# On the Fault Injection Security of White-box Ciphers

Md Alamgir Alam[1,2], Avik Chakraborti[1,2], Takanori Isobe[3], Sajani Kundu[1,4] and Sayandeep Saha[5]

[1]Institute for Advancing Intelligence, TCG CREST, Kolkata, India
[2]Academy of Scientific and Innovative Research (AcSIR), India
[3]Osaka University, Osaka, Japan
[4]Ramakrishna Mission Vivekananda Educational and Research Institute, India
[5] Indian Institute of Technology Bombay, Mumbai, India
md.alamgir.alam.119@tcgcrest.org,avik.chakraborti@tcgcrest.org,takanori.isobe@ist.osaka-u.ac.jp,sajani.kundu.89@tcgcrest.org,sayandeepsaha@cse.iitb.ac.in

**Abstract.** White-box security settings assume an extremely powerful adversary having full visibility and control of the software implementation and internal computations. Leakage-based attacks extract secret information via a local passive attacker (e.g., malware) and transmit it to a remote server. However, an active adversary, who can perform fault injections in a white box setting, has received limited attention, especially in the symmetric-key setting. In this paper, we initiate a formal study of active data-only adversaries in the white-box setting. Such adversaries preserve the control flow of the implementation but corrupt a bounded number of key-embedded lookup-table entries, enabling precise and repeatable manipulation of table values. Unlike leakage-based attacks, which are constrained by the bandwidth and existence of firewalls, such fault attacks can operate entirely locally. We focus on a data-only tampering adversary that preserves the control flow of the white-box implementation, but corrupts a bounded number of key-embedded lookup-table entries. Even under this stealth-preserving restriction, the adversary can cryptographically weaken the implementation and make faulty ciphertexts significantly easier to decrypt. We formalize such an active adversary by defining a new security notion and studying its impact on contemporary table-based white-box implementations. Our analyses reveal a structural disparity between two major design paradigms: Feistel-based white-box ciphers appear significantly more vulnerable to fault injection than SPN-based designs. Finally, we propose a software-based fault detection mechanism that detects fault injections with high probability, strengthening resilience. We provide detailed analysis of the SPN-based cipher WEM (the same analyses also work for other SPN-based ciphers like SPNbox), and two Feistel-based ciphers SPACE and Galaxy. Our analyses reveal that SPACE and Galaxy are significantly more vulnerable than WEM, under our fault-based security setting. Precisely, we show that WEM achieves high security under all the adversarial models, whereas SPACE and Galaxy instances can be attacked with a very high message recovery probability of $2^{-8}$, when the adversary can choose the fault positions and the values and corrupts up to one fourth of the implementation table entries.

**Keywords:** white-box cryptography, fault injection, table-based constructions, Feistel networks, SPN.

# 1 Introduction

Cryptographic primitives are traditionally evaluated under the black-box security model, in which an adversary is restricted to interacting exclusively with the input–output interface of a cipher. However, this modelling assumption becomes insufficient when cryptographic software is deployed in untrusted or adversarial execution environments. The adversary can exploit various physical defaults, such as timing, power, faults, and, in some cases, can read or modify the intermediate results of cryptographic software by means of malware, completely undermining the security guarantees (e.g., leaking the secret key).

To address this limitation, white-box cryptography was introduced by Chow *et al.* [13, 12] with the goal of protecting the secret key embedded within software implementations of AES and DES. In the white-box threat model, the adversary has complete access to the execution environment, including internal states and intermediate computations of the cipher. Under this stronger adversarial model, the adversary may attempt to extract the embedded master key (key extraction) or to compress the implementation while preserving its functionality, thereby violating the notion of incompressibility. Following these foundational works, a number of white-box implementations for DES and AES were introduced [9, 18, 23, 32]. Nevertheless, all such proposed schemes were broken through key extraction or table-decomposition attacks [3, 22, 29, 27, 31]. Biryukov et al. [4] proposed dedicated white-box block ciphers using the ASASA structure, but was broken with decomposition attacks [25]. The security of constructions with multiple secret nonlinear and linear layers has been explored by other works [5, 8, 28]. Bogdanov and Isobe introduced the SPACE family of white-box block ciphers [6], which aims to resist key extraction and code lifting (a variant of incompressibility). Bogdanov et al. [7] later proposed SPNbox, a new family of space-hard block ciphers that improves SPACE [6] in throughput. While SPACE uses the Feistel structure, SPNbox uses the SPN-type structure with small block ciphers as keyed S-boxes.

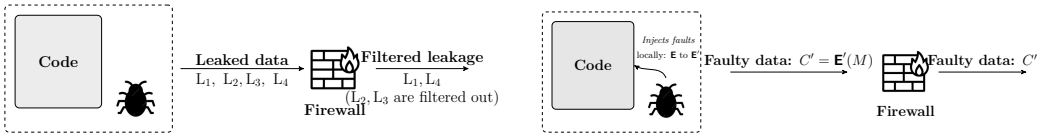
**Existing Real-World Usage of White-Box Cryptography.** White-box cryptography was originally proposed [13, 12] for Digital Rights Management (DRM [1]) applications. With the emergence of software-based approaches such as host card emulation (HCE [2, 26]), which replace hardware secure elements (HSE) in NFC transactions, white-box cryptography has become increasingly relevant for software-only security solutions, such as cloud services. Major payment networks like Mastercard recommend the use of white-box cryptography for token protection [24]. The growing prevalence of advanced malware and Trojans has further reinforced its relevance in security-critical deployments.

**Existing White-Box Security Notions.** Memory leakage constitutes a significant threat in software environments and may arise from malware, Trojans, buffer overflow vulnerabilities, and cold-boot attacks [15]. Recent white-box leakage notions assume two adversarial functionalities: i) a local malware that can read/dump the memory footprint of the cipher; and ii) a remote adversary, to which the memory dump is covertly transmitted. This is called the *code-lifting* scenario. To address such threats, white-box security notions define *Space-Hardness* (targeting one-time leakage) and *Longevity* (targeting continuous leakage) [6, 7, 14, 11, 20, 19, 10]. Space-hard designs limit the feasibility of exploiting low-capacity exfiltration channels (this refers to covertly transferring information at low rates to avoid detection), while longevity targets resilience against gradual information exposure over time Constraints on the exfiltration capacity are practical in the presence of firewalls or anomaly detection systems. However, such memory leaking adversaries are still passive – the local adversary (e.g., malware) does not modify/interrupt computations of the white-box cipher.

In contrast, fault injections are more powerful and active attack techniques against

white-box implementations. Given full control over the execution, the adversary can introduce precise and repeatable perturbations into the implementation. Software-based environments enable fine-grained manipulation with greater spatial and temporal precision than traditional fault models.

**Fault Injection.** Surprisingly, fault injection has not been formally studied for recent symmetric white-box ciphers (although some dedicated white-box ciphers claim to be inherently key extraction resistant against differential fault attacks [6, 7, 11, 20, 10] as the full master key is embedded inside the implementation table). However, the literature is void of weaker but more practical adversarial goals under the white-box fault injection setting (for example, message recovery). The effectiveness of fault injection arises from its active character. Instead of merely observing internal values, the adversary deliberately induces controlled faults within the implementation. These induced faults may significantly weaken the internal structure of the cipher, which would otherwise remain unaffected under passive observation. Moreover, unlike classical fault injection attacks on ciphers, which perform a timed injection by means of voltage/clock/electromagnetic glitches or laser pulses, white-box fault injections can happen through malware and hence can be far more controllable. Classical fault injection has a gray-box view of the cipher execution and has limited control over the value of the injected fault. In contrast, white-box fault injections can happen through malware, and therefore can modify stored table entries or memory-resident data precisely and repeatedly to desired values. Here, precision should not be viewed as an additional idealization inherited from physical fault attacks. In a software white-box setting, a local adversary such as malware may directly patch stored tables or overwrite table entries in memory. Thus, entry-level control over lookup-table entries is a natural abstraction. The main practical restriction is instead stealth: the adversary should preserve the surrounding program behavior and avoid overt control-flow tampering. This motivates a formal study of active, data-only tampering attacks against table-based white-box ciphers.




(a) White-box leakage scenario. (b) White-box fault injection scenario.

**Figure 1:** White-box leakage and fault injection scenarios.

## 1.1 A Potential Fault Adversarial Setting

The recent white-box ciphers with key-embedded tables are unlikely to admit key recovery attack possibilities under fault injections (since keys are securely embedded in the tables). However, there are possibilities of other practical fault-based attacks. Let us simply consider the two-part entity (local and remote) setting, similar to the code-lifting setting, where the local entity could be a malware that can modify the white-box implementation. An immediate possible attack scenario is the cryptographic weakening of the white-box cipher. For example, if several lookup table entries are corrupted to identity mapping, the system could transmit ciphertexts with cryptographically weaker properties. The remote entity may intercept such ciphertexts and easily decrypt the message. In Figure 1, $\mathsf{E}$ is corrupted to $\mathsf{E}'$, and $M$ is encrypted to $C'$ (using $\mathsf{E}'$) that can be captured by an external entity to recover $M$ with a smaller complexity than intended. Interestingly, this setting does not require any code-lifting and very limited transmission to the remote entity, and, therefore, is free of any constraints due to bandwidth limitations.

However, limitations might come through other sources, for example, an aggressive change of the entire table might be easily detectable by the legitimate communication parties, or some security monitors co-residing with the white-box cipher. This precisely means a runtime integrity-checker or anti-tampering component running on the same device as the white-box implementation. We would like to mention that, such monitoring is only a possible deployment mechanism, not an assumption of our security model. Moreover, in some cases, there can be communication between the remote and local adversary to facilitate the decryption of the weakened ciphertexts. All such constraints strongly depend on the nature of the white-box cipher under consideration. Overall, this new adversarial setting must be formalized and studied, which is the goal of this paper.

We want to briefly highlight the practical significance of this setting. The intended setting is client-side encryption on a compromised device, where malware can modify the embedded lookup table but direct data exfiltration is constrained or monitored in the client side network. By weakening the implementation, the malware can cause the legitimate encryption channel to produce ciphertexts that a remote collaborator can decrypt more easily. Relevant examples include client-side encrypted uploads, backups, telemetry, and outgoing application messages.

**Fault Model.** In this work, we consider the fault model with precise, fully controlled faults to be injected only at the key-embedded look-up tables but not at the unkeyed public part of the implementation (for example, mix-column operation for AES). Our model intentionally excludes overt control-flow tampering, such as bypassing the encryption routine or reducing the number of rounds. Such modifications would amount to functionality replacement rather than cryptographic weakening, and they are also more likely to be detected by code-integrity checks, anti-tampering mechanisms, or abnormal program behavior. Instead, we focus on a data-only tampering adversary that modifies a bounded number of key-embedded lookup-table entries while preserving the original code path. This captures a stealth-preserving attack surface specific to table-based white-box implementations: the program continues to execute the intended encryption routine and outputs ciphertexts of the expected format, but the underlying primitive is weakened. Below, we will briefly talk about the counter scenario of Evasion of Control-Flow Integrity (CFI) and Anti-Tampering Protections. Real-world DRM systems and payment applications often incorporate robust tamper-detection mechanisms for executable code, such as code obfuscation, checksum verification, and Control-Flow Integrity (CFI) checks. If an attacker skips a function or reduces the number of rounds by rewriting instructions, the program is likely to crash immediately or trigger a tampering alert. In contrast, modifying a few bytes within a multi-megabyte blob of seemingly random data (i.e., a lookup table) is highly likely to evade these code-integrity monitors.

Our setting may seem related to the work [30] because a modified lookup table defines a cipher related to the original implementation. However, our relation comes from bounded entry-wise modifications of a single-key-embedded table, rather than related-keys or oracle access to related-ciphers.

### 1.1.1 Why Data-Only Table Tampering?

Our restriction to data-only table tampering is not merely a simplifying assumption. It reflects an asymmetry that commonly appears in software deployments of white-box cryptography: executable code and data objects are protected and modified in different ways.

On the defender side, executable regions are natural targets for integrity checks, control-flow monitoring, anti-debugging mechanisms, and anti-tampering protections. Modifying the control flow, skipping an encryption call, or reducing the number of rounds can therefore cause crashes, checksum failures, or other observable deviations from the intended

execution. In contrast, table-based white-box implementations contain large key-embedded data structures whose contents are designed to look pseudorandom. Continuously verifying every table entry at runtime may introduce non-negligible overhead, and therefore, partial modification of such data objects can be a more stealthy attack surface than instruction-level patching.

On the attacker side, the same asymmetry applies. Code-level patching typically requires changing executable pages or altering control-flow behaviour, which may interact with memory protections and code-integrity mechanisms. Data-only tampering, by contrast, only requires the ability to overwrite selected data locations once the adversary has obtained a memory-write primitive. This type of restriction is consistent with the broader systems-security literature on data-only attacks. Data-oriented programming showed that non-control-data attacks can be highly expressive without directly hijacking control flow [16], and block-oriented programming further demonstrated that data-only attacks can be automated even in the presence of strong control-flow defences such as CFI and shadow stacks [17].

Our model instantiates this data-only attack paradigm in the white-box setting: the adversary does not replace the encryption routine, skip rounds, or tamper with public unkeyed operations. Instead, it corrupts a bounded number of key-embedded lookup-table entries while preserving the original code path. This captures a stealth-preserving form of cryptographic weakening that is distinct from generic software patching. This is precisely the setting captured by NdUF: after bounded table-entry corruption, the resulting faulty ciphertext should still remain hard to decrypt.

## 1.2 Our Contributions

In this work, we begin a fine-grained fault-based analysis (with weaker but practical adversarial goals) of white-box ciphers. Our contributions are as follows.

1. We propose a new white-box security notion called NdUF (*Non-decryptability Under Fault Injection*), which captures fault injection scenarios under different adversarial models.

2. We analyze one SPN-based and two Feistel-based white-box ciphers under this notion, and show that Feistel-based ciphers are more vulnerable than SPN-based ones with respect to NdUF.

3. Finally, we describe a software-based fault detection mechanism.

### 1.2.1 Theoretical Modelling.

A key feature of our fault setting is that it enables a clean and structured theoretical model. The interaction between malware and the remote adversary is captured via communication patterns, divided into: (1) selection of fault positions in the implementation table, and (2) selection of fault values.

We define the xP-yV setting, where P and V denote fault positions and values respectively, and $\mathsf{x}, \mathsf{y} \in \{\text{Known (K)}, \text{Chosen (C)}, \text{Adaptive Chosen (AC)}\}$. Here, xP-yV indicates that positions are selected in the 'x' manner and values in the 'y' manner. Details are given in Section 4. In practice, malware can be pre-programmed for various communication behaviours using randomness or control signals (e.g., a seed that allows reconstruction of positions and values), requiring only minimal communication. For instance, in the adaptive case, let $p_0, \ldots, p_{F-1}$ be fault positions, and the adversary provides an initial value $v_0$ for $p_0$. The malware then computes and injects $v_1 = f_1(\cdot), v_2 = f_2(\cdot), \ldots, v_{F-1} = f_{F-1}(\cdot)$, where adaptive functions $f_i$ are embedded in the code, and '·'s are adaptive inputs understood from the context. This structure enables systematic security evaluation.

### 1.2.2 NdUF Notion Under Fault Injection.

We introduce *Non-decryptability Under Fault Injection* (NdUF), a fault-based white-box security notion. The model considers table-based implementations where an adversary can inject a number of controlled faults into tables. We clearly distinguish the original table T and the faulty table T′, as well as the sets of fault positions and values.

The main requirement is that, even after up to $F$ adversarial changes to table entries, a faulty ciphertext of a random plaintext cannot be decrypted with probability greater than $2^{-D}$. This is captured by the notion $(F, D)$-NdUF.

*Remark* 1. We remark that, the adversary is not formally prohibited from modifying the entire table. However, full-table corruption may require locating and overwriting $2^{n_{in}}$ entries, resulting in a large write footprint and a higher risk of detection, particularly for 16- or 32-bit tables. Our model therefore focuses on the practically relevant case where the adversary has a bounded write capability or must limit modifications for stealth. Note that, full corruption is the trivial endpoint, while NdUF quantifies security in the nontrivial partial-corruption regime. Also in practice, modern and future systems would deploy security monitors [21] to track malware activities. With security monitors, malware is not expected to work continuously to bypass the monitors. Therefore, corrupting a large chunk of memory may become harder in practice.

### 1.2.3 NdUF Security Analysis of WEM and SPACE.

We provide the following.

- A detailed NdUF evaluation of WEM, measuring the probability of decrypting a faulty ciphertext. In all models, an adversary must control a sufficiently large number of table entries to meaningfully affect decryption. Since encryption uses many independent table look-ups across several rounds, the success probability decreases as the number of rounds and look-ups increases. Even in the strongest model, where both positions and values are chosen, the structure of WEM limits the adversary's advantage (see Table 1).
- A detailed NdUF analyses of SPACE and Galaxy. In general, these Feistel-based schemes remain secure when the adversary has limited control over fault values, but their security margin decreases under stronger and more structured fault capabilities. In weaker settings (such as KP-KV and CP-KV), exploitation requires consistent control over many rounds, and the adversary's influence decreases quickly as the number of rounds increases, leading to controlled security degradation. In stronger settings (such as KP-CV and CP-CV), the ability to coordinate fault values allows the adversary to use structural dependencies to build valid plaintext–ciphertext pairs more efficiently. Thus, security decreases in the KP-CV model, and in the CP-CV setting, the attack complexity becomes very low and practical (see Table 1).

### 1.2.4 A Software-Based Fault Detection Technique.

We propose a software-based fault detection mechanism for white-box implementations. Unlike hardware redundancy, which is ineffective in the white-box setting due to precise adversarial control, our approach uses a remote trusted server that stores a correct reference implementation.

