Lehrstuhl für Sicherheit
in der Informationstechnik

Exploiting Bus Communication to Improve Cache Attacks on Systems-on-Chips

Johanna Sepúlveda, **Mathieu Gross**, Andreas Zankl and Georg Sigl

TECHNISCHE UNIVERSITÄT MÜNCHEN

Fraunhofer AISEC

Outline

- Introduction
- System-on-Chip (SoC) description and threat model
- AES Access-driven attack on SoCs:
  - Simple attack
  - Bus-enhanced attack
- Countermeasures
  - Shuffling
  - Constant accessed cache lines implementation
- Conclusion

Microarchitectural attacks today…

Powerful attacks: Recover of keystrokes, inference of cryptographic keys, stole user private data.
New attack methods: same-core (2005), cross-core (2014), javascript (2015)

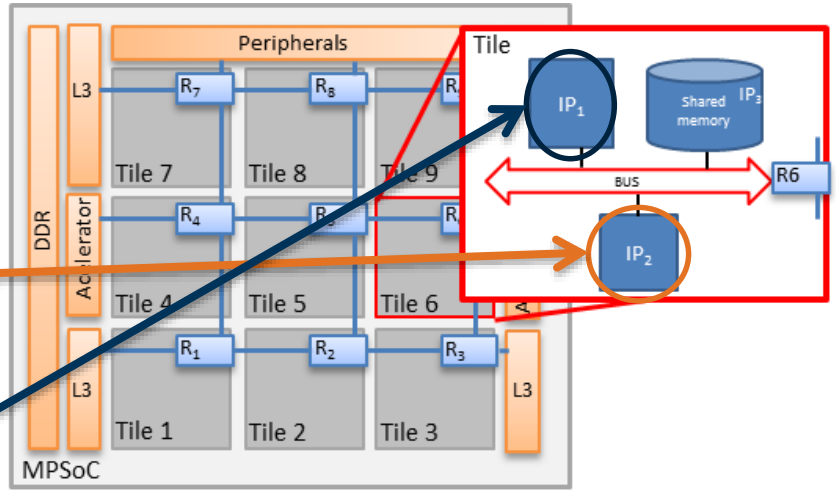
Attack characterisation

Precondtion
- SoC connected to the Internet

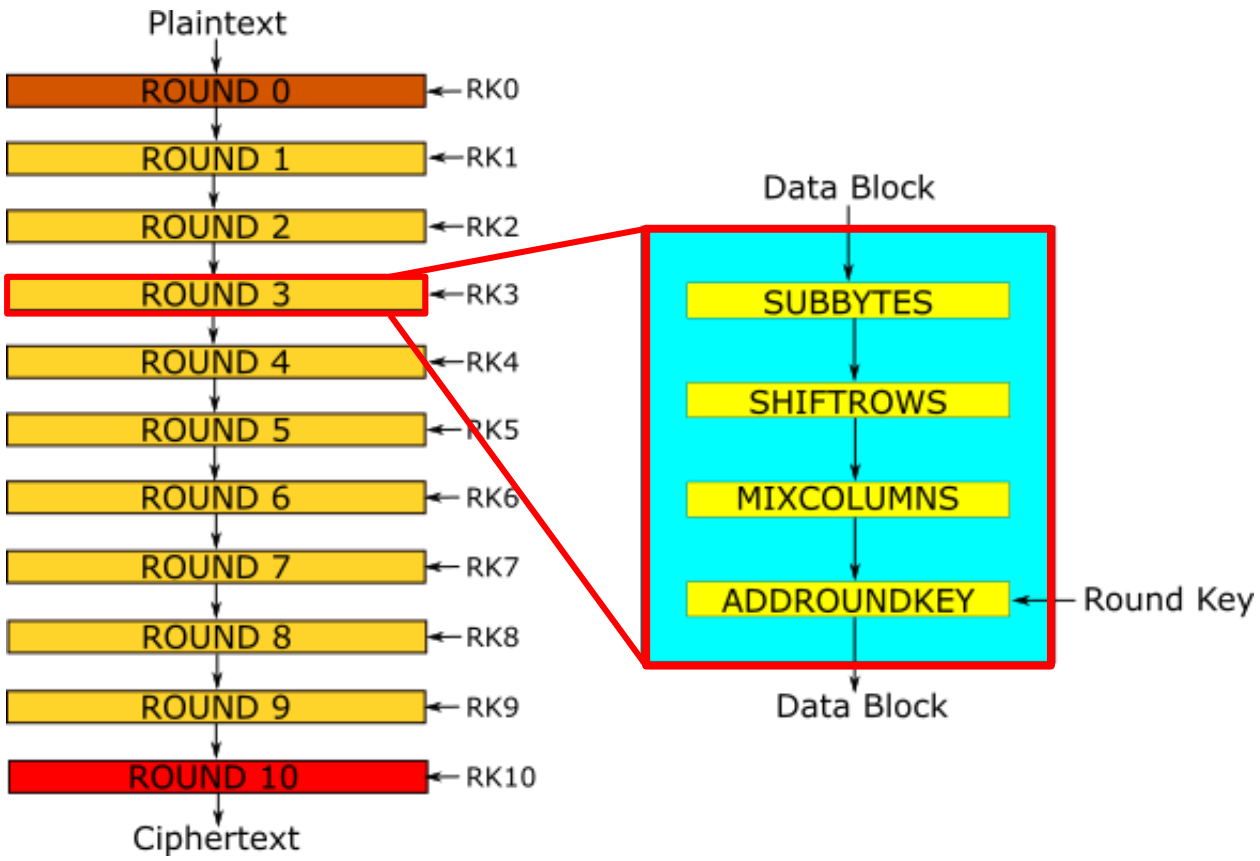
Attack Method
- Malware downloaded from the internet
- Change of the behavior of one IP core

