Novel Key Recovery Attack on Secure ECDSA Implementation by Exploiting Collisions between Unknown Entries

Prerecorded talk for CHES 2021

Sunghyun Jin[1,2], Sangyub Lee[3], Sung Min Cho[4], HeeSeok Kim[5], and Seokhie Hong[1,2]

[1] School of Cyber Security, Korea University
[2] Center for Information Security Technologies, Institute of Cyber Security and Privacy, Korea University
[3] National Institute for Mathematical Sciences
[4] CIOT
[5] Department of Cyber Security, College of Science and Technology, Korea University

Background

Cryptographic Algorithm Laboratory

- ECDSA
  - An elliptic curve cryptography-based digital signature scheme
  - Used in a wide variety of security services
  - Has speed and memory usage advantages with shorter key length, compared to RSA
  - Preferred in constrained environments such as smart cards

- Side-Channel Analysis
  - Known as a practical threat against cryptosystems
  - Recover secret by using leakages occurring from cryptosystems are executing
  - Cryptosystems must be implemented securely against side-channel analysis

Background

- ECDSA signature generation

1. $k \xleftarrow{\$} [1, ord - 1]$
2. $Q = k \cdot P$ // scalar multiplication
3. $r = x_Q \ mod \ ord$
4. $h = hash(m)$
5. $s = k^{-1} \cdot (h + d \cdot r) \ mod \ ord$ // long integer calculation

Power/EM

Timing

Background

- Overview of Side-Channel Attacks against ECDSA

$$k \cdot P$$

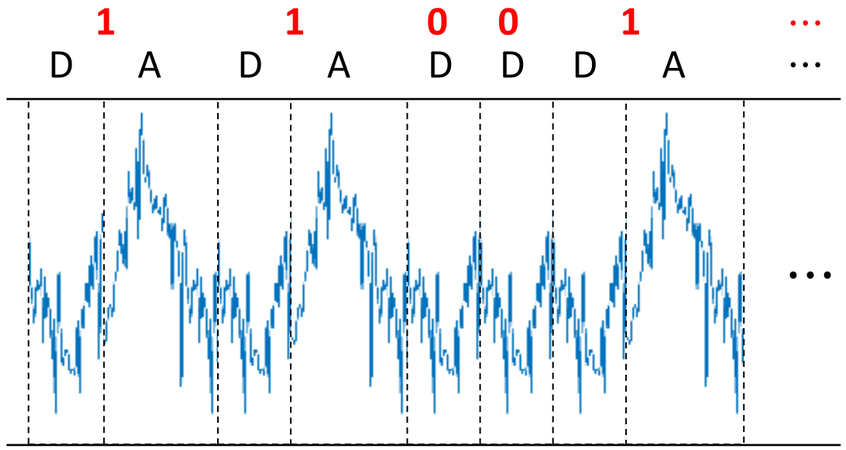
$$s = k^{-1}(h + dr) \;\Leftrightarrow\; d = (ks - h)r^{-1}$$

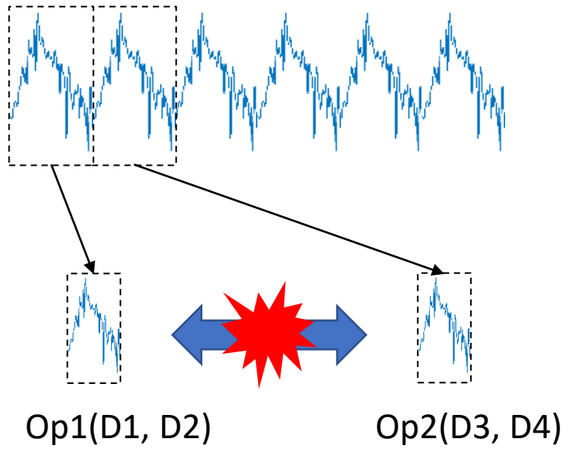
- Single-Trace Attack

  - SPA (Simple Power Analysis)

  - CA (Collision Attack)

    ▹ HCCA, ROSETTA





- Multiple-Trace Attack

  - DPA/CPA (Differential/Correlation Power Analysis)

  - HNP (Hidden Number Problem)

    ▹ convert partial information to CVP (Closest Vector Problem)

Background

- Regular Table-based Scalar Multiplication
  - Regular
    - Perform an identical sequence of operations, independently to scalar
      - ▹ the same number of doubling and addition
  - Table-based scalar multiplication
    - It is widely used for efficiency and security
    - Fixed-Base Comb, NAF windowing, T_SM, …

$\Rightarrow$ This can be easily implemented to be practically secure against known SCAs

Background

- Regular Table-based Scalar Multiplication

Algorithm 2 Preparation of pre-computed tables for regular table-based scalar multiplication

