A Note on Combined Attacks on Fallen Sanctuary

Enanko Basak[1] and Sayandeep Saha[1]

Indian Institute of Technology Bombay, Mumbai, India,
enankobasak@cse.iitb.ac.in,sayandeepsaha@cse.iitb.ac.in

Abstract. Leakage-resilient rekeying schemes aim to maintain cryptographic security in the presence of side-channel leakage by periodically refreshing ephemeral keys before sufficient information can be accumulated by an adversary. Fallen Sanctuary (LR4) is a recent higher-order leakage-resilient rekeying construction that achieves exponential security amplification with respect to the number of primitive encryption invocations and the number of traces required to compromise the physical security of the implementation. Its security, however, relies on the correct maintenance of internal counters and cached intermediate keys that enforce the prescribed trace bounds.

In this work, we investigate the security of LR4 under a combined fault and side-channel attack model. We show that transient faults targeting the counter-update and counter-validation mechanism can prevent the advancement of the rekeying state, causing repeated reuse of temporal keys that are intended to be short-lived. As a consequence, the bounded-trace assumptions underlying the LR4 security proof no longer hold. We demonstrate that an adversary can accumulate an arbitrary number of leakage traces corresponding to the same secret state, effectively reducing the security of the protected primitive to that of a conventional implementation without rekeying.

We evaluate the attack on a fault simulated implementation and analyze its impact on the leakage-resilience guarantees claimed by LR4.

Our findings emphasize that leakage-resilient rekeying schemes must consider fault-induced violations of state evolution assumptions in addition to conventional side-channel leakage.

Keywords: Fault Attacks · Side Channel Attacks · Combined Attacks · Physical Security

1 Introduction

Side-channel attacks (SCAs) pose a significant threat to cryptographic implementations by exploiting physical leakage to recover secret information. Leakage-resilient cryptography addresses this problem by limiting the amount of information an adversary can accumulate from any single secret state.[DEM+20] [DMP22] Among such approaches, fresh rekeying periodically updates temporal keys before sufficient leakage can be collected for successful key recovery.

Fallen Sanctuary (LR4) [UHIM24] is a recent higher-order leakage-resilient rekeying scheme that achieves exponential growth in the number of secure cryptographic operations under a bounded-trace leakage model. Its security relies on the assumption that internal counters and cached intermediate keys evolve correctly, ensuring that no secret state is used beyond its prescribed trace bound.

In recent years, combined fault and side-channel attacks[SBJ+21] [SRJB23] have gained significant attention as practical threats against cryptographic implementations. Unlike traditional attacks that exploit either incorrect computation or physical leakage

independently, combined attacks leverage faults to alter the internal execution state and side-channel analysis to extract sensitive information from the resulting behavior. By disrupting security-critical control flow or state-management operations, fault injection can amplify the effectiveness of side-channel attacks by enabling repeated observations of secret-dependent computations. This motivates the evaluation of leakage-resilient constructions under adversarial fault conditions in addition to conventional leakage models.

In this work, we investigate the security of LR4 under fault attacks targeting the cached counter-update mechanism and cached counter-validation mechanism. We show that faults preventing counter advancement can force repeated reuse of temporal keys, violating the bounded-trace assumptions on which LR4's security guarantees depend. As a result, an attacker can collect an arbitrary number of leakage traces from the same secret state and mount conventional side-channel attacks against the protected implementation.

Our findings reveal an important gap between the cryptographic model of LR4 and the behavior of faulted implementations. We demonstrate that preserving leakage resilience requires not only protection against side-channel leakage but also mechanisms that guarantee the integrity of state updates and counter evolution. To the best of our knowledge, this work is the first analysis of how faults targeting the state-transition mechanisms of leakage-resilient rekeying schemes can invalidate bounded-trace security assumptions.

1.1 Background

Fallen Sanctuary (LR4) is a leakage-resilient rekeying scheme proposed by Ueno et al. [UHIM24] to protect cryptographic implementations against side-channel attacks. Unlike conventional approaches that attempt to eliminate leakage, LR4 limits the amount of information that can be obtained from any individual secret state by periodically deriving fresh temporal keys. The security of the construction is based on the observation that an attacker should never be able to collect more than a bounded number of leakage traces corresponding to the same key.

