Tears for fears, breaking an RFID counter

Pierre Granier[1], Jean-Joseph Marty[2] and Rémy Delion[2]
pierre.granier@univ-rennes.fr
jjmarty@amossys.fr
rdelion@amossys.fr

[1] University of Rennes
[2] Amossys

1 Introduction

In an ITSEF (Information Technology Security Evaluation Facility) laboratory, it is mandatory to keep state-of-the-art skills over time. This can be achieved by adapting existing attacks on novel hardware. This monitoring led us to adapt research on RFID tearing [6] to the monotonic counter of the ST25TB-based transportation tickets, breaking an anti-tearing protection previously deemed safe.

These tickets can be found in several French transportation systems which often offer two main NFC enabled supports for ticketing. The first solution is a smart card which embeds the necessary element to ensure their authenticity and integrity. The second, and the subject of this paper, is a disposable ticket with minimal functionality.

In the latter solution, a certificate to ensure ticket integrity is used. This certificate is computed based on the card UID, content (e.g., rides left on the ticket), and a monotonic counter. Here the monotonic counter's purpose is to prevent reuse of the transport application certificate.[3] This monotonic counter is a memory area designed to only allow decreases of its value while resisting accidental early disconnection of the tag from the reader. The family of card studied here is robust against this scenario. However, we present a way to break these counters by disconnecting intentionally and repeatedly the tag during operations. This defeats all mechanisms trusting the monotonic properties of those cards. Sources of our exploit are available at [4].

In this situation, the monotonic counter was solely designed to resist against accidental tearing, such as the one triggered by a normal user

[3] A more detailed view of the French ST25TB-based transport systems mechanisms is available [1]

removing his card too quickly. However, this hardware was not designed to handle an attacker playing with repeated and brutal tearing.

Noting that the difference between safety and security lies in the approach applied to make critical products. The safety approach ensures that the target behave *at least* as intended. The security approach ensures that the target behave *at most* as intended. While safety is the way critical approaches are designed, it checks that the target works within a nominal situation. In opposition, security takes into account an attacker that deliberately adapts to the protection mechanisms.

2 Related Works

For RFID technologies, *tear-off* refers to the removal of an RFID tag from a reader while the reader is interacting with the card tag [2]. Tearing comes in a few flavors. It can be caused by the user provoking an early removal of the tag from the reader. This slowly cuts the power supply of the card. Such issue is known and mitigated in RFID chip design. Tearing can also be caused by an unexpected shutdown of the reader. Checks are usually made to ensure a valid memory state in such a case.

In our case we use repeated and precise reader shutdowns at specific momentum of memory operations to defeat the mechanism designed to protect against the two above cases. Specific conditions can be triggered with a *Proxmark3* and the corresponding scripts [4]. The most prevalent work on tearing was started in 2019 by P. Teuwen and C. Herrmann in Quarkslab's blog [5] after application to find multiple vulnerabilities. Their work was compiled and published in [6] while also providing support for easing *tear-off* operations in the *Proxmark3* software stack [7].

3 EEPROM Mechanism and Tearing

The main type of memory used in RFID technologies is Electrically Erasable Programmable Read-Only Memory (EEPROM). When doing a memory operation, this type of memory is erased/emptied by setting all memory cells to an empty state. Then, bits are filled/programmed to set the desired value. Note that erased/empty state and programmed/filled state are respectively logic 1 and logic 0 in our examples and targets. When reading an EEPROM cell, its voltage is compared to a reference value which is then associated with logic 1 or 0, but in reality values held by EEPROM cells are analogic.

Hardware and software tools — Tearing operations in this paper are performed using a *Proxmark3* with the tooling provided by [6]. This allows us to define a timing in μs before cutting power to the tag. By increasing this timing, we can influence the power level induced into the cells. With each increase we go further into the memory operation before its interruption.

Properties related to tearing and EEPROM are analyzed in [6], those relevant for our exploit are summarized below.

Weak bits — When the tag's power delivery is shut down during memory operations, undesired behavior might arise. EEPROM erases/empties the cells (which have an associated logic of 1) in batch before programming/filling them individually (each obtaining an associated logic of 0). Interrupting this flow of operations enables the creation of intermediary states. The power level of these bits influences the way they are logically evaluated as they insufficiently express the desired bit value. Such bits are called "Weak bits".