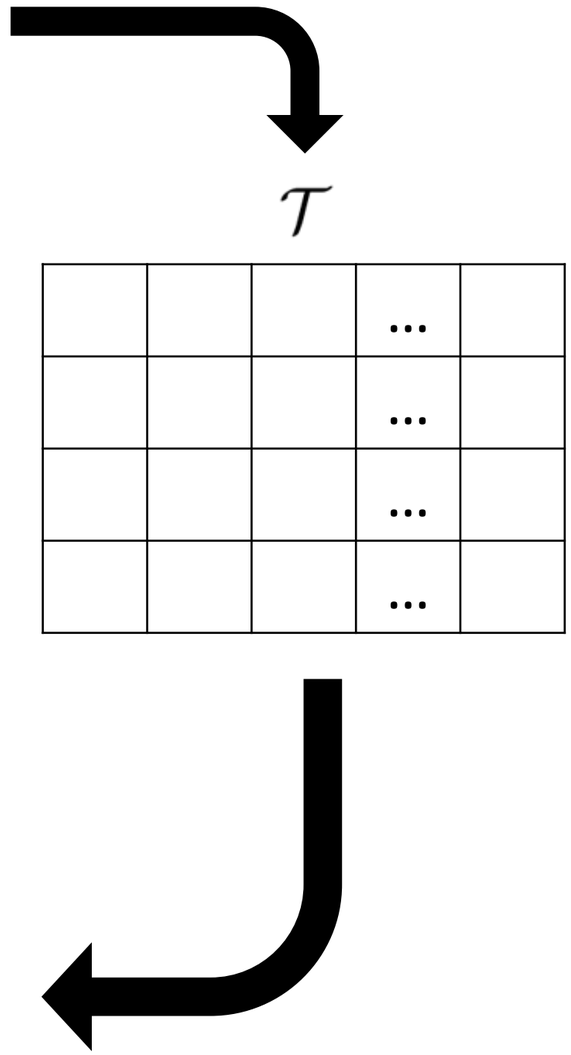
Require: base point P over $\mathcal{E}(\mathbb{F}_p)$ of order ord
Ensure: $r \times c$ pre-computation table $\mathcal{T}$

1: Initialize $r \times c$ table $\mathcal{T}$
2: **for** $j \in 0$ up to $c-1$ **do**
3: **for** $i \in 0$ up to $r-1$ **do**
4: Choose $\alpha_{i,j} \in \mathbb{Z}_{ord}$ that is appropriate for the current table-based scalar multiplication
5: $\mathcal{T}[i,j] \leftarrow \alpha_{i,j} \cdot P$
6: **end for**
7: **end for**
8: Return $\mathcal{T}$

Algorithm 3 Regular table-based scalar multiplication

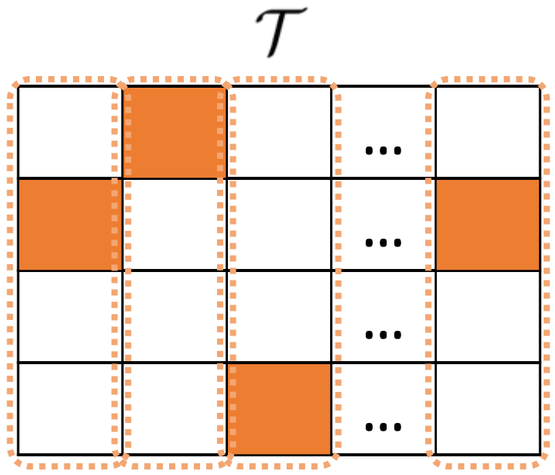
Require: k and $r \times c$ pre-computation table $\mathcal{T}$
Ensure: $Q = k \cdot P$

1: Set $ks(k) = (ks_0, ks_1, ..., ks_{c-1})$ and ws to be appropriate for the current table-based scalar multiplication
2: $Q \leftarrow \infty$
3: **for** $j \in 0$ up to $c-1$ **do**
4: $Q \leftarrow ws(ks_j) \cdot Q$
5: $Q \leftarrow Q + \mathcal{T}[ks_j, j]$
6: **end for**
7: $k \leftarrow k \ mod \ ord$
8: Return k, Q



