The Security Framework of White-Box Cryptography Revisited

Yufeng Tang, Tao Sun, Zheng Gong
cis.gong@gmail.com

[1]School of Computer Science, South China Normal University
[2]Mobile Applications And Security Engineering Center of Guangdong Province

April 29, 2021

Cryptography: the very beginning

"Information theory is about communication in the presence of noise."

-C. Shannon, 1948.[1]

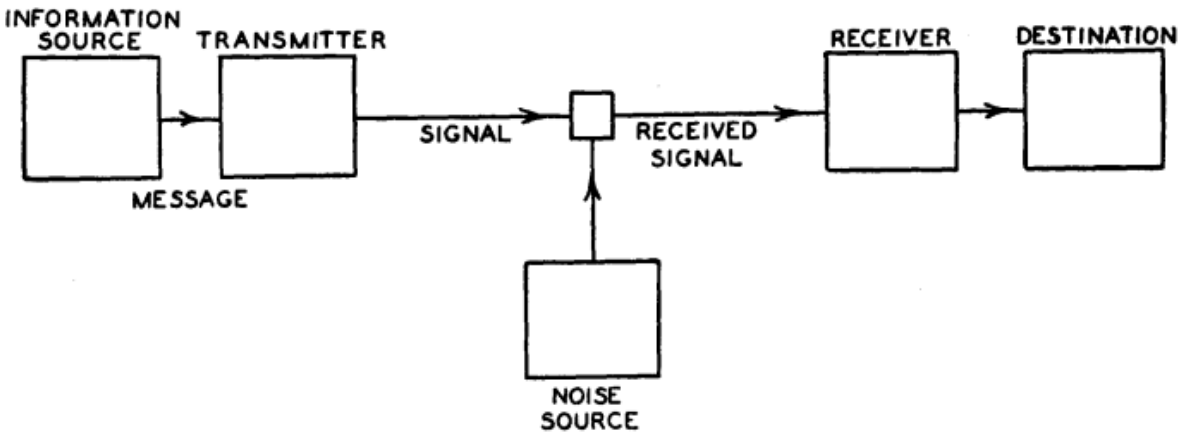


Fig. 1—General communications system.

[1] http://fab.cba.mit.edu/classes/S62.12/docs/Shannon_noise.pdf

Cryptography: an informal definition

"Cryptography is about communication in the presence of adversaries."

-R. Rivest

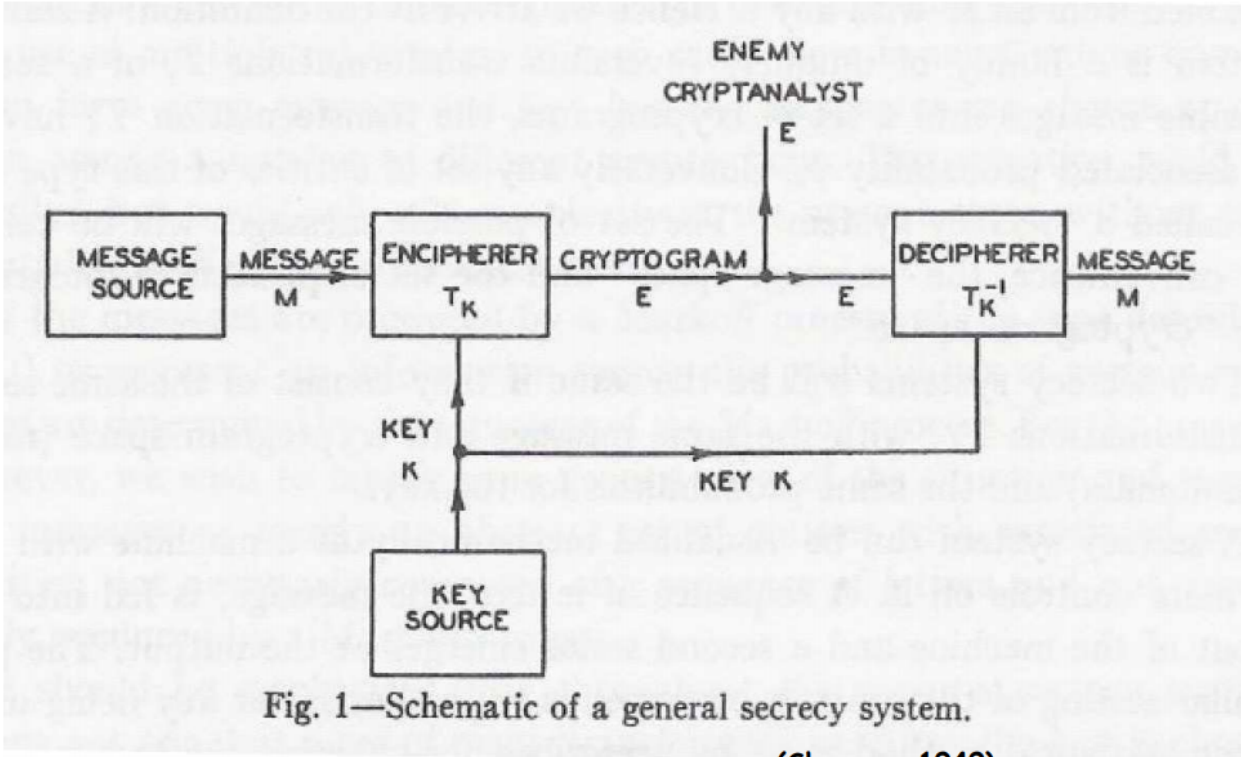


Fig. 1—Schematic of a general secrecy system.

(Shannon, 1949)

Define cryptography in "a mathematically acceptable way"

"As a first step in the mathematical analysis of cryptography, it is necessary to idealize the situation suitably, and to define in a mathematically acceptable way what we shall mean by a secrecy system."

-C. Shannon, 1949. [2]

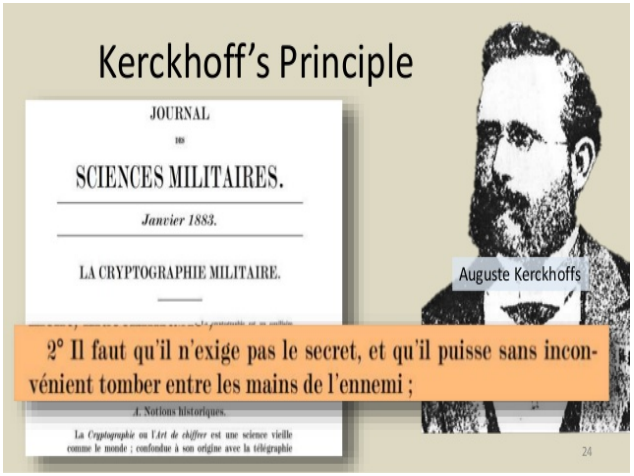
[2] "Communication Theory of Secrecy Systems", Bell System Tech. J., vol. 28, pp. 656-715, Oct., 1949.

The Kerckhoffs' principle of cryptography

A cipher should be secure when the enemy cryptanalyst knows all details of the enciphering process and deciphering process except for the value of the secret key.

- stated in 1881 by the Dutchman Auguste Kerckhoffs (1835-1903).

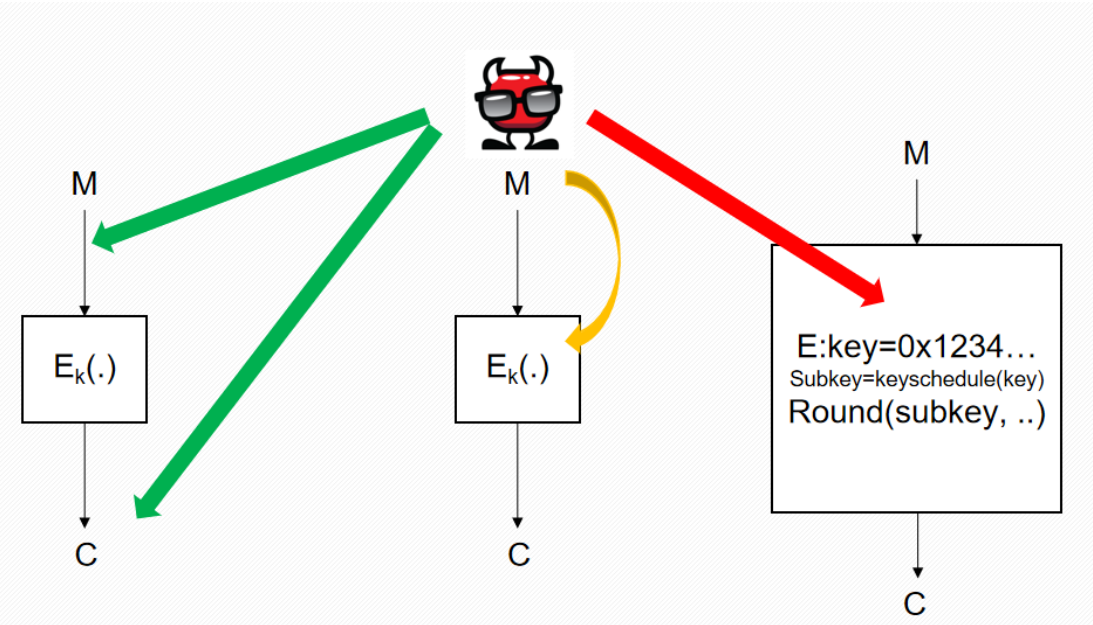
- Have some mathematical foundation for the belief that it will be hard to extract the key.



The goal of modern cryptography

- The goal of modern cryptography is to design, analysis, and implement a cryptosystem which obtains a mathematically acceptable security proofs.
  - Confidentiality / Secrecy
  - Integrity
  - Authenticity
  - Availability
- Thus security must be reduced on an acceptable model!
  - Ideal cipher model
  - Random oracle model/Standard model
  - Indistinguishability model
  - Indifferentiability model

How to modeling attackers in cryptography



Black-box model

- The adversary is able to know, choose or adaptively choose inputs and obtain outputs.
- Given the black-box implementation of the function, the adversary aims to recover the secret values or to misbehave the function.