The method relies on random probe-based verification: the server selects random table positions, computes the correct values, and sends encoded results to the user, who recomputes and compares them locally; any mismatch reveals injected faults.

[1]Note that, the bounds in Table 1 for SPNbox and WEM instances appear similar because both

**Table 1:** Benchmarking[1] of the existing white-box cipher families, with respect to NdUF security. $\sigma(\mathsf{T})$ is code (table) size; K = Known, C = Chosen, AC = Adaptive Chosen, P = Position, V = Value. In "NdUF Bound", "$(\frac{\sigma(T)}{2^c}, Z)$" denotes that even if faults are injected in $\frac{1}{2^c}^{th}$ part of the table, adversary's success probability is $\leq 2^{-Z}$ (hence more the value of $Z$, better the security). The terms "w.p." and "MR" denote "with probability" and "Message Recovery", respectively.

| | | NdUF Bound | | | |
|---|---|---|---|---|---|
| Cipher | Reference | KP-KV/ CP-KV | ACP-KV | KP-CV | CP-CV |
| WEM-16 | Section 5 | $(\frac{\sigma(T)}{2^{1.33}}, 128)$ | $(\frac{\sigma(T)}{2^{1.33}}, 119.56)$ | $(\frac{\sigma(T)}{2^{1.33}}, 128)$ | $(\frac{\sigma(T)}{2^{1.33}}, 118.47)$ |
| WEM-32 | Section 5 | $(\frac{\sigma(T)}{2^{2.67}}, 128)$ | $(\frac{\sigma(T)}{2^{2.67}}, 104.3)$ | $(\frac{\sigma(T)}{2^{2.67}}, 128)$ | $(\frac{\sigma(T)}{2^{2.67}}, 103.86)$ |
| SPNbox-16 | Section 5 | $(\frac{\sigma(T)}{2^{1.6}}, 128)$ | $(\frac{\sigma(T)}{2^{1.6}}, 119.64)$ | $(\frac{\sigma(T)}{2^{1.6}}, 128)$ | $(\frac{\sigma(T)}{2^{1.6}}, 118.88)$ |
| SPNbox-32 | Section 5 | $(\frac{\sigma(T)}{2^{3.2}}, 128)$ | $(\frac{\sigma(T)}{2^{3.2}}, 04.44)$ | $(\frac{\sigma(T)}{2^{3.2}}, 128)$ | $(\frac{\sigma(T)}{2^{3.2}}, 104.26)$ |
| SPACE-16 | Section 6 | $(\frac{\sigma(T)}{2}, 128)$ | $(\frac{\sigma(T)}{2}, 119.53)$ | MR Attack w.p. $2^{-119.43}$ when $F = \frac{\sigma(T)}{4}$ | MR Attack w.p. $2^{-16}$ when $F = \frac{\sigma(T)}{4}$ |
| SPACE-32 | Section 6 | $(\frac{\sigma(T)}{2}, 128)$ | $(\frac{\sigma(T)}{2}, 103.53)$ | MR Attack w.p. $2^{-103.43}$ when $F = \frac{\sigma(T)}{4}$ | MR Attack w.p. $2^{-8}$ when $F = \frac{\sigma(T)}{4}$ |
| Galaxy-16 | Appendix B | $(\frac{\sigma(T)}{2^2}, 128)$ | $(\frac{\sigma(T)}{2^2}, 118.80)$ | MR Attack w.p. $2^{-117.91}$ when $F = \frac{\sigma(T)}{4}$ | MR Attack w.p. $2^{-16}$ when $F = \frac{\sigma(T)}{4}$ |
| Galaxy-32 | Appendix B | $(\frac{\sigma(T)}{2^2}, 128)$ | $(\frac{\sigma(T)}{2^2}, 102.80)$ | MR Attack w.p. $2^{-102.95}$ when $F = \frac{\sigma(T)}{4}$ | MR Attack w.p. $2^{-8}$ when $F = \frac{\sigma(T)}{4}$ |

### 1.3 Paper Organization

Section 2 introduces the fundamental notation, conventions, and formal definitions that are used throughout the paper. Section 3 presents concise specifications of our target constructions. In Section 4, we define the new white-box fault-based security notion NdUF, together with the associated adversarial models and their relationships. Section 5 provides a detailed NdUF analysis of WEM, while Section 6 presents the corresponding NdUF analysis of SPACE. In Section 7, we propose a software-based fault detection technique capable of identifying fault injections with a high probability. Finally, we conclude at Section 8.

## 2 Preliminaries

### 2.1 Notations

For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$ and $\{0,1\}^n$ denote all $n$-bit strings. For a bit string $x$, let $|x|$ be its length in bits, and $\mathsf{lsb}_l(x)$, $\mathsf{msb}_l(x)$ its least/most significant $l$ bits ($0 \leq l \leq |x|$). The concatenation of two bit strings $x$ and $y$ is denoted by $x \| y$. For a string $S$ of length $n$, $S[a:b]$ denotes the $(b-a)$-bit substring starting at position $a$, where $0 \leq a \leq b \leq n-1$. For a string $S$ of length $n$, $S[a:b]$ denotes the substring of $S$ starting at position $a$ and ending at position $b-1$, with length $b-a$, where $0 \leq a \leq b \leq n$. The notation $a\%b$ denotes the remainder of $a$ modulo $b$. In the white-box setting, $\mathsf{T}'$ denotes the table obtained after injecting faults into $\mathsf{T}$, and $F$ is the total number of faults. Let $\mathcal{P} = \{p_0, \ldots, p_{F-1}\}$ and $\mathcal{V} = \{v_0, \ldots, v_{F-1}\}$ be the sets of fault positions and values, and $\mathcal{F} = \{(p_0, v_0), \ldots, (p_{F-1}, v_{F-1})\}$ the corresponding position–value pairs. Finally, $\Pr[X]$ denotes the probability of event $X$.

constructions share a closely related SPN structure consisting of repeated layers of SBoxes followed by diffusion layers. The NdUF bounds primarily depend on the total number of SBox invocations. SPNbox is slightly weaker because it employs ten rounds, whereas WEM employs eleven rounds. Consequently, WEM contains more SBox invocations and achieves the same security level under a slightly larger number of faults.

## 2.2 White-Box Encryption

**Definition 1** (Encryption Scheme)**.** An encryption scheme is a mapping $\mathsf{E} : \mathcal{K} \times \mathcal{R} \times \mathcal{M} \rightarrow \mathcal{C}$, that takes as input a key $\mathsf{K} \in \mathcal{K}$, a plaintext $M \in \mathcal{M}$, and optionally some randomness $r \in \mathcal{R}$ and outputs a ciphertext $C \in \mathcal{C}$. Additionally, $\mathsf{E}$ must be invertible so that there exists a decryption function $\mathsf{D} : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$, for which $\mathsf{D}(\mathsf{K}, \mathsf{E}(\mathsf{K}, r, M)) = M$, $\forall \mathsf{K}, r, M$. For simplicity, we will avoid randomized definitions below, as our candidate ciphers are deterministic.

**Definition 2** (White-box Compiler)**.** A compiler $\mathsf{Com}$ for an encryption scheme $\mathsf{E} : \mathcal{K} \times \mathcal{R} \times \mathcal{M} \rightarrow \mathcal{C}$ is a function from $\mathcal{K}$ to $\mathcal{T}$. $\mathsf{Com}$ outputs a table $\mathsf{T} \in \mathcal{T}$, where $\mathcal{T}$ is called the table space. Then there exists a program $\tilde{\mathsf{E}}_{\mathsf{T}}$, functionally the same as $\mathsf{E}_{\mathsf{K}}$ (i.e., $\tilde{\mathsf{E}}_{\mathsf{T}}(M) = \mathsf{E}_{\mathsf{K}}(M)$, $\forall M \in \mathcal{M}$). We denote the size of the table (code) by $\sigma(\mathsf{T})$. Notably, $\sigma(\mathsf{T})$ is significantly larger than the size of $\mathsf{K}$.

**Definition 3** (White-box Encryption Scheme [14] )**.** A white-box encryption scheme is a tuple of an encryption scheme and its implementation, i.e, $(\mathsf{E}, \tilde{\mathsf{E}})$ such that, $\mathsf{E} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, $\tilde{\mathsf{E}} : \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{C}$, along with a white-box compiler $\mathsf{Com} : \mathcal{K} \rightarrow \mathcal{T}$, such that $\forall \mathsf{K} \in \mathcal{K}$, $\mathsf{E}(\mathsf{K}, \cdot)$ (or simply $\mathsf{E}_{\mathsf{K}}(\cdot)$) is functionally equivalent to $\tilde{\mathsf{E}}(\mathsf{T}, \cdot)$ (or simply $\tilde{\mathsf{E}}_{\mathsf{T}}(\cdot)$) where $\mathsf{T} = \mathsf{Com}(\mathsf{K})$.

From Definition 3, $\mathsf{E}$ can be viewed as a traditional encryption scheme based on a master key $\mathsf{K} \in \mathcal{K}$, and $\tilde{\mathsf{E}}$ can be viewed as its white-box implementation version based on the table $\mathsf{T} \leftarrow \mathsf{Com}(\mathsf{K})$. To differentiate, $\mathsf{E}$ can be called the cipher, and $\tilde{\mathsf{E}}$ is the white-box implementation.

## 2.3 Black-Box Attack Model

The black-box model is a traditional attack model for symmetric key ciphers.

### 2.3.1 Adversarial Abilities.

The adversary is assumed to have black-box access to the cipher. The adversary can make adaptive encryption or decryption queries with chosen plaintexts or ciphertexts, respectively.

### 2.3.2 Security Requirements.

The adversary targets the master key recovery or distinguishes the cipher from a randomly sampled counterpart. The security requirements are as follows. (1) It is computationally hard to compute the master secret key, and (2) it is computationally hard to distinguish the cipher from a randomly sampled permutation. The complexity is given by the time, data, and memory complexities. For example, an 128-bit secure cipher depicts that the time complexity for key recovery is at least $2^{128}$ encryptions. Also, the distinguishing bound is evaluated by the number of necessary black-box queries.

## 2.4 White-box Attack Model With Respect to Leakage

In traditional white-box setting, the adversary aims to extract the secret key or lift the implementation code. The difficulty of code lifting attack is evaluated by *Space-Hardness*, which is a variant of the *incompressibility* notion.

### 2.4.1 Space-Hardness.

If we assume that the adversary can not extract the key given full access to $\tilde{\mathsf{E}}(\mathsf{Com}(\mathsf{K}), \cdot, \cdot)$. As the adversary can not compute the ciphertext without $\mathsf{K}$, alternatively, she can directly use $\tilde{\mathsf{E}}(\mathsf{Com}(\mathsf{K}), \cdot, \cdot)$ as a larger key. This opens an avenue of a new attack, where the adversary lifts $\tilde{\mathsf{E}}(\mathsf{Com}(\mathsf{K}), \cdot, \cdot)$ (instead of $\mathsf{K}$) to another collaborative adversary. This is called a *Code-Lifting* attack. To address the code-lifting scenario, Bogdanov *et al.* [6], introduced the security notion denoted by *Space-Hardness*.

**Definition 4.** ***((A, B) Space-Hardness)[6, 7]***
An implementation of a block cipher $\mathsf{E}_\mathsf{K}$ is $(A, B)$ space-hard if it is infeasible to encrypt (decrypt) any randomly chosen plaintext (ciphertext) with a probability of more than $2^{-B}$, given any leaked code (through the table entries) of size less than $A$ bits.

Intuitively, $(A, B)$ Space-Hardness provides an estimate on the leakage size $A$ from the code, that can be isolated from the white-box environment to encrypt (or decrypt) any plaintext (or ciphertext) with probability more than $2^{-B}$.

Generally, the adversary is allowed to leak any part of the code, but the master secret key is embedded inside the table; hence, it is reasonable to assume the adversary can leak only the table entries. To analyze the bounds for space hardness, the following 3 adversarial models are considered based on the adversarial ability as presented in [7].

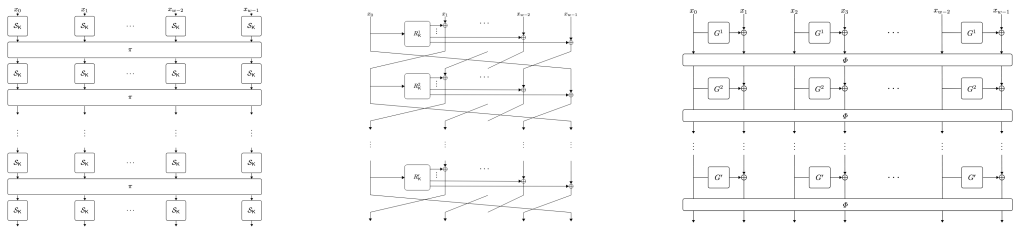
1. **Known-Space (KS) Attack:** The adversary has the knowledge of $t$ known input-output pairs from the tables. We denote the adversary by $\mathcal{A}_{KS}$.

2. **Chosen-Space (CS) Attack:** The adversary has the knowledge of $t$ previously chosen input-output pairs from the tables. We denote the adversary by $\mathcal{A}_{CS}$.

3. **Adaptively-Chosen-Space (ACS) Attack:** The adversary obtains $t$ adaptively chosen input-output pairs from the tables, which means she gets the output of an input before choosing the next input. We denote the adversary by $\mathcal{A}_{ACS}$.

# 3 Specifications of Our Target Constructions

In this section, we compactly specify our target constructions (1) WEM [11], (2) SPACE [6], and (3) Galaxy [20]. These three constructions cover both the SPN and Feistel cases and are reasonable candidates for our analysis. We only provide a generic description of the constructions and do not specify the underlying components, as the specific description is not significant for our work.



**(a)** WEM. **(b)** SPACE. **(c)** Galaxy.

**Figure 2:** Our target ciphers: (a) WEM, (b) SPACE, (c) Galaxy.

## 3.1 Black-Box and White-Box Specification of WEM

### 3.1.1 Black-Box Specification of WEM.

WEM (see Figure 2a), is a family of white-box block ciphers that is based on incompressible permutations and the classical Even-Mansour construction. WEM uses a keyed SBox $\mathcal{S}_{\mathsf{K}} : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{in}}$ (can be viewed as a small $n_{in}$-bit block cipher) in the substitution layer $\mathsf{Sub}_{\mathsf{K}} : \{0,1\}^{n} \rightarrow \{0,1\}^{n}$; where the key $\mathsf{K}$ is sampled uniformly at random, and a random public permutation $\pi : \{0,1\}^{n} \rightarrow \{0,1\}^{n}$ in the diffusion layer. $\mathsf{Sub}_{\mathsf{K}}$ is $w = \frac{n}{n_{in}}$ applications of $\mathcal{S}_{\mathsf{K}}$ in parallel. The version of WEM with $\mathsf{r}$ rounds, and $n_{in}$-bit $\mathcal{S}_{\mathsf{K}}$ is denoted by

$$\mathsf{WEM}^{\mathsf{r}}_{n_{in}} = \mathsf{Sub}_{\mathsf{K}} \circ \underbrace{(\pi \circ \mathsf{Sub}_{\mathsf{K}}) \circ \cdots \circ (\pi \circ \mathsf{Sub}_{\mathsf{K}})}_{\mathsf{r}\ times}(M).$$

### 3.1.2 White-Box Implementation of WEM.

In the white-box implementation of WEM, the underlying $\mathcal{S}_{\mathsf{K}}$ is implemented by a table $\mathsf{T}$ with $2^{n_{in}}$ elements, such that $\mathsf{T}(i) = \mathcal{S}_{\mathsf{K}}(i)$, with $0 \leq i \leq 2^{n_{in}} - 1$. Note that, the underlying $\mathcal{S}_{\mathsf{K}}$ needs to be an incompressible permutations such that $\mathsf{T}$ represents an incompressible implementation of $\mathcal{S}_{\mathsf{K}}$.

### 3.1.3 Instances for WEM.

The instances of WEM proposed in [11] are as follows.

- WEM-8 with $\mathsf{r} = 11$, $n = 128$, $n_{in} = 8$, $w = 16$
- WEM-16 with $\mathsf{r} = 11$, $n = 128$, $n_{in} = 16$, $w = 8$
- WEM-32 with $\mathsf{r} = 11$, $n = 128$, $n_{in} = 32$, $w = 4$

*Remark* 2. Note that, [11] proposes the instances WEM-8, and WEM-16, but not WEM-32, possibly for memory efficiencies. We intend to use WEM-32 in our analysis, as it uses a large 32-bit table and hence is more applicable and reasonable for the fault injection scenario (as it can resist more faults). Also, we want to exclude WEM-8 from our analysis, as WEM-8 uses a small table, and it may be easier to fault the full table (though our analysis will hold for WEM-8). Note that, theoretically our proof depends on the fraction of corrupted-entries and applies to any table size. Practically, 8-bit-tables have only $2^8$-entries and are easier to corrupt, whereas 32-bit-tables have $2^{32}$-entries,making large-scale-corruption harder.