Vulnerability
- Not protected software AES implementation

Effect
- Leakage of key bytes through cache accesses

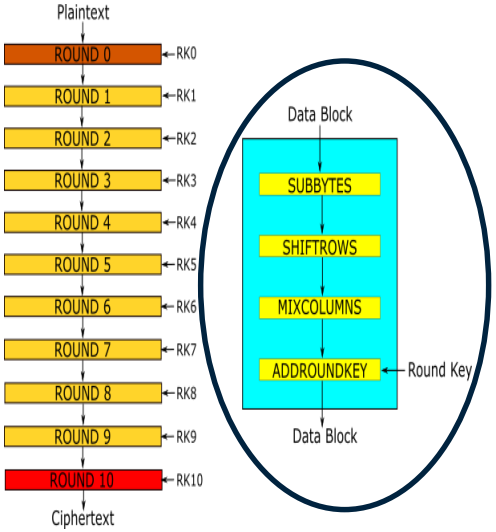


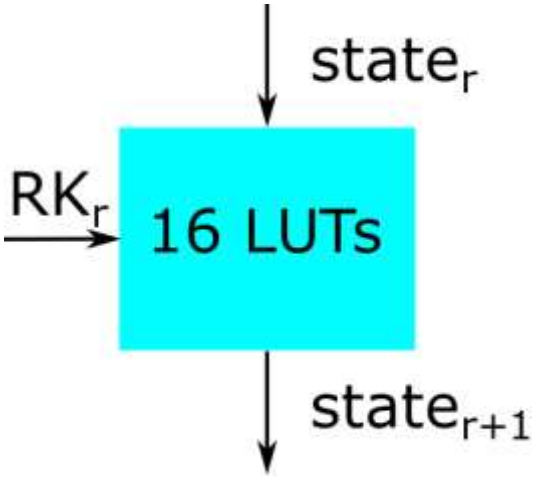
The AES algorithm



AES T-Tables implementation