White-box model

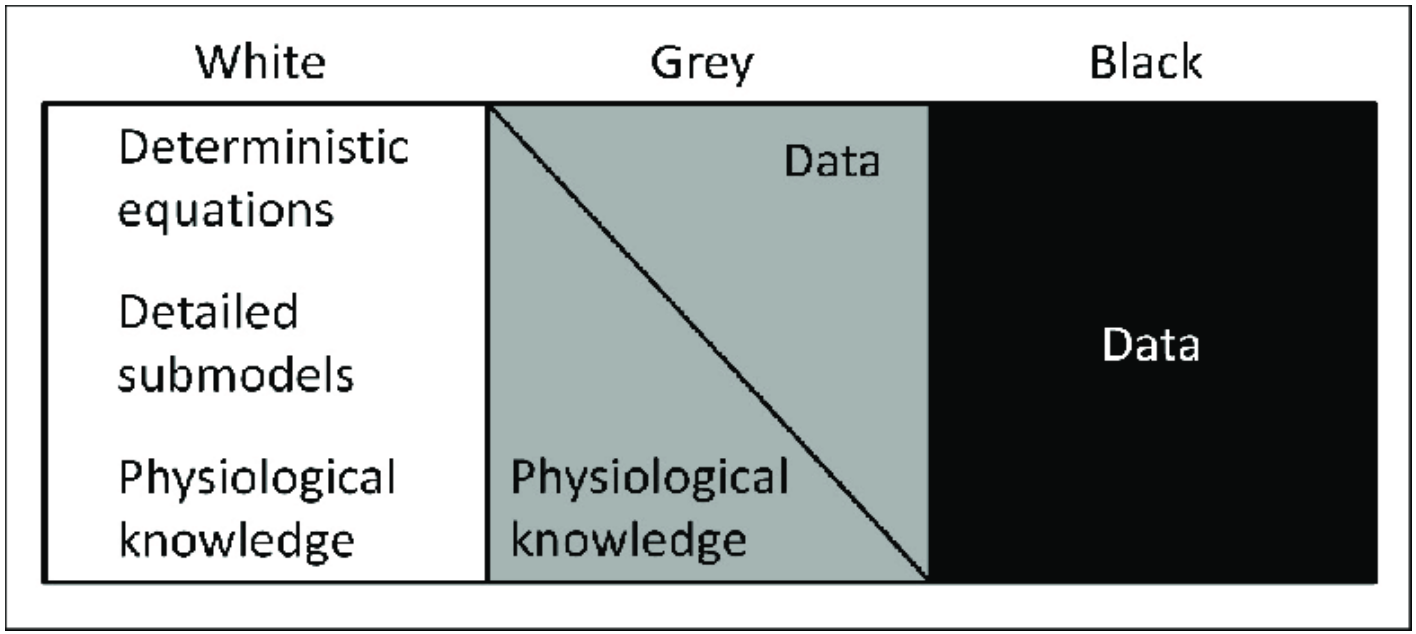
- Informally speaking, an adversary in white-box model can tamper, modify, manipulate all intermediate values and processes of the implementation of a secrecy system.
- It can be looked as a superset of black-box and grey-box models.

Grey-box model

According to the black/white-box modeling, the grey-box adversary is able to

- know, choose or adaptively choose inputs and outputs of the function;
- tamper, modify, manipulate intermediate values with specific (physical) knowledge.

An illustration of the Black/White/Grey-box modeling



WBC Implementations of DES/AES/SM4/... (All broken!)

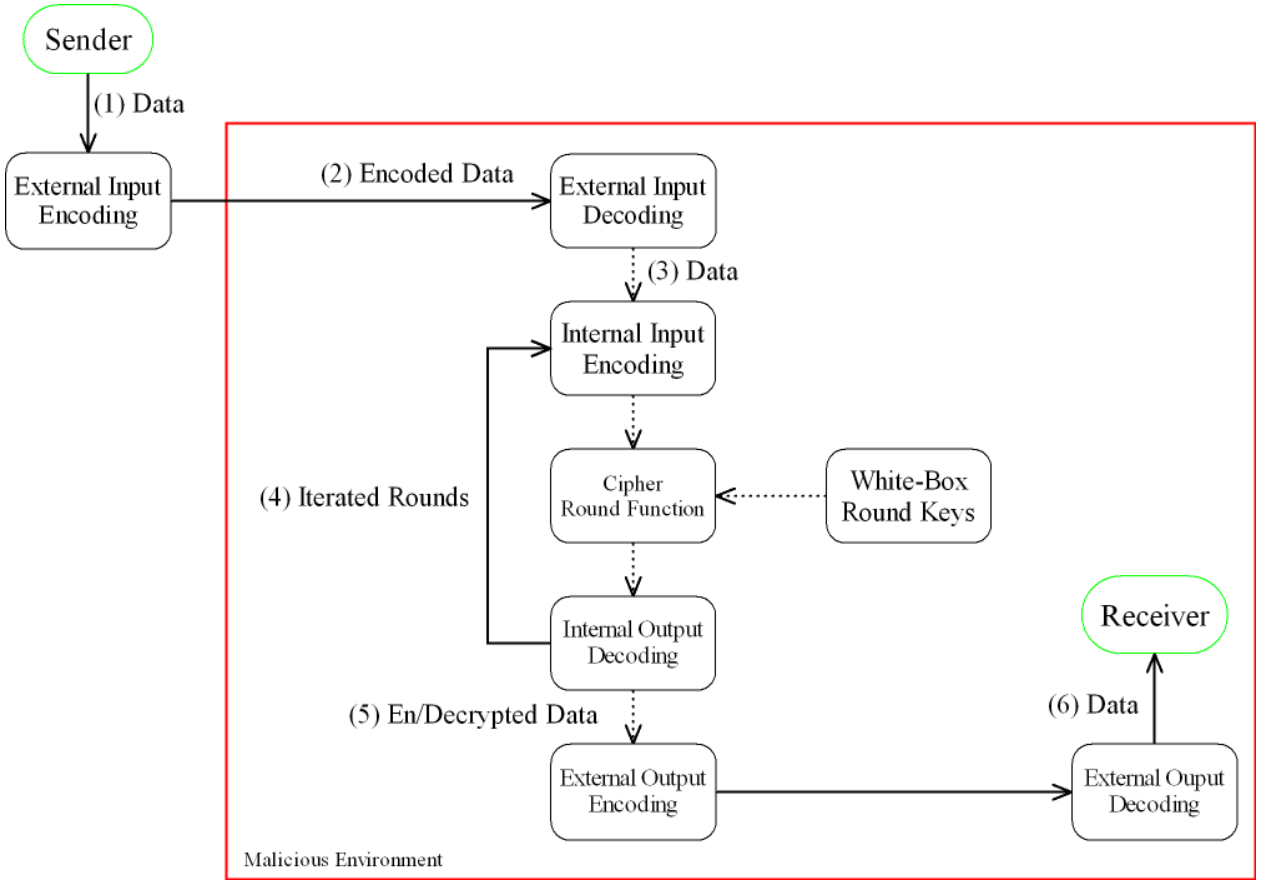
- Chow *et al.* Tbox-based WBDES/AES, 2002
- Link and Neumann. Tbox-based WBTDES, 2005
- Bringer. Isomorphism of Polynomials, WBAES, 2006
- Xiao and Lai. 16-bit Tbox WBAES, 2009
- Xiao and Lai. Affine-function WBSM4, 2009
- Karroumi *et al.* Dual-cipher WBAES, 2011
- Shi *et al.* Tbox-based WBSHARK, 2013
- Luo *et al.* ShiftRows table WBAES, 2014
- Su *et al.* Random number WBCLEFIA, 2014
- Shi *et al.* Dual-cipher WBSM4, 2015

- Baek *et al.* Larger Tbox WBAES, 2016
- Bai and Wu. Uncombinable encodings WBSM4, 2016
- Lee *et al.* Masked WBAES, 2018
- Xu *et al.* Obfuscated round boundaries WBAES/SM4, 2018
- Bai *et al.* Affine-function WBAES, 2018
- Zhou *et al.* Lightweight WBC, 2018
- Lv *et al.* WB-KMAC, 2019
- Lee and Kim. Table redundancy WBAES, 2019
- Lee and Kim. Improved masked WBAES, 2020
- Yao and Chen. Random state WBSM4, 2020
- Yao *et al.* Affine-function WBCLEFIA, 2020
- Yao *et al.* Affine-function WBGIFT, 2021

Dedicated WBC Proposals

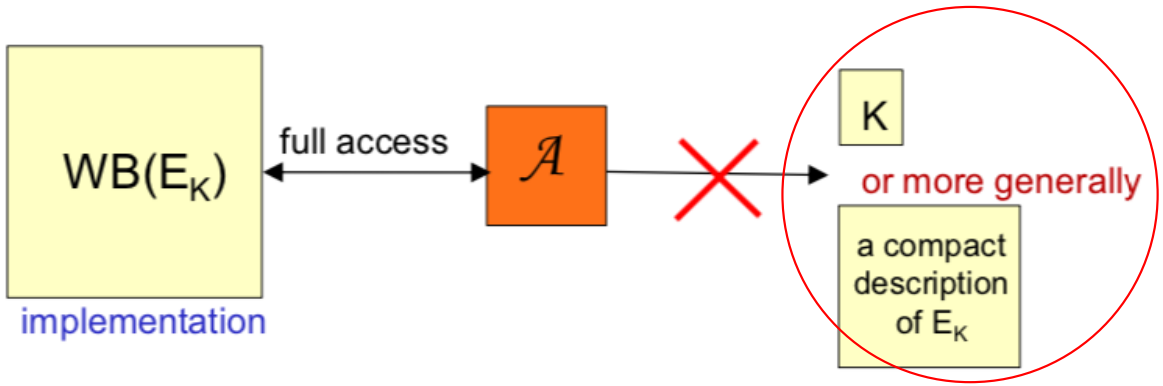
- Biyukov *et al.* ASASA structure, 2014
- Bogdanov and Isobe. SPACE, 2015
- Bogdanov and Isobe. SPNbox, 2016
- Fouque *et al.* WhiteBlock, 2016
- Bai and Wu. AES-Like cipher, 2016
- Cho and Dinur. WEM, 2017
- Lin *et al.* ASASASA structure, 2017
- Xu *et al.* AES-like cipher, 2017
- Shi *et al.* SDSRS, 2020
- Kwon *et al.* FPL, 2020
- Koike *et al.* Galaxy, 2020