## 3.2 Black-Box and White-Box Specification of SPACE

### 3.2.1 Black-Box Specification of SPACE.

SPACE (see Figure 2b) is a family of white-box secure $n$-bit block ciphers, based on the generalized Feistel structure. We provide a generic black-box description needed to address our work. SPACE uses the keyed round function $RF^{j}_{\mathsf{K}} : \{0,1\}^{n} \rightarrow \{0,1\}^{n}$, where the key $\mathsf{K}$ is sampled uniform at random from the respective key space such that

$$RF^{j}_{\mathsf{K}}(\mathsf{X}^{j-1}) = (R^{j}_{\mathsf{K}}(\mathsf{X}^{j-1}_{0}) \oplus (\mathsf{X}^{j-1}_{1} \| \mathsf{X}^{j-1}_{2} \| \cdots \| \mathsf{X}^{j-1}_{w-1})) \| \mathsf{X}^{j-1}_{0}.$$

We denote the $n$-bit input state of $j^{th}$ round as $\mathsf{X}^{j-1} = (\mathsf{X}^{j-1}_{0}, \mathsf{X}^{j-1}_{1}, \cdots, \mathsf{X}^{j-1}_{w-1})$, with $\mathsf{X}^{j-1}_{i} \in \{0,1\}^{n_{in}}$ for $0 \leq i \leq w-1$ (i.e, $w = \frac{n}{n_{in}}$ is the number of branches). $R^{j}_{\mathsf{K}} : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n-n_{in}}$, for the $j^{th}$ round is defined as

$$R^{j}_{\mathsf{K}}(x) = R_{\mathsf{K}}(x) \oplus (0^{(w-2)n_{in}} \| j), \text{ where } R_{\mathsf{K}}(x) = msb_{n-n_{in}}(\mathsf{E}_{\mathsf{K}}(0^{n-n_{in}} \| x)).$$

Here, $\mathsf{E}_\mathsf{K}$ is an $n$-bit block cipher with a $k$-bit key $\mathsf{K}$. Note that the length of $x$ in bits is $n_{in}$. The version of SPACE with $\mathsf{r}$ rounds and round functions $R^j_\mathsf{K}$'s is denoted as $\mathsf{SPACE}^\mathsf{r}_{n_{in}}$ and it is defined as

$$\mathsf{SPACE}^\mathsf{r}_{n_{in}}(\mathsf{X}^0) = RF^\mathsf{r}_\mathsf{K} \circ RF^{\mathsf{r}-1}_\mathsf{K} \circ \cdots \circ RF^1_\mathsf{K}(\mathsf{X}^0)$$

#### 3.2.2 White-Box Implementation of SPACE.

In the white-box implementation of SPACE, the underlying function $R_\mathsf{K} : \{0,1\}^{n_{in}} \to \{0,1\}^{n-n_{in}}$ is implemented by a table $\mathsf{T}$ with $2^{n_{in}}$ elements each of $n - n_{in}$ many bits, such that $\mathsf{T}(i) = R_\mathsf{K}(i)$, with $0 \leq i \leq 2^{n_{in}} - 1$.

#### 3.2.3 Instances for SPACE.

The instances of SPACE proposed in [6] are as follows.

- SPACE-8 with $\mathsf{r} = 300$, $n = 128$, $n_{in} = 8$, $w = 16$
- SPACE-16 with $\mathsf{r} = 128$, $n = 128$, $n_{in} = 16$, $w = 8$
- SPACE-24 with $\mathsf{r} = 128$, $n = 128$, $n_{in} = 24$, $w = 16$
- SPACE-32 with $\mathsf{r} = 128$, $n = 128$, $n_{in} = 32$, $w = 4$

*Remark* 3. We describe our analyses for SPACE-16, and SPACE-32. From a practical perspective, SPACE-8 uses a small table of size in bytes, and it may be easier to fault the full table. Precisely, SPACE-8 maintains an 8-bit table with only $2^8$ entries and can therefore be corrupted more easily. In contrast, a 32-bit table contains $2^{32}$ entries, making large-scale corruption substantially more difficult in practice. Hence, it is reasonable to exclude SPACE-8. In addition, SPACE-24 has a different structure (three branches are fed to the underlying lookup table, and the branches are unbalanced) as the other variants and traditional Feistel. Hence, to keep things simpler, we exclude SPACE-24 as well. However, from a theoretical perspective, our analyses idea is the same for all the variants.

### 3.3 Black-Box and White-Box Specification of Galaxy

#### 3.3.1 Black-Box Specification of Galaxy.

Galaxy (see Figure 2c) is a family of white-box secure $n$-bit block ciphers with $k$-bit secret key, based on type-2 generalized Feistel construction. Galaxy uses the round function $RF^j_{n_{in},\phi} : \{0,1\}^n \to \{0,1\}^n$, such that

$$RF^j_{n_{in},\phi}(\mathsf{X}^{j-1}) = \Phi(\mathsf{X}^{j-1}_0, G^j(\mathsf{X}^{j-1}_0) \oplus \mathsf{X}^{j-1}_1, \cdots, \mathsf{X}^{j-1}_{w-2}, G^j(\mathsf{X}^{j-1}_{w-2}) \oplus \mathsf{X}^{j-1}_{w-1}).$$

We denote the $n$-bit input state of $j^{th}$ round as $\mathsf{X}^{j-1} = (\mathsf{X}^{j-1}_0, \mathsf{X}^{j-1}_1, \cdots, \mathsf{X}^{j-1}_{w-1})$, with $\mathsf{X}^{j-1}_i \in \{0,1\}^{n_{in}}$ for $0 \leq i \leq w-1$ (i.e, $w = \frac{n}{n_{in}}$ is the number of branches). The function $\Phi : \{0,1\}^n \to \{0,1\}^n$ is defined as

$$\Phi(\mathsf{X}^j_0, \mathsf{X}^j_1, \cdots, \mathsf{X}^j_{w-1}) = (\mathsf{X}^j_{\phi(0)}, \mathsf{X}^j_{\phi(1)}, \cdots, \mathsf{X}^j_{\phi(w-1)}),$$

where $\phi : \{0, 1, \cdots, w-1\} \to \{0, 1, \cdots, w-1\}$ is a positional permutation. $G^j : \{0,1\}^{n_{in}} \to \{0,1\}^{n_{in}}$, for the $j^{th}$ round is defined as

$$G^j(i) = G(i) \oplus j, \text{ where } G(i) = \phi_{n_{in}}(keygen(\mathsf{K}), i).$$

Here, $keygen(\mathsf{K})$ denotes the sequence of key stream $S$ derived from the underlying stream cipher with $k$-bit secret key $\mathsf{K}$ (where the key $\mathsf{K}$ is sampled uniform at random from the

respective key space) and $\phi_{n_{in}}(S, i) = S[i.n_{in} : (i+1).n_{in}]$ is the $n_{in}$-bit substring of $S$, starting from the $(i \cdot n_{in})$-th bit, where $0 \leq i \leq 2^{n_{in}} - 1$. Note that the length of $i$ in bits is $n_{in}$. The version of Galaxy with r rounds and the functions $G^j$s is denoted as $\mathsf{Galaxy}^{\mathsf{r}}_{n_{in},\phi}$ and it is defined as

$$\mathsf{Galaxy}^{\mathsf{r}}_{n_{in},\phi}(\mathsf{X}^0) = RF^{\mathsf{r}}_{n_{in},\phi} \circ RF^{\mathsf{r}-1}_{n_{in},\phi} \circ \cdots \circ RF^{1}_{n_{in},\phi}(\mathsf{X}^0)$$

#### 3.3.2 White-Box Implementation of Galaxy.

In the white-box implementation of Galaxy, the underlying function $G : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{in}}$ is implemented by a table T with $2^{n_{in}}$ elements each of $n_{in}$ many bits, such that $\mathsf{T}(i) = G(i)$, with $0 \leq i \leq 2^{n_{in}} - 1$.

#### 3.3.3 Instances for Galaxy.

The instances of Galaxy proposed in [20] are as follows.

- Galaxy-8 with $\mathsf{r} = 8$, $n = 128$, $n_{in} = 8$, $w = 16$, and $\phi = \{5, 0, 1, 4, 7, 12, 3, 8, 13, 6, 9, 2, 15, 10, 11, 14\}$
- Galaxy-16 with $\mathsf{r} = 16$, $n = 128$, $n_{in} = 16$, $w = 8$, and $\phi = \{3, 0, 1, 4, 7, 2, 5, 6\}$
- Galaxy-32 with $\mathsf{r} = 32$, $n = 128$, $n_{in} = 32$, $w = 4$, and $\phi = \{3, 0, 1, 2\}$

*Remark* 4. For a similar reason as in Remark 3, we describe our analyses with respect to Galaxy-16, and Galaxy-32 only.

## 4 A New Fault-Based White-Box Security Notion: NdUF

In this section, we introduce the **white-box attack models with respect to fault injections**. Note that the adversary in the white-box setting is extremely powerful and is assumed to inject a fault with complete precision. Hence, it is reasonable to assume that the adversary will be able to replace any table entry with any value it wants, though there is an upper bound on the number of entries it can alter. The security requirement for the adversary is as follows.

### 4.1 Security Requirement

The adversary, with its abilities, targets to inject a certain number of faults into the implementation tables. In particular, he can put his own values into the program table such that the adversary has control over how the table should behave. The white-box implementation should ensure that, even if a part of the table is corrupted with adversary-chosen values, a *faulty* ciphertext computed with this fault-injected implementation can not be decrypted with high probability. We denote this notion as ***N**on-**d**ecryptability **U**nder **F**ault Injection* or ***NdUF***.

Intuitively, the adversary injects a bounded but large number of faults into the implementation table for a single session. We denote this fault type as a *one-time* fault. A construction secure in this fault setting is denoted as NdUF under single-session fault or ***NdUF***. Below, we formally define the security notion under this fault injection setting.

### 4.2 The NdUF Security Notion

To show the tradeoff between the code injected with fault and the advantage of the attacker, we introduce a new security notion named *(F, D)-NdUF*, defined below.

**Definition 5.** ***((F, D)-NdUF)*** A white-box cipher $(\mathsf{E_K}, \tilde{\mathsf{E}}_\mathsf{T})$ is $(F, D)$-NdUF if it is infeasible to decrypt the faulty ciphertext corresponding to any randomly chosen plaintext with a probability of more than $2^{-D}$, given less than or equal to $F$ entries of the implementation table have been fault-injected.

Intuitively, $(F, D)$-NdUF provides an estimate of the number of fault-injected table entries that can be exploited to decrypt any faulty ciphertext for a random plaintext with probability more than $2^{-D}$.

Note that, our notion intentionally models persistent pre-challenge corruption. The fault injections are given before the challenge. This ordering allows us to isolate the effect of pre-positioned faults and establish the basic framework clearly. Allowing challenge-dependent corruption defines a stronger and qualitatively different model, which may trivialize some chosen-value settings. We would like to consider post-challenge corruption as an extended work outside the scope of the present results.

## 4.3 Fault-Based Attack Model

In this section, we describe the fault-based attack model in which faults are injected only into the table entries. Under this model, the local adversary (e.g, malware) that injects faults is incorporated into the challenger $Ch$, and the remote adversary is considered as the adversary $\mathcal{A}$. Now, the attack model, depicted in Figure 3, is simply described as follows:

- First, the challenger selects a key $\mathsf{K}$ uniformly at random from the key space $\mathcal{K}$ and computes the table $\mathsf{T}$ by executing the white-box compiler $Com(\mathsf{K})$ for the white-box implementation $\tilde{\mathsf{E}}_\mathsf{T}$.
- Next, the set of fault positions and corresponding values, denoted by $\mathcal{F}$, is determined through the interaction between the challenger and the adversary, according to the specified adversarial model.
- Then those faults are injected into the table $\mathsf{T}$, resulting in a modified table $\mathsf{T}'$ and the white-box implementation becomes $\tilde{\mathsf{E}}_{\mathsf{T}'}$.
- Finally, the challenger computes the ciphertext $C = \tilde{\mathsf{E}}_{\mathsf{T}'}(M)$, for a random plaintext $M$, and sends the challenge ciphetext $C$ to the adversary. Then, the ultimate goal of the adversary is to recover the original plaintext $M$ from the ciphertext $C$, that is, to output an $M'$ so that $M = M'$.

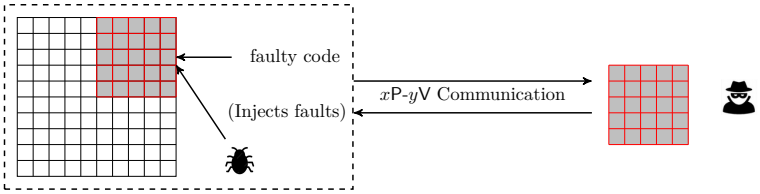



**Figure 3:** Fault-based attack model.

## 4.4 Adversarial Models of NdUF

We introduce several adversarial settings. Depending on the power of the adversary, six adversarial models, denoted as **(xP-yV)**, where **x, y** $\in$ {**K, C, AC**}, with **K** being *known*, **C** being *chosen*, and **AC** being *adaptively chosen*. Note that, *ideally there should be nine such models. However, we observe that it is not reasonable to consider the three models ACP-CV, CP-ACV, and ACP-ACV, since the remote adversary is computing the positions and the values all by herself, and the term "**adaptive**" does not carry any impact.* The six models are described below and illustrated in Figure 4 for better understanding. To describe this, we assume that $p_i$ denotes the index (i.e., the position) of the table

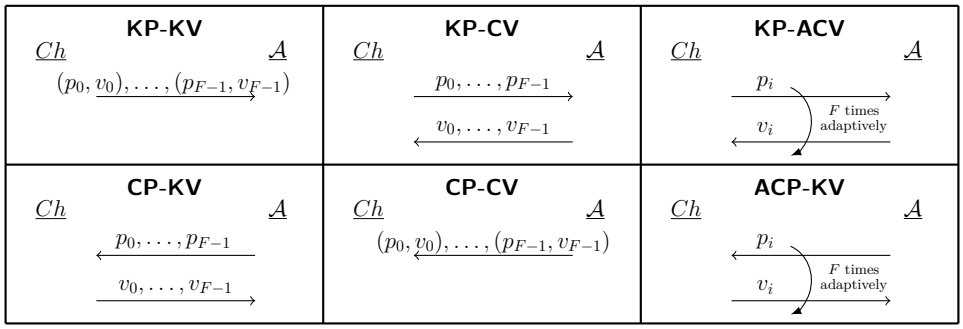



**Figure 4:** Six adversary models depending on the communication between the adversary $\mathcal{A}$ and the challenger $Ch$ to create faulted data.

$\mathsf{T}$ and the fault values $v_i$ are injected into $\mathsf{T}(p_i)$ for $i \in \{0, 1, \cdots, F-1\}$ by the fault-injector.

1. **Known Position Known Value (KP-KV):** The adversary is provided with fault position–value pairs $(p_i, v_i)$ for $i \in \{0, 1, \ldots, F-1\}$, without having any control over the fault positions and the injected values.
2. **Known Position Chosen Value (KP-CV):** The adversary is given a set of fault positions $p_i$, where $i \in \{0, 1, \ldots, F-1\}$, and can arbitrarily select the corresponding fault values $v_i$, thereby defining $\mathsf{T}(p_i) = v_i$ for each $i$.
3. **Known Position Adaptively Chosen Value (KP-ACV):** The adversary has no control over fault positions, but is allowed to choose corresponding fault values adaptively. For each $i \geq 1$, $v_i$ may depend on the previously revealed positions and assigned values. Thus, the adversary receives the fault positions $p_i$, $i \in \{0, 1, \cdots, F-1\}$ sequentially and selects appropriate values for each position based on the information obtained so far.
4. **Chosen Position Known Value (CP-KV):** The adversary first selects a sequence of fault positions $p_i$, where $i \in \{0, 1, \ldots, F-1\}$, and is subsequently provided with the corresponding fault values $v_i$.
5. **Chosen Position Chosen Value (CP-CV):** The adversary selects a sequence of fault position–value pairs $(p_i, v_i)$, where $i \in \{0, 1, \ldots, F-1\}$ of her own choice.
6. **Adaptively Chosen Position Known Value (ACP-KV):** The adversary is allowed to choose the fault positions $p_i$ adaptively, for $i \in \{0, 1, \ldots, F-1\}$, while there is no control over the corresponding fault values $v_i$s. For $i \geq 1$, the choice of $p_i$ depends on the previously selected positions and the values $v_0, v_1, \ldots, v_{i-1}$. Thus, the adversary selects positions sequentially, adapting each choice based on the information obtained so far, while having no control over the injected fault values.

### 4.4.1 Reductions of the Above Models

In this section, we denote that a model $A$ reduces to model $B$ by $A \prec B$, i.e., when $A$ is not secure implies $B$ is not secure.

**KP-ACV $\prec$ KP-CV.** To show that KP-ACV reduces to the KP-CV model (see Figure 8a), we construct an adversary for KP-CV, denoted $\mathcal{A}_{\mathsf{KP\text{-}CV}}$, that breaks security in the KP-CV model by using as a subroutine an adversary for KP-ACV, denoted by $\mathcal{A}_{\mathsf{KP\text{-}ACV}}$, which is assumed to break security in the KP-ACV model. The reduction is as follows:

1. After obtaining the positions $p_i$ for $i \in \{0, 1, \cdots, F-1\}$, the adversary $\mathcal{A}_{\mathsf{KP\text{-}CV}}$ invokes the adversary $\mathcal{A}_{\mathsf{KP\text{-}ACV}}$ as a subroutine on input $p_i$'s. For each such $p_i$, $\mathcal{A}_{\mathsf{KP\text{-}ACV}}$ sets the value $v_i$ adaptively, to $\mathsf{T}'(p_i)$, which are set accordingly by $\mathcal{A}_{\mathsf{KP\text{-}CV}}$. Afterwards, $\mathcal{A}_{\mathsf{KP\text{-}CV}}$ receives the challenge ciphertext $C$ and invokes $\mathcal{A}_{\mathsf{KP\text{-}ACV}}$ on input $C$.
2. The adversary $\mathcal{A}_{\mathsf{KP\text{-}ACV}}$ outputs a plaintext $M'$ corresponding to the ciphertext $C$.
3. Finally, the adversary $\mathcal{A}_{\mathsf{KP\text{-}CV}}$ outputs $M'$.