Distance dependency — Adding distance between the card and reader will influence the evaluation threshold (it is speculated that it is due to a lower operating voltage in the card due to a low coupling between card and reader) and increases the probability of a weak bit being read at 1.

Biased bits — With the chip manufacturing process, each memory has its own access time. During the manufacturer wafer test, the dispersion of this electrical characteristic is controlled to remain in the boundary of the card's specifications. However, this difference between cells makes the bit flip duration unique per cell.

Progressive tearing — Repeating a write operation with the same tearing delay can see its effects stack on EEPROM cells. Thus, when tearing a write operation, it is better to re-use the same delay before increasing it. This methodology allows us to very slowly step into a write operation and provide better control when shaping weak bits. We refer to this procedure as "progressive tearing (value)". This operation is stopped when an intermediary state between the value of origin and "(value)" is read back.

4 Target properties

All tests were performed on ST25TB512-AT tickets' ICs using various RF front ends. These IC's are part of the ST25TB family (see Figure 1). In this family of cards, two 32-bit monotonic counters located at blocks 5 and 6 are designed to only allow their contained values to decrease. Any update to the values stored in these blocks must be lower than the ones they previously contained. Both blocks are independent from one another, each respectively containing values ranging between 0xFFFFFFFE and 0 (block 5) and 0xFFFFFFFF and 0 (block 6) [8].

Address	ST25TB04K	ST25TB02K	ST25TB512-AC	ST25TB512-AT
[0:4]	Resettable OTP			Lockable EEPROM
[5:6]	Monotonic Counters			
[7:15]	Lockable EEPROM			
[16:23]	EEPROM	EEPROM		
[64:127]				
255	System OTP bits			
UID0 UID1	64 bits UID ROM D0 02 1F + serial	64 bits UID ROM D0 02 3F + serial	64 bits UID ROM D0 02 1B + serial	64 bits UID ROM D0 02 33 + serial

Fig. 1. ST25TB family

Note that through this paper, and to ease legibility we will display only two bytes out of the 4 in the original counter. All notations have higher order bytes/bits on their leftmost side. How blocks provide the monotonic property of the counter is not described in the data sheet. However, two elements helped initiate our research.

No canary — In [6], a counter technology made by NXP was defeated. In this implementation, a reserved byte indicated that a tearing event occurred when its value was different than 0xBD (serving the purpose of a canary) [3]. Like all standard blocks, 4 bytes are available for writing, making it unlikely that a part of these blocks is reserved to act to that effect.

OSINT — A patent registered by STMicroelectronic [9] in 2019 described a method for decreasing the value of a counter on a chip of which the power supply is controlled by the user. This patent aims to provide a tear-off resisting mechanism using two sub-counters. When reading, the lowest sub-counter value was used. When writing, the highest value was overwritten.

While both of the previous elements provide hints towards what mechanisms to expect, we wish to confirm them using the following observations.

OBS:1: Shadow counters —

```
1. Min  Range & Write                    (11111111 00000000)
2. Min  Range & Progressive Tearing  (11111110 00000000)
3. Min  Range & Read Output ->        11111110 10101010
```

When progressively tearing during a write operation, 0's overwritten by 0's can flip to one. Despite this, no value increase occurs.

This confirms that an erasing of each cell happened. It also informs us that the block we are shaping with our progressive tearing only becomes the counter's value once it is inferior to the original value.

OBS:2: Block overwriting conditions? — Bits at 1 will never flip when overwritten by a 1. Bits at 0 will flip during progressive tearing no matter what new value is affected.

```
1. Min  Range & Write                    (11111111 00001111)
2. Min  Range & Progressive Tearing  (11111110 00000000)
3. Min  Range & Read Output ->        11111110 00000000
4. Max  Range & Read Output ->        11111111 00001111
5. Max  Range & Write                    (11111111 00001111)
6. Min  Range & Read Output ->        11111110 00000000
7. Max  Range & Read Output ->        11111111 00001111
```

Here, we set up the counter with a known value (line 1), then set a weak bit at position 9 (line 2). By playing with distance (line 3,4), we control the read value of the bit at position 9. In other terms, we can now decide to read either CNTA or CNTB as we can influence the value of a high order bit.

From this we can confirm that it was the shadow counter holding the maximum value that was overwritten (line 2), otherwise we would not read `11111111 00001111` at the maximum distance.