A typical white-box block cipher in a nutshell



Basic security definitions for WBC (Chow *et al.*)

- In SAC 2002, the security notions have been informally described for white-box cryptography by Chow *et al.*. First the key recovery problem is informally defined by the weak white-box security .



informal definition: (T, S)-incompressible implementation of E_K.
an adversary with full access to WB(E_K) must be unable to derive
an equivalent* representation of E_K of size lower than S in time T. **

Basic security definitions for WBC

Definition

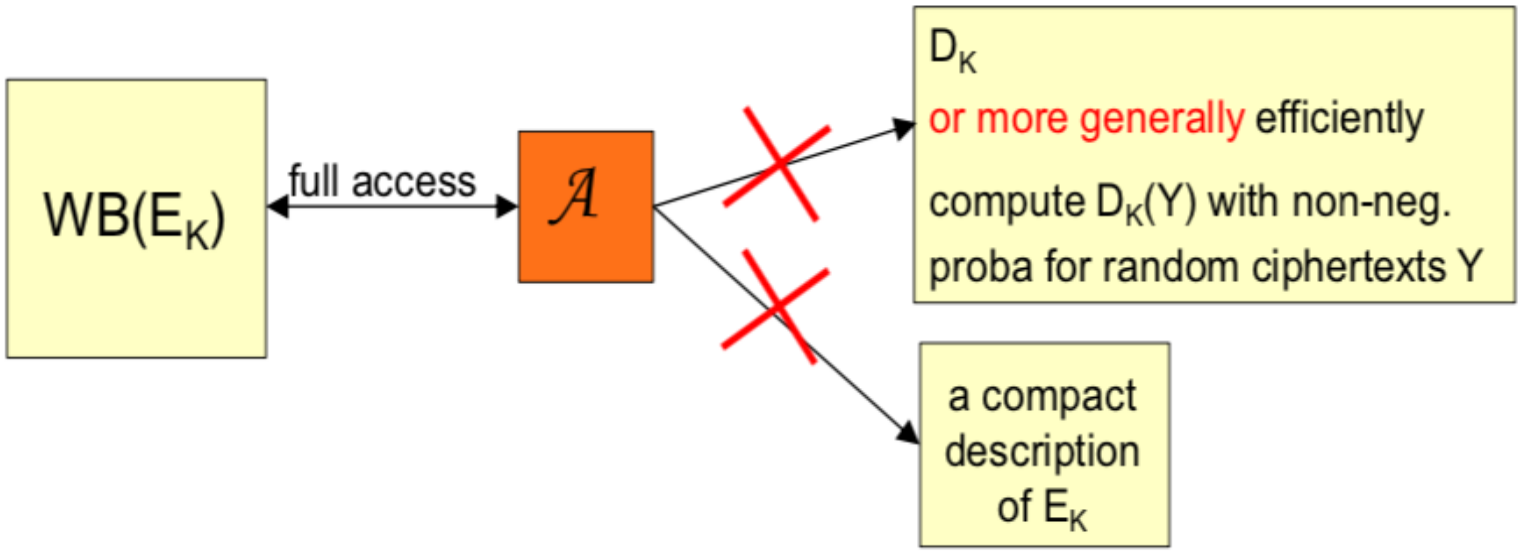
(Secret key recovery (KR-Security)). A white-box implementation of a key-instantiated block cipher E_k (or D_k) is called *KR-insecure* if an attacker extracts the secret key k and furthermore has access to the plaintext P.

Definition

(White-box key recovery (WBKR-Security)). A white-box implementation of an encoded version of a key-instantiated block cipher E_k (or D_k) is called *WBKR-insecure* if the attacker extracts the secret key k and the inverse mappings of the applied external encodings.

Basic security definitions for WBC (Biryukov *et al.*)

- For more general security, the strong white-box security has been defined by [Biryukov *et al.* 2014] (which connects to KR/WBKR-Security).



The problems of the weak/strong white-box security

- It only works for white-box encryption/decryption, while does not suitable for MAC/signature.
- "A compact description of E_k/D_k", it is very hard to measure under this informal definition.

The negative results on virtual black-box property (VBBP)

- At Crypto 2001, Barak *et al.* provided an insight on the impossibility of code obfuscation for generic programs.
- The virtual black-box property (VBBP) has been proposed for defining the ideal code/program obfuscation.
- The negative results are given by counter-examples that cannot satisfy the virtual black-box property (VBBP) in any circumstance.

Recall the definitions of VBBP [Saxena2009] (1)

Correctness. $\mathcal{O}$ is an obfuscator for a polynomial time function Q if the follow properties hold.

1. *(Functionality)*.
   $\forall k, \forall (q, a) \in \mathcal{K}_Q^k \times \mathcal{I}_Q^k : \Pr[\mathcal{O}(Q, k)(a) \neq Q[k](a)] \leq negl(k)$
2. *(Polynomial slowdown and expansion)*.
   $\exists p \in \mathbb{P}, \forall k, \forall q \in \mathcal{K}_Q^k$:
   - $|\mathcal{O}(Q, q)| \leq p(k)$;
   - $\forall a : Q[q](a) \leq t \rightarrow \mathcal{O}(Q, q)(a) \leq p(t)$

Recall the definitions of VBBP [Saxena2009] (2)

- Let $Q[q]$ be a random instance of a polynomial Turing machine family (PTMF) Q under key q.
- The VBBP requires that whatever information that adversary $\mathcal{A}$ extracts from $\mathcal{O}(Q, q)$, simulator $\mathcal{S}$ can also extract with black-box access to $Q[q]$.
- The existing notions of VBBP can be categorized in the following two directions.
  - *Predicate VBBP*
  - *Indistinguishability*

[Saxena 2009] noted that *Predicate VBBP(pvbbp)* is too weak for practice, whilst *Indistinguishability* is too strong to be achievable.

Recall the definitions of VBBP [Saxena2009] (3)

Soundness. $\mathcal{O}$ is sound if at least one of the following properties holds.

1. *Predicate VBBP:*
   $\forall A \in \mathbb{PPT}, \exists S \in \mathbb{PPT} : Adv^{pvbbp}_{A,S,\mathcal{O},Q} \leq negl(k)$, where
   $Adv^{pvbbp}_{A,S,\mathcal{O},Q}(k) = |\Pr_{q \xleftarrow{R} \mathcal{K}^k_Q}[A^{Q[q]}(1^k, \mathcal{O}(Q,q)) = 1 \wedge S^{Q[q]}(1^k) \neq 1]|$.
2. *Indistinguishability:*
   $\forall A \in \mathbb{PPT}, \exists S \in \mathbb{PPT} : Adv^{ind}_{A,S,\mathcal{O},Q} \leq negl(k)$, where
   $Adv^{ind}_{A,S,\mathcal{O},Q}(k) = |\Pr_{q \xleftarrow{R} \mathcal{K}^k_Q}[A^{Q[q]}(1^k, \mathcal{O}(Q,q)) = 1 \wedge A^{Q[q]}(1^k, S^{Q[q]}(1^k)) \neq 1]|$.

Saxena *et al.*'s WBC proposal

1. Encryption: $E[k] : (m, r) \mapsto (H(\hat{e}(k^r, g) \oplus m), g^r) \mapsto (c_1, c_2)$
2. Decryption: $D[k] : (c_1, c_2) \mapsto H(\hat{e}(c_2, k)) \oplus c_1 \mapsto m$
3. White-box encryption: Let $y = \hat{e}(k, g)$,
   $\mathcal{O}(E[k]) : (m, r) \mapsto (H(y^r) \oplus m, g^r) \mapsto (c_1, c_2)$

The problem:

- Only CPA-security, to achieve CCA-Security requires MAC on message. (which means another authenticated key!)
- Black-box security can be reduced to the discrete logarithm problem (k^r), whilst white-box security is the pairing inversion problem. ($y = \hat{e}(k, g)$)

Chow *et al.*'s White-box metrics

[CEJO2002] introduced a few metrics that try to measure the achieved level of white-box security for CEJO-framework.

- White-box diversity: How many distinct ways a particular unencoded LUT can be encoded. For key-dependent LUTs, the variation of the embedded key-material needs to be considered.

$$\mathcal{T}' = g \circ T \circ f, \text{WB-div} = \#g \times \#T \times \#f$$

- White-box ambiguity: How many distinct ways a specific encoded LUT can be interpreted.

$$\text{WB-ambT'} = \text{WB-div(T')}/\#\text{T'}$$

- Local security: Indistinguishability on $\mathcal{T}'$ with different keys and encodings.

The problems of Chow *et al.*'s White-box metrics

- White-box diversity/ambiguity and local security cannot imply to KR/WBKR-security.
- The increase number of white-box diversity/ambiguity cannot be related to the achieved security level.
- It is very hard to accurately count white-box diversity/ambiguity due to the bijection restriction.

Tamper resistance for DRM

- Michiels and Gorissen [DRM'07] proposed a method to protect the integrity of software which depends on the correct operation of the white-box implementation of a block cipher.

security goal

- The proposed method assume that it is the goal of an attacker to modify the protected software without losing the ability to decrypt/encrypt properly.

White-box security objectives

For the application in which a white-box implementation is deployed, [Delerablée *et al.* 2013] discussed the following security objectives.

- One-wayness: implies the strong white-box security.
- Incompressibility: prevents a functionally equivalent deduction with a significantly smaller memory footprint.
- Traceability: make a white-box implementation traceable.

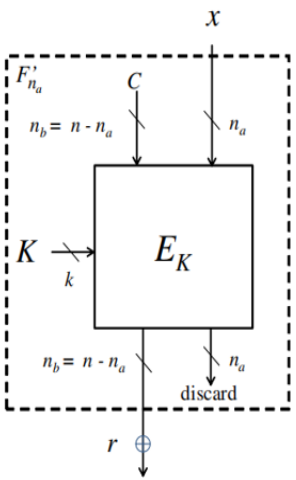
Space-hard ciphers: SPACE

- In CCS 2015, Bogdanov and Isobe proposed a family of white-box block cipher SPACE with several novel features.
  - SPACE-hardness: The implementation of E_K is (M, Z)-space hard if it is infeasible to en/decrypt any randomly drawn plain/ciphertext with probability of more than 2^{-Z} given any code (LUT) of size less than M.
  - Strong $(M, Z) - Space\ hardness$: existential possibility version of $(M, Z) - Space\ hardness$.