**CP-KV $\prec$ ACP-KV.** To show that CP-KV reduces to ACP-KV (see Figure 8b), we construct an adversary $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$ that breaks the ACP-KV model by using as a subroutine an adversary for CP-KV, denoted by $\mathcal{A}_{\mathsf{CP\text{-}KV}}$, which is assumed to break security in the CP-KV model. The reduction is described as follows:

1. After choosing a series of positions $p_i$ for $i \in \{0, 1, \cdots, F-1\}$, the adversary $\mathcal{A}_{\mathsf{CP\text{-}KV}}$ which is invoked by the adversary $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$ as a subroutine sends the positions to $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$. Then, $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$ obtains the corresponding values $v_i$ adaptively, for the positions $p_i$, for $i \in \{0, 1, \cdots, F-1\}$. Afterwards, $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$ receives the challenge ciphertext $C$ and invokes $\mathcal{A}_{\mathsf{CP\text{-}KV}}$ on input the values $v_i$ for $i \in \{0, 1, \cdots, F-1\}$ and the challenge ciphertext $C$.
2. The adversary $\mathcal{A}_{\mathsf{CP\text{-}KV}}$ outputs a plaintext $M'$ corresponding to $C$.
3. Finally, the adversary $\mathcal{A}_{\mathsf{ACP\text{-}KV}}$ outputs $M'$.

### 4.4.2 Ideas Behind Our NdUF Analysis.

In this section, we discuss the NdUF analysis idea. An interesting aspect of this NdUF notion is that it can be viewed as a generalization of the Space Hardness notion (see Section 2.4.1), or in other words, Space Hardness is theoretically a special case of this notion. Precisely, consider the xP-KV scenarios with $\mathsf{x} \in \{\mathsf{K}, \mathsf{C}, \mathsf{AC}\}$. In all these scenarios, the malware sets the fault values and gives them to the remote adversary. In case, the malware does not alter (that means zero fault) the table entries and gives it to the remote adversary, then KP-KV, CP-KV, and ACP-KV correspond to the Space Hardness notion (captures white-box leakage scenario) in the $KS$, $CS$, and the $ACS$ settings respectively as defined in Section 2.4.1. We provide our NdUF analysis of WEM and SPACE. We also provide NdUF analysis of Galaxy but move it to Appendix B due to the shortage of space.

- **NdUF Analysis of WEM Instances:** For WEM, when the fault-injected table is a permutation, then the adversary can optimize the message recovery probability, because if the faulty table is not a permutation, then the faulty challenge ciphertext may have multiple preimages. Hence, we consider only the case that the faulty table is a permutation.
  - We first consider the xP-KV scenarios, with $\mathsf{x} \in \{\mathsf{K}, \mathsf{C}, \mathsf{AC}\}$. For KP-KV, the analysis assumes the challenger can easily maintain permutation in the faulty table, and hence, the analysis will be similar to the Space Hardness in the KS model. For CP-KV, the challenger again can maintain the permutation property, and the analysis will be similar to that of KP-KV. However, in the ACP-KV case, the positions are requested by the remote adversary to the challenger adaptively, and hence it is hard to maintain permutation by injecting faults. The only way the adversary can maintain permutation is that it does not inject any fault, only receives leaked table entries from the challenger (that means zero faults). Hence, in this case, the analysis will be similar to the Space Hardness analysis under the ACS setting.

  - We next talk about the analyses in KP-CV, KP-ACV, and CP-CV. We provide dedicated security proofs for that. Note that, KP-ACV $\prec$ KP-CV, hence, it is sufficient to upper bound KP-CV to provide an upper bound for KP-ACV. CP-CV bound should be computed independently.

- **NdUF Analysis of SPACE Instances:** The SPACE instances, on the other hand, do not need the faulty table to be a permutation. The NdUF analysis under the xP-KV settings should be similar to that of the Space Hard analysis for leakage. We also provide dedicated NdUF analyses under KP-CV, KP-ACV, and CP-CV.

# 5 NdUF Analysis of WEM

In this section, we provide the NdUF analysis of WEM under the KP-KV, CP-KV, ACP-KV, KP-CV, and CP-CV settings. We use $\mathsf{T}'$ to denote the fault-injected $\mathsf{T}$.

*Remark* 5. The analyses for the other existing SPN-based ciphers, such as SPNbox (with a similar structure as WEM, except SPNbox has 10 rounds, whereas WEM has 11 rounds), are exactly the same, and the same observations hold for SPNbox as well. Hence, we provide the NdUF bounds for both WEM and SPNbox.

## 5.1 NdUF Analysis of WEM Under KP-KV

**Theorem 1.** *Given $F$ distinct fault positions and their corresponding values under KP-KV adversarial model for the table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of a white-box implementation of WEM, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability of higher than $\left(\frac{F}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)}$, where $w$ is the number of table look-ups in each round and $\mathsf{r}$ is the total number of rounds.*

*Proof.* Given $F$ distinct fault positions and their corresponding values, the probability that each look-up for the table $\mathsf{T}'$ is included in the faulted entries is $\frac{F}{2^{n_{in}}}$. Suppose the adversary wants to derive all the faulty intermediate states sequentially, starting from the challenge ciphertext. Observe that the adversary can derive the intermediate faulty state before one round from a ciphertext with probability $\left(\frac{F}{2^{n_{in}}}\right)^{w}$, as $w$ is the number of table look-ups in one round. Again, the same $\mathsf{T}'$ table is used in each round for the $(\mathsf{r}+1)$ rounds of invocations of the substitution layer. Hence, the adversary can't decrypt a faulty ciphertext with a probability of more than $\left(\frac{F}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)}$. □

This bound is the same as the bound for Space Hard analysis under the KS setting. A comparative summary of NdUF bounds for the SPN ciphers under the KP-KV fault model is presented in Table 2.

## 5.2 NdUF Analysis of WEM Under CP-KV

Since the adversary has no view of the implementation table $\mathsf{T}$ in both the KP and the CP models. Thus, this model has no advantage over the KP scenario. We obtain this bound to be $(\frac{F}{2^{n_{in}}})^{w(\mathsf{r}+1)}$ (same as the KP-KV bounds in Table 2).

## 5.3 NdUF Analysis of WEM Under ACP-KV

The technique we use to upper bound the NdUF security is to compute the message recovery probability $p$ when the challenger maintains the permutation property in the faulty table. This is because, when the faulty table is a permutation, then $p$ is optimized. In the ACP-KV setting, we observe that the challenger can maintain permutation in the

faulty table only when the challenger injects zero fault, i.e., the leakage scenario. Hence, we will follow a similar technique as in the proof for the Space Hardness bound under the ACS setting in [7]. The bound is computed by Theorem 2. We mention Lemma 1 first.

**Lemma 1.** *For $m$ table look-ups, the expected number of used entries in the table with $2^{n_{in}}$ entries is estimated as $2^{n_{in}} \cdot (1 - (\frac{2^{n_{in}}-1}{2^{n_{in}}})^m)$.*

Here, we omit the proof of the lemma, already described in [7].

**Theorem 2.** *Given $F$ distinct fault positions and their corresponding values under ACP-KV adversarial model for the table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of a white-box implementation of WEM, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability higher than*

$$N \cdot 2^{-n} + (1 - N \cdot 2^{-n})(F/2^{n_{in}})^{w(r+1)}, \text{ where } N = \lceil log_{e_{in}}(1 - F/2^{n_{in}})/w(r+1) \rceil,$$

*where $e_{in} = \frac{2^{n_{in}}-1}{2^{n_{in}}}$, $w$ is the number of table look-ups per round and $\mathsf{r}$ is the total number of rounds.*

*Proof.* From Lemma 1, we observe that $2^{n_{in}} \left(1 - (e_{in})^{Nw(\mathsf{r}+1)}\right)$ positions are needed to encrypt $N$ plaintexts, as $Nw(\mathsf{r}+1)$ table look-ups are required to encrypt $N$ plaintexts. This ensures that $F$ distinct fault positions enable to compute $N = \lceil log_{e_{in}}(1 - F/2^{n_{in}})/w(\mathsf{r}+1) \rceil$ plaintext-ciphertext pairs. Then, the challenge ciphertext is included in the set of precomputed plaintext-ciphertext pairs with probability $N \cdot 2^{-n}$. Otherwise, for $F$ given distinct fault positions and their corresponding values, the challenge ciphertext can be decrypted with the probability upper bounded by $(F/2^{n_{in}})^{w(\mathsf{r}+1)}$, as Theorem 1. □

This bound is exactly the same as the bound for Space Hard analysis under the ACS setting. A comparative summary of NdUF bounds for the SPN ciphers under the ACP-KV fault model is presented in Table 2.

**Table 2:** NdUF bounds for SPN ciphers under the KP-KV and ACP-KV fault model. The results are provided for both WEM and SPNbox constructions.

| | WEM-16 | WEM-32 | SPNbox-16 | SPNbox-32 |
|---|---|---|---|---|
| KP-KV | $(\frac{\sigma(\mathsf{T})}{2^{1.33}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{2.67}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{1.6}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{3.2}}, 128)$ |
| ACP-KV | $(\frac{\sigma(\mathsf{T})}{2^{1.33}}, 119.56)$ | $(\frac{\sigma(\mathsf{T})}{2^{2.67}}, 104.13)$ | $(\frac{\sigma(\mathsf{T})}{2^{1.6}}, 119.64)$ | $(\frac{\sigma(\mathsf{T})}{2^{3.2}}, 104.44)$ |

## 5.4 NdUF Analysis of WEM Under KP-CV

In this section, we evaluate the bound for $(F, D)$-NdUF security of WEM in the KP-CV adversarial model and then provide a comparative summary of NdUF bounds for the SPN ciphers under this fault model in Table 3 at the end of the proof. The adversary first obtains fault positions and chooses the values corresponding to those positions. In this setting, the faulty table fails to be a permutation with high probability, as the actual values in the non-faulty positions are unknown to the adversary. It is also assumed that all the faulty values corresponding to the given positions are distinct. However, these faulty values may also occur in non-faulty positions. We next propose the following theorem.

**Theorem 3.** *Given $F$ distinct fault values under KP-CV adversarial model for the table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of a white-box implementation of WEM, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability higher than $\left(\frac{F+1}{2^{n_{in}}}\right)^{w(r+1)}$ with the given condition $\left(\frac{F}{2} - \frac{F}{2^{n_{in}}-F} - \frac{(2^{n_{in}}-F)\ln 2}{2}\right) \leq 0$, where $w$ is the number of table look-ups in each round and $\mathsf{r}$ is the total number of rounds and* ln *denotes logarithm base $e$.*

*Proof.* For the proof, we first define collision as the fact that there exists $x \neq x'$ such that $\mathsf{T}(x) = \mathsf{T}'(x')$, where $\mathsf{T}'$ is the table obtained from $\mathsf{T}$ after fault injection. Here, $x'$ lies in the set of faulted positions, whereas $x$ does not. Since $\mathsf{T}$ corresponds to a permutation and under the assumption that the adversary injects distinct fault values (otherwise, it may create more collisions, contradicting that the adversary tries to maximize its winning (decryption) probability) at all $F$-many specified positions, any collision in $\mathsf{T}'$ can involve **at most two** distinct inputs $x$ and $x'$ (one in a faulty and the other in a non-faulty position). Let the event **Decrypt** denotes that a faulty ciphertext (encrypted with the white-box implementation using faulty $\mathsf{T}'$) obtained from a random plaintext can be **uniquely** decrypted to that plaintext, and the event $\mathbf{Coll} = q$ denotes that there are $q$ collision pairs in $\mathsf{T}'$. Then the upper bound of the probability of **Decrypt** is the desired bound. Hence, we would like to show that, $Pr[\mathbf{Decrypt}] \leq \left(\frac{F+1}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)}$. Clearly,

$$Pr[\mathbf{Decrypt}] = \sum_{q=0}^{F} Pr[\mathbf{Decrypt}|\mathbf{Coll} = q] \cdot Pr[\mathbf{Coll} = q]. \tag{1}$$

For simplicity, let $\mathbf{A_q}$ denotes the event $(\mathbf{Decrypt}|\mathbf{Coll} = q)$. Let us use the event $(\mathbf{Miss} = i)$ to denote that $i$-many table look-ups are non-faulty during the decryption of a ciphertext corresponding to the faulty implementation with $q$ collisions in the table $\mathsf{T}'$. Now, the total number of table look-ups for a single decryption is $w(\mathsf{r}+1)$. Hence,

$$Pr[\mathbf{A_q}] = \sum_{i=0}^{w(\mathsf{r}+1)} Pr[\mathbf{A_q}|\mathbf{Miss} = i] \cdot Pr[\mathbf{Miss} = i]. \tag{2}$$

Since, there are $q$ collisions in the table $\mathsf{T}'$, then there are $(2^{n_{in}} - q)$ distinct values $\mathsf{T}'$. Then a value is included in the fault values with probability $\frac{F}{2^{n_{in}}-q}$ and the value is not included in the fault values with probability $(1 - \frac{F}{2^{n_{in}}-q})$. For each of the $i$ table look-up values are not included in the fault values with probability $(1 - \frac{F}{2^{n_{in}}-q})^i$. For the event $(\mathbf{Miss} = i)$, $(w(\mathsf{r}+1) - i)$ table look-up values are faulty, and this occurs with probability $(\frac{F}{2^{n_{in}}-q})^{(w(\mathsf{r}+1)-i)}$. In addition, there are $\binom{w(\mathsf{r}+1)}{i}$ possibilities for $i$ table look-ups, not included in the fault values. Thus,

$$Pr[\mathbf{Miss} = i] = \binom{w(\mathsf{r}+1)}{i} \cdot \left(1 - \frac{F}{2^{n_{in}} - q}\right)^i \cdot \left(\frac{F}{2^{n_{in}} - q}\right)^{(w(\mathsf{r}+1)-i).} \tag{3}$$

Since $i$ table look-ups are not included in the fault values, each of those look-ups has to be guessed from $(2^{n_{in}} - F)$ values, and the guessing probability will be $\frac{1}{2^{n_{in}}-F}$ for each of those table look-ups.

Again, for $(w(\mathsf{r}+1)-i)$ table look-ups that are included in the fault values, first we find the expected number of table look-ups that have more than one preimage under $\mathsf{T}'$. For this, the probability of obtaining table look-ups that have more than one preimage under $\mathsf{T}'$ is $\frac{q}{F}$, and there are $(w(\mathsf{r}+1)-i)$ table look-ups that are faulty. Hence, the expected number of table look-ups that have more than one preimage under $\mathsf{T}'$ is $\frac{(w(\mathsf{r}+1)-i)q}{F}$. Moreover, there are at most two possible preimages for one colliding value for the image of $\mathsf{T}'$. Thus, there exist $2^{\frac{(w(\mathsf{r}+1)-i)q}{F}}$ plaintexts corresponding to one ciphertext. Therefore,

$$Pr[\mathbf{A_q}|\mathbf{Miss} = i] = \frac{1}{2^{\frac{(w(\mathsf{r}+1)-i)q}{F}}} \cdot \left(\frac{1}{2^{n_{in}} - F}\right)^i \tag{4}$$

Now, assuming $t = w(\mathsf{r}+1)$, from the equations 2, 3, and 4, we obtain

$$\begin{aligned} Pr[\mathbf{A_q}] &= \sum_{i=0}^{t} \binom{t}{i} \frac{1}{2^{\frac{(t-i)q}{F}}} \cdot \left(\frac{1}{2^{n_{in}} - F}\right)^i \cdot \left(1 - \frac{F}{2^{n_{in}} - q}\right)^i \cdot \left(\frac{F}{2^{n_{in}} - q}\right)^{(t-i)} \\ &= \sum_{i=0}^{t} \binom{t}{i} \left(\frac{2^{n_{in}} - F - q}{(2^{n_{in}} - F)(2^{n_{in}} - q)}\right)^i \cdot \left(\frac{F}{(2^{n_{in}} - q) \cdot 2^{\frac{q}{F}}}\right)^{(t-i)} \\ &= \left(\frac{2^{n_{in}} - F - q}{(2^{n_{in}} - F)(2^{n_{in}} - q)} + \frac{F}{(2^{n_{in}} - q) \cdot 2^{\frac{q}{F}}}\right)^t \\ &= \left(\frac{1}{2^{n_{in}} - q} - \frac{q}{(2^{n_{in}} - F)(2^{n_{in}} - q)} + \frac{F}{(2^{n_{in}} - q) \cdot 2^{\frac{q}{F}}}\right)^t \\ &= \left(\frac{f(q)}{2^{n_{in}} - q}\right)^t, \end{aligned} \tag{5}$$

where, $f(q) = 1 - \frac{q}{2^{n_{in}} - F} + \frac{F}{2^{\frac{q}{F}}}$. To find the upper bound of $Pr[\mathbf{A_q}]$, let $S(q) = Pr[\mathbf{A_q}]$ and $g(q) = \frac{f(q)}{2^{n_{in}} - q}$. Then, $S(q) = (g(q))^t$ and for the upper bound of $S(q)$, it is sufficient to find the upper bound of $g(q)$. Now, differentiating $f$ and $g$ with respect to $q$, we get

$$f'(q) = -\frac{1}{2^{n_{in}} - F} - \frac{\ln 2}{2^{\frac{q}{F}}} \text{ and } g'(q) = \frac{(2^{n_{in}} - q) \cdot f'(q) + f(q)}{(2^{n_{in}} - q)^2}.$$

Again, let $h(q) = (2^{n_{in}} - q) \cdot f'(q) + f(q)$ and, differentiating it with respect to $q$, we obtain

$$\begin{aligned} h'(q) &= (2^{n_{in}} - q) \cdot f''(q) - f'(q) + f'(q) = (2^{n_{in}} - q) \cdot f''(q) \\ &= (2^{n_{in}} - q) \cdot \frac{(\ln 2)^2}{F \cdot 2^{\frac{q}{F}}} \geq 0, \end{aligned} \tag{6}$$

where, $f'$ and $f''$ denote the first and second order derivatives of $f$ respectively. Hence, $h$ is an increasing function, i.e., $h(0) \leq h(q) \leq h(F)$, for $0 \leq q \leq F$. Therefore,

$$\begin{aligned} g'(q) &= \frac{h(q)}{(2^{n_{in}} - q)^2} \leq \frac{h(F)}{(2^{n_{in}} - q)^2} \\ &= \frac{1}{(2^{n_{in}} - q)^2} \cdot \left(\frac{F}{2} - \frac{F}{2^{n_{in}} - F} - \frac{(2^{n_{in}} - F)\ln 2}{2}\right) \end{aligned} \tag{7}$$

Hence, $g$ is a decreasing function if and only if $g'(q) \leq 0$, i.e.,

$$\left(\frac{F}{2} - \frac{F}{2^{n_{in}} - F} - \frac{(2^{n_{in}} - F)\ln 2}{2}\right) \leq 0 \tag{8}$$

As $g$ is decreasing, for $0 \leq q \leq F$, $g(q) \leq g(0)$, i.e., $S(q) \leq S(0)$. From Equations 1 and 5, $Pr[\mathbf{Decrypt}] \leq S(0) \cdot \sum_{q=0}^{F} Pr[\mathbf{Coll} = q] = S(0) = \left(\frac{F+1}{2^{n_{in}}}\right)^t = \left(\frac{F+1}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)}$. □

The NdUF bound summary under KP-CV is given in Table 3. The NdUF bounds in KP-ACV are bounded by those in KP-CV (see Section 4.4.1).

