FSE 2023 (Beijing and Kobe)

# Hybrid Code Lifting on Space-Hard Block Ciphers

## --Application to Yoroi and SPNbox--

NTT Social Informatics Laboratories, University of Hyogo

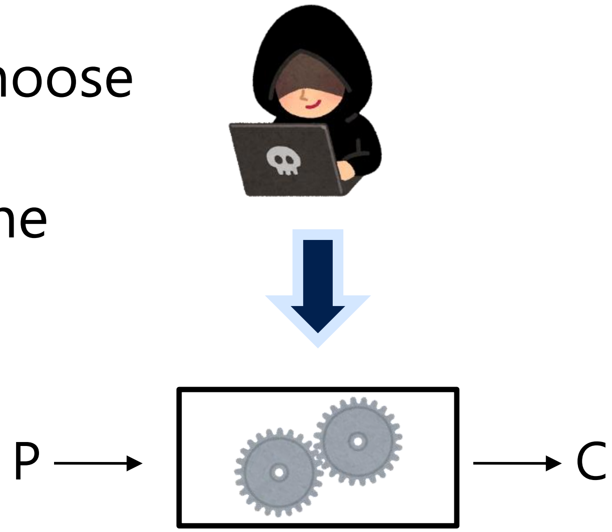
Yosuke Todo and Takanori Isobe

# Whitebox cryptography

NTT

### Blackbox attack

- An attacker can observe/choose plaintexts and ciphertexts.
- The attacker never watch the inside of the encryption.




Block cipher

### Whitebox attack

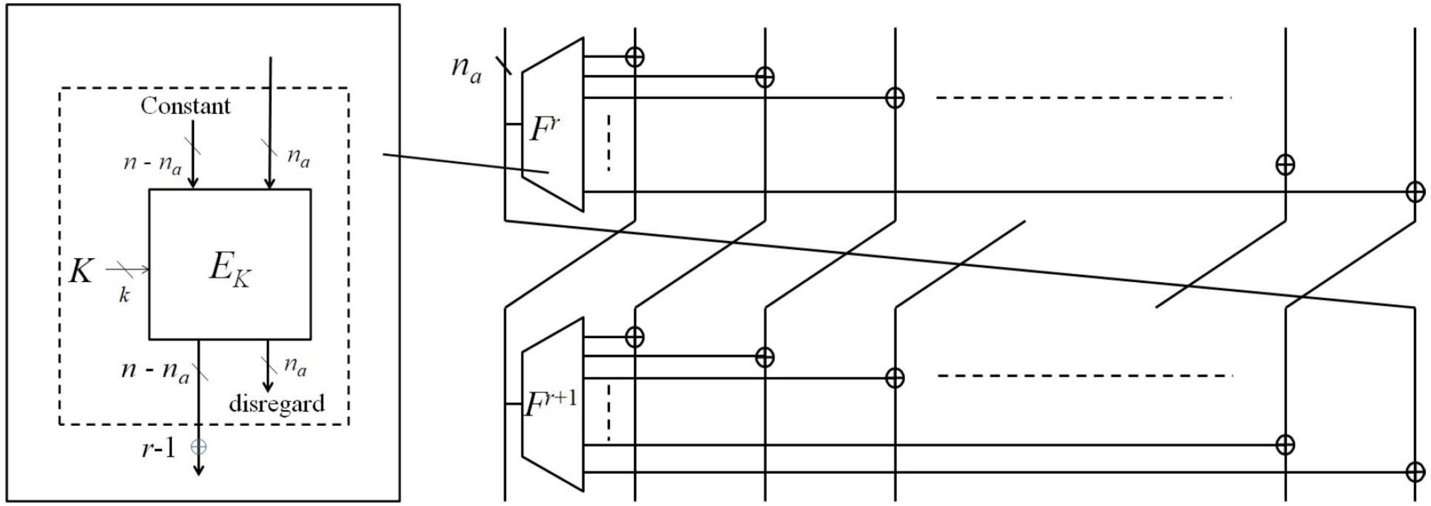
- An attacker can observe everything including the inside of the encryption.
- Demanded security when the encryption can be used on untrusted environment.

**Goal of whitebox block ciphers**

- Primary goal is to resist the key extraction attack.
- Secondary goal is to resist the code lifting.

# Space-hard block ciphers

- Proposed by Bogdanov and Isobe at CCS 2015.




Block cipher, SPACE.

- *Security against key extraction attack.*
  - Extracting the short secret key is as difficult as the blackbox attack against AES.
- *Security against code lifting.*
  - So-called space hardness.

# Space hardness

- (M,Z)-space hardness
  - The implementation of a block cipher is (M,Z)-space hard if it is infeasible to encrypt (decrypt) any randomly drawn plaintext (ciphertext) with probability higher than $2^{-Z}$ given any code (tale) of size less than M.

- Attack models
  - **Known space (KS)** leaks *M* table entries randomly.
  - **Chosen space (CS)** leaks *M* chosen table entries.
  - **Adaptive chosen space (ACS)** leaks *M* adaptively chosen table entries.
  - **Arbitrary** leakage.