Space-hard ciphers: SPACE

- GFN structure

- Build the F function from a block cipher E with key K
  - F:{0,1}^{8}->{0,1}^{120}



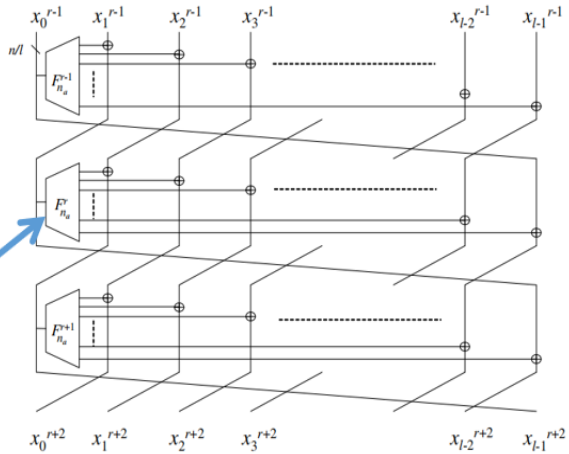


Figure 4: Ciphers with fixed code size: SPACE

Problem: GFN is self-inversable without the secret key

Space-hard ciphers: SPNBox

In Asiacrypt 2016, Bogdanov and Isobe proposed SPNBox, which can be looked as the SPN version of SPACE.

Problem 2: How to build an 8bit to 8bit 128-bit key-dependent sbox?

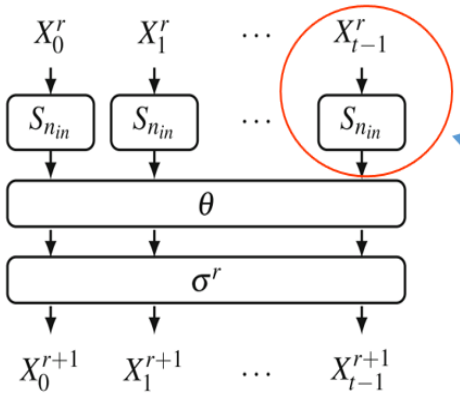


Fig. 3. Round transformation of SPNbox.

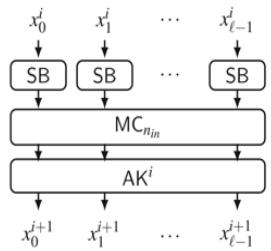


Fig. 4. Round transformation of the underlying block ciphers $S_{n_{in}}$.

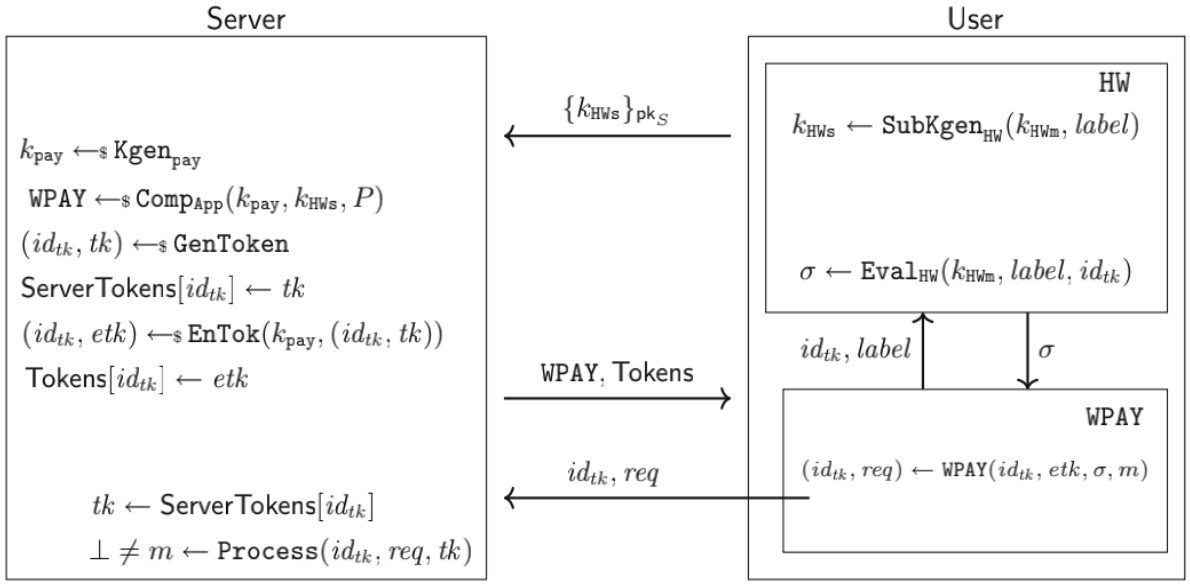
Problem 1: All the components are bijective on small finite fields, thus their inverse functions are easy to compute.

The problems of Space-hard ciphers

- Performance: The costs of LUTs and computation are higher than CEJO-like constructions.
- Weak WBC: only achieves the weak white-box security (add encodings might be helpful but even worse on performance).
- Accuracy of space-hardness: For example, if give 1000 bytes out of a 1024 bytes LUT, does attacker need to exhaustively search the remain 24 bytes?
- Compatibility: only works for new designs (especially block ciphers).

Bock *et al.*'s white-box security with hardware/application-binding

In Asiacrypt 2020 and CHES 2020, Bock *et al.* proposed a white-box security key derivation function with hardware/application-binding, and then built a white-box payment application on it.



The problems of Bock *et al.*'s hardware/application bound white-box security

- It is not truly white-box security anymore if the precondition is a secure hardware. It becomes trusted computing.
- How to achieve hardware-binding is missing in the paper, the authors suggest using Android Keystore (which is insecure when it is rooted.)
- If we already had a trusted hardware, why we need white-box cryptography? (e.g., using TEE is a better choice)

SCA model on WBC

In CHES 2016, Bos *et al.* proposed that WBC can be attacked by SCA techniques under grey-box model.

- Using SCA techniques can avoid the reverse engineering work, which always be time-costly in practice.
- For a WBC implementation,
  - Differential computation analysis (DCA) focuses on the inputs and outputs of the first round;
  - Differential fault analysis (DFA) exploits the faulty inputs and outputs of the last (diffusion) round.

Revisiting DFA/DCA and their countermeasures

- DFA
  - Building a model and performing analysis between the fault-free and faulty ciphertexts to exploit the key.
- DCA
  - The software counterpart of the Differential Power Analysis with noise-free intermediate values.
  - Instead of physical measurements, it employs instrumentation tools to capture the memory traces.

Their countermeasures:

- The obfuscation and the runtime protections (e.g., the desynchronization, virtual machine, dummies, and duplicates).
- The well-known SCA countermeasures (e.g., redundancy, masking, and shuffling).

The adaptive side-channel analysis model on WBC

Although some improved white-box schemes can prevent the generalized SCA attacks (e.g., DFA and DCA), a white-box adversary can combine the powerful ability in the *white-box context* with the technique of well-studied SCA attacks.

An **Adaptive side-channel analysis** is an SCA attack scenario in which the attacker has the ability to make her choice of the inputs to the cryptographic function based on the previous chosen plaintexts queries and their corresponding intermediates.

The abilities of adaptive SCA attacker

- (For choosing inputs/plaintexts.) The adversary has in-depth reverse engineering and cryptographic skills to de-obfuscate the white-box implementation.
- (For practical SCA attack.) The adversary is capable of the generalized SCA attacks (e.g., DFA and DCA) on the white-box implementation.

This is assumed that the adversary has the identical abilities to a white-box attacker when querying the cryptographic function for choosing inputs/plaintexts. For example, the adversary can deconstruct step by step the algorithms and collect/modify certain intermediate values before the SCA attack.

The steps of adaptive SCA attack

- The adversary de-obfuscates the white-box implementation with reverse engineering effort and therefore exposes the intellectual property.
- The adversary collects the target inputs/plaintexts by analyzing the intermediate values of the pinpointed sub-function.
- The adversary mounts SCA attacks on the white-box implementation with the chosen inputs/plaintexts.

Lee *et al.* improved masking method against DCA

Improved masking method [Lee *et al.* 2020] applies the masking technique to the key-dependent intermediate value. Such scheme adds the **random masks** on the Ty_i outputs and encodes the masked values and the masks used.

The unmasking is designed to combine with input decoding of the second round.

TypeII_MO table

Masks are randomly picked for each input, and the 8×8 linear transformations are applied to the mask in *TypeII_MO* table.

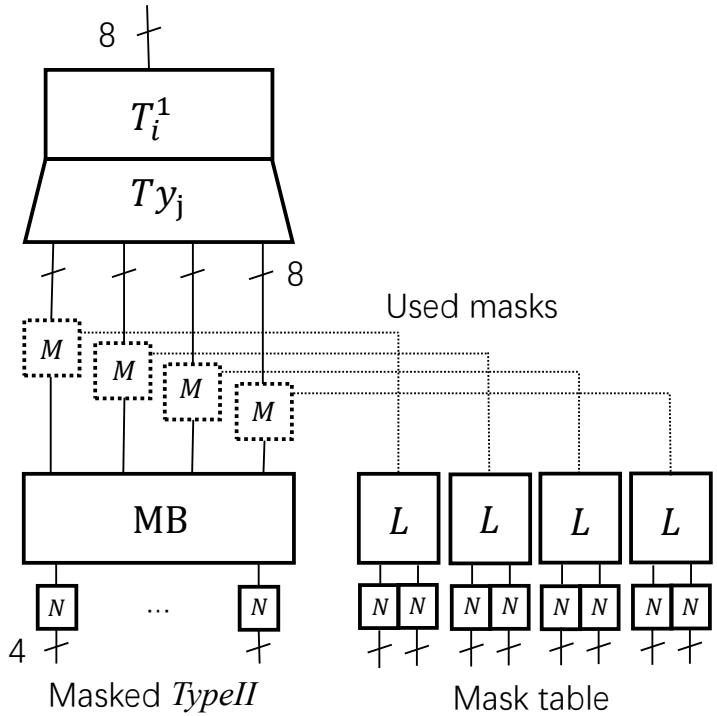


Figure: *TypeII_MO* table.