## 5.5 NdUF Analysis of WEM Under CP-CV

In the CP-CV attack model, the adversary itself can prepare a set of plaintext-ciphertext pairs. To compute the bound, we first need to estimate how many table entries are required to compute $A$ plaintexts in advance. Note that, to encrypt $A$ plaintexts, $w(r+1)A$ table look-ups are needed. After injecting $F$ distinct fault values, the table is no longer a permutation with a high probability, and let $q$ distinct values appear twice (note that, we assume the adversary chooses distinct values to minimize collisions in the faulty table $\mathsf{T}'$) in the faulty table $\mathsf{T}'$. We therefore first apply the following lemma.

**Lemma 2.** *For $m$ table lookups and $q$ colliding values (as mentioned above) of $\mathsf{T}'$, the expected number of distinct values used is estimated as*

$$(2^{n_{in}} - 2q)\left(1 - \left(1 - \frac{1}{2^{n_{in}}}\right)^m\right) + q\left(1 - \left(1 - \frac{2}{2^{n_{in}}}\right)^m\right).$$

*Proof.* With distinct $q$ colliding values in $\mathsf{T}'$, $(2^{n_{in}} - 2q)$ many values occur once, and $q$ many values occur twice (i.e., collision). A value which occurs once in the faulty table is observed during $m$ table look-ups with probability $(1 - (1 - \frac{1}{2^{n_{in}}})^m)$. As there are total $(2^{n_{in}} - 2q)$ such values, the expected number for those values is $(2^{n_{in}} - 2q)\left(1 - \left(1 - \frac{1}{2^{n_{in}}}\right)^m\right)$. Again, a value which occurs twice (i.e., a collision) in the faulty table is observed during $m$ table look-ups with probability $\left(1 - \left(1 - \frac{2}{2^{n_{in}}}\right)^m\right)$. As there are total $q$ such values, the expected number for those values is $q\left(1 - \left(1 - \frac{2}{2^{n_{in}}}\right)^m\right)$. Hence, the result follows. □

Using Lemma 2, we now prove the following theorem.

**Theorem 4.** *Given $F$ distinct fault values under the CP-CV attack for the table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of a white-box implementation of WEM, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability higher than $A \cdot 2^{-n} + (1 - A \cdot 2^{-n}) \cdot \left(\frac{F}{2^{n_{in}}} + \frac{1}{2^{n_{in}} - F}\right)^{w(r+1)}$, where $A \leq \left\lceil -\frac{2^{n_{in}}}{w(r+1)} \cdot \ln\left(1 - \frac{F}{2^{n_{in}} - F}\right)\right\rceil$, $w$ is the number of table look-ups in each round and $\mathsf{r}$ is the total number of rounds.*

*Proof.* For the proof, we assume that there are $q$ collisions in the faulty table. Therefore, $0 \leq q \leq F$. Then from Lemma 2, we observe that to compute $A$ plaintext-ciphertext pairs, we require,

$$(2^{n_{in}} - 2q)\left(1 - \left(1 - \frac{1}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)A}\right) + q\left(1 - \left(1 - \frac{2}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)A}\right)$$

many distinct values. Since $F$ distinct fault values are given, then we have,

$$\begin{aligned}
&(2^{n_{in}} - 2q)\left(1 - \left(1 - \frac{1}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)A}\right) + q\left(1 - \left(1 - \frac{2}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)A}\right) = F \\
&\implies F \geq (2^{n_{in}} - 2q)\left(1 - e^{-\frac{w(\mathsf{r}+1)A}{2^{n_{in}}}}\right) + q\left(1 - e^{-\frac{2 \cdot w(\mathsf{r}+1)A}{2^{n_{in}}}}\right) \\
&\implies F \geq (2^{n_{in}} - q)\left(1 - e^{-\frac{w(\mathsf{r}+1)A}{2^{n_{in}}}}\right) \\
&\implies A \leq -\frac{2^{n_{in}}}{w(\mathsf{r}+1)} \ln\left(1 - \frac{F}{2^{n_{in}} - q}\right).
\end{aligned}$$

As $q \leq F$, we have $A \leq -\frac{2^{n_{in}}}{w(\mathsf{r}+1)} \ln\left(1 - \frac{F}{2^{n_{in}} - F}\right)$. Again, the probability of observing a value that occurs twice in the faulty table is $\frac{q}{F}$. Thus, for observing total $w(\mathsf{r}+1)$ values to compute one plaintext-ciphertext pair, the expected number of values that occur twice is $\frac{w(\mathsf{r}+1)q}{F}$. Therefore, there are $2^{\frac{w(\mathsf{r}+1)q}{F}}$ plaintexts corresponding to one ciphertext. This shows that the random ciphertext is included in the precomputed set of plaintext-ciphertext pairs with the probability bounded by $\frac{A}{2^n}$ and those ciphertexts will be decrypted with the probability upper bounded by $2^{-\frac{w(\mathsf{r}+1)q}{F}}$ which is upper bounded by 1, while $q = 0$.

The random ciphertext is not in the precomputed set if at least one value among $w(\mathsf{r}+1)$ values is not in the fault values. Preimage of those values has to be guessed

with probability $\frac{1}{2^{n_{in}}-F}$ to obtain the plaintext. Then the probability that the random ciphertext is not included in the precomputed set of plaintext-ciphertext pairs is $1-\frac{A}{2^n}$. Also, the probability of decrypting that ciphertext is

$$\sum_{i=0}^{w(\mathsf{r}+1)} \binom{w(\mathsf{r}+1)}{i} \left(\frac{F}{2^{n_{in}}}\right)^{w(\mathsf{r}+1)-i} \cdot \left(\frac{1}{2^{n_{in}}-F}\right)^{i} = \left(\frac{F}{2^{n_{in}}}+\frac{1}{2^{n_{in}}-F}\right)^{w(\mathsf{r}+1)}.$$

Hence, the result follows. □

A comparative summary of NdUF bounds for the SPN ciphers under the CP-CV fault model is presented in Table 3.

**Table 3:** NdUF bounds for SPN ciphers under the KP-CV and CP-CV fault model. The results are provided for both WEM and SPNbox constructions.

| | WEM-16 | WEM-32 | SPNbox-16 | SPNbox-32 |
|---|---|---|---|---|
| KP-CV | $(\frac{\sigma(\mathsf{T})}{2^{1.33}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{2.67}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{1.6}}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^{3.2}}, 128)$ |
| CP-CV | $(\frac{\sigma(\mathsf{T})}{2^{1.33}}, 118.47)$ | $(\frac{\sigma(\mathsf{T})}{2^{2.67}}, 103.86)$ | $(\frac{\sigma(\mathsf{T})}{2^{1.6}}, 118.88)$ | $(\frac{\sigma(\mathsf{T})}{2^{3.2}}, 104.26)$ |

# 6 NdUF Analysis of SPACE

In this section, we provide the NdUF analysis of SPACE under the KP-KV, CP-KV, and attack under KP-CV, CP-CV adversarial models.

*Remark* 6. We also provide detailed NdUF analyses of Galaxy in Appendix B. Note that, both SPACE and Galaxy follow Feistel structure, but they differ in the number of table look ups per round. Hence, we provide a separate analysis for Galaxy.

## 6.1 NdUF Analysis of SPACE Under KP-KV

We now analyze the bound for $(F, D)$-NdUF security of SPACE in KP-KV attack setting. To evaluate NdUF security bounds under the KP-KV attack, we propose Theorem 5. Note that, we use the table $\mathsf{T}'$ to denote the fault-injected table $\mathsf{T}$ corresponding to the function $R_{\mathsf{K}}$.

**Theorem 5.** *Given $F$ distinct fault values and their corresponding positions in each table under KP-KV attack setting for the table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $(n-n_{in})$-bits) of a white-box implementation of SPACE, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability of more than $\left(\frac{F}{2^{n_{in}}}\right)^{\mathsf{r}}$, where $\mathsf{r}$ is the total number of rounds.*

*Proof.* Given $F$ distinct fault values and their corresponding positions, the probability that a table look-up for $\mathsf{T}'$ is in the faulty entries is $\frac{F}{2^{n_{in}}}$. Since $\mathsf{T}'$ is used in each round (total number of rounds is $\mathsf{r}$), the adversary can decrypt a faulty ciphertext for a random plaintext with probability $\left(\frac{F}{2^{n_{in}}}\right)^{\mathsf{r}}$. □

A comparative summary of NdUF bounds is presented in Table 4.

## 6.2 NdUF Analysis of SPACE Under CP-KV

Since the adversary has no view of the challenge ciphertext in advance, the positions used in the table look-ups to decrypt the challenge ciphertext are unpredictable in advance. Therefore, this model has no advantage over the KP-KV model. Hence, bounds for specific instances are the same as Table 4.

## 6.3 NdUF Analysis of SPACE Under ACP-KV

The NdUF bound under the ACP-KV model is described in Theorem 6, which is similar to the Space Hard bound under the ACS setting.

**Theorem 6.** *For $F$ distinct fault positions under the ACP-KV model for table $\mathsf{T}$ to $\mathsf{T}'$ (containing $2^{n_{in}}$ entries each of $(n-n_{in})$-bits) of the white-box implementation of SPACE, it is infeasible for any adversary to decrypt the challenge ciphertext corresponding to any randomly chosen plaintext with probability higher than $N \cdot 2^{-n} + (1 - N \cdot 2^{-n}) \cdot (F/2^{n_{in}})^r$, where $\mathsf{r}$ is the total number of rounds, $N = \left\lceil \log_{e_{in}} \left(1 - \frac{F}{2^{n_{in}}}\right)/r \right\rceil$ and $e_{in} = \frac{2^{n_{in}}-1}{2^{n_{in}}}$.*

*Proof.* From Lemma 1, $2^{n_{in}} \cdot \left(1 - (\frac{2^{n_{in}}-1}{2^{n_{in}}})^{N \cdot \mathsf{r}}\right)$ positions are needed to encrypt $N$ plaintexts, as $N\mathsf{r}$ table look-ups are required to encrypt $N$ plaintexts. This shows $F$ fault positions enable to compute $N = \left\lceil \log_{e_{in}} \left(1 - \frac{F}{2^{n_{in}}}\right)/\mathsf{r} \right\rceil$ plaintext-ciphertext pairs in advance. The challenge ciphertext is then included in the precomputed set of plaintext-ciphertext pairs with probability $N \cdot 2^{-n}$. Otherwise, for given $F$ distinct fault positions, the challenge ciphertext can be decrypted with probability upper bounded by $(F/2^{n_{in}})^{\mathsf{r}}$, as Theorem 5. □

A comparative summary of NdUF bounds is presented in Table 4.

**Table 4:** NdUF bounds for SPACE ciphers under KP-KV and ACP-KV fault model.

| | SPACE-16 | SPACE-32 |
|---|---|---|
| KP-KV | $(\frac{\sigma(\mathsf{T})}{2}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2}, 128)$ |
| ACP-KV | $(\frac{\sigma(\mathsf{T})}{2}, 119.53)$ | $(\frac{\sigma(\mathsf{T})}{2}, 103.53)$ |

## 6.4 NdUF Analysis of SPACE Under KP-ACV and KP-CV

In this section, we explain the NdUF analysis of SPACE under the KP-ACV and KP-CV models. We start with our observation in the KP-ACV setting. Note that, this observation will also work in the KP-CV setting, because in KP-CV, the adversary has more power as she gets all the fault positions together (need not be adaptive), whereas in KP-ACV, the adversary receives the fault positions one by one adaptively. In more details, in KP-ACV, each position $p_i$ is revealed only after the adversary has already committed to $v_{i-1} = T(p_{i-1})$. In contrast, in KP-CV, all positions $p_0, p_1, p_2, ...$ are revealed before any fault values are selected. For Feistel constructions, these positions correspond to inputs to the underlying PRF table. Therefore, in KP-CV, knowledge of all future positions enables the adversary to carefully choose current fault values in a way that influences subsequent table inputs. This additional foresight is unavailable in KP-ACV. We later provide a dedicated attack in the KP-CV setting.

### 6.4.1 NdUF Analysis of SPACE Under KP-ACV.

In KP-ACV, the adversary obtains a fault position and then chooses corresponding fault values adaptively.

**Observation from the Adversarial Model.** Let $\mathcal{P} = \{p_0, p_1, \ldots, p_{F-1}\}$ denote the $F$ fault positions obtained by the adversary in the KP-ACV model with adaptively chosen fault values. Also, let $\mathsf{T}'$ is faulty table corresponding to $\mathsf{T}$. The position $p_i$ is received by the adversary after setting the fault values at $\mathsf{T}'(p_j)$ for all $j < i$. Again, assume that for a message $x = (x_0, x_1, \ldots, x_{w-1})$, $I_i$ denotes the input to the table $\mathsf{T}'$ in the $i^{th}$ round, for $1 \leq i \leq \mathsf{r}$ (where $\mathsf{r}$ is the number of rounds). Here $I_1 = x_0$. Note that, the adversary can traverse upto round $i$ deterministically, if all the $I_1$, $I_2$ to $I_i$ are in $\mathcal{P}$. Hence, the

adversary's goal is to control the $I_i$s (control means to ensure they are in $\mathcal{P}$) through the fault values and the chosen message. We observe that,

**Observation 1.** For a chosen message $x = (x_0, x_1, \ldots, x_{w-1})$, the values $I_i$s can be controlled maximum upto $(2w-1)$ (i.e., upto $I_{2w-1}$) rounds by adaptively chosen fault values, with only two known fault positions for SPACE.

*Proof.* For the sake of simplicity, we prove this observation using the example SPACE-32 (here, $\mathsf{r} = 128$ and $w = 4$), as it has only four branches and the construction is simpler; note that, exactly the same reasoning applies to SPACE-16 as well. We need to show for SPACE-32, that the adversary can control upto $I_7$. For $I \in \mathcal{P}$, if $I = p_i$, we denote $\mathsf{Order}(I) = i$. Let, for a chosen plaintext $x = (x_0 \| x_1 \| x_2 \| x_3)$, the inputs to the table corresponding to the first 8 rounds be given by:

$$I_1 = x_0, \tag{9}$$
$$I_2 = \mathsf{T}'(I_1)_1 \oplus x_1, \tag{10}$$
$$I_3 = \mathsf{T}'(I_2)_1 \oplus \mathsf{T}'(I_1)_2 \oplus x_2, \tag{11}$$
$$I_4 = \mathsf{T}'(I_3)_1 \oplus \mathsf{T}'(I_2)_2 \oplus \mathsf{T}'(I_1)_3 \oplus x_3 \oplus 1, \tag{12}$$
$$I_5 = \mathsf{T}'(I_4)_1 \oplus \mathsf{T}'(I_3)_2 \oplus \mathsf{T}'(I_2)_3 \oplus I_1 \oplus 2, \tag{13}$$
$$I_6 = \mathsf{T}'(I_5)_1 \oplus \mathsf{T}'(I_4)_2 \oplus \mathsf{T}'(I_3)_3 \oplus I_2 \oplus 3, \tag{14}$$
$$I_7 = \mathsf{T}'(I_6)_1 \oplus \mathsf{T}'(I_5)_2 \oplus \mathsf{T}'(I_4)_3 \oplus I_3 \oplus 4, \tag{15}$$
$$I_8 = \mathsf{T}'(I_7)_1 \oplus \mathsf{T}'(I_6)_2 \oplus \mathsf{T}'(I_5)_3 \oplus I_4 \oplus 5. \tag{16}$$

Here, $\mathsf{T}'(I_i)_j$ represents the $j^{\text{th}}$ $n_{in}$-bit word of $\mathsf{T}'(I_i)$, where $j \in \{1, 2, 3\}$. The adversary can simulate the faulty encryption corresponding to the chosen $x$ deterministically when all $I_i \in \mathcal{P}$. We show that, by controlling $x$ and the fault values of $\mathsf{T}'$, the adversary can traverse a maximum of 7 rounds (i.e., ensure $I_1, \ldots, I_7 \in \mathcal{P}$), but has no control over $I_8$. Note that the adversary can traverse to round $i$ if she can ensure $I_1, \ldots, I_i \in \mathcal{P}$.

By appropriately choosing $x_0, x_1, x_2$, and $x_3$, the adversary can ensure $I_i \in \mathcal{P}$ for all $i \in \{1, 2, 3, 4\}$. We now analyze the possible cases.