Background

- Regular Table-based Scalar Multiplication



$ks(k) = (1,0,3,\dots,1)$

$$Q_0 = ws(ks_0) \cdot \infty + \mathcal{T}[ks_0, 0]$$

$$Q_1 = ws(ks_1) \cdot Q_0 + \mathcal{T}[ks_1, 1]$$

$$Q_2 = ws(ks_2) \cdot Q_1 + \mathcal{T}[ks_2, 2]$$

$$\vdots$$

$$Q_{c-1} = ws(ks_{c-1}) \cdot Q_{c-2} + \mathcal{T}[ks_{c-1}, c-1]$$

$$kP = ws(ks_{c-1})(\dots(ws(ks_1)(ws(ks_0) \cdot \infty + \mathcal{T}[ks_0, 0]) + \mathcal{T}[ks_1, 1])\dots) + \mathcal{T}[ks_{c-1}, c-1]$$

$$\begin{aligned} kP &= W_0 \cdot \mathcal{T}[ks_0, 0] + W_1 \cdot \mathcal{T}[ks_1, 1] + \cdots + W_{c-1} \cdot \mathcal{T}[ks_{c-1}, c-1] \\ &= W_0 \cdot T_0 \cdot P + W_1 \cdot T_1 \cdot P + \cdots + W_{c-1} \cdot T_{c-1} \cdot P \end{aligned}$$

Proposed Attack

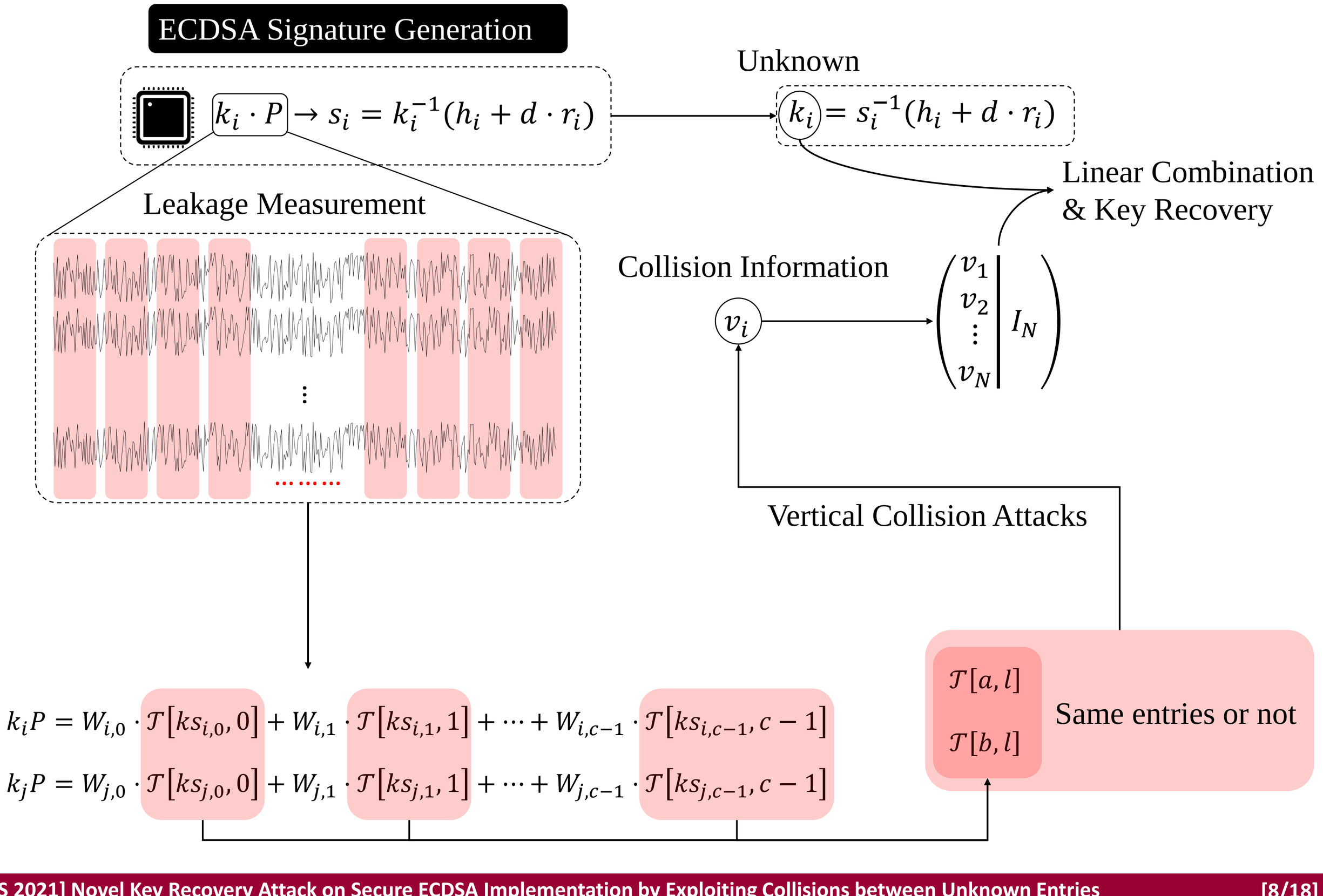
- Assumptions

  1. ws is a function of index j

     - Doubling operation depends only on loop index

  2. Side-channel attacker can cause multiple ECDSA signature generations and collect corresponding traces with condition that private key d and table $\mathcal{T}$ are fixed