TypeII_MIMO table

The encoded masked 1-st round output and the mask are combined by XOR to unmask in the input decoding phase of the second round.

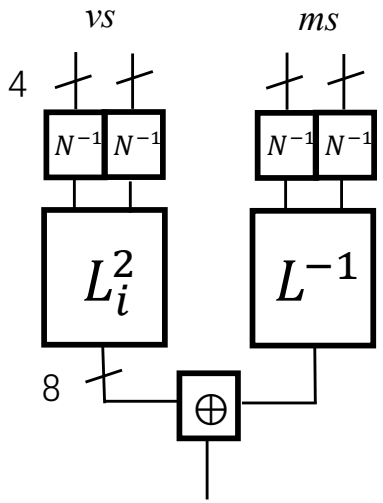


Figure: Input decoding phase of *TypeII_MIMO* table, the following components of T-box are omitted.

The data flow of improved masking scheme

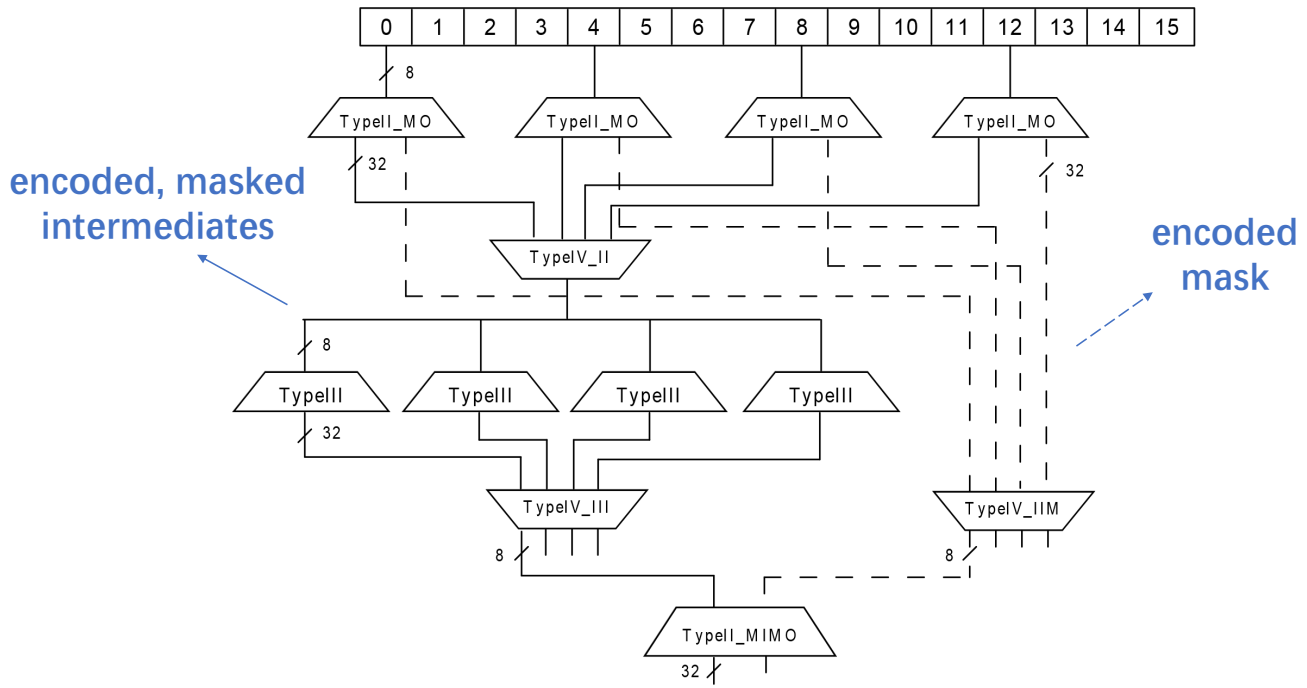


Figure: Lookup sequence from the first column in the first round to the first byte at the second round. (Solid line represents the data flow of masked values while the dotted line denotes the one of mask used.)

DCA on improved masking method

DCA fails to analyze the correlation between intermediate and hypothesis values.

- The mask is selected uniformly at random, and is independent with the input and key.
- The intermediates in the first round are all masked, and the unmasking is combined with the input decoding phase of the LUT in the second round.

Adaptive DCA against improved masking method

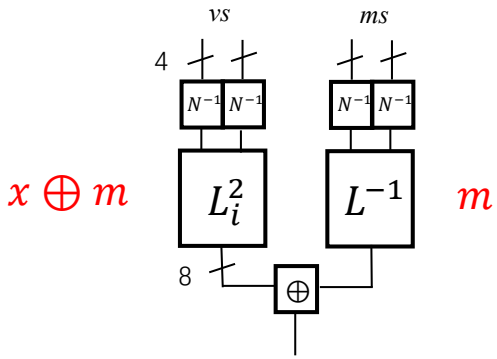


Figure: The underlying mask *m* of *vs* and *ms* is identical to each other.

If the mask of different plaintexts is **constant**, the combination of the mask and linear encodings are converted into **an affine function**. Hence, the mask is fail to hide the correlation between the key-dependent output and hypothetical value.

Review the data flow of improved masking scheme

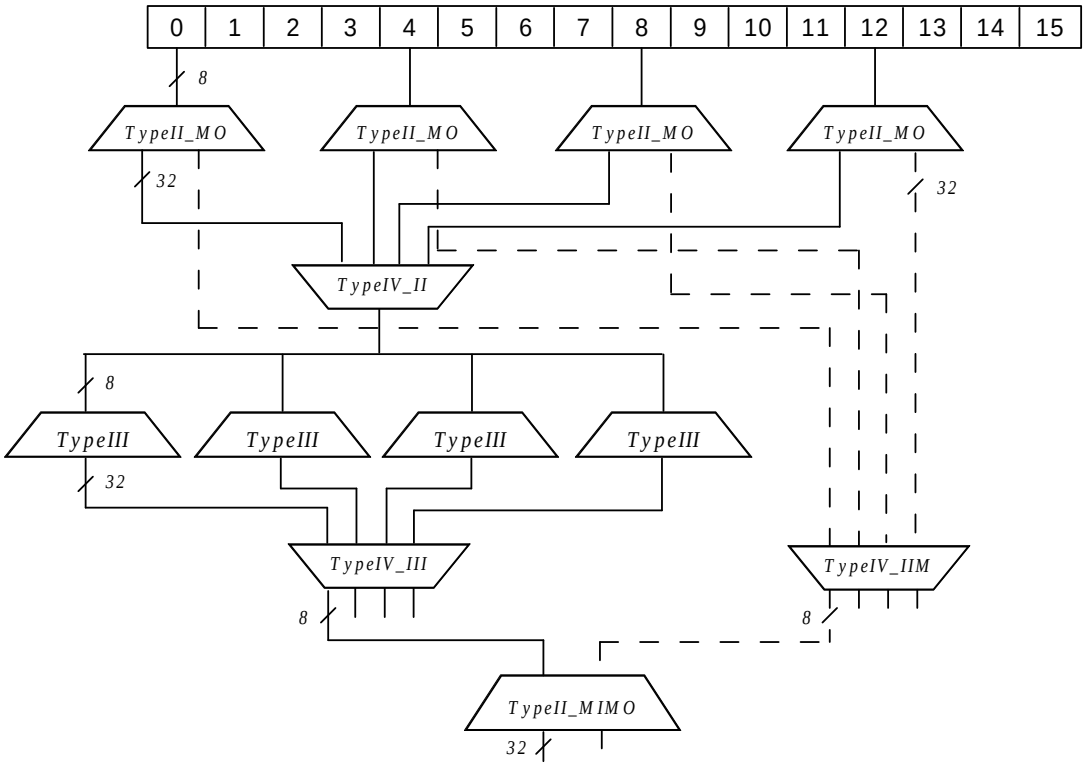


Figure: Lookup sequence from the first column in the first round to the first byte at the second round. (Solid line represents the data flow of masked values while the dotted line denotes the one of mask used.)

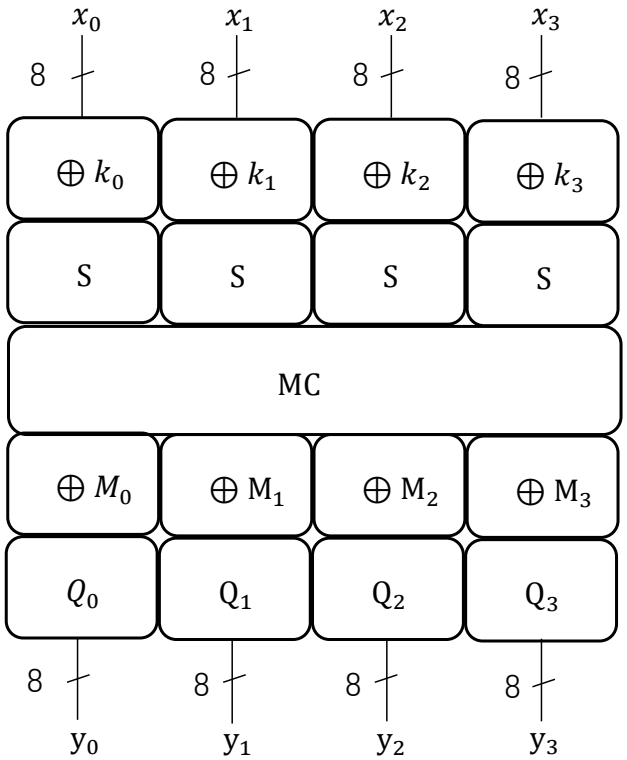


Figure: Composition of the encoded masked 1-st round output (solid line).