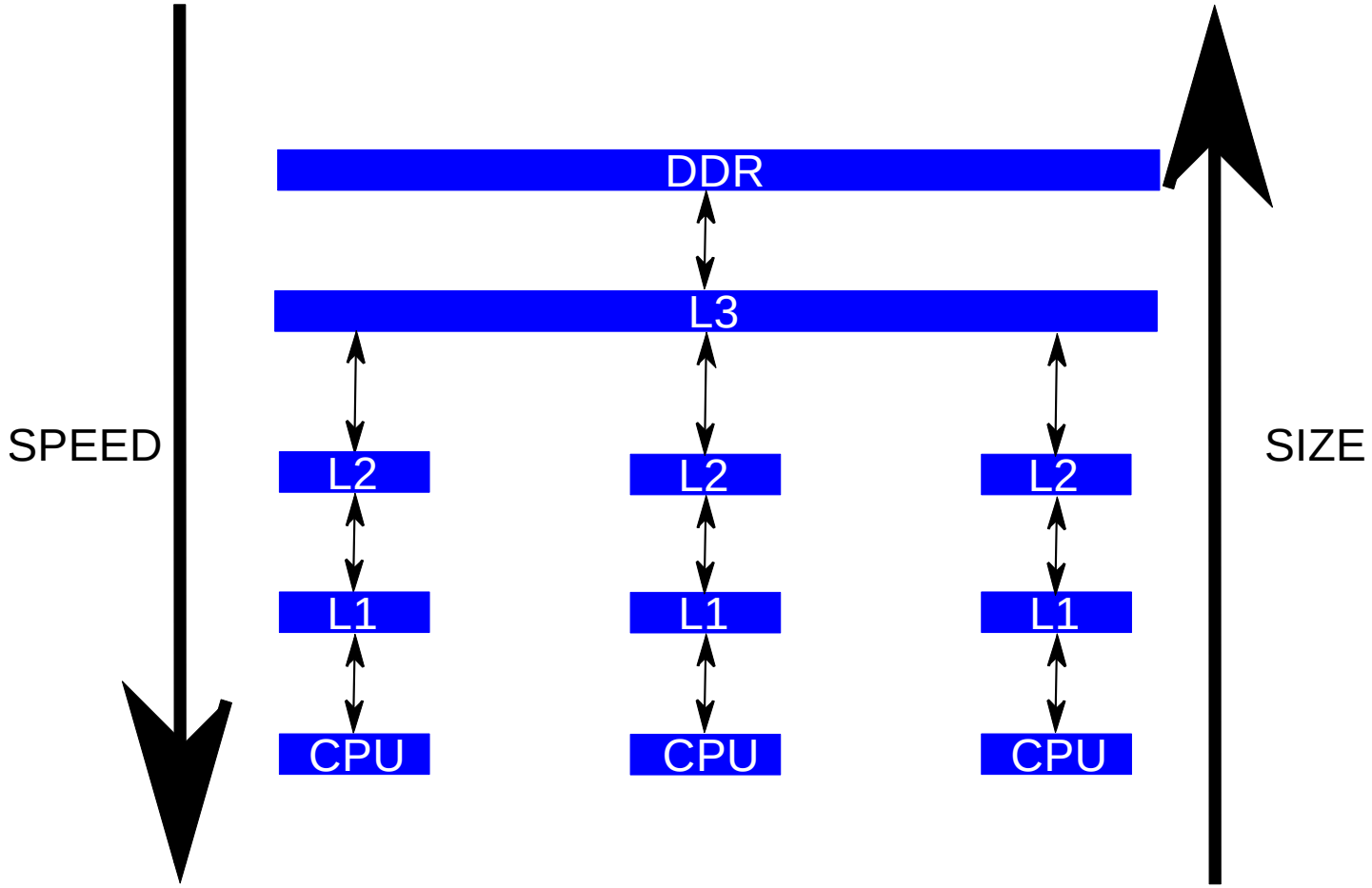
- Performance-oriented software implementation of AES
- Use 5 Tables of 1 kB containing 32 bits words: T0,T1, T2,T3, T4





- In the first round, the accessed indices are simply: $x_i^0 = PT_i \oplus KEY_i$