Unlike conventional masking-based countermeasures, where the objective is to decorrelate physical leakage from the secret value, LR4 limits the adversary's measurement capability by ensuring that each secret-dependent computation is executed only a bounded number of times. The construction achieves this property by deriving a sequence of short-lived temporal keys from a long-term master secret.

The LR4 rekeying mechanism is organized as a hierarchical derivation tree. Starting from a long-term master key, a sequence of intermediate keys is derived using a public counter value and a family of rekeying functions. The final derived key is used as a temporal encryption key for the underlying cryptographic primitive. As the counter advances, the construction moves through different branches of the derivation tree, ensuring that fresh keys are generated throughout the lifetime of the device.

To improve efficiency, LR4 employs a caching mechanism that stores previously derived intermediate keys together with their associated counter values. When a new counter is processed, only the portions of the derivation tree that have changed need to be recomputed. This significantly reduces the computational overhead while preserving the leakage-resilience properties of the scheme.

A simplified view of the cache-based rekeying procedure is shown in Algorithm 1. Before deriving a temporal key, the algorithm verifies that the supplied counter value is valid with respect to the cached state. This check prevents replayed counters and ensures that the internal state progresses monotonically. If the counter is accepted, the required intermediate keys are recomputed, a fresh temporal key is generated, and the cached counter state is updated to reflect the most recently processed value.

The security analysis of LR4 relies critically on the correctness of these state-management operations. In particular, the bounded-trace guarantees assume that replayed counters are rejected and that the cached state is updated correctly after every successful derivation.

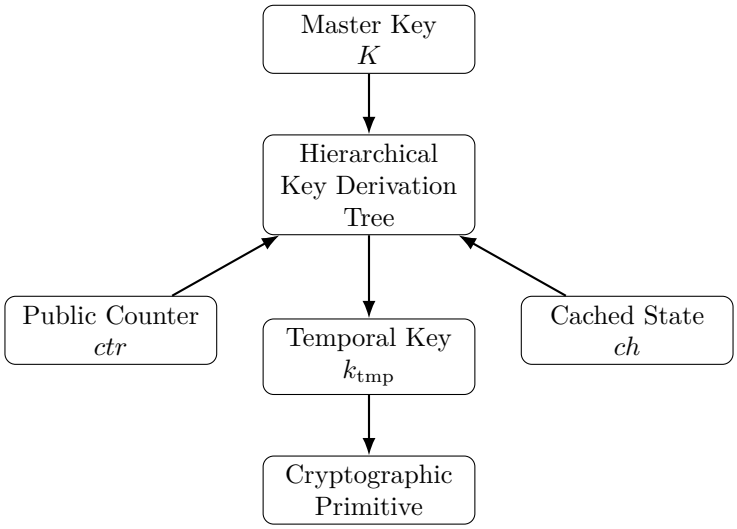

Figure 1: Simplified overview of the LR4 rekeying mechanism. The public counter determines the derivation path, while cached intermediate states improve efficiency. The resulting temporal key is used by the underlying cryptographic primitive.

Under these assumptions, each secret state is evaluated only a limited number of times, preventing an adversary from accumulating enough leakage observations to recover key material through side-channel analysis.

Algorithm 1: Algorithm $R_C(\mathrm{ctr}^{d+1}; \mathrm{kh}^{d+1}, \mathrm{ch}^{d+1})$

if $\sum_{i=1}^{d+1} 2^{n_{\mathrm{ctr}(i-1)}} \mathrm{ctr}_{d+2-i} \leq \sum_{i=1}^{d+1} 2^{n_{\mathrm{ctr}(i-1)}} \mathrm{ch}_{d+2-i}$ *or* $\exists i, \mathrm{ctr}_i \geq m$ *or* $\mathrm{ctr}_{d+1} \geq m'$ **then**
  return $\perp$ (Abort encryption/decryption)
else
  flag $\leftarrow 0$;
  for $i = 1$ **to** d **do**
    if $\mathrm{ctr}_i > \mathrm{ch}_i$ *and* flag $= 0$ **then**
      flag $\leftarrow 1$;
  if flag $= 1$ **then**
    $\mathrm{kh}_{i+1} \leftarrow G_i(\mathrm{kh}_i, \mathrm{ctr}_i)$;
  $k_{\text{tmp}} \leftarrow \mathrm{kh}_{d+1}$;
  $\mathrm{ch}^{d+1} \leftarrow \mathrm{ctr}^{d+1}$;
  return $k_{\text{tmp}}; \mathrm{kh}^{d+1}, \mathrm{ch}^{d+1}$