Another crucial property appears when we try to write either the value `11111111 00001111` or higher (line 5). Irrelevant of bit 9 being interpreted at 1, this value is overwritten neither in `CNTA` nor `CNTB`. This effectively prevents us from replacing a weakly set value by its strongly set counterpart. This also excludes more simple and effective exploit strategies.

paragraphOBS:3: Timing oddity While we think that our understanding of the counters behaviors is correct, when testing the functionalities of the ST25TB512, timings of bits flips are oddly quick in comparison to "standard" blocks.

As described in [6]: “Content dependency” the content of a block may influence its erasing speed, other factors also play a role in this (temperature / distance), those factors may introduce 100 μs delay at most. However, counters block present a delay for programming in the [150;220] μs range against [2150;2200] μs for standard blocks (timing may vary depending on Proxmark hardware and/or cards).

We don’t have an explanation for this discrepancy between standards and counter EEPROM cells but it need to be taken into consideration as to not use the timing of standard blocks as a reference point for testing or exploiting counter blocks.

OBS:4: Chained’s operations pitfall Another timing issue that should be considered is the delay between operations, when manipulating weak bits (writing and reading) too short of a delay between operations can bias the outcome.

For instance, in our exploit algorithm we sometimes use multiple reads to check the probability of reading a weak bit at 1 or 0. Without added delay, all but the first read will be biased toward reading value at 0, writing operations are also impacted and may lead to difficulty when setting weak bits.

It is not known to us how much hardware both of the reader and card play a role here but we recommend not neglecting this point in the frame of this exploit or others.

Even if not explicitly said in the description of our exploit mechanisms, it is advisable to have the shortest delay between instruction when trying to detect early bit flip to 0 when setting up a weak bit. And on the contrary, we will want to wait a few milliseconds when trying to interpret weak bit as 1.

Counter Behavior : Considering the sus-mentioned observations, we confirm that the counter is composed of two sub-counter `CNTA` and `CNTB`, which operates as follows (see Figure 2):

— When reading, the counter will return the minimal value between `CNTA` and `CNTB`.

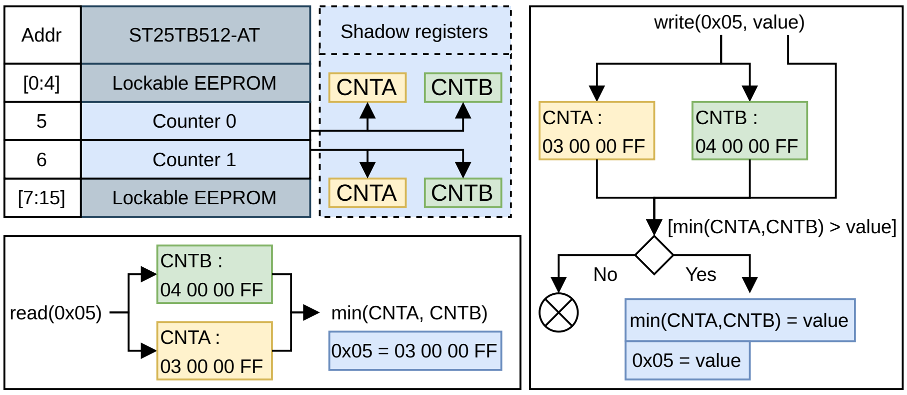

Fig. 2. ST25TB counter architecture

— When writing, the counter will overwrite the maximal value between `CNTA` and `CNTB` only if the value is strictly inferior to both `CNTA` and `CNTB`.

5 Attack

Quarkslab paper [6] also presented an attack on monotonic counter, however, due to the implementation differences between the counters, we cannot apply their already documented methodology. A comparison of how our methods diverge is detailed in the online version of this paper.

We present two complementary strategies for reverting a counter's state.

Both strategies' examples assume the following starting values:

— `CNTA = 11111111 00000000    CNTB = 11111111 11000000`

In our examples `"?"` are weak bits. They are interpreted at 0 at a minimal distance and 1 at a maximal distance.

Strategy 1: — This strategy relies on only 1 bit of high order above all bits we wish to set back at 1 (we label this bit as n, bit 9 in our example).