Proposed Attack



Proposed Attack

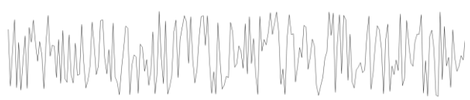
- Step 1 : Preparation of Collision Information

$\mathcal{T}$

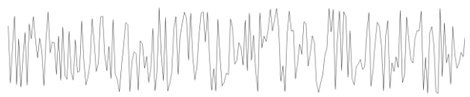
1	4	7
2	5	8
3	6	9

Scalar multiplication

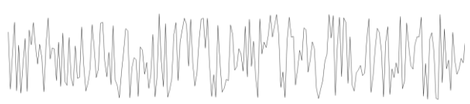
Vertical Clustering



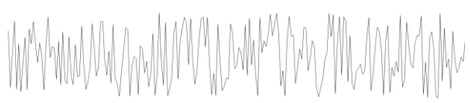
a d g $\quad k_1 = s_1^{-1}(h_1 + r_1 d)$



b e h $\quad k_2 = s_2^{-1}(h_2 + r_2 d)$

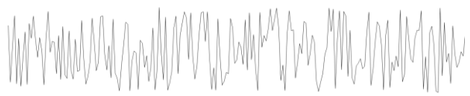


a f i $\quad k_3 = s_3^{-1}(h_3 + r_3 d)$



b e h $\quad k_4 = s_4^{-1}(h_4 + r_4 d)$

⋮ ⋮

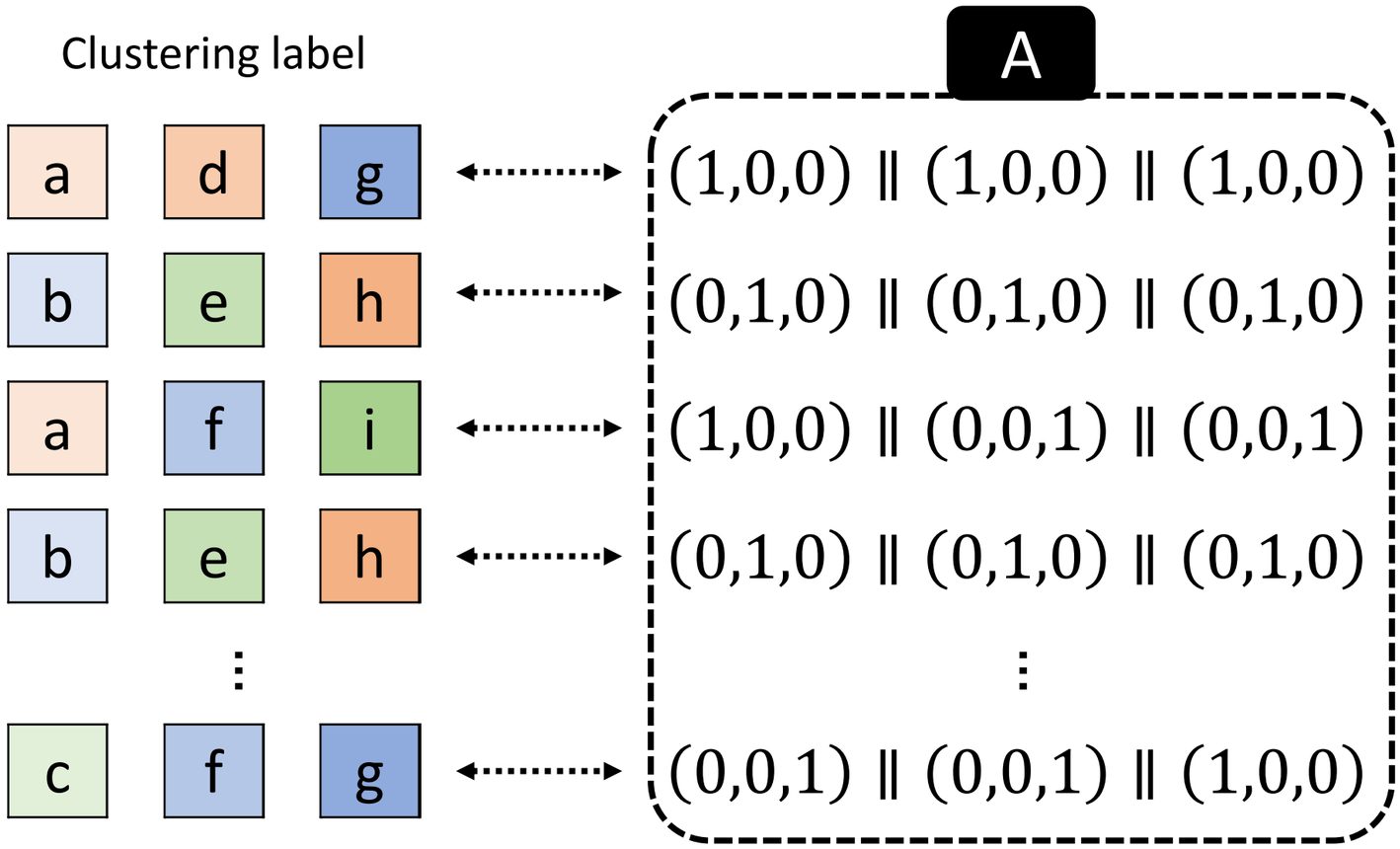


c f g $\quad k_N = s_N^{-1}(h_N + r_N d)$

Goal : Find Linear Dependency

Proposed Attack

- Step 2 : Key Recovery by Identifying Linearly Dependent Nonces
  - Attacker has no information on entries of pre-computation table
  - Convert clustering label into one-hot representation
    - Component of converted vector represents whether usage of each entry of table
  - Find linearly dependency of nonces using converted vector



$$M = [A \parallel I_N] \in Mat_{N\times(r\cdot c+N)}(\mathbb{Z}_{ord})$$

$$\downarrow \textbf{Gaussian elimination}$$

$$M' = [A' \parallel B] \in Mat_{N\times(r\cdot c+N)}(\mathbb{Z}_{ord})$$

$$\begin{aligned} 0 &= \sum_i b_{g,i}\cdot k_i \\ &= \sum_i b_{g,i}\cdot(s_i^{-1}\cdot h_i + s_i^{-1}\cdot r_i\cdot d) \\ &= \sum_i (b_{g,i}\cdot s_i^{-1}\cdot h_i) + (\sum_i b_{g,i}\cdot s_i^{-1}\cdot r_i)\cdot d \\ \Leftrightarrow d &= \left(\sum_i b_{g,i}\cdot s_i^{-1}\cdot r_i\right)^{-1}\cdot\left(-\sum_i b_{g,i}\cdot s_i^{-1}\cdot h_i\right) \end{aligned}$$

Proposed Attack

Cryptographic Algorithm Laboratory

- Discussion on the Proposed Attack
  - This attack can be possible even when there is no information on entries
    - Attacker can use collision between unknown entries
  - For finding linear dependency, there must be no error in clustering