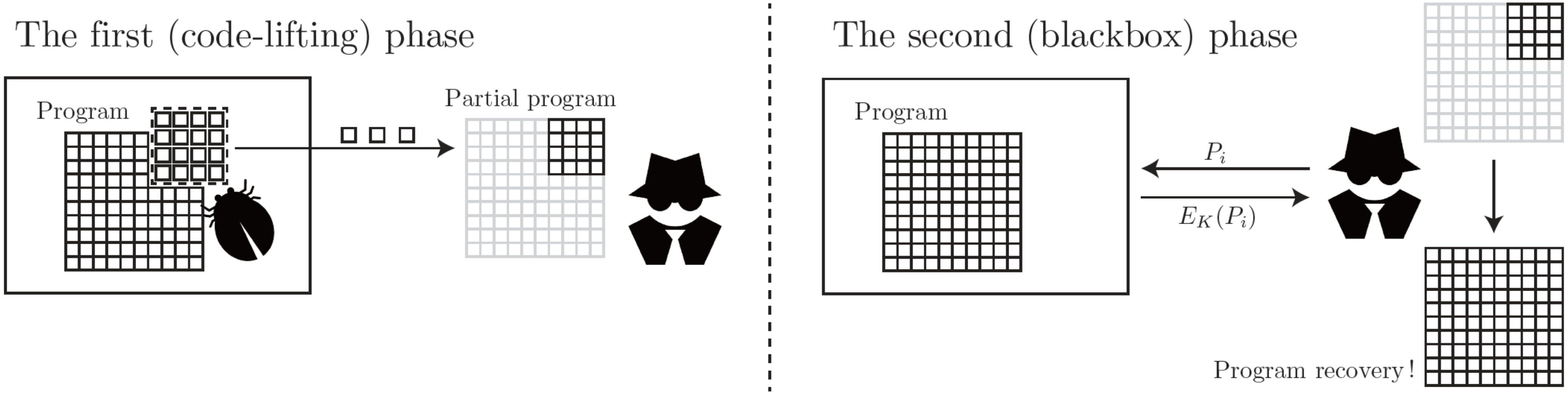
# Behind intention of space hardness

NTT

- Even if a whitebox attacker can successfully extract the M code from the implementation, the attacker can't imitate the cipher.

- Is this intention true??
  - Space hardness doesn't suppose the blackbox attacker receiving the leakage.
  - It doesn't satisfy the intention if slight leakage allows the blackbox attacker to recover the full program!!

# Hybrid scenario




- The first phase is code lifting by a whitebox attacker.
  – The attacker analyzes the implementation like “known-key(table) attack”, and outputs leakage whose size is up to M.
- The second phase is a classical blackbox attacker.
  – They can exploit the leakage generated by the whitebox attacker.

# Let’s discuss hybrid scenario

- Yoroi (from CHES2021)
  - Yoroi has very unique functionality called longevity.
  - The implementation is updatable while maintaining the functionality.

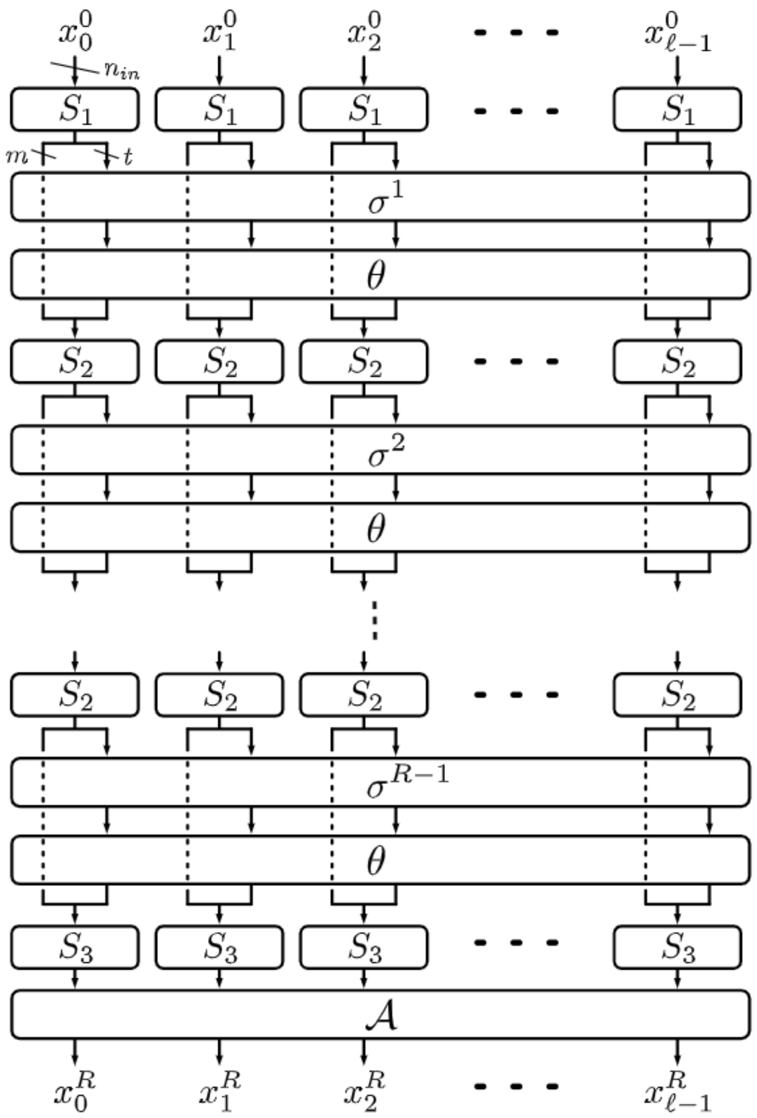
- SPNbox (from Asiacrypt2016)
  - As far as we know, SPNbox is the most efficient space-hard ciphers.
  - In other words, it doesn’t have enough security margin.

**We consider a new attack model**
**taking the intention of the space hardness into consideration.**

**Note that the authors of existing ciphers don't claim such security.**

# Security of Yoroi with hybrid scenario

# Yoroi




- Three S-boxes, S1, S2, and S3 are used.
- $\sigma$ is constant addition
- $\theta$ is the multiplication of the MDS.
  - $\sigma$ and $\theta$ are only applied to the last t bits.
- Finally, AES $\mathcal{A}$ is applied.