1. **$\mathsf{Order}(I_1) \geq \mathsf{Order}(I_2) \geq \mathsf{Order}(I_3) \geq \mathsf{Order}(I_4)$.**

   In this case, by choosing $x_0, x_1, x_2$, and $x_3$, we can set up to $I_4$. However, in the equation for $I_5$, if $\mathsf{Order}(I_1)$ is greater than both $\mathsf{Order}(I_2), \mathsf{Order}(I_3)$, and $\mathsf{Order}(I_4)$, then we cannot ensure $I_5 = \mathsf{T}'(I_4)_1 \oplus \mathsf{T}'(I_3)_2 \oplus \mathsf{T}'(I_2)_3 \oplus I_1 \oplus 2$, since $I_1$ is received after $\mathsf{T}'(I_4)_1$, $\mathsf{T}'(I_3)_2$, and $\mathsf{T}'(I_2)_3$ were set.

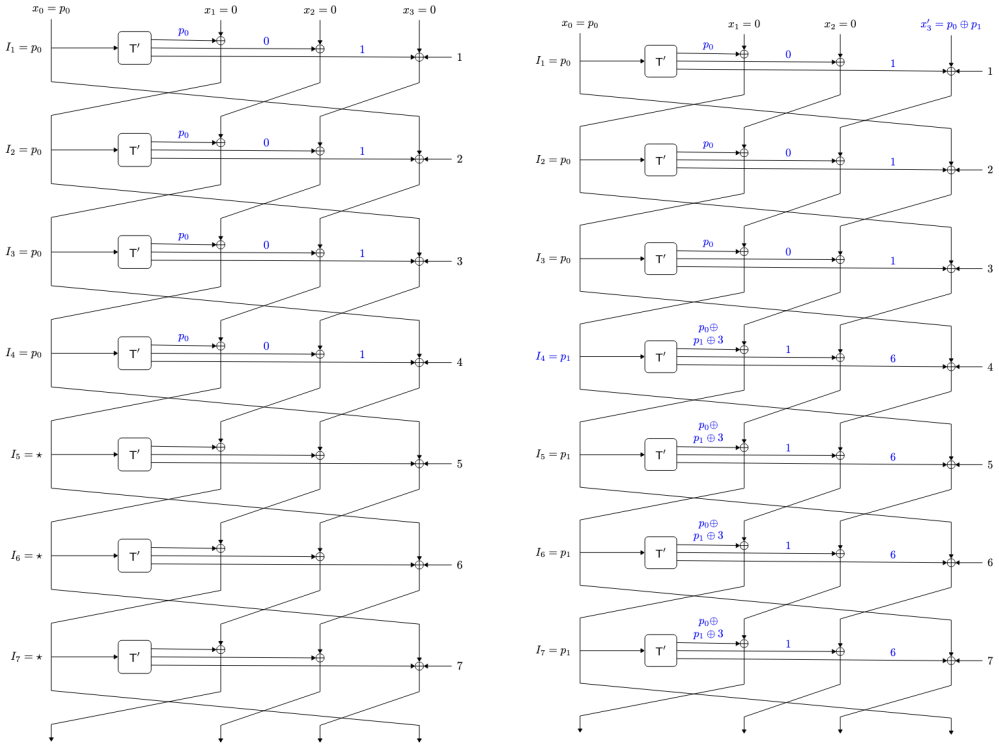
   If $\mathsf{Order}(I_1)$ equals one of the others, say $\mathsf{Order}(I_2)$, then $\mathsf{T}'(I_2)_3$ is already fixed while $I_1 = I_2$ was received, but $\mathsf{T}'(I_4)_1$ and $\mathsf{T}'(I_3)_2$ cannot be set as $\mathsf{Order}(I_1)$ is greater than both $\mathsf{Order}(I_4)$ and $\mathsf{Order}(I_3)$. The reasoning is similar for other subcases. Hence, using one fault (**in the case $\mathsf{Order}(I_1) = \mathsf{Order}(I_2) = \mathsf{Order}(I_3) = \mathsf{Order}(I_4)$**), the adversary can traverse at most four rounds (i.e., up to $I_4$).

2. **$\mathsf{Order}(I_1)$ is the maximum of $\{\mathsf{Order}(I_1), \mathsf{Order}(I_2), \mathsf{Order}(I_3), \mathsf{Order}(I_4)\}$.**

   The same reasoning applies. Whether $\mathsf{Order}(I_1)$ is equal to or greater than the others, the entry $\mathsf{T}'(I_2)_3$ cannot be used to set $I_5$. Hence, the adversary cannot go beyond the fourth round. (**note that, Case 1 is included in Case 2, but we separate it to simply show that one fault suffices.**)

3. **$\mathsf{Order}(I_2)$ is the maximum of $\{\mathsf{Order}(I_1), \mathsf{Order}(I_2), \mathsf{Order}(I_3), \mathsf{Order}(I_4)\}$.**

   Consider Equation (13). In this case, by appropriately setting $\mathsf{T}'(I_2)_3$, the adversary can ensure $I_5 \in \mathcal{P}$ with $\mathsf{Order}(I_5) \leq \mathsf{Order}(I_2)$. However, in Equation (14) for $I_6$, if $\mathsf{Order}(I_5) = \mathsf{Order}(I_2)$, then $\mathsf{T}'(I_5)_1 = \mathsf{T}'(I_2)_1$ was already fixed (see Equation (11)).




**(a)** First four rounds with first fault position $p_0$.

**(b)** Total 7-rounds with fault positions $p_0$ and $p_1$. $p_1$ is received by the adversary after $p_0$.

**Figure 5:** Example for KP-ACV (SPACE-32).

If $\mathsf{Order}(I_5) < \mathsf{Order}(I_2)$, then $\mathsf{T}'(I_5)_1$ cannot be used either, as $I_2$ is received after $I_5$.For a similar reason, $\mathsf{T}'(I_3)_3$ and $\mathsf{T}'(I_4)_2$ cannot be used. Hence, traversal stops at round five.

4. **$\mathsf{Order}(I_3)$ is the maximum of $\{\mathsf{Order}(I_1), \mathsf{Order}(I_2), \mathsf{Order}(I_3), \mathsf{Order}(I_4)\}$.**

   Following the same reasoning as in Case 3, the adversary can traverse up to the sixth round.

5. **$\mathsf{Order}(I_4)$ is the maximum of $\{\mathsf{Order}(I_1), \mathsf{Order}(I_2), \mathsf{Order}(I_3), \mathsf{Order}(I_4)\}$.**

   Following the same reasoning as in Case 3, the adversary can traverse up to the seventh round.

Thus, the adversary can traverse at most seven rounds (i.e., ensure $I_7 \in \mathcal{P}$), but has no control over $I_8$. Following exactly the same way, this result can be shown for SPACE-16 as well. A pictorial example corresponding to this observation is given below in Figure 5. □

Suppose the adversary receives the first fault position $p_0$. She selects a plaintext of the form $x_0 \| x_1 \| x_2 \| x_3$, where $x_0 = p_0$ and the remaining components are arbitrary. For simplicity, let $x_1 = x_2 = x_3 = 0^{32}$. The adversary then chooses a fault value at position $p_0$ such that the subsequent inputs satisfy $I_2 = I_3 = I_4 = p_0$. It can be verified that assigning the fault value $p_0 \| 0 \| 1$ at position $p_0$ achieves this objective (see Figure 5a). Note that beyond this point, the input $I_5$ may deviate from $p_0$, and hence the adversary may lose control over the subsequent inputs. Next, the adversary receives the second

fault position $p_1$. To regain control, she modifies the plaintext by updating $x_3$ from 0 to $x'_3 = p_0 \oplus p_1$, thereby forcing $I_4 = p_1$. The resulting plaintext becomes $p_0\|0\|0\|(p_0 \oplus p_1)$. Subsequently, the adversary selects an appropriate fault value at position $p_1$ such that $I_5 = I_6 = I_7 = p_1$. In particular, choosing the fault value $(p_0 \oplus p_1 \oplus 3)\|1\|6$ ensures this and hence she deterministically computes $7^{th}$ rounds (see Figure 5b).

The above Observation depicts that the adversary can maintain deterministic control over the internal variables for at most $2w-1$ rounds. Beyond this point, the structure of the state evolution prevents further deterministic control. Any additional influence after $2w-1$ rounds is therefore no longer fully controlled by the adversary and can probabilistically occur. In other words, the attack transitions from being structurally driven to being purely probabilistic.

Deriving a tight and explicit bound for this remaining probabilistic influence requires carefully tracking dependencies across multiple rounds. We leave the formulation of a concrete bound as a future extension. Importantly, this does not weaken our main results. Our observation identifies the precise limit of deterministic control in the adaptive setting.

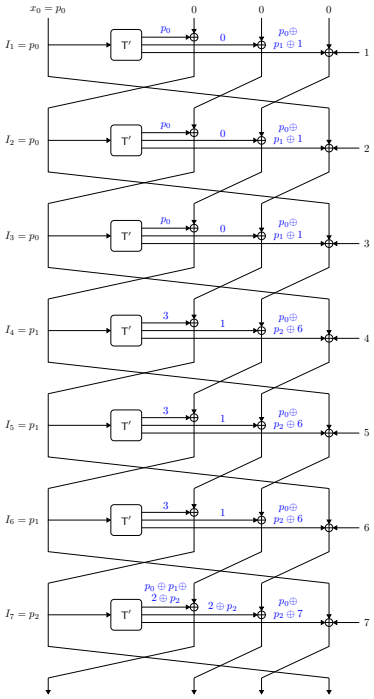
### 6.4.2 A Dedicated Attack on SPACE in the KP-CV Setting.

In this section, we describe a dedicated attack on SPACE under KP-CV. This attack won't work in the KP-ACV setting. In KP-CV, the adversary is first given a sequence of fault positions (need not be adaptive), and then it chooses the fault values. Below, we provide an informal description of the attack.

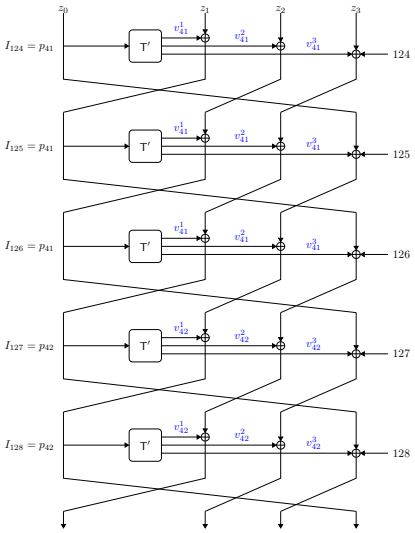
1. Select $t_1$ distinct positions $p_0, \ldots, p_{t_1-1}$ from $\mathcal{P}$.
2. Consider the plaintext $p_0\|0^{n_{in}}\|0^{n_{in}}\|\cdots\|0^{n_{in}}$. By construction, the first input to $\mathsf{T}'$ is $I_1 = p_0$.
3. Inject a fault at position $p_0$ such that $I_2 = I_3 = \cdots = I_{w-1} = p_0$ and $I_w = p_1$.
4. Next, inject a fault at the position $p_1$ so that $I_{w+1} = I_{w+2} = \cdots = I_{2(w-1)} = p_1$ and $I_{2(w-1)+1} = p_2$.
5. Continue this procedure in the same manner. In general, when the input becomes $p_j$ and no fault has been injected yet at $p_j$, then we inject an appropriate fault so that the next $w-2$ inputs remain $p_j$, and the following input transitions to $p_{j+1}$ and if a fault is already injected at $p_j$ then skip.
6. Repeating this process ensures $I_{(t_1-1)(w-1)} = p_{t_1-2}$ and $I_{(t_1-1)(w-1)+1} = p_{t_1-1}$.
7. Finally, inject a suitable fault at the position $p_{t_1-1}$ so that the remaining inputs satisfy $I_{(t_1-1)(w-1)+2} = I_{(t_1-1)(w-1)+3} = \cdots = I_{\mathsf{r}} = p_{t_1-1}$.
8. If additional positions remain in the set $\mathcal{P}$, select the next $t_1$ unused positions and repeat the same procedure starting from Step 2 to compute another PT-CT pair.

**Key Observation.** Let $I_i$ denote the input to the table $\mathsf{T}'$ at round $i$, where $1 \leq i \leq \mathsf{r}$. Observe that if $I_i \in \mathcal{P}$ for some $i \in [(t_1-1)(w-1)+1]$ and no fault has been injected yet at the position $I_i$, then by injecting a suitable fault at that position, we can control the next $\mathsf{r} - ((t_1-1)(w-1)+1)$ consecutive inputs to $\mathsf{T}'$, where $t_1 = \lfloor \frac{\mathsf{r}}{w-1} \rfloor$, for $\mathsf{r}\%(w-1) \leq 1$ and $t_1 = \lceil \frac{\mathsf{r}}{w-1} \rceil$, for $\mathsf{r}\%(w-1) > 1$. In particular, it is possible to force the inputs $I_{i+1}, I_{i+2}, \cdots, I_{i+(w-1)}$ to take predetermined values. For example, suppose that $I_1 = p_0$ for some $p_0 \in \mathcal{P}$ and no fault has been injected at $p_0$ yet. If we want to enforce $I_2 = I_3 = \cdots = I_w = p_0$, then it suffices to inject the fault value $(p_0 \oplus x_1)\|(x_1 \oplus x_2)\|\cdots\|(x_{w-3} \oplus x_{w-2})\|(x_{w-2} \oplus x_{w-1} \oplus 1)$ at the position $p_0$. This observation allows the adversary to control multiple future inputs to the table $\mathsf{T}'$ using a single fault injection. Using this property, the adversary aims to compute as many plaintext–ciphertext pairs as possible.

**Attack Procedure.** We consider a plaintext $p\|0^{n_{in}}\|0^{n_{in}}\|\cdots\|0^{n_{in}}$, where $p \in \mathcal{P}$. We show how the adversary simulates ciphertext computation for such a plaintext. The attack is presented in Algorithm 1. Here, an adversary can precompute up to $\left\lfloor \frac{F}{t_1} \right\rfloor$ plaintext–ciphertext pairs in advance. We provide a diagram demonstrating the ciphertext computation for SPACE-32, in Figure 6




(a) First 7-rounds.




(b) Last 5-rounds. Here, $v^1_{41} = z_1 \oplus p_{41}$, $v^2_{41} = z_1 \oplus z_2$, $v^3_{41} = z_2 \oplus z_3 \oplus 124 \oplus p_{41} \oplus p_{42}$, $v^1_{42} = z_1 \oplus z_3 \oplus 124 \oplus 125$, and $v^2_{42}$, $v^3_{42}$ are any 32-bit value.

**Figure 6:** Attack under KP-CV for SPACE-32.

**Algorithm 1** Attack for SPACE-$n_{in}$ with $n_{in} = 16, 32$ in the KP-CV Setting. Here, $w = \frac{n}{n_{in}}$ and $t_1 = \lfloor \frac{\mathsf{r}}{w-1} \rfloor$, for $\mathsf{r}\%(w-1) \leq 1$ and $t_1 = \lceil \frac{\mathsf{r}}{w-1} \rceil$, otherwise.

Suppose $P = \{p_0, p_1, \cdots, p_{F-1}\}$ be the set of positions where the adversary can inject faults
$\mathcal{F} \leftarrow \emptyset$, $j = 0$
**while** $j < F$ **do**
  $X^0 \leftarrow p_j\|0^{n_{in}}\|0^{n_{in}}\|\cdots\|0^{n_{in}}$
  **for** $i = 0$ to $(w-1)(t_1-1)$ **do**
    **if** $i\%(w-1) = 0$ **then**
      **if** $i < (w-1)(t_1-1)$ **then**
        $v_j \leftarrow temp\|(X^i_0 \oplus X^i_{w-2} \oplus X^i_{w-1} \oplus (i+1) \oplus p_{j+1})$
        ▷ where, $temp \leftarrow (X^i_0 \oplus X^i_1)\|(X^i_1 \oplus X^i_2)\|\cdots\|(X^i_{w-3} \oplus X^i_{w-2})$
      **else**
        $v_j \leftarrow temp\|(X^i_{w-2} \oplus X^i_{w-1} \oplus (i+1))$
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{(p_j, v_j)\}$
      $j \leftarrow j+1$
      **if** $j \geq F$ **then**
        **break**
    $X^i \leftarrow X^i \oplus (0^{n_{in}}\|v_j) \oplus (0^{n-n_{in}}\|(i+1))$
    $X^{i+1} \leftarrow (X^i << n_{in}) \oplus (X^i >> (n-n_{in}))$
**return** $\mathcal{F}$

**Attack Probability.** The attack procedure prepares a set of $\lfloor \frac{F}{t_1} \rfloor$ many PT-CT pairs among $2^{128}$ many pairs. Thus, the challenge ciphertext is in the prepared set with probability

$\lfloor \frac{F}{t_1} \rfloor \cdot 2^{-n}$. In particular, the adversary obtains at least one ciphertext with high probability from the prepared set, after getting $(\lfloor \frac{F}{t_1} \rfloor \cdot 2^{-n})^{-1}$ ciphertexts. A summary of attack complexities is presented in Table 5.

**Table 5:** Attack probabilities for SPACE under KP-CV for different values of $F$.

| Cipher | $F = \frac{\sigma(T)}{2}$ | $F = \frac{\sigma(T)}{4}$ | $F = \frac{\sigma(T)}{8}$ | $F = \frac{\sigma(T)}{16}$ |
|---|---|---|---|---|
| SPACE-16 | $2^{-118.43}$ | $2^{-119.43}$ | $2^{-120.43}$ | $2^{-121.43}$ |
| SPACE-32 | $2^{-102.43}$ | $2^{-103.43}$ | $2^{-104.43}$ | $2^{-105.43}$ |

## 6.5 A Practical NdUF Attack of SPACE Under CP-CV

In this section, we present a highly practical attack technique against SPACE assuming fault injections into $F$ table entries. In the CP-CV model, the adversary chooses a sequence of fault positions as well as the corresponding fault values. So our target is to select a set of fault positions $\mathcal{P}$ in such a way that, by injecting appropriate fault values at these positions, we can deterministically compute the ciphertext for any plaintext of the form $x_0 \| x_1 \| \cdots \| x_{w-1}$, where $x_i \in \mathcal{P}$ for all $i \in \{0, 1, \cdots, w-1\}$. In particular, all inputs $I_i$ to $\mathsf{T}'$ remain within the set $\mathcal{P}$, ensuring that the overall computation proceeds deterministically. Consequently, the adversary aims to compute $F^w$ plaintext-ciphertext (PT-CT) pairs.