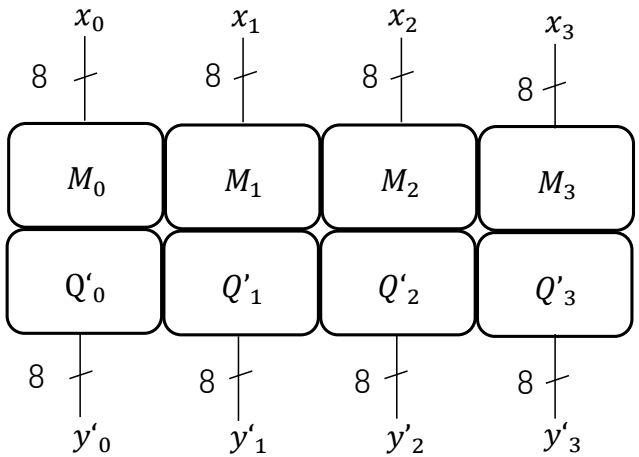


Figure: Composition of the encoded mask in 1-st round (dotted line).

The encoded and masked 1-st round functions are shown in below.

$$y_i(x_0, x_1, x_2, x_3) = Q_i \circ (mc_{i,0} \cdot S[x_0 \oplus k_0] \oplus mc_{i,1} \cdot S[x_1 \oplus k_1] \oplus mc_{i,2} \cdot S[x_2 \oplus k_2] \oplus mc_{i,3} \cdot S[x_3 \oplus k_3] \oplus M_i).$$

The encoded output of mask table can be represented as follows.

$$y'_i(x_0, x_1, x_2, x_3) = Q'_i \circ (M_i),$$

where M_i denotes the mask used, mc_i denotes the **MixColumns** coefficient, $0 \leq i \leq 3$ denote the byte index of the column.

Note that $x_0||x_1||x_2||x_3 \in \mathbb{F}_2^{32}$, M_i take all the values in $\mathbb{F}_2^8$. One can sort the inputs by the identical output y_i' of mask table, i.e., to get the input set $\mathcal{W} = \{x|y'(x) = c, x \in \mathbb{F}_2^{32}\}$, where c is a constant.

To reduce the key space, the random plaintext can be selected with fixed value for x_2 and x_3 (specifically $x_2 = x_3 = 0$).

In this way, for any input $x \in \mathcal{W}$, the mask used of $y(x)$ is a constant.

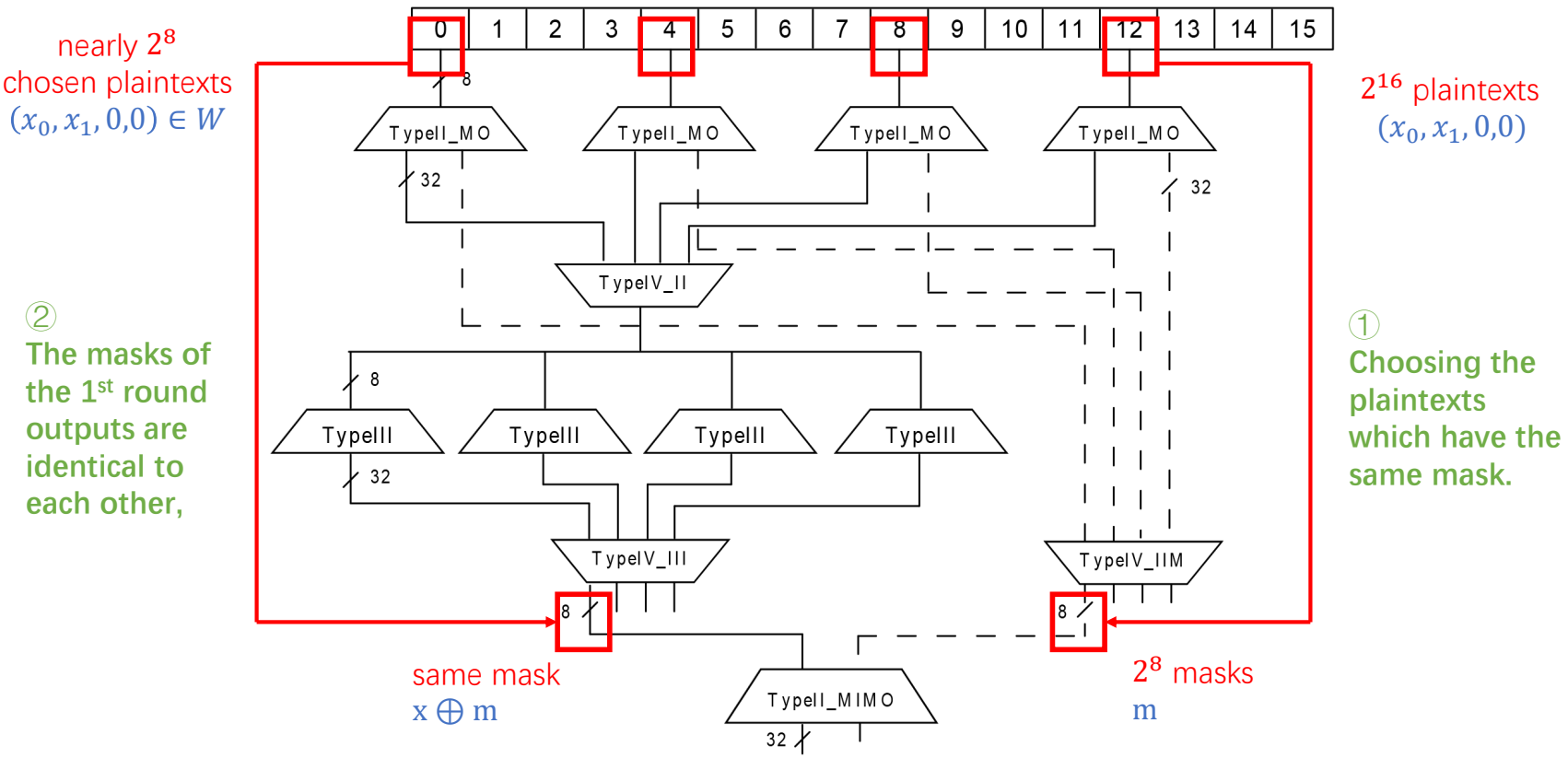


Figure: The chosen plaintexts phase of an adaptive DCA attack.

The detailed attack can be represented by the following steps.

- Collecting the plaintexts to form the set $\mathcal{W}$ which have the same output of y_0' by enumerating the first two bytes of the inputs and fixing the other bytes of y_0' (e.g., $y_0'(x_0, x_1, 0, 0)$).
- Collecting the bit traces υ which include the values of the outputs of y_0 by choosing the plaintexts in $\mathcal{W}$.

- Selecting the XOR of the first two outputs of MixColumns as the hypothetical value b, that is

$$b = mc_{i,0} \cdot S(x_0 \oplus k'_0) \oplus mc_{i,1} \cdot S(x_1 \oplus k'_1),$$

  where k'_0 and k'_1 are key guess.
- Mounting the DCA attack on analyzing the correlation between υ and b to recover k_0 and k_1.

Work factor of adaptive DCA on improved masking

After the reverse engineering, the experimental results show that,

1. at least 25 plaintexts $(x_0, x_1, 0, 0) \in \mathcal{W}$ help to recover the 2-byte key k_0 and k_1,
2. nearly 3,240 encryptions can help to collect 25 elements in $\mathcal{W}$.

Such that,

1. at least about 3,265 encryptions (3,240 times for obtaining $\mathcal{W}$ corresponding to four 4-byte plaintexts and 25 times for collecting computation) can recover four 2-byte subkeys of the first round,
2. totally at least about 6,530 encryptions can extract the 16-byte first-round key.

Table redundancy method for protecting against DFA

In 2019, Lee *et al.* [eprint 2019/959] proposed a table redundancy scheme based on Chow *et al.*'s white-box AES for protecting against DFA.

Table redundancy method uses **multiple branches of LUTs** for the vulnerable rounds (Round 7, 8, and 9) and xor the output to detect the modification.