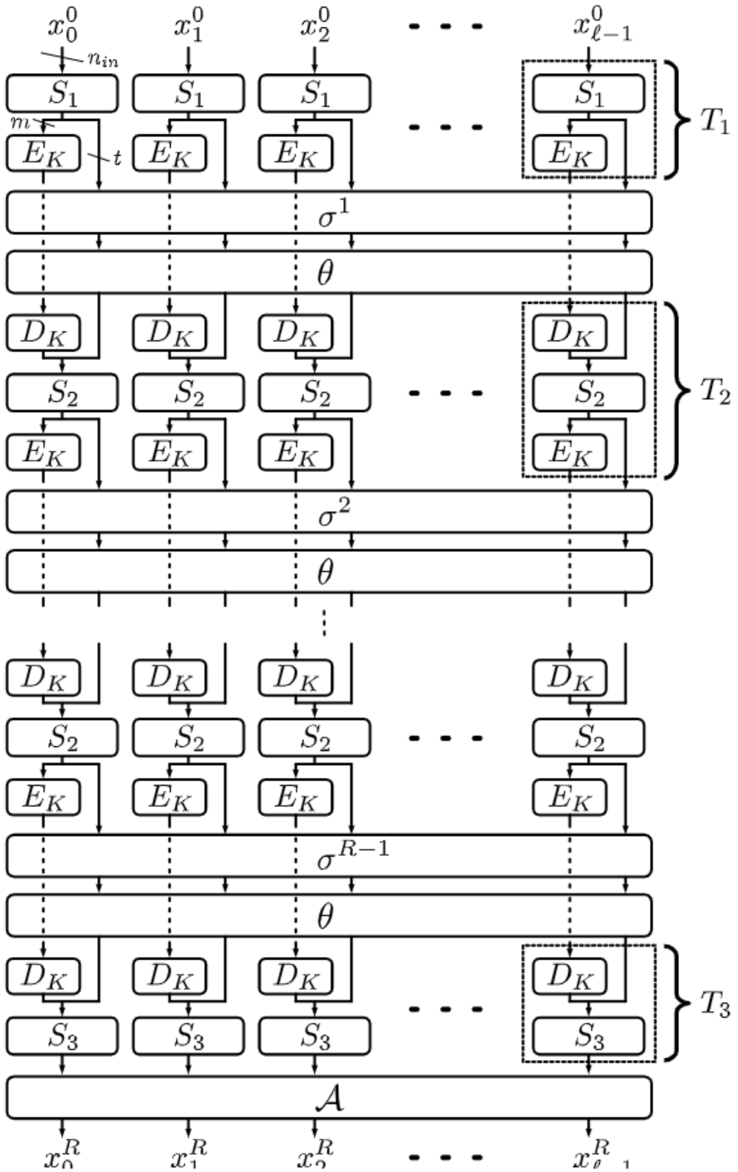
- Security claims.
  - 128-bit security against blackbox attacker.
  - 128-bit security against key extraction.
  - $\left(\frac{2^{n_{in}}}{4}, 128\right)$-space hard against KSA (the ability of the whitebox attacker is limited to random table entry extraction.).

# Unique property: longevity of Yoroi

- Yoroi was designed to aim for the unique property, longevity.

- Longevity: updatable implementation.
  - The functionality is maintained.
  - Once the implementation (table) is updated, attackers need to re-leak the updated table from the beginning to copy the functionality.
  - It can be promising countermeasure against the following attack.
    - Leak slight data every day such that it’s not detectable by anomaly detection.
    - Leak much data by spending many days.
      - e.g., 10MB / day. Then, we can collect 1GB in 100 days.

# Yoroi – How to update implementation



- Apply m-bit block cipher $E_K$ to top m bits of output of each S-box.
- They are cancelled out in the next round.

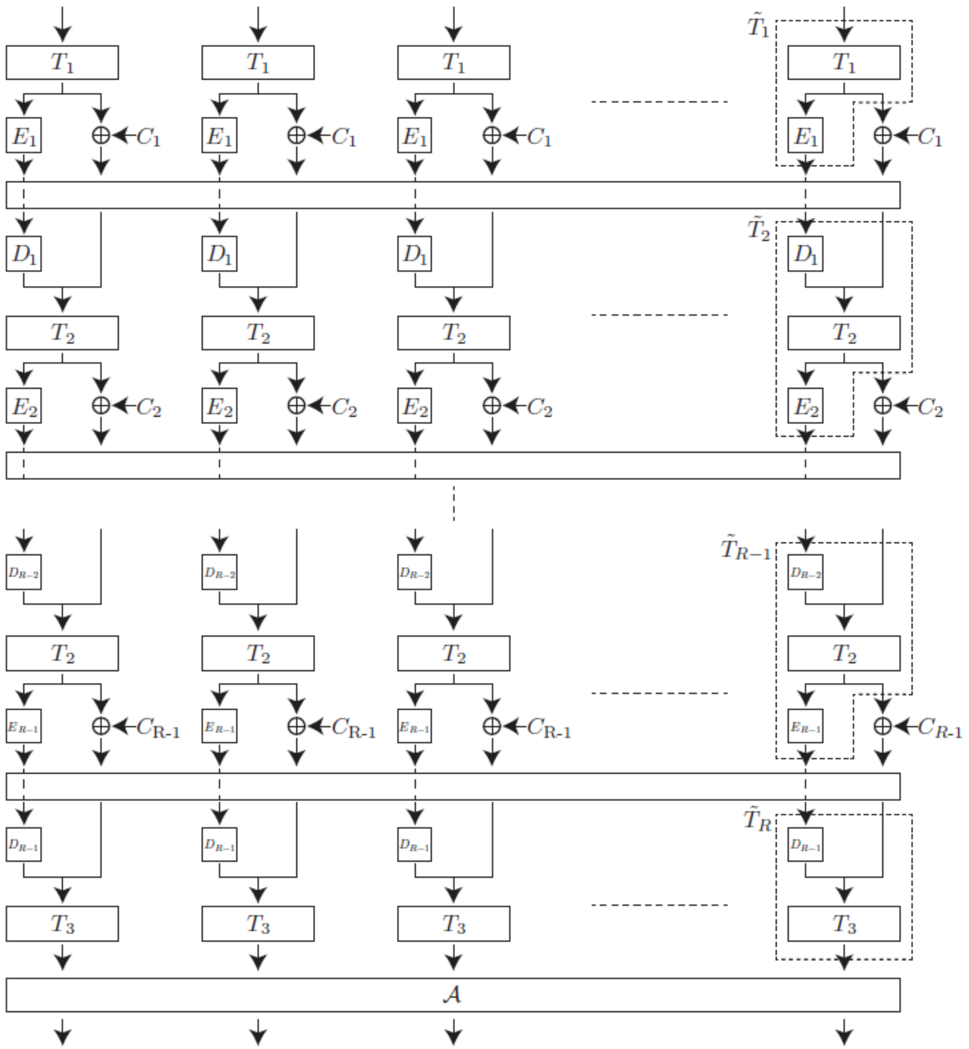
- Security claim.
  - $\left(\frac{2^{n_{in}}}{64}, 128\right)$-space hard against KSA each table update.

