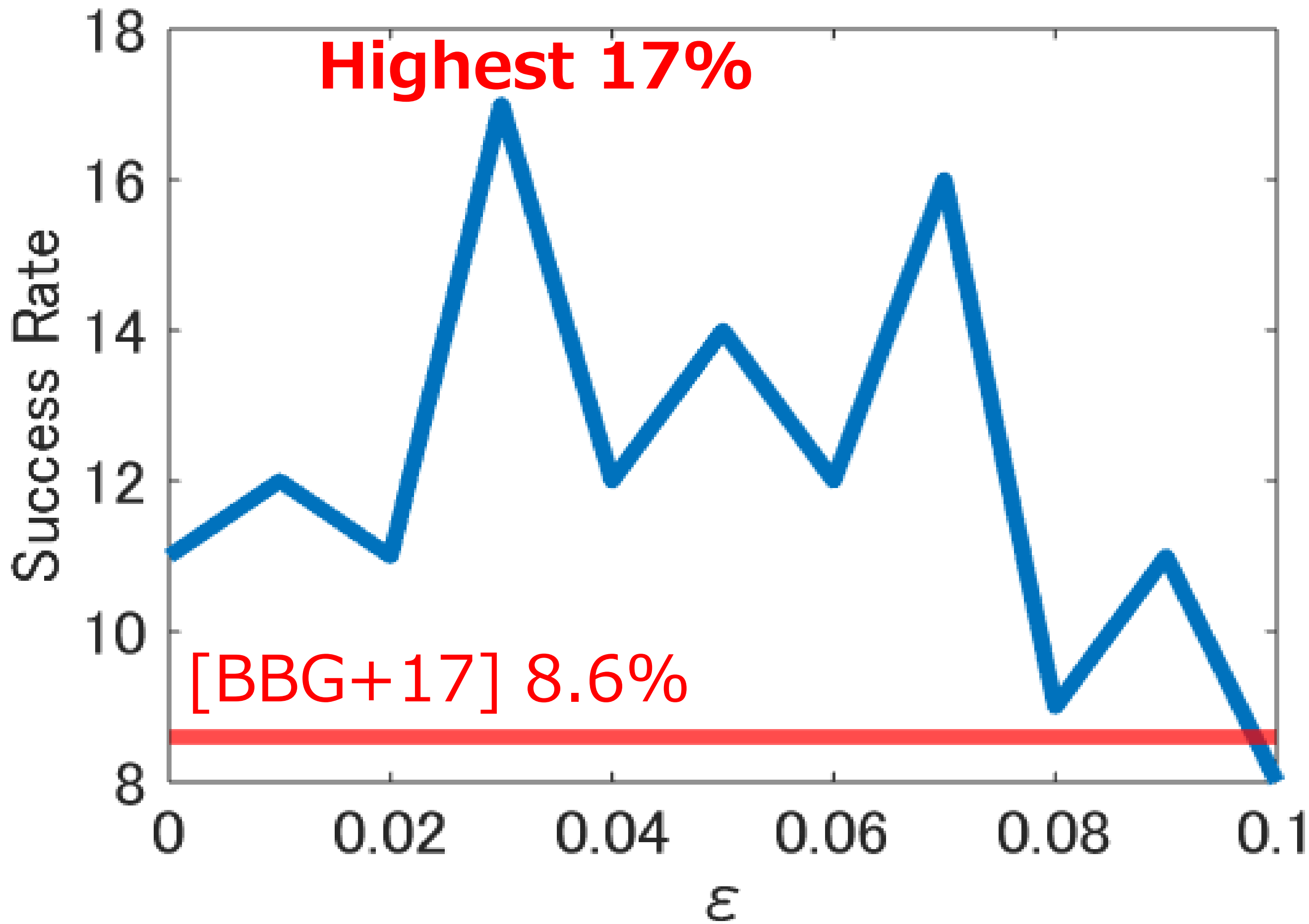
We set the parameter $0 \leq \varepsilon \leq 0.1$ at 0.01 intervals.

In each ε,

- We determine additional bits based on 1,000 samples.
- Implement by Depth first search (DFS)
- If we generate L leaves, return failure.
- We generate 100 CRT-RSA keys randomly, and measure success rate.