**Key Observation.** To achieve this goal, we need to select a suitable set of fault positions along with the corresponding fault values. Suppose that we inject the fault value $0^{n-n_{in}}$ at each of the selected positions. Then, the problem reduces to constructing the fault position set $\mathcal{P}$.

Before providing the exact description of the set $\mathcal{P}$, consider the following observation. Let us assume that we want to compute the ciphertexts corresponding to the plaintexts $x_{j_0} \| x_{j_1} \| \cdots \| x_{j_{w-1}}$, where $j_i \in \{0, 1, \cdots, w-1\}$, $\forall 0 \leq i \leq w-1$. Then observe that we can deterministically compute the ciphertexts corresponding to these plaintexts if $x_i \oplus z \in \mathcal{P},\ \forall 0 \leq i \leq w-1, 0 \leq z \leq r$. That is, for each component $x_{j_i}$, all the positions obtained by XORing $x_i$ with every $z \in \{0, 1, \ldots, r\}$ must belong to $\mathcal{P}$. In particular, this implies that if we select a fault position $p \in \mathcal{P}$, then we must also include all positions of the form $p \oplus z,\ \forall 0 \leq z \leq r$, in the set $\mathcal{P}$. Observe that this operation affects only the $t_2 = (\lfloor \log r \rfloor + 1)$ least significant bits of $p$, while the remaining higher-order bits remain unchanged. Hence, we can construct the set $\mathcal{P}$ as follows. Select an arbitrary string $s$ from the set $\{0,1\}^{n_{in}-t_2}$, and include all positions of the form $s \| i,\ 0 \leq i \leq 2^{t_2} - 1$, in $\mathcal{P}$. Thus, each choice of $s$ contributes a block of size $2^{t_2}$ positions. Consequently, to construct the set $\mathcal{P}$ with $F$ elements, it suffices to select $\frac{F}{2^{t_2}}$ distinct elements from the set $\{0,1\}^{n_{in}-t_2}$ and append all possible $t_2$-bit strings to each of these elements. The exact attack procedure is described below.

**Attack Procedure.** For this attack, to deterministically compute the ciphertext corresponding to any plaintext of the form $x_0 \| x_1 \| \cdots \| x_{w-1}$, where $x_i \in \mathcal{P}$ for all $i \in \{0, 1, \cdots, w-1\}$, the fault positions and values are selected as follows:

1. Choose $\frac{F}{2^{t_2}}$ distinct elements $s_0, s_1, \ldots, s_{(\frac{F}{2^{t_2}}-1)}$ from $\{0,1\}^{n_{in}-t_2}$, where $t_2 = (\lfloor \log r \rfloor + 1)$ is the number of bits used for round constant.
2. For each $s_i$, include all the strings $s_i \| s$ in the set $\mathcal{P}$, where $s \in \{0,1\}^{t_2}$. It means, for every fixed $s_i$, we include all $2^{t_2}$ extensions obtained by appending a $t_2$-bit suffix.
3. Inject the fault value $0^{n-n_{in}}$ at each position belonging to the set $\mathcal{P}$.

**Algorithm 2** Attack for SPACE in the CP-CV Setting. Here $t_2 = (\lfloor \log \mathsf{r} \rfloor + 1)$.

1: $\mathcal{F} \leftarrow \emptyset$
2: Choose distinct elements $s_0, s_1, \cdots, s_{\frac{F}{2^{t_2}}-1}$ from $\{0,1\}^{n_{in}-t_2}$
3: **for** $i = 0$ to $\frac{F}{2^{t_2}} - 1$ **do**
4: **for** $s = 0$ to $2^{t_2} - 1$ **do**
5: $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s_i \| s, 0^{n-n_{in}})\}$
6: **return** $\mathcal{F}$

**Table 6:** Attack probabilities for SPACE under CP-CV for different values of $F$.

| Cipher | $F = \frac{\sigma(T)}{2}$ | $F = \frac{\sigma(T)}{4}$ | $F = \frac{\sigma(T)}{8}$ | $F = \frac{\sigma(T)}{16}$ |
|---|---|---|---|---|
| SPACE-16 | $2^{-8}$ | $2^{-16}$ | $2^{-24}$ | $2^{-32}$ |
| SPACE-32 | $2^{-4}$ | $2^{-8}$ | $2^{-12}$ | $2^{-16}$ |

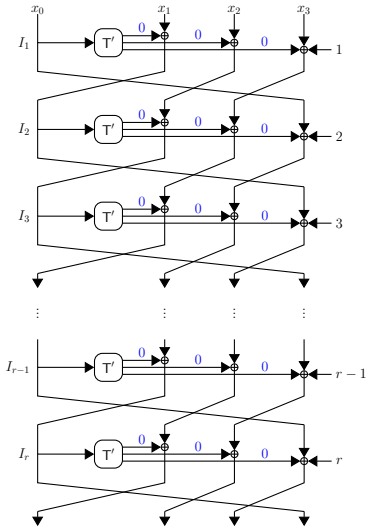



**Figure 7:** Attack under CP-CV for SPACE-32. Here, $x_0$, $x_1$, $x_2$, and $x_3$ belong to $\mathcal{P}$.

By injecting the fault $0^{n-n_{in}}$ at every position in $\mathcal{P}$, for any input $x_0 \| \ldots \| x_{w-1}$ with $x_i \in \mathcal{P}$, all intermediate $I_j$s are ensured in $\mathcal{P}$ for every $j \in [\mathsf{r}]$. Consequently, the adversary can deterministically compute the corresponding ciphertext. Hence, the adversary can compute $F^w$ many PT-CT pairs in advance. The procedure is described in Algorithm 2. The attack is illustrated in Figure 7.

**Attack probability.** Here, $F^w$ many PT-CT pairs can be prepared in advance (total PT-CT pairs is $2^n$). Thus, the challenge ciphertext is in the prepared set with probability $F^w \cdot 2^{-n}$. In particular, with high probability, the adversary obtains at least one ciphertext from the prepared set after obtaining $(F^w \cdot 2^{-n})^{-1}$ ciphertexts. A comparative summary is presented in Table 6.

# 7 Software-Based Fault Detection Technique

A popular hardware-based fault detection technique is hardware redundancy (i.e., dual modular redundancy), where two identical modules perform the same operation with the same inputs. As it is difficult to inject the same fault in both simultaneously, faults are detected from mismatched outputs. In the white-box setting, this technique may be ineffective as the adversary has full control over fault injection, making it easier to inject the same fault into multiple copies.

We present a procedure that can detect the presence of faults with a high probability. We assume the server hosts the correct implementation which is inaccessible to the adversary, and the user initiates detection by sending a Start signal to the server; the remaining process is as follows.

The server proceeds as follows:

1. It selects $\ell$ positions $i_0, \ldots, i_{\ell-1}$ independently and uniformly at random, and let $\mathsf{T}(i_j) = x_{i_j}$ for each $j \in \{0, 1, \ldots, \ell-1\}$. Here, $n_{\text{in}}$ and $n_{\text{out}}$ denote the input and output sizes of the table $\mathsf{T}$, respectively. We employ a white-box secure block cipher $\mathsf{E}$ with block size $n$; for simplicity, we may assume $n = 128$.
2. $\forall j \in \{0, \ldots, \ell-1\}$, it computes $c_j \leftarrow \mathsf{E}_{\mathsf{K}}^{-1}\left(i_j \| x_{i_j} \| 0^{\,n-n_{\text{in}}-n_{\text{out}}}\right)$.
3. It then forms $c \leftarrow c_0 \| \cdots \| c_{\ell-1}$ and sends $c$ to the user.

Upon receiving $c$, the user performs the following steps:

1. $\forall j \in \{0, \ldots, \ell-1\}$, compute $\left(i_j \| x_{i_j} \| 0^{\,n-n_{\text{in}}-n_{\text{out}}}\right) \leftarrow \mathsf{E}_{\mathsf{K}}(c_j)$.
2. Verify whether $x_{i_j} = x'_{i_j}$ holds for all $j \in \{0, 1, \ldots, \ell-1\}$, where $x'_{i_j}$ denotes the value stored in the user's table at position $i_j$.
3. If $\exists\ j$ such that $x_{i_j} \neq x'_{i_j}$, the algorithm outputs that a fault has been injected.

**Lemma 3.** *For given $F$ faulted entries of the user's table with $2^{n_{in}}$ entries, the above procedure can detect a fault with the probability lower bounded by $\frac{\ell \cdot F}{2^{(n_{in}+1)}}$.*

*Proof.* Let us assume that $\mathbf{A}$ is the event of fault detection and $\mathcal{P}$ is the set of fault positions. From the above algorithm, the fault will be detected if $x_{i_j} \neq x'_{i_j}$ for at least one $j \in \{0, 1, \cdots, \ell-1\}$. Then, the probability of detecting fault is,

$$\begin{aligned} Pr[\mathbf{A}] &= \sum_{j=0}^{\ell-1} Pr[i_j \in \mathcal{P} \wedge x_{i_j} \neq x'_{i_j}] \\ &= \sum_{j=0}^{\ell-1} Pr[x_{i_j} \neq x'_{i_j} | i_j \in \mathcal{P}] \cdot Pr[i_j \in \mathcal{P}] \\ &= \sum_{j=0}^{\ell-1} (1 - \frac{1}{2^{n_{in}}}) \cdot \frac{F}{2^{n_{in}}} \\ &\geq \sum_{j=0}^{\ell-1} \frac{1}{2} \cdot \frac{F}{2^{n_{in}}} = \frac{\ell \cdot F}{2^{(n_{in}+1)}} \end{aligned}$$

□

For each case listed in Table 7, the probability of fault detection satisfies $\Pr[\mathbf{A}] \geq \frac{1}{2}$. Thus, if the user's table contains faults in a sufficiently large number of positions, then it will be detected with high probability.

**Table 7:** Cases where fault detection succeeds with probability at least $\frac{1}{2}$.

| $F$ | $\frac{\sigma(\mathsf{T})}{2}$ | $\frac{\sigma(\mathsf{T})}{4}$ | $\frac{\sigma(\mathsf{T})}{8}$ | $\frac{\sigma(\mathsf{T})}{16}$ |
|---|---|---|---|---|
| $\ell$ | 2 | 4 | 8 | 16 |

**Overhead:**

We would like to mention that, in our fault detection technique, the overhead consists of $\ell$ decryptions on the server side and $\ell$ encryptions plus verification on the client side, where $\ell$ can be as small as 2 (as mentioned in Table 7). Furthermore, it could be reasonable to assume that the verification is performed periodically rather than after every execution, which may reduce the practical cost.

# 8 Conclusion

In this work, we propose a fault-injection framework in the white-box setting, where the adversary can inject precise faults into the implementation locally. This model is practically stronger than the leakage-based ones, since the malware does not need to communicate a large amount of data to its remote adversary. Instead, it can locally inject faults to weaken the cipher. In contrast, in leakage-based settings, the malware must extract and transmit sensitive information to a remote entity.

Although existing designs have been proven secure in the white-box leakage-based incompressibility setting, we show that they exhibit a different behaviour under fault injection. In particular, we demonstrate that dedicated white-box ciphers based on SPN structures, such as WEM and SPNbox, achieve strong security even in this fault-based setting. On the other hand, white-box ciphers based on Feistel structures, such as SPACE and Galaxy, become highly insecure when the adversary can choose fault locations and values. This model is practical as the remote adversary can preprogram its malware to choose fault positions and values of its choice.

# References

[1] Adobe Systems Incorporated. Adobe primetime technical primer for operators. Technical report, Adobe Systems Incorporated, 2014.

[2] Smart Card Alliance et al. Host card emulation (hce) 101. *A Smart Card Alliance Mobile and NFC Council White Paper*, 2014.

[3] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240, Waterloo, Ontario, Canada, August 9–10, 2004. Springer Berlin Heidelberg, Germany.

[4] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 63–84, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer Berlin Heidelberg, Germany.

[5] Alex Biryukov and Adi Shamir. Structural cryptanalysis of SASAS. *Journal of Cryptology*, 23(4):505–518, October 2010.

[6] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1058–1069, Denver, CO, USA, October 12–16, 2015. ACM Press.

[7] Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser. Towards practical white-box cryptography: Optimizing efficiency and space hardness. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 126–158, Hanoi, Vietnam, December 4–8, 2016. Springer Berlin Heidelberg, Germany.

[8] Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Søren S. Thomsen. Slender-set differential cryptanalysis. *Journal of Cryptology*, 26(1):11–38, January 2013.

[9] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. *Cryptology ePrint Archive*, 2006.

[10] Avik Chakraborti, Shibam Ghosh, Takanori Isobe, and Sajani Kundu. EWEMrl: A white-box secure cipher with longevity. *IACR Communications in Cryptology (CiC)*, 2(4):21, 2026.

[11] Jihoon Cho, Kyu Young Choi, Itai Dinur, Orr Dunkelman, Nathan Keller, Dukjae Moon, and Aviya Veidberg. WEM: A new family of white-box block ciphers based on the Even-Mansour construction. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 293–308, San Francisco, CA, USA, February 14–17, 2017. Springer, Cham, Switzerland.

[12] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In Joan Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[13] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer Berlin Heidelberg, Germany.

[14] Pierre-Alain Fouque, Pierre Karpman, Paul Kirchner, and Brice Minaud. Efficient and provable white-box primitives. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 159–188, Hanoi, Vietnam, December 4–8, 2016. Springer Berlin Heidelberg, Germany.

[15] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security 2008: 17th USENIX Security Symposium*, pages 45–60, San Jose, CA, USA, July 28 – August 1, 2008. USENIX Association.

[16] Hong Hu, Shweta Shinde, Sendroiu Adrian, Zheng Leong Chua, Prateek Saxena, and Zhenkai Liang. Data-oriented programming: On the expressiveness of non-control data attacks. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP)*, pages 969–986. IEEE Computer Society, 2016.

[17] Kyriakos K. Ispoglou, Bader AlBassam, Trent Jaeger, and Mathias Payer. Block oriented programming: Automating data-only attacks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, pages 1868–1882, New York, NY, USA, 2018. Association for Computing Machinery.

[18] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *ICISC 10: 13th International Conference on Information Security and Cryptology*, volume 6829 of *Lecture Notes in Computer Science*, pages 278–291, Seoul, Korea, December 1–3, 2011. Springer Berlin Heidelberg, Germany.

[19] Yuji Koike and Takanori Isobe. Yoroi: Updatable whitebox cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):587–617, 2021.

[20] Yuji Koike, Kosei Sakamoto, Takuya Hayashi, and Takanori Isobe. Galaxy: A family of stream-cipher-based space-hard ciphers. In Joseph K. Liu and Hui Cui, editors, *ACISP 20: 25th Australasian Conference on Information Security and Privacy*, volume 12248 of *Lecture Notes in Computer Science*, pages 142–159, Perth, WA, Australia, November 30 – December 2, 2020. Springer, Cham, Switzerland.

[21] William Kosasih, Yusi Feng, Chitchanok Chuengsatiansup, Yuval Yarom, and Ziyuan Zhu. Sok: Can we really detect cache side-channel attacks by monitoring performance counters? In *Proceedings of the 2024 ACM Asia Conference on Computer and Communications Security (ASIA CCS '24)*, pages 171–185, New York, NY, USA, 2024. Association for Computing Machinery.

[22] Thomas Lepoint, Mathieu Rivain, Yannick Mulder, Peter Roelse, and Bart Preneel. Two attacks on a white-box aes implementation. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography – SAC 2013*, volume 8282 of *Lecture Notes in Computer Science*, pages 265–285. Springer, Heidelberg, 2014.

[23] Hamilton E Link and William D Neumann. Clarifying obfuscation: Improving the security of white-box des. In *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, volume 1, pages 679–684. IEEE, 2005.

[24] Mastercard. Mobile SDK Integration . 2026. https://developer.mastercard.com/mastercard-gateway/documentation/integrations-types/mobile-integration/integrate-mobile-payments/mob-sdk-int/.

[25] Brice Minaud, Patrick Derbez, Pierre-Alain Fouque, and Pierre Karpman. Key-recovery attacks on ASASA. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 3–27, Auckland, New Zealand, November 30 – December 3, 2015. Springer Berlin Heidelberg, Germany.

[26] Mobey Forum. The host card emulation in payments: Options for financial institutions. White paper, Mobey Forum, 2014.

[27] Yannick Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the xiao–lai white-box aes implementation. In Lawrie R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer, Heidelberg, 2013.

[28] Tyge Tiessen, Lars R. Knudsen, Stefan Kölbl, and Martin M. Lauridsen. Security of the AES with a secret S-box. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 175–189, Istanbul, Turkey, March 8–11, 2015. Springer Berlin Heidelberg, Germany.

[29] Workgroup Mobey, H.C.E. Forum. The host card emulation in payments: Options for financial institutions. Technical report, Mobey Forum, 2014.