# Attack overview

- Canonical representation
  - Introduce a **canonical representation of Yoroi** that can be quickly reconstructed from implementation.
- Leakage
  - Leak the AES key (128 bits) and slight table entries.
- Blackbox attack using the leakage
  - Construct efficient truncated differential.
  - Recover table entries of the canonical representation.

# Canonical representation of Yoroi

NTT




- We choose $E_r$ such that $\widetilde{T}_r$ satisfies Property 1.

**Property 1**

$x_i \xrightarrow{\widetilde{T}_r} y_i$ (m bits) | fixed (t bits)

$x_j \xrightarrow{\widetilde{T}_r} y_j$ (m bits) | fixed (t bits)

If $x_i < x_j$, $y_i < y_j$ holds.

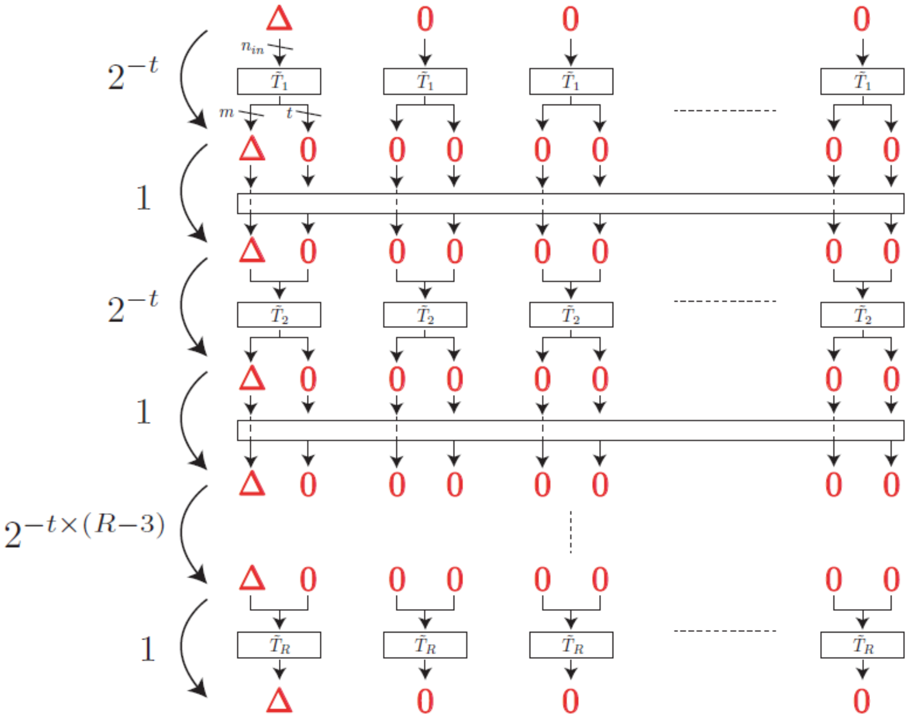
- The converted table is unique independent of the table update.

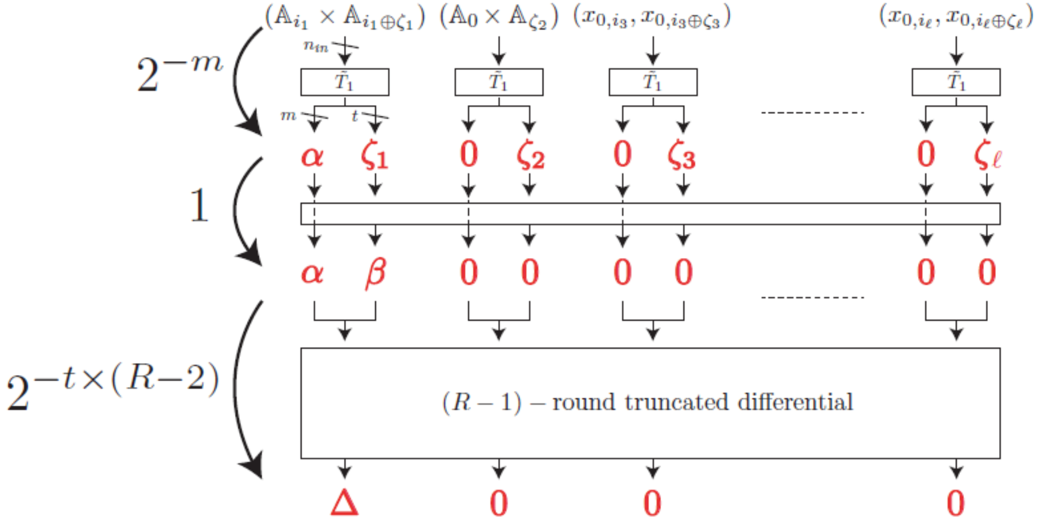
**Yoroi has a unique canonical representation independent of the implementation.**

# How to recover the canonical representation?

NTT

- High-probability truncated differential allows us to detect the partial collision of each table entry.
- It's useful to recover the table of the canonical representation.