2 Fault-Induced Violation of the Bounded-Trace Assumption

The security of Fallen Sanctuary (LR4) is based on the observation that each secret state involved in the rekeying hierarchy is evaluated only a bounded number of times. The security proof models this property through the bounded-trace assumption, under which an adversary cannot obtain more than a prescribed number of leakage observations from the same intermediate key. Consequently, even a theoretically optimal side-channel adversary

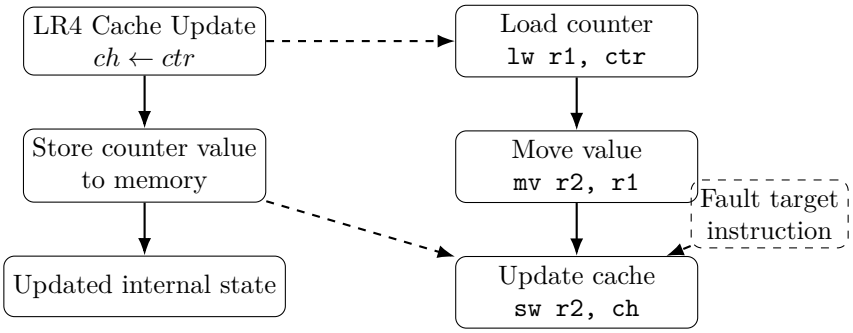

Figure 2: Mapping of the LR4 cache-update operation to assembly-level state transition. The cache update $ch \leftarrow ctr$ is implemented through a memory write instruction. Skipping this instruction prevents state advancement and enables repeated reuse of the same temporal key.

is prevented from accumulating sufficient information to recover secret key material.

In this work, we show that this assumption can be invalidated through faults targeting the state-management logic of the implementation. In particular, we identify two fault targets that independently enable repeated execution of the same derivation path and, therefore, allow the collection of an effectively unbounded number of side-channel traces from identical secret-dependent computations.

2.1 Attack A: State-Update Bypass

The first attack targets the cache-update operation performed after successful temporal-key derivation in line 11 of Algorithm 1. The exact assembly instructions is described in Figure 2. In the cache-based LR4 construction, the internal cached counter value is updated to match the externally supplied counter after the derivation procedure is completed. This update ensures that the internal state accurately reflects the most recently processed counter and allows future invocations to determine which portions of the derivation tree have already been evaluated.

Suppose that an adversary induces a fault that prevents this update operation from completing while allowing the remainder of the computation to proceed normally. In such a scenario, the device returns a valid cryptographic output, but its internal state remains unchanged. From the perspective of the cache, the processed counter value appears never to have been used.

As a consequence, subsequent invocations may repeatedly traverse the same derivation path and recompute identical intermediate keys. Since the affected secret states are no longer retired after use, the adversary can repeatedly trigger the same secret-dependent computations and collect an arbitrary number of side-channel measurements.

Importantly, the attack does not directly expose secret master key through the fault itself. Instead, the fault prevents the implementation from advancing its internal state, causing the same temporal key to be generated and reused across multiple invocations. Since the attacker is free to vary the plaintext input while keeping the counter value fixed, an arbitrary number of encryptions can be performed with the same ephemeral key.

This behavior fundamentally violates the bounded-trace assumption underlying the LR4 security analysis. Under normal operation, each temporal key is intended to be used only a limited number of times before the rekeying mechanism advances to a new state. After a successful state-update bypass, however, the attacker can repeatedly trigger computations involving the same secret key material and collect an effectively unbounded number of side-channel measurements.