  - If errors occur, the only trial-and-error solution is possible in present
    - Choose attack traces from a sufficient pool of traces until attack success

Proposed Attack

- Discussion of Issues regarding the Proposed Attack
  - Probability of incorrect clustering between different entries
    - Point $\mathrm{k} \cdot P = (X, Y)$ be determined by scalar k
    - Let 256-bit scalar is loaded 32-bit-wisely and ideal leakage assumption
      - i.e., Hamming weight model
    - Then, the target leakage consists of 8 points

- The probability of two Hamming weights of the two different words being equal

$$p = \frac{1}{2^{64}} \sum_{i=1}^{31} \binom{32}{i} \cdot \left(\binom{32}{i} - 1 \right)$$

- The probability of two leakages being equal when two different entries are chosen

$$P = \sum_{i=1}^{8} \binom{8}{i} \cdot p^i \cdot \left(\frac{1}{2^{32}} \right)^{8-i}$$

Case Studies

- Fixed-Base Comb Scalar Multiplication
  - Representative table-based scalar multiplication
  - For the sake of simplicity, we consider the only original version

$$ws(x) = 2 \text{ for any } x, \quad r = 2^w, \quad c = d$$

Table 1: For the fixed-base comb method, the size, the number of entries in the pre-computation table, and attack parameter according to security parameter.

secp256r1 [Bro10] (128-bit security)		Pre-computation table		Attack phase	
w	d	Size	# of entries	# of group per loop	Dimension of v_i
2	128	128 B	4	4	512
4	64	512 B	16	16	1,024
8	32	8 KB	256	128	8,192
16	16	2 MB	65,536	65,536	1,048,576

Case Studies

Cryptographic Algorithm Laboratory

- T_SM Scalar Multiplication
  - designed to be resistant against STA
  - outputs both random nonce k and corresponding result point $k \cdot P$
  - employed only in specific settings such as ECDSA signature generation
  - All entries in table is chosen randomly