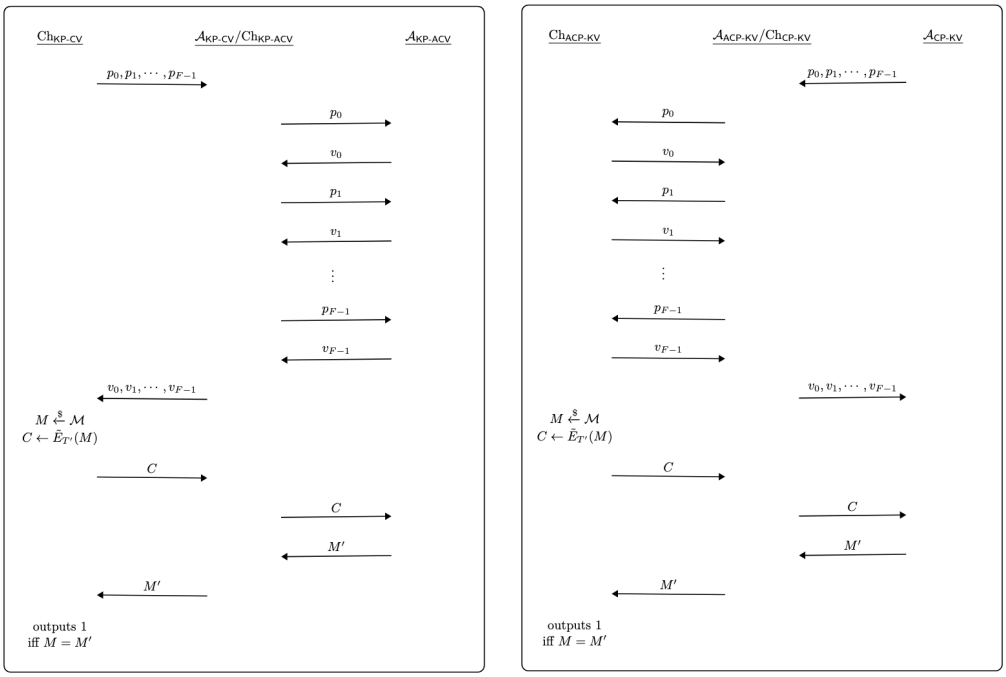
[30] Hongjun Wu. Related-cipher attacks. In Robert H. Deng, Sihan Qing, Feng Bao, and Jianying Zhou, editors, *ICICS 02: 4th International Conference on Information and Communication Security*, volume 2513 of *Lecture Notes in Computer Science*, pages 447–455, Singapore, December 9–12, 2002. Springer Berlin Heidelberg, Germany.

[31] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007: 14th Annual International Workshop on Selected Areas in Cryptography*, volume 4876 of *Lecture Notes in Computer Science*, pages 264–277, Ottawa, Canada, August 16–17, 2007. Springer Berlin Heidelberg, Germany.

[32] Yaying Xiao and Xuejia Lai. A secure implementation of white-box aes. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6. IEEE, 2009.

# Appendix

## A Relation Between Various Adversarial Models




**(a)** KP-ACV reduces to KP-CV. **(b)** CP-KV reduces to ACP-KV.

**Figure 8:** Reduction of KP-ACV to KP-CV and CP-KV to ACP-KV.

## B NdUF Analysis of Galaxy

In this section, we provide the NdUF analysis of Galaxy under the KP-KV, CP-KV, ACP-KV, and attack under KP-CV, CP-CV adversarial models. We first specify Galaxy below.

### B.1 NdUF Analysis of Galaxy Under KP-KV

In this section, we analyze the bound for $(F, D)$-NdUF security of Galaxy in KP-KV attack setting. To evaluate NdUF bounds under the KP-KV attack, we propose Theorem 7.

**Theorem 7.** *Given $F$ distinct fault values and their corresponding positions in each table under KP-KV attack setting for the table $\mathsf{T}$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of a white-box implementation of Galaxy, it is infeasible for any adversary to decrypt a faulty ciphertext corresponding to any randomly chosen plaintext with a probability of more than $\left(\frac{F}{2^{n_{in}}}\right)^{\frac{w}{2} \cdot \mathsf{r}}$, where $\mathsf{r}$ is the total number of rounds and $w$ is the number of $n_{in}$-bit blocks in each round.*

*Proof.* Given $F$ distinct fault values and their corresponding positions, each look-up of the table is included in the fault values with probability $\frac{F}{2^{n_{in}}}$. As the number of $n_{in}$-bit branches in each round is $w$, the same table $\mathsf{T}'$ is used $\frac{w}{2}$ times in each of the $\mathsf{r}$ rounds. Thus, the adversary can decrypt a faulty ciphertext corresponding to any random plaintext with probability $\left(\frac{F}{2^{n_{in}}}\right)^{\frac{w}{2}\cdot\mathsf{r}}$. □

A comparative summary of NdUF bounds for the Galaxy ciphers under the KP-KV fault model is presented in Table 8.

**Table 8:** NdUF bounds for Galaxy ciphers under KP-KV fault model.

| | Galaxy-16 | Galaxy-32 |
|---|---|---|
| KP-KV | $(\frac{\sigma(\mathsf{T})}{2^2}, 128)$ | $(\frac{\sigma(\mathsf{T})}{2^2}, 128)$ |

**Table 9:** NdUF bounds for Galaxy ciphers under ACP-KV fault model.

| | Galaxy-16 | Galaxy-32 |
|---|---|---|
| ACP-KV | $(\frac{\sigma(\mathsf{T})}{2^2}, 118.80)$ | $(\frac{\sigma(\mathsf{T})}{2^2}, 102.80)$ |

## B.2 NdUF Analysis of Galaxy Under CP-KV

For similar reasons, described in Section 6.2, the NdUF bound of Galaxy is $\left(\frac{F}{2^{n_{in}}}\right)^{\frac{w}{2}\cdot\mathsf{r}}$ and the bounds for specific instances are the same as Table 8.

## B.3 NdUF Analysis of Galaxy Under ACP-KV

In this section, we evaluate NdUF security bounds under the ACP-KV model and propose Theorem 8, which is similar to Theorem 6 except for the number of table look-ups.

**Theorem 8.** *For $F$ distinct fault positions under the ACP-KV model for table $\mathsf{T}$ (containing $2^{n_{in}}$ entries each of $n_{in}$-bits) of the white-box implementation of Galaxy, it is infeasible for any adversary to decrypt the challenge ciphertext corresponding to any randomly chosen plaintext with probability higher than $N \cdot 2^{-n} + (1 - N \cdot 2^{-n}) \cdot (\frac{F}{2^{n_{in}}})^{\frac{w}{2}\cdot\mathsf{r}}$, where $e_{in} = \frac{2^{n_{in}}-1}{2^{n_{in}}}$, $\mathsf{r}$ is the total number of rounds and $N = \left\lceil \frac{2\cdot\log_{e_{in}}\left(1-\frac{F}{2^{n_{in}}}\right)}{w\cdot\mathsf{r}} \right\rceil$.*

*Proof.* From Lemma 1, we observe that $2^{n_{in}} \cdot \left(1 - (\frac{2^{n_{in}}-1}{2^{n_{in}}})^{N\cdot\frac{w}{2}\cdot\mathsf{r}}\right)$ positions are needed to encrypt $N$ plaintexts, as $N \cdot \frac{w}{2} \cdot \mathsf{r}$ table look-ups are required to encrypt $N$ plaintexts. This shows that $F$ fault positions enable to compute $N = \left\lceil \frac{2\cdot\log_{e_{in}}\left(1-\frac{F}{2^{n_{in}}}\right)}{w\cdot\mathsf{r}} \right\rceil$ plaintext-ciphertext pairs in advance. The challenge ciphertext is then included in the precomputed set of plaintext-ciphertext pairs with probability $N \cdot 2^{-128}$. Otherwise, for given $F$ distinct fault positions, the challenge ciphertext can be decrypted with probability $(F/2^{n_{in}})^{\frac{w}{2}\cdot\mathsf{r}}$, as Theorem 7.

□

A comparative summary of attack complexities for the Galaxy ciphers under the ACP-KV fault model is presented in Table 9.

## B.4 NdUF Analysis of Galaxy Under KP-CV

In this section, we present an attack technique for Galaxy against the KP-CV model. The attack strategy in this construction is conceptually similar to the KP-CV attack described for SPACE in Section 6.4.2. However, there are two important differences in the construction of Galaxy. First, in each round, the construction invokes the function $G$, $\frac{w}{2}$ times. Second, both the input and output sizes of the function $G$ are of $n_{in}$ bits. These differences affect how the adversary can control the inputs to the function during the attack. Also, the total number of rounds is $\mathsf{r} = n_{in}$. Consequently, the attack technique for Galaxy differs slightly from that for SPACE.

**Key Observation.** In this attack, we use the same input for all $\frac{w}{2}$ tables within a round, allowing a plaintext–ciphertext pair to be computed using a smaller number of table inputs. Let $I_i$ denote the input to all tables $\mathsf{T}$ in the $i^{th}$ round, for $1 \leq i \leq (\mathsf{r}-1)$. For this construction, we observe that if $I_i \in \mathcal{P}$ for some $i \in [\mathsf{r}-1]$ and no fault has been injected yet at the position $I_i$, then by injecting a suitable fault at that position for the plaintext of the form $I_1 \| x_1 \| I_1 \| x_1 \| \ldots \| I_1 \| x_1$, we can control the input to the table $\mathsf{T}$ only for the next round. Thus, the input $I_{i+1}$ can only be forced to take the predetermined values. For example, we assume that $I_1 = p_0$ for some $p_0 \in \mathcal{P}$ and no fault value is injected at the position $p_0$ yet. If we want $I_2 = p_0$, then we have to inject the fault value $p_0 \oplus x_1 \oplus 1$ at the position $p_0$. In the case of the same input for each $\frac{w}{2}$ tables within a round, the remaining $\frac{w}{2}$ many $n_{in}$-bits of that round must be the same. With this property, the adversary tries to compute as many plaintext-ciphertext pairs as possible.

**Attack Procedure.** In this attack, we consider the plaintexts which are of the form $p \| 0^{n_{in}} \| p \| 0^{n_{in}} \| \cdots \| p \| 0^{n_{in}}$, where $p \in \mathcal{P}$. The procedure of computing the ciphertext corresponding to such a plaintext is described as follows.

1. Select $(\mathsf{r}-1)$ positions $p_0, p_1, \cdots, p_{\mathsf{r}-2}$ from the set of fault positions $\mathcal{P}$.
2. Consider $p = p_0$ i.e., the plaintext is $p \| 0^{n_{in}} \| p \| 0^{n_{in}} \| \cdots \| p \| 0^{n_{in}}$. Thus, the input to the table for $G$ in the first round is $I_1 = p_0$.
3. Inject a suitable fault value at the position $p_0$ such that $I_2 = p_1$.
4. Next, inject a suitable fault value at the position $p_1$ to obtain $I_3 = p_2$ and so on.
5. Repeating this process ensures that $I_{\mathsf{r}-2} = p_{\mathsf{r}-3}$ and $I_{\mathsf{r}-1} = p_{\mathsf{r}-2}$.
6. Finally, inject a suitable fault value at the position $p_{\mathsf{r}-2}$ so that the next input $I_{\mathsf{r}-1} = p_{\mathsf{r}-2}$.
7. If there exist positions in $\mathcal{P}$ other than the above-mentioned $(\mathsf{r}-1)$ positions, select another $(\mathsf{r}-1)$ unused position and repeat from Step 2 in order to compute another plaintext-ciphertext pair.

Using this attack procedure, $(\mathsf{r}-1)$ different fault positions are required to compute one plaintext-ciphertext pair. Hence, $\lfloor \frac{F}{(\mathsf{r}-1)} \rfloor$ many plaintext-ciphertext pairs can be computed, for $F$ faults. The algorithm for choosing fault values corresponding to the given positions is given in the Algorithm 3.

**Algorithm 3** Attack for Galaxy in the KP-CV Setting.

Suppose $P = \{p_0, p_1, \cdots, p_{F-1}\}$ be the set of positions where the adversary can inject faults
$\mathcal{F} \leftarrow \emptyset$, $j = 0$
**while** $j < F$ **do**
  $X^0 \leftarrow p_j \| 0^{n_{in}} \| \cdots \| p_j \| 0^{n_{in}}$
  **for** $i = 0$ to $(\mathsf{r}-2)$ **do**
    **if** $i = 0$ **then**
      $v_j \leftarrow p_{j+1} \oplus (i+1)$
    **else if** $0 < i < (\mathsf{r}-2)$ **then**
      $v_j \leftarrow p_{j+1} \oplus p_{j-1} \oplus (i+1)$
    **else**
      $v_j \leftarrow p_{j-1} \oplus p_j \oplus (i+1)$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{(p_j, v_j)\}$
    $j \leftarrow j+1$
**return** $\mathcal{F}$

**Algorithm 4** Attack for Galaxy in the CP-CV Setting. Here $t_2 = (\lfloor \log \mathsf{r} \rfloor + 1)$.

$\mathcal{F} \leftarrow \emptyset$
Choose distinct elements $s_0, s_1, \cdots, s_{\frac{F}{2^{t_2}}-1}$ from the set $\{0,1\}^{n_{in}-t_2}$
**for** $i = 0$ to $\frac{F}{2^{t_2}} - 1$ **do**
  **for** $s = 0$ to $2^{t_2} - 1$ **do**
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{(s_i \| s, 0^{n_{in}})\}$
**return** $\mathcal{F}$

**Attack Probability.** The attack procedure prepares a set of $\lfloor \frac{F}{(\mathsf{r}-1)} \rfloor$ many PT-CT pairs among $2^n = 2^{128}$ many PT-CT pairs. Thus, the challenge ciphertext is included in the prepared set with probability $\lfloor \frac{F}{(\mathsf{r}-1)} \rfloor \cdot 2^{-n}$. In particular, the adversary obtains at least one ciphertext from the prepared set, after getting $(\lfloor \frac{F}{(\mathsf{r}-1)} \rfloor \cdot 2^{-n})^{-1}$ ciphertexts. A comparative

summary of attack complexities for the Galaxy ciphers under the KP-CV fault model is presented in Table 10.

**Table 10:** Attack probability for Galaxy under KP-CV for different values of $F$.

| Cipher | $F = \frac{\sigma(T)}{2}$ | $F = \frac{\sigma(T)}{4}$ | $F = \frac{\sigma(T)}{8}$ | $F = \frac{\sigma(T)}{16}$ |
|---|---|---|---|---|
| Galaxy-16 | $2^{-116.91}$ | $2^{-117.91}$ | $2^{-118.91}$ | $2^{-119.91}$ |
| Galaxy-32 | $2^{-101.95}$ | $2^{-102.95}$ | $2^{-103.95}$ | $2^{-104.95}$ |

**Table 11:** Attack probability for Galaxy under CP-CV for different values of $F$.

| Cipher | $F = \frac{\sigma(T)}{2}$ | $F = \frac{\sigma(T)}{4}$ | $F = \frac{\sigma(T)}{8}$ | $F = \frac{\sigma(T)}{16}$ |
|---|---|---|---|---|
| Galaxy-16 | $2^{-8}$ | $2^{-16}$ | $2^{-24}$ | $2^{-32}$ |
| Galaxy-32 | $2^{-4}$ | $2^{-8}$ | $2^{-12}$ | $2^{-16}$ |

## B.5 NdUF Analysis of Galaxy Under CP-CV

In this section, we present an attack technique for Galaxy assuming fault injections into $F$ table entries against the CP-CV model. This attack strategy is conceptually similar to the attack for SPACE described in Section 6.5.

**Key Observation.** For this model, the adversary can choose the set of fault positions $\mathcal{P}$ with cardinality $F$ such that it is possible to deterministically compute the ciphertext corresponding to any plaintext of the form $x_0\|x_1\|\cdots\|x_{w-1}$, where $x_i \in \mathcal{P}$ for all $i \in \{0, 1, \cdots, w-1\}$, by injecting appropriate fault values at these positions. In particular, all inputs to the table corresponding to $G$ for a total of $\mathsf{r}$ rounds are included in the set $\mathcal{P}$. Consequently, the adversary aims to compute $F^w$ plaintext-ciphertext (PT-CT) pairs.

**Attack Procedure.** In this attack, to ensure deterministic computation of the ciphertext for any plaintext of the form $x_0\|x_1\|\cdots\|x_{w-1}$, where $x_i \in \mathcal{P}$ for all $i \in \{0, 1, \cdots, w-1\}$, the fault positions are selected as follows:

1. Choose $\frac{F}{2^{t_2}}$ distinct elements $s_0, s_1, \ldots, s_{(\frac{F}{2^{t_2}}-1)}$ from the set $\{0,1\}^{n_{in}-t_2}$, where $t_2 = (\lfloor \log \mathsf{r} \rfloor + 1)$ is the maximum number of bits used for round constant.
2. For each $s_i$, include all string of the form $s_i\|s$ in the set $\mathcal{P}$, where $s \in \{0,1\}^{t_2}$, as the maximum number of round is $\mathsf{r}$, a $t_2$-bits number. In other words, for every fixed prefix $s_i$, all possible $2^{t_2}$ many positions are obtained by appending $t_2$-bit suffixes.
3. Inject the fault value $0^{n_{in}}$ at each of the positions of $\mathcal{P}$.

With this choice of $\mathcal{P}$ and injecting all fault values $0^{n_{in}}$, all the intermediate inputs for any plaintext of the form $x_0\|x_1\|\cdots\|x_{w-1}$ with $x_i \in \mathcal{P}$ for all $i \in \{0, 1, \cdots, w-1\}$. Consequently, those plaintexts can be encrypted deterministically. Hence, the adversary can compute $F^w$ many PT-CT pairs in advance. The algorithm for choosing fault positions and values is given in the Algorithm 4.

**Attack Probability.** From the attack, $F^w$ many PT-CT pairs can be computed in advance. Thus, the challenge ciphertext is included in the prepared set with probability $F^w \cdot 2^{-n}$. In particular, after $(F^w \cdot 2^{-n})^{-1}$ ciphertexts, the adversary obtains at least one ciphertext from the prepared set. A comparative summary is presented in Table 11.