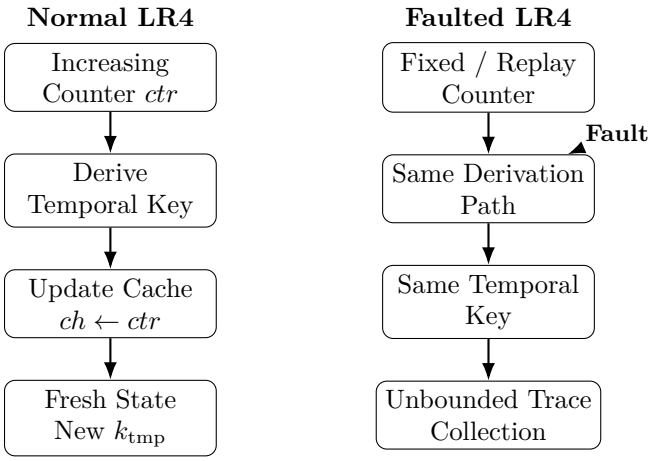

Figure 3: Effect of state-management faults on LR4. Faults preventing counter evolution or cache updates allow repeated reuse of the same temporal key, violating the bounded-trace assumption.

Consequently, the protected implementation effectively degenerates into a conventional fixed-key cryptographic implementation. Standard side-channel techniques, such as Correlation Power Analysis (CPA), can then be applied to recover the final ephemeral key. Once this key has been extracted, all encryptions performed under the affected counter value can be decrypted, defeating the primary security objective of the leakage-resilient rekeying mechanism.

2.2 Attack B: Replay-Check Bypass

The second attack targets the counter-validation procedure performed at the beginning of the cache-based rekeying in Line 1 of Algorithm 1. Before deriving a temporal key, LR4 verifies that the supplied counter value is consistent with the cached state. Counters that correspond to previously processed values are rejected in order to prevent replay attacks and repeated evaluation of the same derivation path.

If a fault causes this validation procedure to be skipped, the implementation may continue execution even when the supplied counter should have been rejected. An adversary can therefore repeatedly submit the same counter value and force the device to derive the corresponding temporal key multiple times.

In contrast to Attack A, the cache state may continue to be updated correctly. Nevertheless, replay protection is no longer enforced, and the implementation repeatedly evaluates the same secret-dependent computations. The adversary can therefore obtain an arbitrary number of leakage traces associated with identical intermediate keys and temporal keys.

From the perspective of side-channel security, the consequences are equivalent to those of Attack A. The replay-detection mechanism is responsible for enforcing the bounded-trace assumption in the presence of repeated counter submissions. Once this mechanism is bypassed, the assumption no longer holds.

2.3 Trace Averaging

The proposed faults allow the attacker to repeatedly execute the same key derivation path and regenerate the same temporal key. Consequently, the adversary can collect multiple power traces corresponding to identical secret-dependent operations.

For a fixed secret state, the measured trace can be represented as

$$T_i = L(K) + N_i$$

where $L(K)$ represents the leakage dependent on the secret key K, and N_i represents measurement noise. By collecting N traces of the same computation and averaging them,

$$\bar{T} = \frac{1}{N} \sum_{i=1}^{N} T_i,$$

the noise component is reduced while the key-dependent leakage is preserved.

In the faulted LR4 implementation, the attacker can therefore amplify the leakage associated with the repeated temporal key derivation process, by increasing the number of collected traces. As the temporal key is repeatedly regenerated from the same master key under the faulted execution, the averaged leakage on the master key can enable the recovery of the master key. The original LR4 security analysis assumes a trace-bounded leakage model; however, the introduced fault allows the attacker to collect an unbounded number of traces. Consequently, trace averaging becomes a practical attack strategy that was not considered in the original security evaluation.

The fault effectively transforms the bounded-trace leakage model into a data-bounded and unbounded-trace setting, where trace averaging provides the attacker with a practical advantage for master key recovery.

Thus, the trace-bounded security assumption of the LR4 algorithm does not hold; by enabling the collection of an effectively unbounded number of traces, the attacker can perform extensive trace averaging to enhance the leakage signal and recover key-dependent information. Therefore, the original trace bound remains valid only when the counter is restricted to a sufficiently small value.