Algorithm 6 T_SM scalar multiplication [SCM+18]

Require: security parameter $\lambda = m \cdot n \in \mathbb{Z}^+$, the order ord of base point $P \in \mathcal{E}(\mathbb{F}_q)$, pre-computation tables $\mathcal{T}_k$ and $\mathcal{T}_P$

Ensure: $k, Q = k \cdot P$

1: $k \leftarrow 0, Q \leftarrow \infty$
2: **for** $j \in 0$ up to $n-1$ **do**
3: $\quad row \xleftarrow{R} \mathbb{Z}_{2^m}$ // Choose an m-bit random integer
4: $\quad k \leftarrow k + \mathcal{T}_k[row, j]$
5: $\quad Q \leftarrow Q + \mathcal{T}_P[row, j]$
6: **end for**
7: $k \leftarrow k \ mod \ ord$
8: Return k, Q

Case Studies

- T_SM Scalar Multiplication

$$ws(x) = 1 \text{ for any } x, \;\; r = 2^m, \;\; c = n$$

Table 2: The size, the number of entries of pre-computation table, and attack parameter according to security parameter of T_SM method.

$\lambda = m \cdot n = 256$		Pre-computation table		Attack phase	
m	n	Size	# of entries	# of groups per loop	Dimension of v_i
2	128	48 KB	512	4	512
4	64	96 KB	1,024	16	1,024
8	32	768 KB	8,192	128	8,192
16	16	96 MB	1,048,576	65536	1,048,576

Experiment

- A proof of work
  - 256-bit T_SM scalar multiplication for security parameter $\lambda = 2 \times 128$

  $\Rightarrow$ parameters of table size : $r = 4$ & $c = 128$

- Setup
  - Chipwhisperer CW308T-STM32F target board, 5MHz
  - LeCroy HDO6104A oscilloscope, 250 Msamples/s
  - Collect $513 (= 4 \times 128 + 1)$ power consumption traces