Example (Illustrated in Figure 3):

— `Min  Range & Progressive Tearing  (11111110 11111111)`

— `CNTA = 11111111 00000000    CNTB = 1111111? 11111111`

— `Min  Range & Progressive Tearing  (11111110 11110000)`

— `CNTA = 1111111? 1111????    CNTB = 1111111? 11111111`

— `Max  Range & reinforcement`

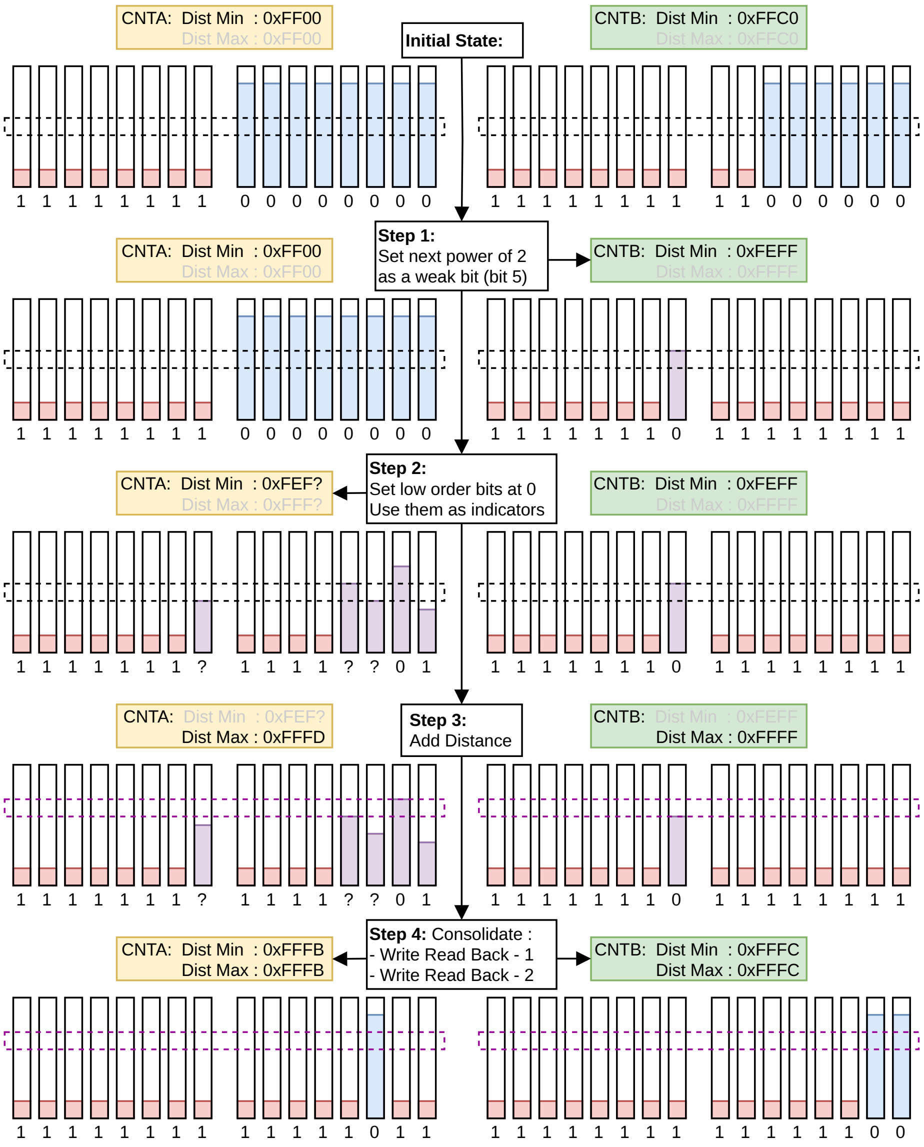

Fig. 3. Example of EEPROM evolution during strategy 1

First, we make a progressive tearing write operation with bit n at 0 and the remainder at 1. This leverage bit n to have a fully reset CNTB at max distance, or to write in CNTA at min distance. We then write the same value but with low-order bits at 0. Setting at least one low-order bit at 0 is an obligation for our write operation to be effective (see OBS:2). As these low-order bits is our sole indication to assert if bit 9 is starting to become weak or no, the more we have the earlier we will be warned.

If we write too few low-order bits at 0, we risk having a retarded indication of bit 9's becoming weak, thus ceding our ability to influence its value by increasing distance from the emitter.

In both strategies, "reinforcement" refers to decrementing twice the read-back value. These two decrements impose that the two least significant bits are respectively set to 1. The aim of this operation is to avoid getting probabilistic or distance dependent results. Additionally the second decrement can be teared to fully reset the other sub-counter, providing a final value of 0xFFFFFFFF - 1.