Thus, the trace-bounded security assumption of the LR4 algorithm is violated under the proposed fault scenario. By enabling the collection of an effectively unbounded number of traces, the attacker can exploit trace averaging to enhance the leakage signal and recover key-dependent information. Therefore, the original trace bound remains valid only when the counter is restricted to a sufficiently small value.

2.4 Security Implications

The two attacks target different implementation components but ultimately achieve the same objective: repeated reuse of secret states that were intended to be evaluated only a bounded number of times. Attack A prevents the internal state from recording progress, whereas Attack B prevents the implementation from verifying that progress has occurred. In both cases, the attacker gains the ability to repeatedly execute the same derivation path.

The security proof of LR4 assumes that the cache state evolves correctly and that replayed counters are rejected. Under these assumptions, the number of leakage observations available for any secret state remains bounded. However, once either assumption is violated, the adversary can collect an effectively unbounded number of side-channel traces corresponding to the same secret-dependent computation.

This fundamentally changes the threat model. Instead of facing a bounded-trace adversary it reduces to a bounded-data adversary, the implementation now faces a conventional side-channel adversary capable of accumulating arbitrarily many measurements. Consequently, standard techniques such as Correlation Power Analysis (CPA), template attacks, and higher-order leakage analysis become applicable against intermediate keys and temporal keys that were originally protected by the rekeying mechanism.

Therefore, while LR4 remains secure under its stated assumptions, our analysis demonstrates that the integrity of the state-management logic is a critical prerequisite for

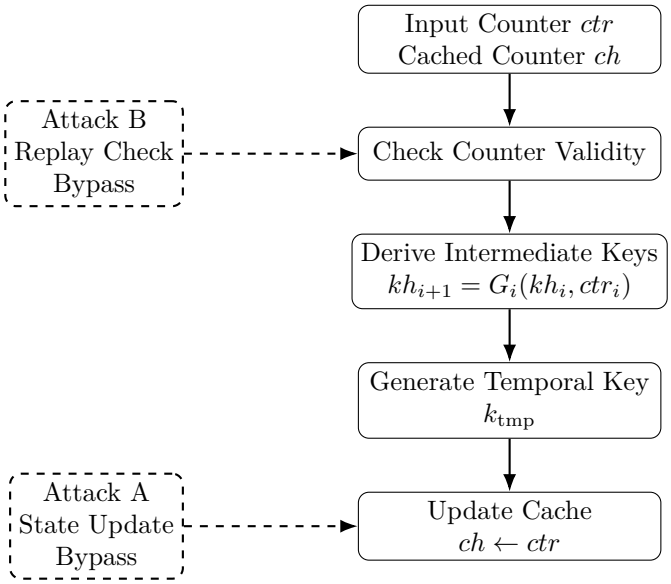

Figure 4: Fault injection targets in the LR4 key derivation procedure. Attack B bypasses counter validation, while Attack A prevents cache state advancement.

maintaining its leakage-resilience guarantees. Faults affecting either cache updates or replay detection effectively collapse the bounded-trace model and reduce the practical security of the implementation to that of a conventional cryptographic implementation without fresh rekeying protection.

3 Implementation Results

To evaluate the practical impact of the proposed attacks, we implemented the LR4 algorithm on a NewAE ChipWhisperer CW308 [1] target platform with an STM32F415 Microcontroller and performed side-channel measurements using the NewAE ChipWhisperer Husky [2]capture setup.

3.1 Baseline Side-Channel Evaluation

We first evaluated the leakage characteristics of the target board under repeated execution of the same secret-dependent computation. Power traces were collected from the target device and analyzed using Correlation Power Analysis (CPA).

Our experiments show that the targeted intermediate value can be recovered with as few as 16 traces. Thus, a temporal key for this platform must be used less than 16 times, before it becomes susceptible to standard CPA attacks.

3.2 Fault Simulation

To evaluate the impact of the proposed attacks, a fault simulation is performed by simulating instruction-level faults in the state-management logic at every invocation of the algorithm.