Experiment

- Identification and Extraction of Target Operation Traces

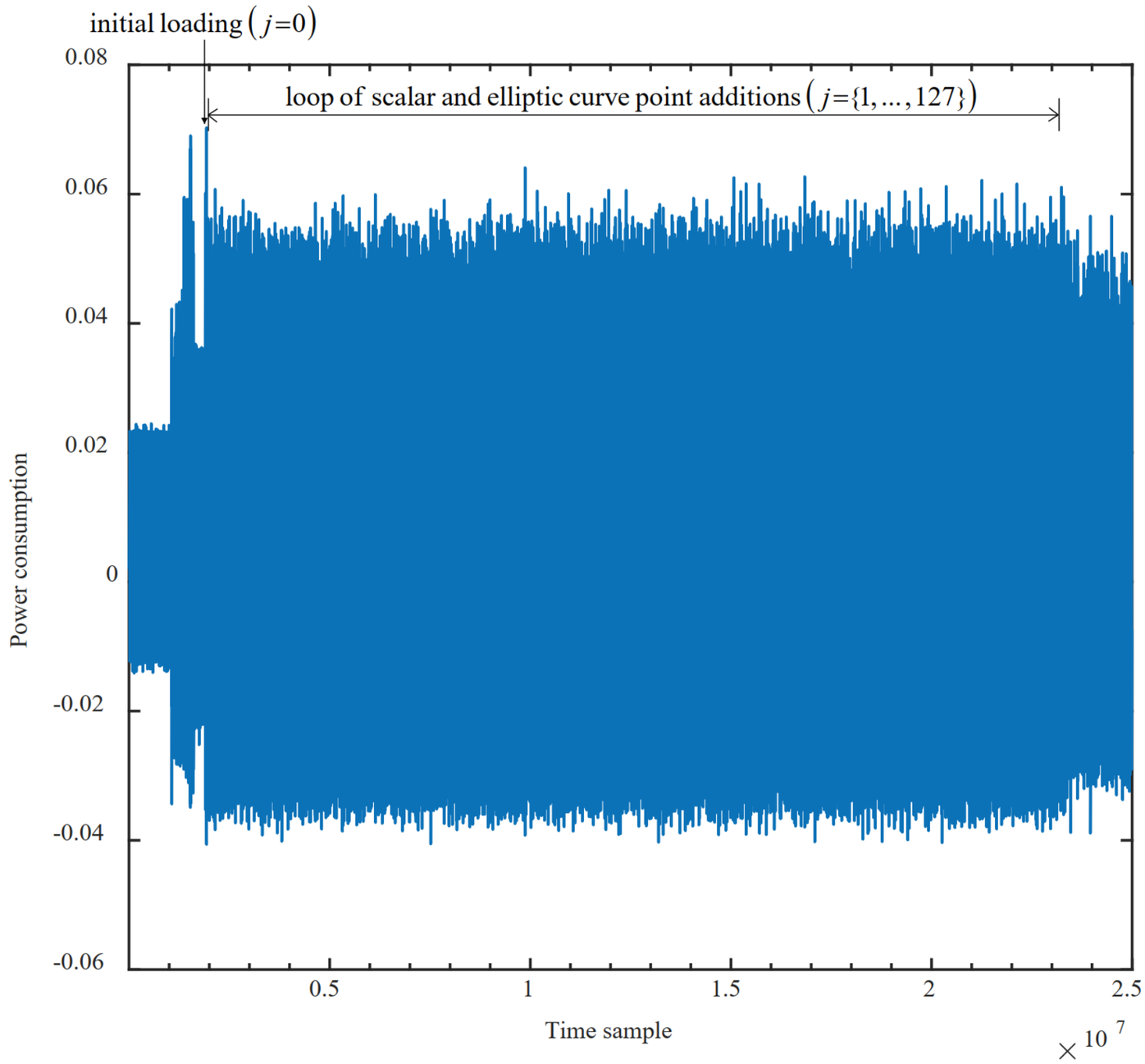
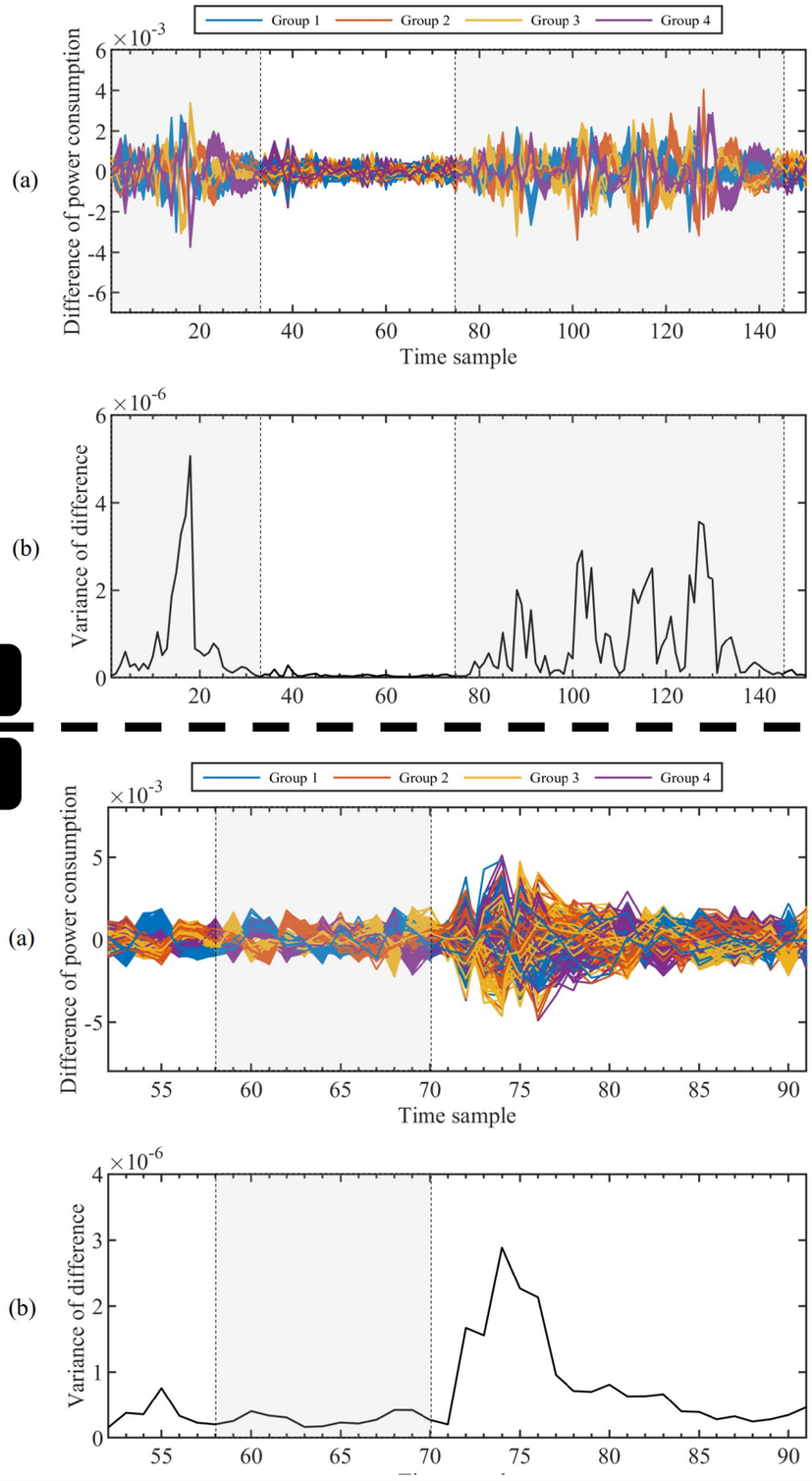