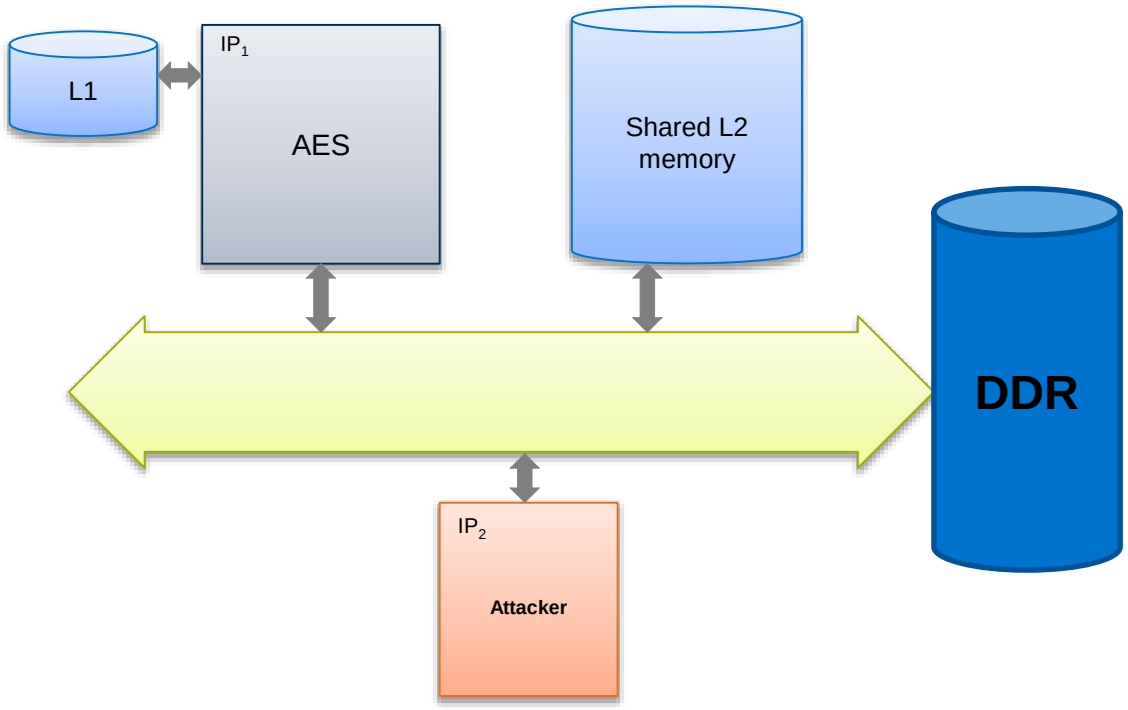
Memory hierarchy in a SoC



The cache side channel leakage

- Exploit key dependent memory accesses of cryptographic primitives
- Information can be obtained from observed hits and misses occuring during cipher execution
- **Access-driven attack**
  - Exploit the information obtained from the accessed cache sets during the cryptographic execution

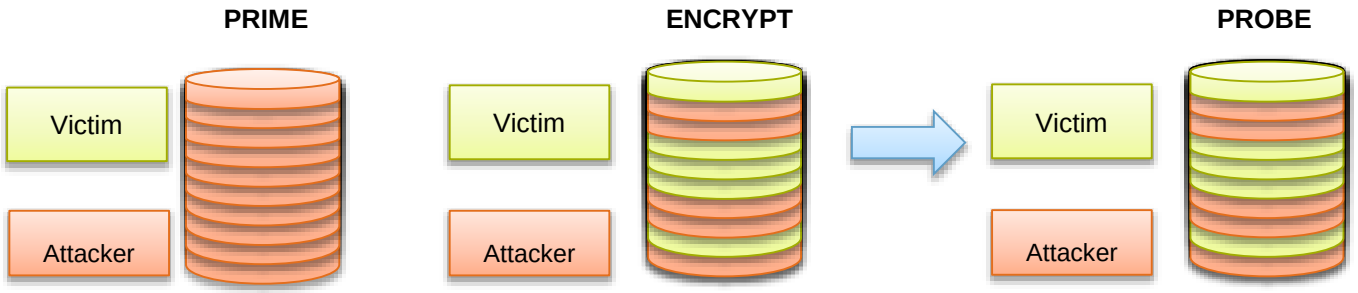
Measurement set up



Attack assumptions

1. Knowledge of the plaintext used for each encryption
2. The attacker can trigger an encryption
3. The L2 cache is inclusive of L1 cache
4. The attacker knows the characteristics of the L2 cache (size, associativity, replacement policy)
5. The attacker knows the lookup tables cache set mapping