# Summary of results

NTT

**Table 1:** Summary of hybrid code lifting on YOROI and SPNBOX.

| target | code-lifting phase time | code-lifting phase leak bit size (ratio) | blackbox phase complexity‡ | remark | reference |
|---|---|---|---|---|---|
| YOROI-16 | $2^{18.8}$ | 800 ($2^{-11.94}$) | $2^{33}$ | verified practically | Sect. 5 |
| YOROI-32 | $2^{35.9}$ | 3008 ($2^{-27.03}$) | $2^{65.5}$ | | Sect. 5 |
| SPNBOX-16 | $2^{14}$ | $16 \times 2^{14}$ (1/4) | $2^{124.09}$ | | Sect. 7 |
| SPNBOX-24 | $2^{22}$ | $24 \times 2^{22}$ (1/4) | $2^{102.27}$ | | Sect. 7 |
| SPNBOX-32 | $2^{30}$ | $32 \times 2^{30}$ (1/4) | $2^{95.84}$ | | Sect. 7 |

Complexity‡ represents the time and data complexities to recover the encryption program from the leaked information.

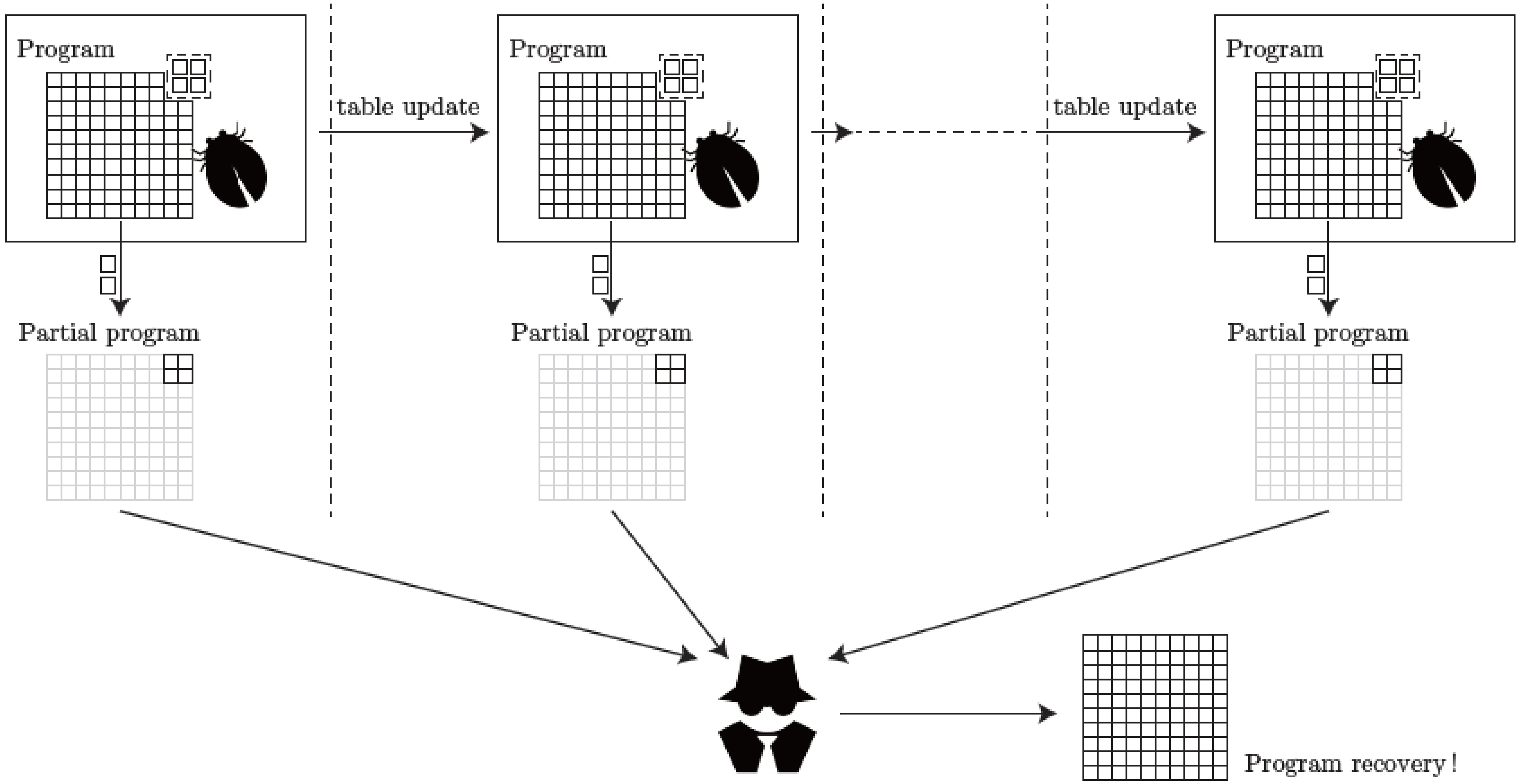
- Only 800-bit leakage (the ratio is $2^{-11.94}$) is enough to recover the full program of Yoroi-16 with practical time complexity!!
- SPNbox is not catastrophic like Yoroi, but impossible to maintain 128-bit security.

# Attack against Longevity

# Motivation

NTT

- Hybrid attack doesn't break the authors' security claim.
- Discuss the longevity, which was the design motivation of Yoroi.



# Three leakage assumptions

- Arbitrary leakage.
  - Just copy the old program and leak it.
  - It's **impossible** to ensure such security in general.