Different types of LUTs in Chow *et al.*'s WBAES

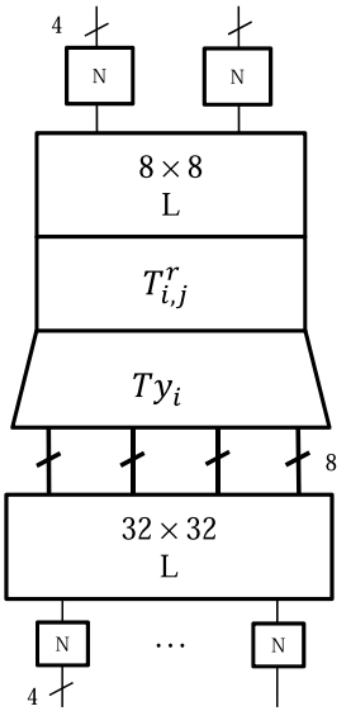


Figure: Type II: key-dependent T-boxes/Ty_i table

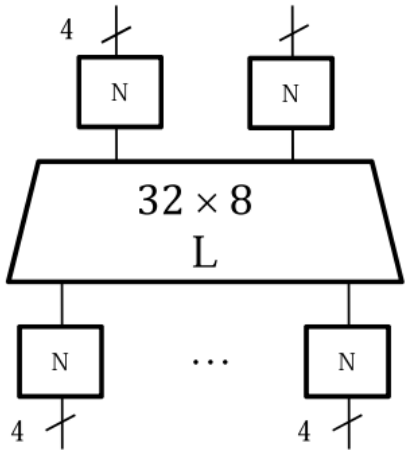


Figure: Type III: compatibility of encodings between consecutive rounds

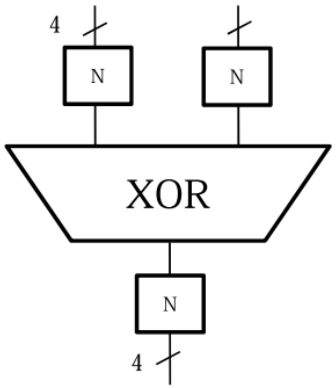
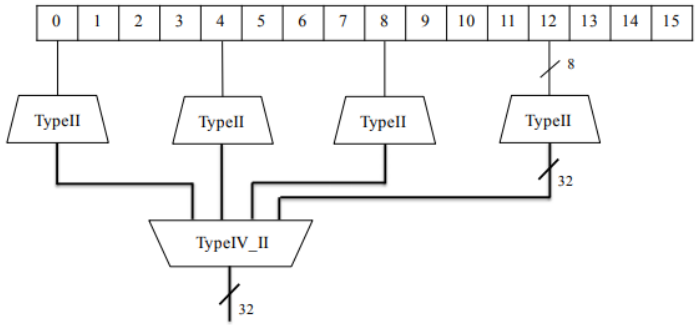
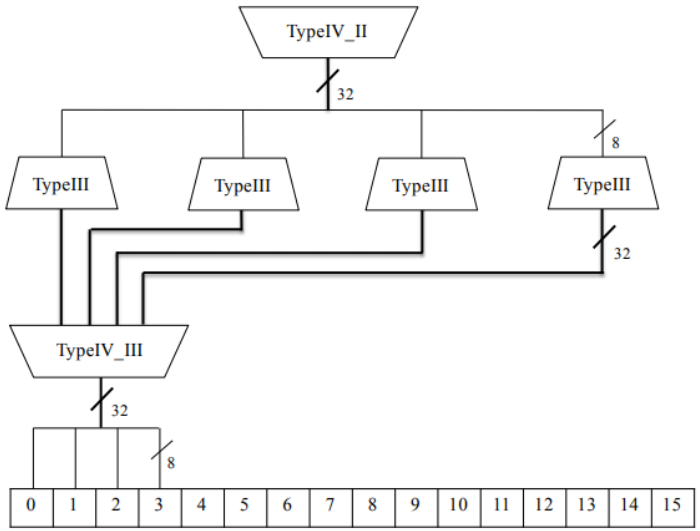


Figure: Type IV: encoded nibble XOR table



(a) *TypeII* and *TypeIV_II* lookup.



(b) *TypeIII* and *TypeIV_III* lookup

Figure: An overview of *TypeII*, *TypeIII* and *TypeIV* table lookups.

Table redundancy protection

The distribution of the redundant computation can be shown as four parts as follows.

- Sharing the look-up tables from Round 1 to 6.
- Transforming independently in parallel with two sets of look-up tables from Round 7 to *TypeII* in Round 9.
- XOR (by a type of *TypeIV*) the output of two redundant computations.
- Sharing the look-up tables from 9-th *TypeIV_II* to Round 10.

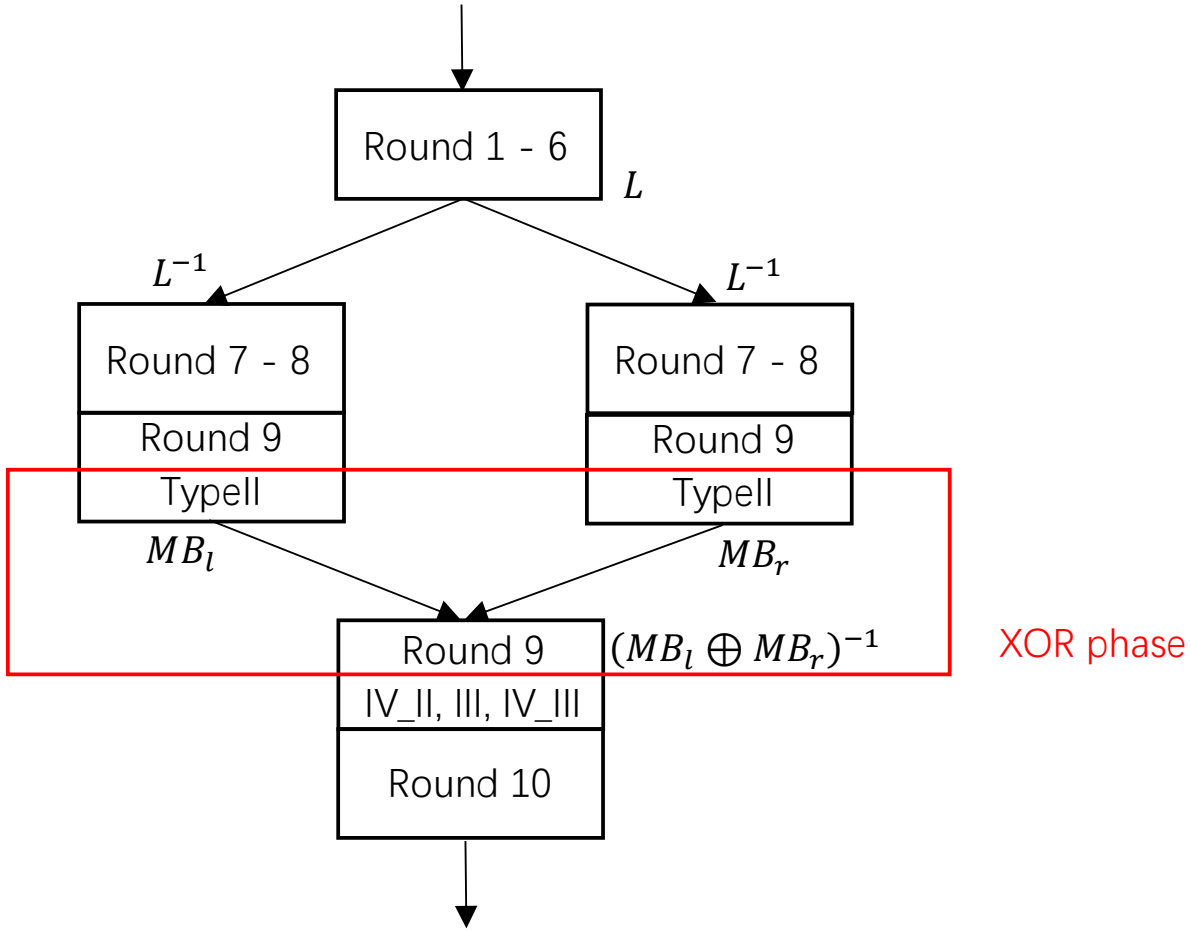


Figure: The lookup sequence of table redundant scheme.

- $MB_l,\ MB_r \in \mathbb{F}_2^{32\times 32}$, the left and right 32-bit linear encoding.
- $x_l,\ x_r \in \mathbb{F}_2^{32}$, the underlying state of 9-th *TypeII* output.

The XOR phase can be shown as

$$MB_l \cdot x_l \oplus MB_r \cdot x_r = (MB_l \oplus MB_r) \cdot x,\ \ iff\ x_l = x_r = x.$$

The inverse Mixing Bijection $MB^{-1} \in \mathbb{F}_2^{32\times 32}$ in 9-th *TypeIII* can be depicted as

$$MB^{-1} = (MB_l \oplus MB_r)^{-1}.$$

The 9-th round byte-fault model

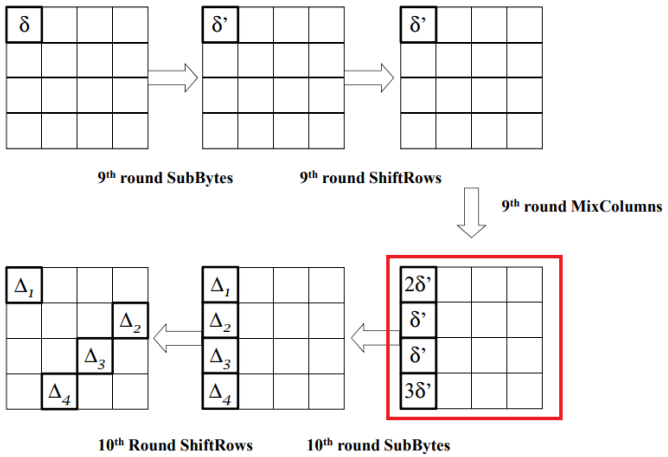


Figure: The 9-th round faulty model of AES. (Injection between 8th and 9th MixColumns.)

- C_i, fault-free ciphertexts.
- C_i^*, faulty ciphertexts.
- k_i^*, key candidates.
- 2 faults can recover 4-byte subkey.

$$2\delta' = S^{-1}(C_0 \oplus K_0^*) \oplus S^{-1}(C_0^* \oplus K_0^*),$$
$$\delta' = S^{-1}(C_7 \oplus K_7^*) \oplus S^{-1}(C_7^* \oplus K_7^*),$$
$$\delta' = S^{-1}(C_{10} \oplus K_{10}^*) \oplus S^{-1}(C_{10}^* \oplus K_{10}^*),$$
$$3\delta' = S^{-1}(C_{13} \oplus K_{13}^*) \oplus S^{-1}(C_{13}^* \oplus K_{13}^*).$$

DFA on table redundancy method

- $x \in \mathbb{F}_2^8$, the fault-free unencoded state.
- $x^* \in \mathbb{F}_2^8$, the faulty one.

The XOR phase now can be shown as

$$\mathrm{MB}_l \cdot \mathrm{T}/\mathrm{Ty_i}(x) \oplus \mathrm{MB}_r \cdot \mathrm{T}/\mathrm{Ty_i}(x^*) \neq (\mathrm{MB}_l \oplus \mathrm{MB}_r) \cdot \mathrm{T}/\mathrm{Ty_i}(x).$$

In this way, the decoding phase $(MB_l \oplus MB_r)^{-1}$ cannot work in the following transformation and thus the faulty ciphertexts cannot be exploited for DFA attack.