Strategy 2: — In this strategy we need two bits of high order n and $n + 1$.

Example (Illustrated in Figure 4):

- `Min  Range & Progressive Tearing  (11111110 11111111)`
- `CNTA = 11111111 00000000    CNTB = 1111111? 11111111`
- `Min  Range & Progressive Tearing  (11111101 11111111)`
- `CNTA = 111111?1 11111111    CNTB = 1111111? 11111111`
- `Max  Range & reinforcement`

Similarly to *strategy 1*, we set a bit n as weak and the remainder at 1. We then set bit $n + 1$ as weak; here we don't need the low-order bits as indicators because as soon as bit $n + 1$ become weak, a new value is read back.

Tradeoff — Both strategies rely on setting a bit in each sub-counter as leverage, increasing distance then allowing for a near full-counter reset.

Strategy 2 allows for easier reset of low-order bit but consume 2 high-order bits at each attempt.

Comparison with Quarkslab exploit on NXP counters Quarkslab presented before us an exploit on a counter technology, this one developed by NXP. The following is summarized from Quarkslab paper, a more thorough explanation of their discoveries and exploit is available in their paper [6]

The counter technology defeated in their work is characterized by the following:

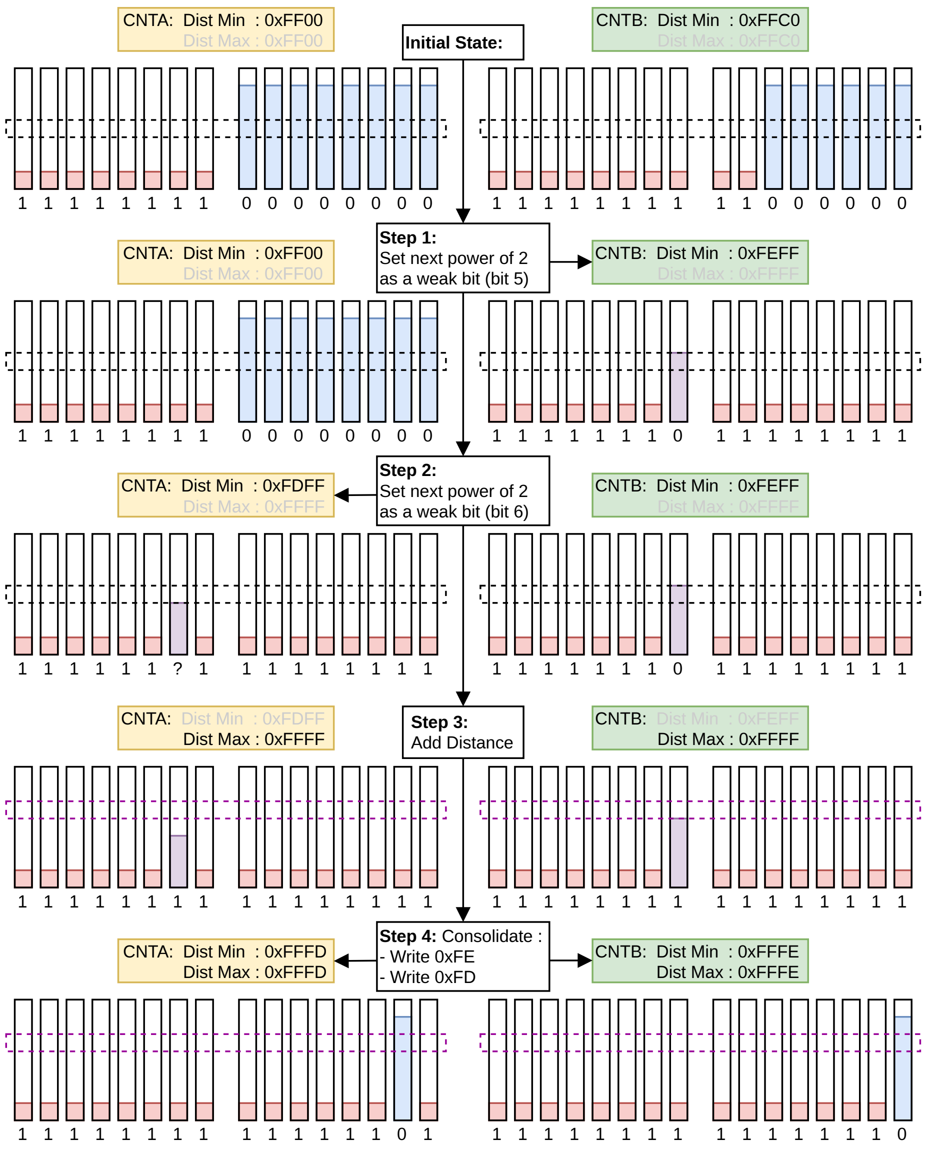

Fig. 4. Example of EEPROM evolution during strategy 2

- Counter blocks are monotonic, only increasable.
- Like in our case two shadow registers are in use.
- Both shadow registers hold a 24-bit value.
- The remaining 8 bits are used as a canary, canary value should be 0xBD, otherwise the block will be flagged as teared.

Counters are increased by issuing a `INCR_CNT(value)` which will increase the counter by the value given in parameters. The following seems to happen when issuing a `INCR_CNT(value)`:

- Find the current counter value: if one flag is corrupted, read the other counter slot, else consider the highest of the two counters slots;
- Add to the highest counter the value sent as parameters of `INCR_CNT`;
- Erase the other counter slot, set both the flags and the counter value to zero;
- Write the erased counter slot with the new value and the flag 0xBD.

It is possible to propagate the highest valid value (which would be returned by a write) to both shadow registers by issuing a `INCR_CNT`, this is referred to as a dummy increment.

When issuing a `READ_CNT` an equivalent to the first step of a `INCR_CNT` decide the value read back:

- Find the current counter value: if one flag is corrupted, return the value of the other counter slot, or else return the highest of the two counter slots.

The command `CHECK_TEARING_EVENT` read back the value of a corrupted canary if any, it return the canary of the smallest shadow register values otherwise.

In summary, the steps for reverting the value of a counter take the following steps:

- Like for our exploit a high order power of 2 is leveraged to reset a subcounter.
- This controlled value is propagated in both sub counter through the use of dummy increments.
- The original value reliant on a weak bit is refreshed through another dummy increment.

Both Quarkslab and our exploit leverage a power of two weakly set for reverting both shadow counters. The difference of implementation in both counters technology implied that we could not leverage dummy increments to propagate an advantageously interpreted value to the whole counter. As a replacement we use two weakly set powers of two, alternatively we use only one power of two and low-order bit as indicators of an exploitable memory state.

Due to the absence of dummy increment functionality, it is harder when using our exploit to revert the counter fully, one or two low-order bit are necessarily set at 0 for ensuring a stable read of the manipulated counter value. Remediation for both our exploits and exploits developed by Quarkslab are the same.

6 Limitations and impacts

The needed attacker hardware is easily available : a *proxmark3 easy* can be acquired for around 40 euros. The time required to perform a full reset is variable and depends on the success rate of the attack. Usually, it takes half a minute up to a few to perform the full attack. Mixing both presented strategies and depending on the availability of high-order bit at 1 available, an attacker can reliably control a counter value. As a potential mitigation, and like described in [6], instead of decreasing the binary encoded number, it is recommended to decrease the number of bits at 1 starting from those of high order. This means that the counter range goes from 2^{32} to 32 values. Such a countermeasure is efficient against Amossys' manipulation for the targeted product and as well as the whole product family. STMicroelectronics recommendations regarding the affected family of cards were updated after Amossys' finding [10].

Our work has provided evidence of the sensitivity of another implementation of monotonic counter to *tear-off* attacks. In general, if no countermeasures are applied, different manipulations exist to derail applications relying on monotonic counter mechanism. As presented in Figure 5, in the context of travel tickets a user could:

1. read the state of the ticket,
2. use the card ticket normally,
3. reset the counters to their maximal value,
4. restore the ticket to its *previous* state, including the counter value and the corresponding certificate.

This gives the user, the opportunity to travel without limitation for the cost of a single ride as illustrated. If the ST25TB-based ticket's integrity is based solely on the monotonic counter, the system is untrustworthy.