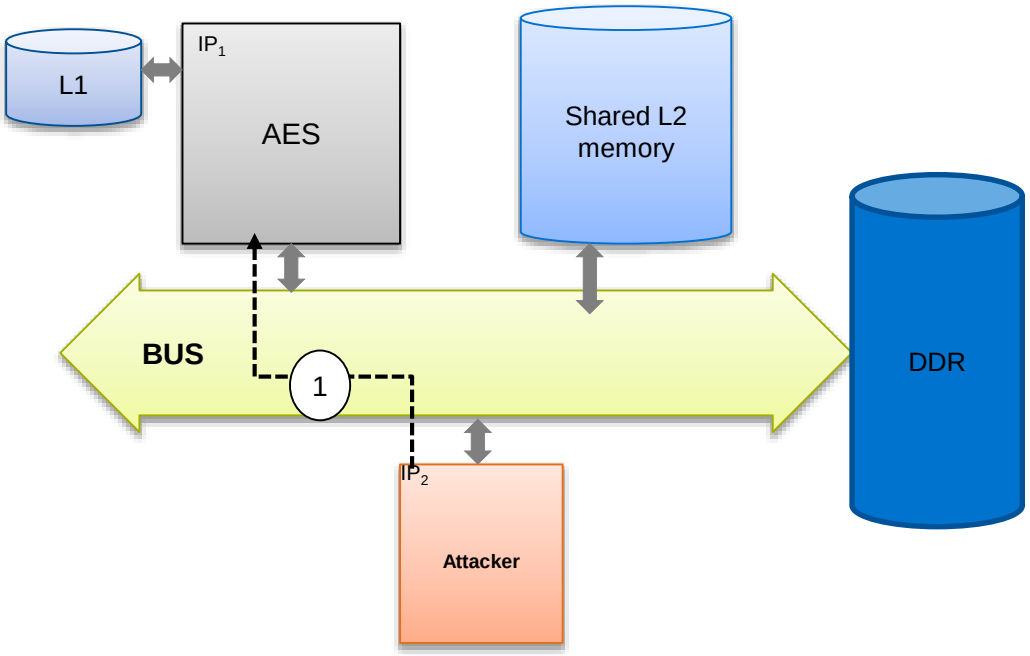
Prime+Probe attack



With a L2 inclusive of L1, the attacker evicts the victim's data from the L1 when priming the L2