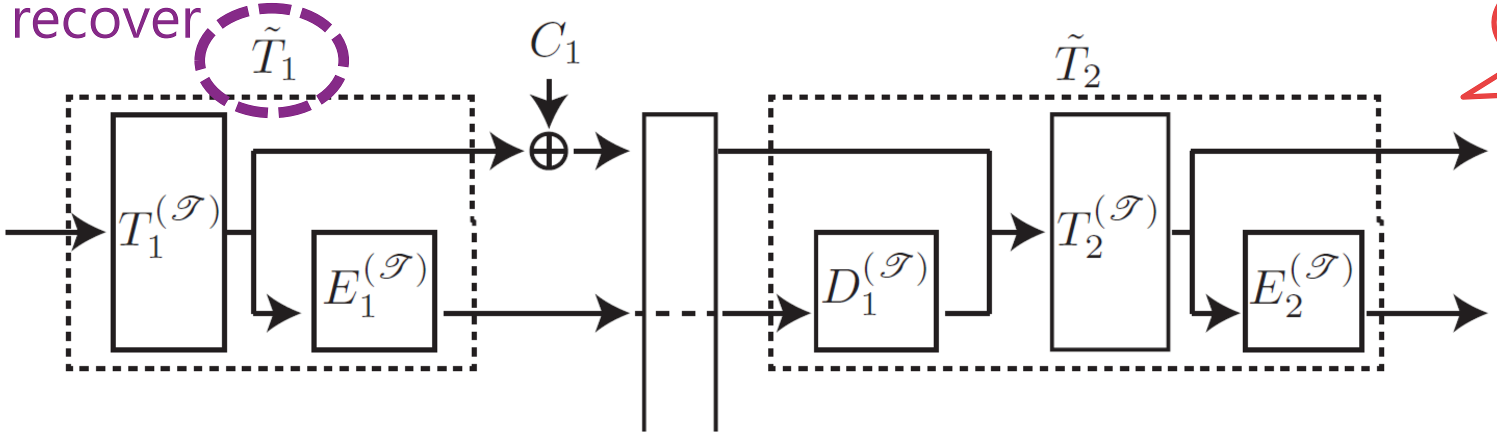
- Arbitrary leakage without non-volatile memory.
  - Compute the unique canonical representation and leak it.
  - Since Yoroi has the canonical representation,
    it's **impossible** to ensure such security.

- KSA leakage.
  - Designers' claim.
  - Is it possible to recover the full program only by this assumption?

# Attack by KSA leakage: recover $\tilde{T}_1$

Try to recover

Use the canonical representation again.

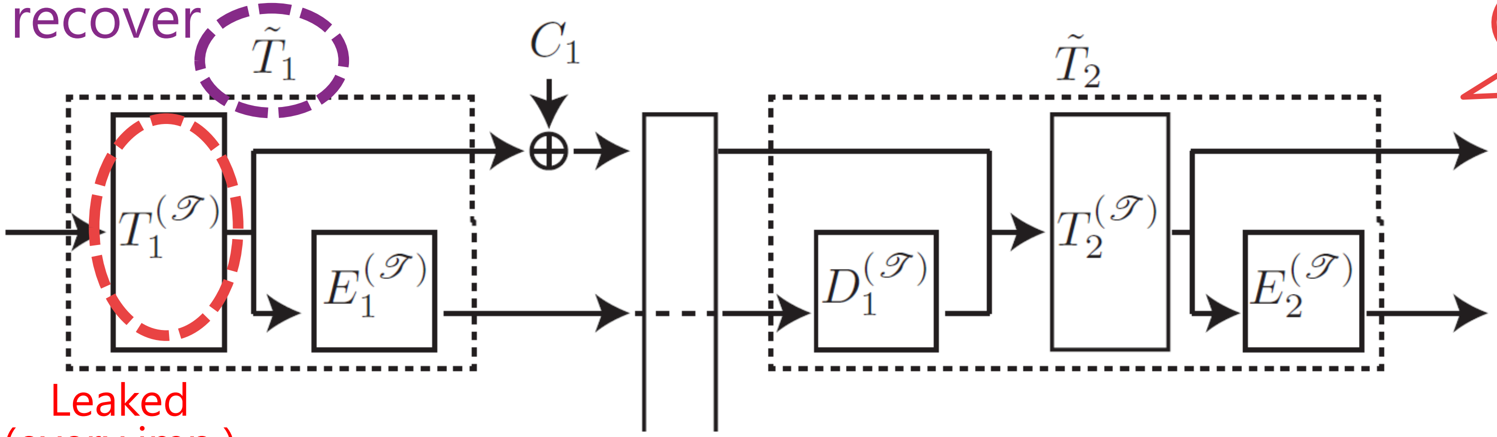


# Attack by KSA leakage: recover $\tilde{T}_1$

NTT

Use the canonical representation again.

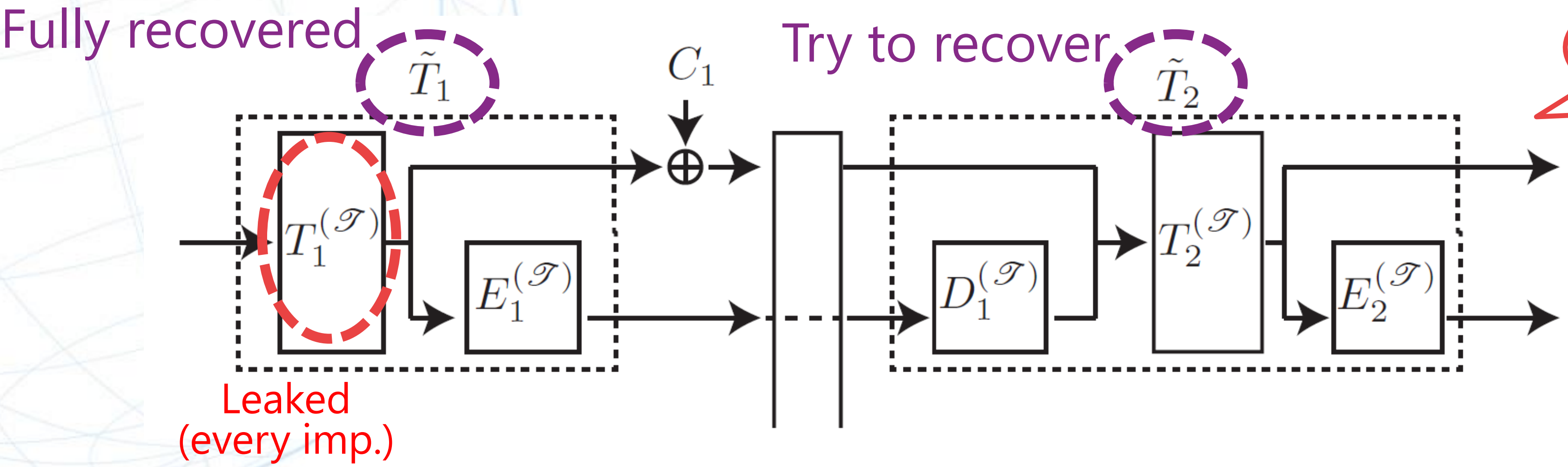
Try to recover




Leaked
(every imp.)