Figure 3: Power consumption trace for T_SM scalar multiplication

Experiment

- Identification and Extraction of Target Operation Traces

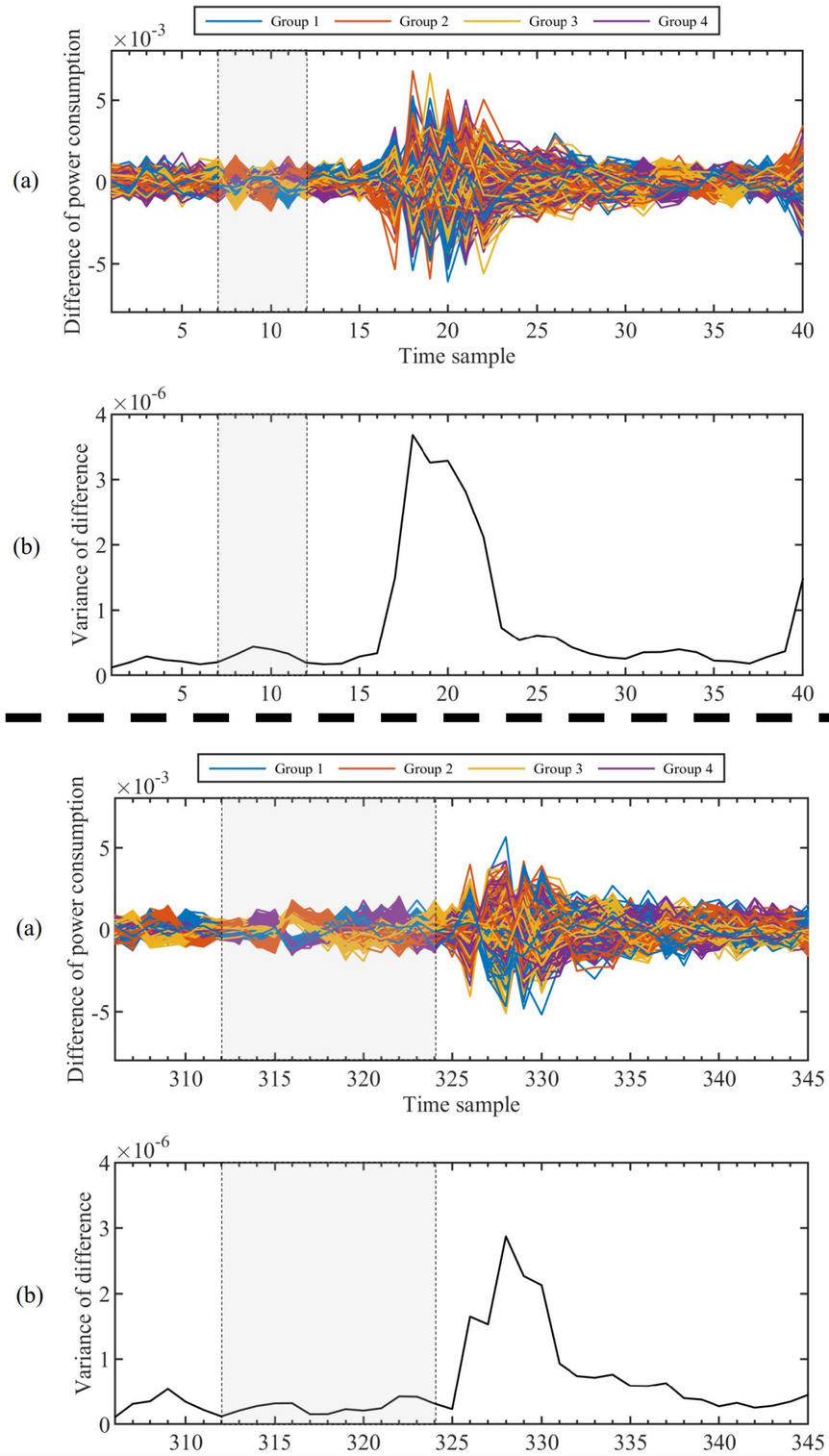
Actual clustering
Variance trace



j=0

j=127

1st long integer multiplication



Long integer addition

j=127

j=127

2nd long integer multiplication

Q&A

Thanks

sunghyunjin@korea.ac.kr