For Attack A, we skip the cache-update operation that synchronizes the internal cache state with the processed counter value. Consequently, the cache remained stale after each

[1] https://rtfm.newae.com/Targets/CW308UFO/
[2] https://rtfm.newae.com/Capture/ChipWhisperer-Husky/

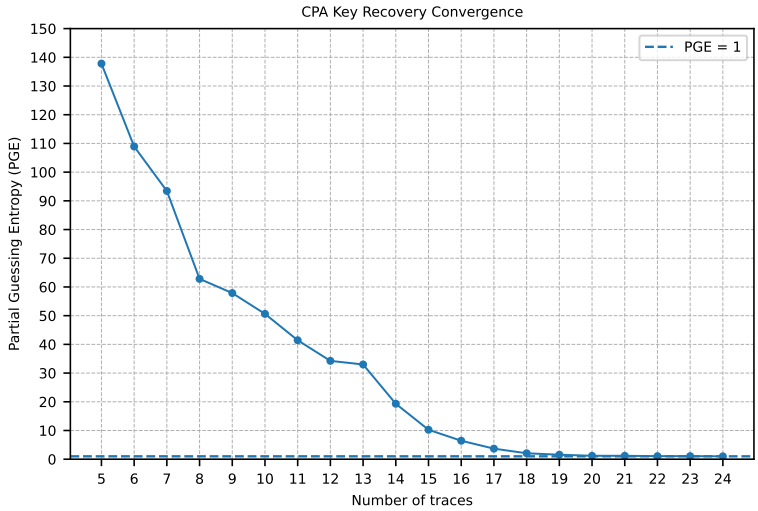

Figure 5: GE convergence for ephemeral key recovery under the cache-update bypass attack (Attack A). Reuse of the temporal key enables accumulation of leakage traces, reducing the key rank to 1 after 24 traces.

invocation, forcing the implementation to repeatedly evaluate the same derivation path and generate an identical temporal encryption key. Since arbitrary plaintexts could be encrypted under this reused key, an effectively unbounded number of side-channel measurements became available to the attacker. Using the collected power traces, we performed a standard Correlation Power Analysis (CPA) attack and successfully recovered the temporal encryption key. This result confirms that bypassing the state-update mechanism eliminates the bounded-trace protection provided by LR4 and reduces the implementation to a conventional fixed-key side-channel target.

For Attack B, we skip the counter-validation procedure responsible for detecting and rejecting previously processed counter values. Consequently, replayed counters were accepted indefinitely, allowing the implementation to repeatedly execute the same key-derivation path. Since identical temporal keys and intermediate states could be regenerated on demand, an effectively unbounded number of power traces became available to the attacker. This behavior defeats the replay-protection mechanism central to the LR4 security model and effectively transforms the bounded-trace leakage setting into an unbounded-trace one, thereby allowing standard side-channel attacks against the temporal encryption key.

In both cases, the resulting behavior was identical from the perspective of side-channel analysis: the same secret-dependent computations could be triggered repeatedly using the same counter value.

3.3 Discussion

The experimental results validate the central claim of this work. LR4 derives its leakage resilience from the assumption that each secret state is observed only a bounded number of times. Our fault simulations show that violations of the state-update or replay-detection mechanisms invalidate this assumption and allow for an unbounded reuse of secret-dependent computations.

The attacks illustrate how a fault adversary can transform a bounded-trace leakage model into an unbounded-trace one. Consequently, the practical security of the implementation is reduced to that of a conventional implementation that lacks fresh rekeying

protection.

References

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2. 0. *IACR Transactions on Symmetric Cryptology*, pages 390–416, 2020.

[DMP22] Christoph Dobraunig, Bart Mennink, and Robert Primas. Leakage and tamper resilient permutation-based cryptography. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 859–873, 2022.

[SBJ+21] Sayandeep Saha, Arnab Bag, Dirmanto Jap, Debdeep Mukhopadhyay, and Shivam Bhasin. Divided we stand, united we fall: Security analysis of some sca+ sifa countermeasures against sca-enhanced fault template attacks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 62–94. Springer, 2021.

[SRJB23] Sayandeep Saha, Prasanna Ravi, Dirmanto Jap, and Shivam Bhasin. Non-profiled side-channel assisted fault attack: A case study on domrep. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[UHIM24] Rei Ueno, Naofumi Homma, Akiko Inoue, and Kazuhiko Minematsu. Fallen sanctuary: A higher-order and leakage-resilient rekeying scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(1):264–308, 2024.