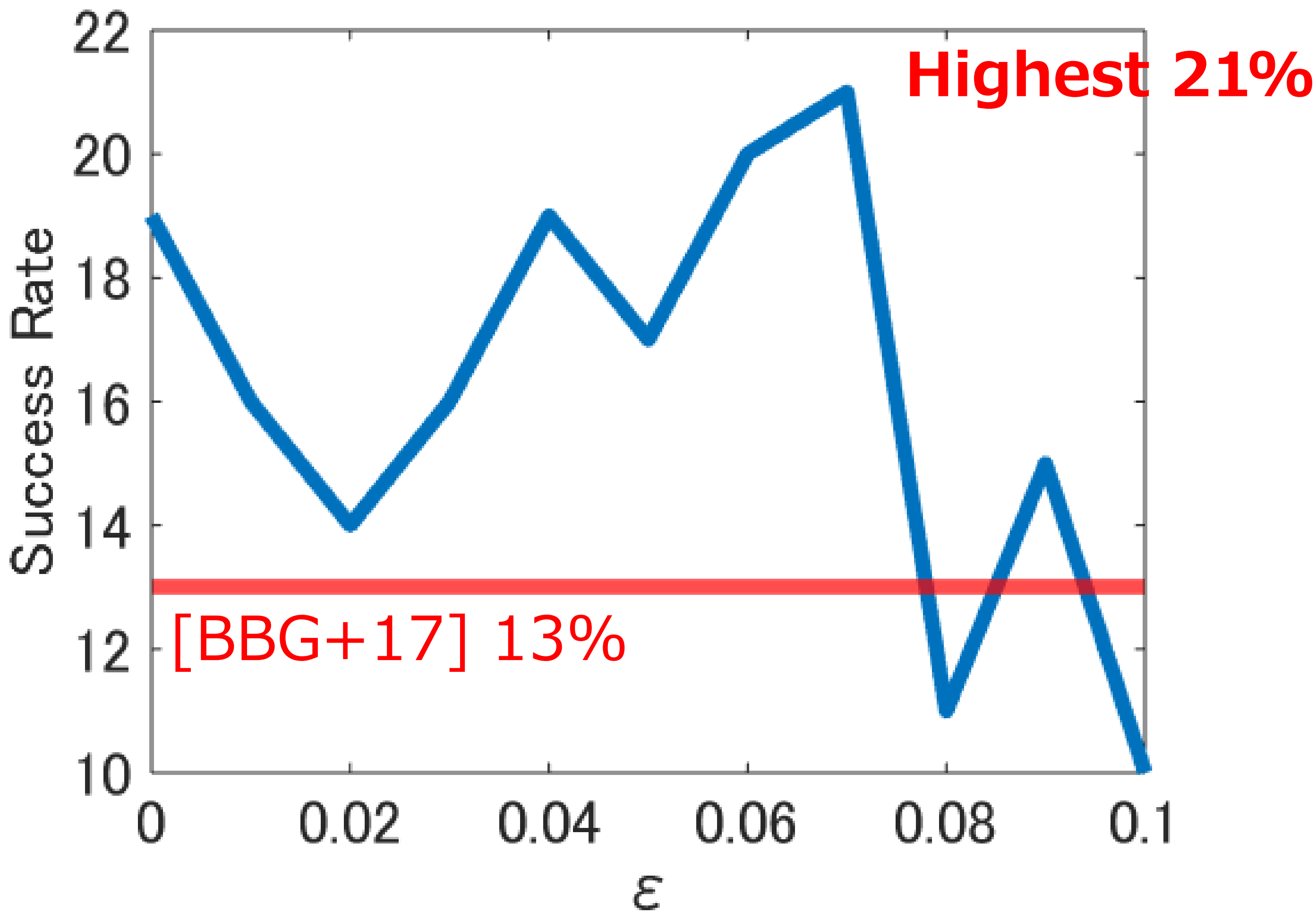
Experimental Result ($L = 1{,}000{,}000$)

Our method recovers more secret keys!



Experimental Result ($L = 2{,}000{,}000$)

Our method recovers more secret keys!



Conclusion

1. Formularize **the exact bit recovery rate** from Square & Multiply sequence on Sliding Window method

2. **Propose the new method** for recovering CRT-RSA secret keys from Square & Multiply sequences

3. **Experiment** of proposed method

L	[BBG+17]	[Ours]
1,000,000	8.6%	17%
2,000,000	13%	21%