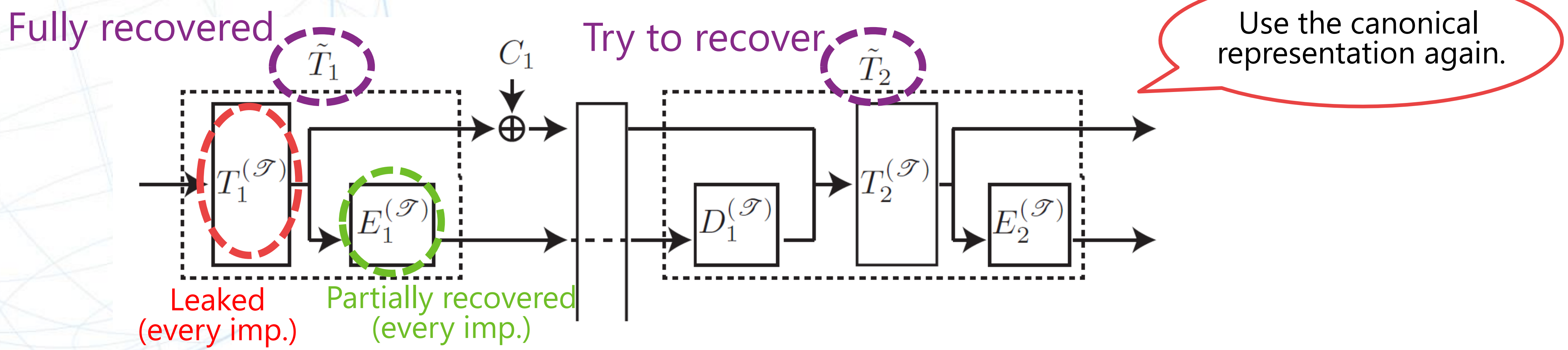
1. Observe the partial entries of $T_1^{(\mathcal{T})}$ as leakage.
   - The canonical representation, $\tilde{T}_1$, is independent of $E_1^{(\mathcal{T})}$.
   - It's not difficult to recover $\tilde{T}_1$.

# Attack by KSA leakage: recover $\tilde{T}_2$



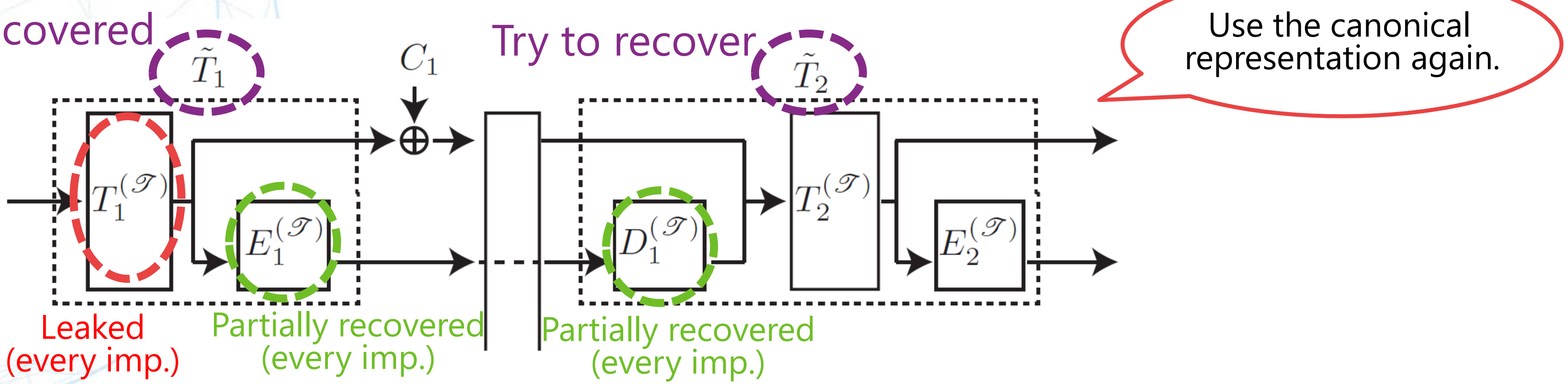

Use the canonical representation again.