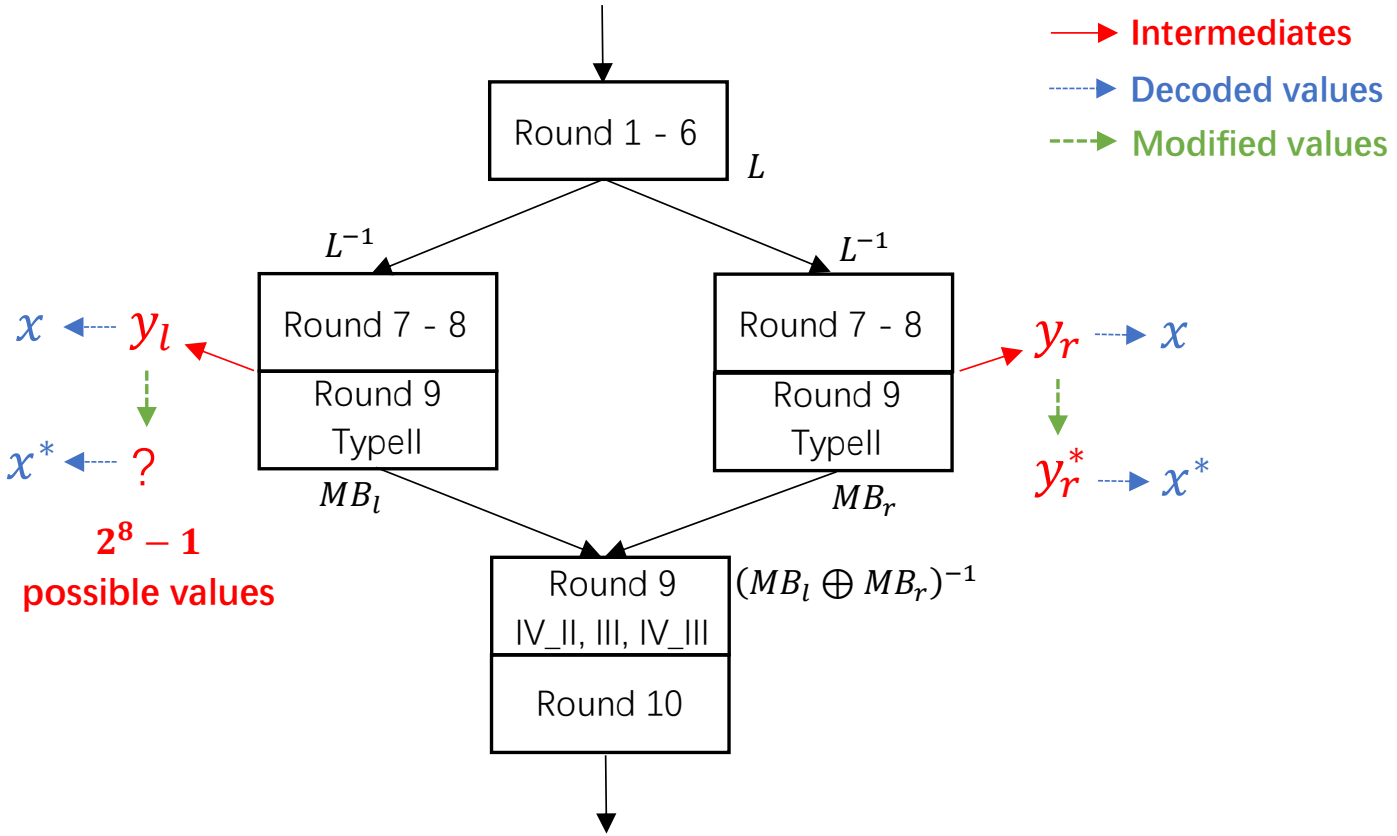


Figure: A brute-force fault injection at one of the 9-th input while fixing the other faulty input in the table redundant scheme.

Adaptive DFA against table redundancy method

- $y_l, y_r \in \mathbb{F}_2^8$, an 8-bit input value of the two 9-th *TypeII* respectively.
- $x \in \mathbb{F}_2^8$, an 8-bit underlying state of y_l and y_r.
- $P_l, P_r \in \mathbb{F}_2^8$, an 8-bit input decoding.

For the decoding process of 9-th round input, we have

$$x = P_l(y_l) = P_r(y_r).$$

Since x takes 256 different values and the encoding is a fixed bijection, the 256 pairs of each (y_l, y_r) can be collected.

Let $\mathcal{T}$ be a set of pairs values, such that

$$\mathcal{T} = \{(y_l, y_r) \mid y_l \in \mathbb{F}_2^8,\ y_r \in \mathbb{F}_2^8\} \textit{ with } \#\mathcal{T} = 256.$$

Replacing the intermediate pairs (y_l, y_r) with the other pairs in $\mathcal{T}$ can induce the faults at 9-th input.

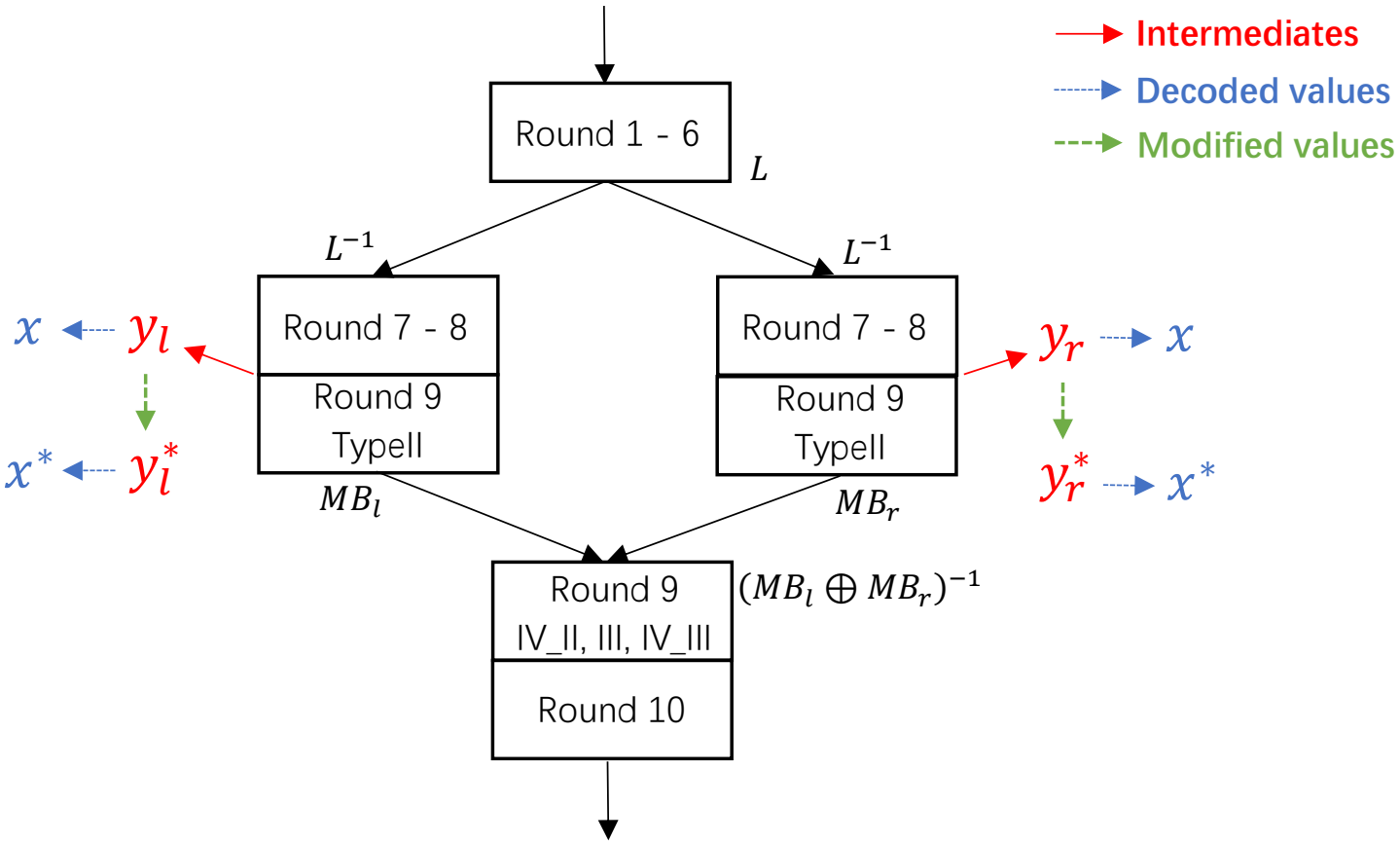


Figure: The replacement of 9-th round input in the table redundant scheme, $(y_l^*, y_r^*) \in \mathcal{T} \backslash (y_l, y_r)$.

The injection at the entry of the 9-th round can be concluded by the following steps.

- Collecting the pairs values of $\mathcal{T}$ by repeatedly running the cryptographic program with random plaintexts.
- Getting a fault-free ciphertext with a chosen plaintext P.
- Replacing the (y_l, y_r) with other pairs in $\mathcal{T} \backslash (y_l, y_r)$ and collecting the faulty ciphertexts by repeatedly running the cryptographic program with P.
- Using the 9-th round byte-fault model to perform the analysis between the fault-free and faulty ciphertexts.

Work factor of adaptive DFA on table redundancy

After the reverse engineering, the experimental results show that,

1. at least 5 times encryption (3 times for collecting $\mathcal{T}$ and 2 times for the replacement of fault injection) can recover the 4-byte subkey (in a column) of the tenth round,
2. repeat the steps for the remain columns, such that totally at least about 20 times encryption can extract the 16-byte tenth-round key.

Adaptive DCA on the CHES 2021 Proposal

In CHES 2021, Seker *et al.* proposed a masking scheme for resisting DCA and Algebraic DCA attacks.

- $\tilde{x}_0, ..., \tilde{x}_d, x_1, ..., x_n$, shares of sensitive variable x.
- Masking: $x = \prod_{j=0}^{d} \tilde{x}_j \oplus \sum_{i=1}^{n} x_i$.

Note that,

$$Pr(\prod_{j=0}^{d} \tilde{x}_j = 0) = 1 - (\frac{1}{2})^{d+1}.$$

Hence, $x = \sum_{i=1}^{n} x_i$ with a high probability for the increasing d, there exists n nodes that are linearly correlated with a predictable vector.

A higher-order DCA attack [Bogdanov *et al.* 2019] can be mounted on the circuit but with high complexity.

How can an adaptive higher-order DCA attacker reduce the complexity?

The practical perspective of white-box security

Since the secret values are the most important in the white-box security model. A white-box cryptography should provide the following security properties.

- **Algorithm level: Key recovery.**
- **Module level: Function integrity.**
- **System level: Code lifting.**

Our considerations

- Security notions and definitions for white-box cryptography still need precise analysis.
  - How to make a unified security framework for DCA/DFA/BGE?
  - How to find a protection method in such a unified framework?
- Only key extraction might not suitable for some applications, e.g., MACs and signatures.
- A secure white-box cryptography does not imply the white-box security of a secrecy system based on it.
  - Do not forget misbehaving attacks.

Thanks for your attentions!