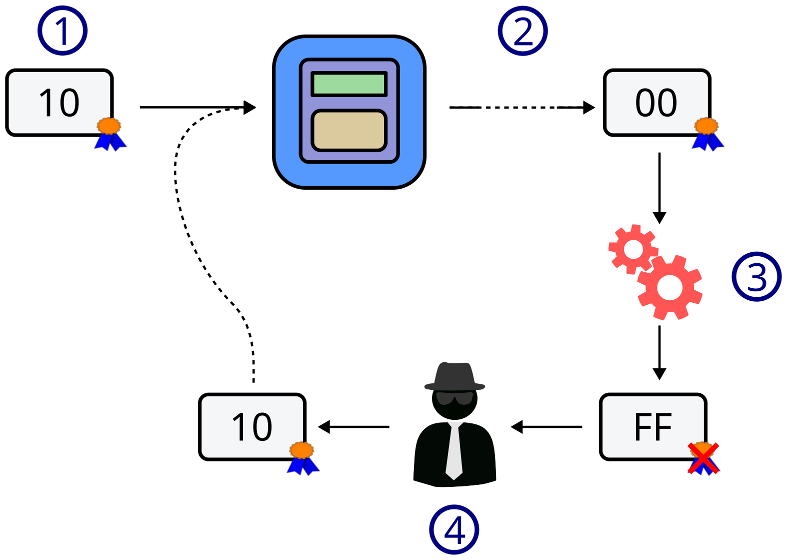

Fig. 5. Attack scenario

7 Conclusion

Our work describes a tearing vulnerability found on ST25TB-based monotonic counter. This result enables attackers to reset the monotonic counter and break the broader system relying on this mechanism. This unwanted behavior is confirmed to be functional on the whole ST25TB family by the manufacturer. STMicroelectronics's updated their recommendations regarding the ST25TB family usage after being notified [10]. Amossys as an ITSEF laboratory followed the ethical vulnerability disclosure process with the manufacturer.

Methodology — The vulnerability was discovered while looking to reproduce [6]. Once identified, contact was established with STMicroelectronics's PSIRT in May 2023. Then, we started to discuss the implications of the vulnerability.

Acknowledgment — We want to thank Philippe Teuwen and Christian Herrmann for their extensive work [6] and for the tools they provided to the community. Without their contributions this work would not have been possible. We particularly want to thank Philippe Teuwen for his technical feedback on the paper and Steven Redfern for his feedback on

the form. We hope that other people will continue to contribute to this field of research.

References

1. Benjamin Delpy. ST25TB series NFC tags for fun in French* public transports, 2023.
2. Michael Hutter, Jörn-Marc Schmidt, and Thomas Plos. Rfid and its vulnerability to faults. In *Cryptographic Hardware and Embedded Systems–CHES 2008: 10th International Workshop, Washington, DC, USA, August 10-13, 2008. Proceedings 10*, pages 363–379. Springer, 2008.
3. NXP. MIFARE Ultralight EV1 tag Datasheet. `https://www.nxp.com/products/rfid-nfc/mifare-hf/mifare-ultralight/mifare-ultralight-ev1:MF0ULX1`, 2012.
4. R. Delion P. Granier, J.J. Marty. Tears For Tears. `https://gitlab.com/SiliconOtter/tears4fears`, 2024.
5. C. Herrmann P. Teuwen. EEPROM: When Tearing-Off Becomes a Security Issue . `https://blog.quarkslab.com/eeprom-when-tearing-off-becomes-a-security-issue.html`, 2019.
6. C. Herrmann P. Teuwen. EEPROM: It Will All End in Tears. `https://www.sstic.org/media/SSTIC2021/SSTIC-actes/eeprom_it_will_all_end_in_tears/SSTIC2021-Article-eeprom_it_will_all_end_in_tears-herrmann_teuwen.pdf`, 2021.
7. C. Herrmann et al P. Teuwen. Iceman Fork - Proxmark3. `https://github.com/RfidResearchGroup/proxmark3`.
8. STMicroelectronics. ST25TB series NFC tags Datasheet. `https://www.st.com/en/nfc/st25tb-series-nfc-tags/documentation.html`, 2016.
9. STMicroelectronics. Method for modifying a counter value of a counter of an electronic chip (Patent). `https://patents.google.com/patent/FR3103925B1/en`, 2019.
10. STMicroelectronics. Best practices for security and privacy with ST25 NFC / RFID Tags. `https://www.st.com/resource/en/application_note/an5493-best-practices-for-security-and-privacy-with-st25-nfc--rfid-tags---stmicroelectronics.pdf`, 2024.