# Attack by KSA leakage: recover $\tilde{T}_2$



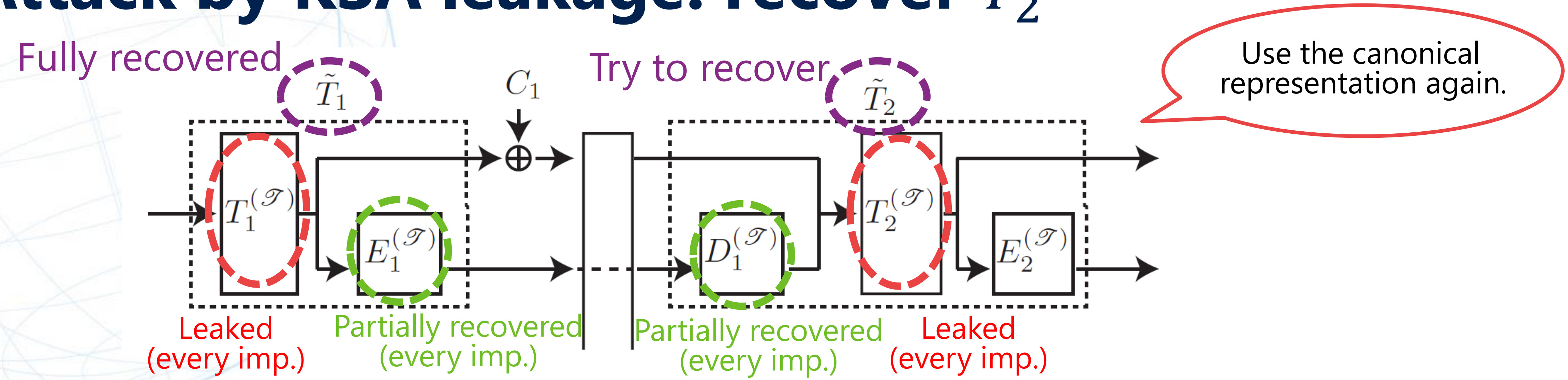

1. Recover the partial entries of $E_1^{(\mathcal{T})}$ by using $\tilde{T}_1$ and $T_1^{(\mathcal{T})}$ (leakage).

# Attack by KSA leakage: recover $\tilde{T}_2$




1. Recover the partial entries of $E_1^{(\mathcal{T})}$ by using $\tilde{T}_1$ and $T_1^{(\mathcal{T})}$ (leakage).
2. Get the partial entries of $D_1^{(\mathcal{T})}$.

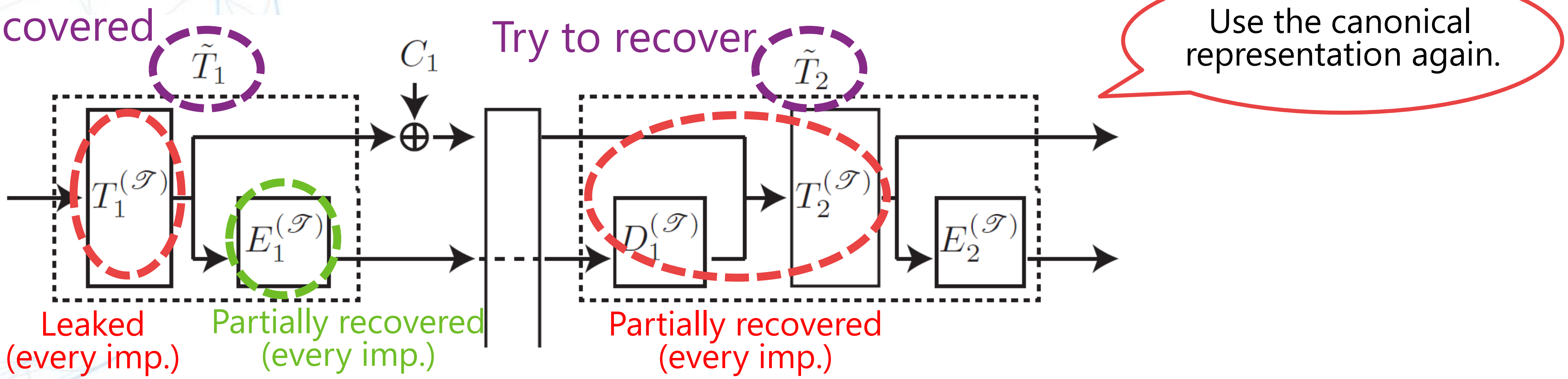
# Attack by KSA leakage: recover $\tilde{T}_2$




1. Recover the partial entries of $E_1^{(\mathcal{T})}$ by using $\tilde{T}_1$ and $T_1^{(\mathcal{T})}$ (leakage).
2. Get the partial entries of $D_1^{(\mathcal{T})}$.
3. Observe the partial entries of $T_2^{(\mathcal{T})}$ as leakage.

# Attack by KSA leakage: recover $\tilde{T}_2$

NTT




1. Recover the partial entries of $E_1^{(\mathcal{T})}$ by using $\tilde{T}_1$ and $T_1^{(\mathcal{T})}$ (leakage).
2. Get the partial entries of $D_1^{(\mathcal{T})}$.
3. Observe the partial entries of $T_2^{(\mathcal{T})}$ as leakage.
4. Get the partial entries of $T_2^{(\mathcal{T})} \circ (I || D_1^{(\mathcal{T})})$.
   - The canonical representation, $\tilde{T}_2$, is independent of $E_2^{(\mathcal{T})}$.
   - It's not difficult to recover $\tilde{T}_2$.

# Summary of attacks

Table 2: Summary of attacks on the longevity of Yoroi.

| target | code-lifting phase model | time | #updates | complexity‡ | remark | reference |
|---|---|---|---|---|---|---|
| Yoroi-16 | arbitrary† | $2^{18.8}$ | 171 | negl. | | Sect. 6.2 |
| Yoroi-32 | arbitrary† | $2^{35.9}$ | 342 | negl. | | Sect. 6.2 |
| Yoroi-16 | known space | - | $2^{35.97}$ | $2^{48.78}$ | break claimed security | Sect. 6.3 |
| Yoroi-32 | known space | - | $2^{68.95}$ | $2^{98.86}$ | break claimed security | Sect. 6.3 |

Arbitrary† represents whitebox adversaries w/o nonvolatile memory.
Complexity‡ represents the time complexity to recover the encryption program from collected leakages, and a query is not required.

# Conclusion

- We propose the hybrid code lifting and demonstrate the impact.
- We break the security claim about the longevity of Yoroi.
  - With complexity of $2^{48.78}$, we can recover the full program.
- Countermeasure?
  - Increasing number of rounds.
    - It's useful only for the attack using the KSA leakage.
    - It's difficult to ensure the security on not only arbitrary but also ACSA leakage.
- The open question is how to design an updatable space-hard cipher, ensuring security against arbitrary leakage.