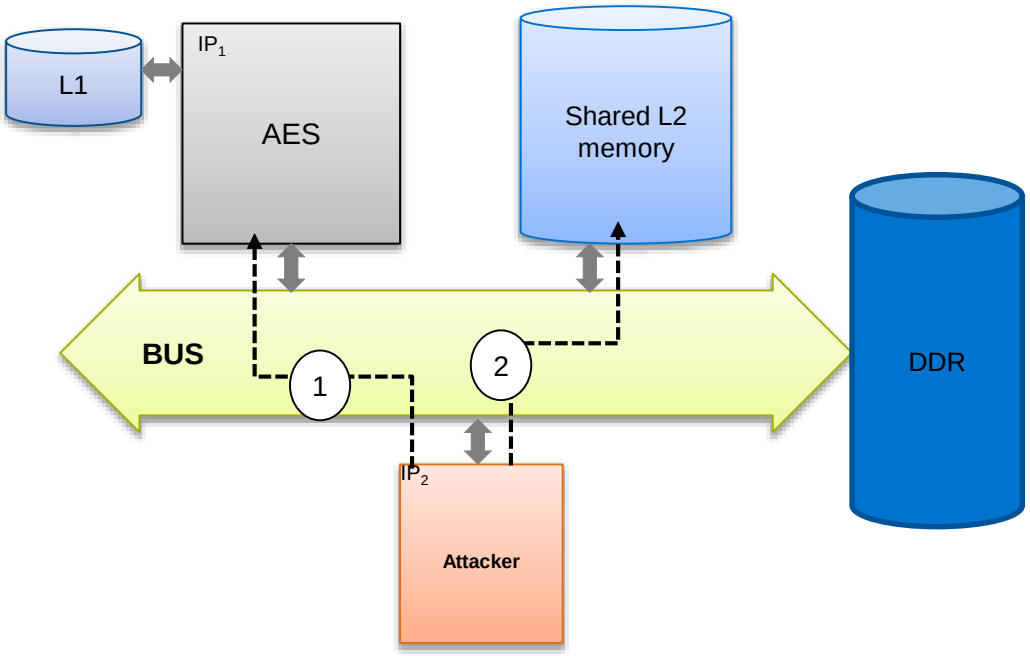
Our approach

Bus-based Prime+Probe attack



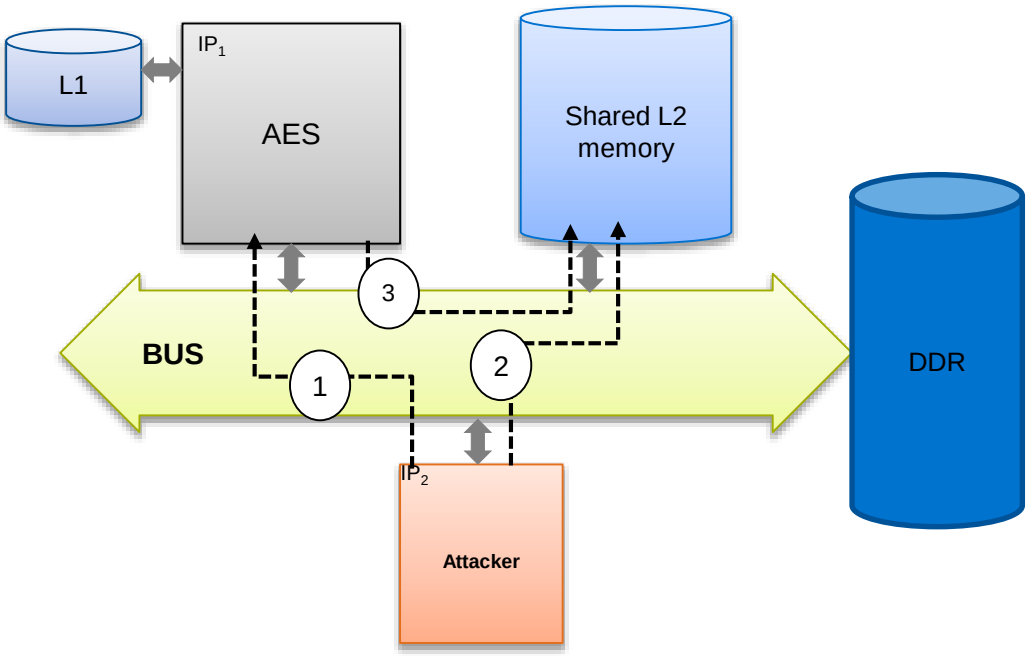
1 Trigger encryption

Bus-based Prime+Probe attack



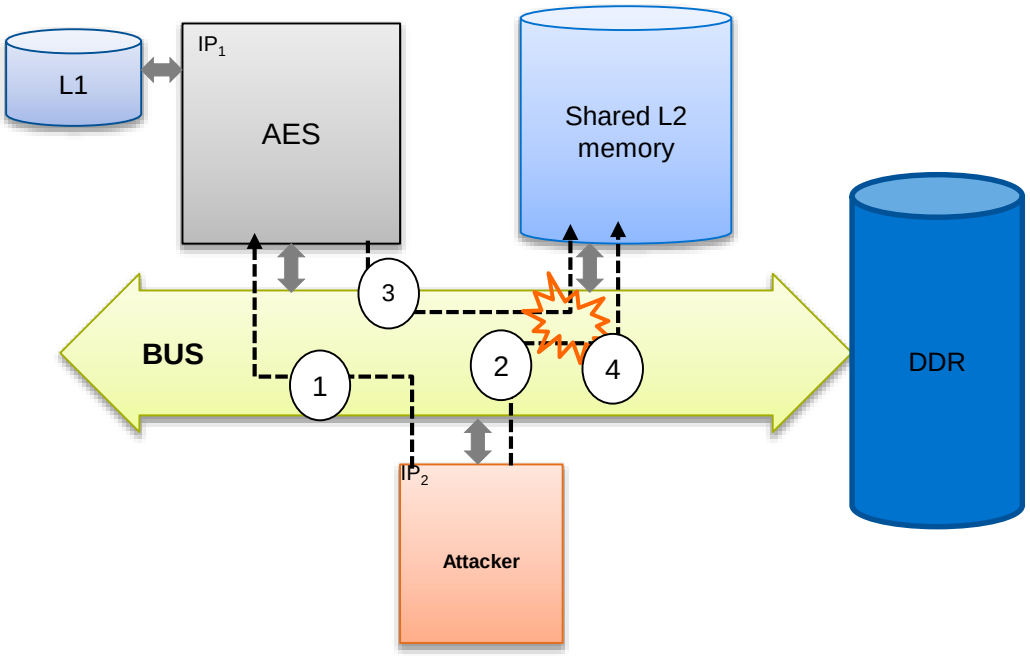
1. Trigger encryption
2. Perform continuously a read request and check completion time

Bus-based Prime+Probe attack



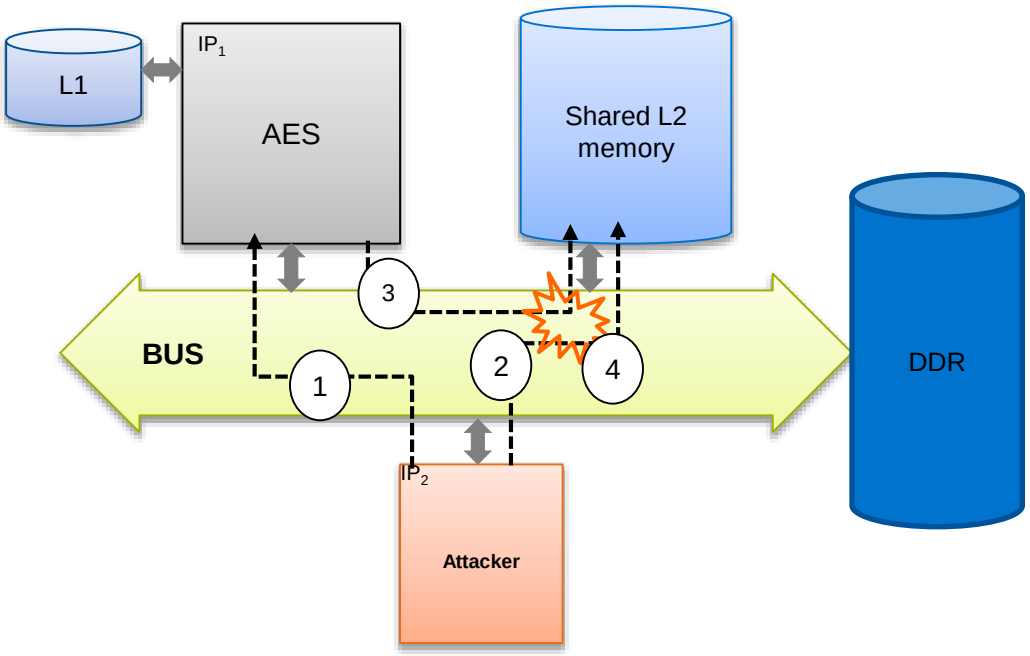
1. Trigger encryption
2. Perform continuously a read request and check completion time
3. T-Tables access from the cryptocore

Bus-based Prime+Probe attack



1. Trigger encryption
2. Perform continuously a read request and check completion time
3. T-Tables access from the cryptocore
4. Increase of attacker's request delay=> Detection of access

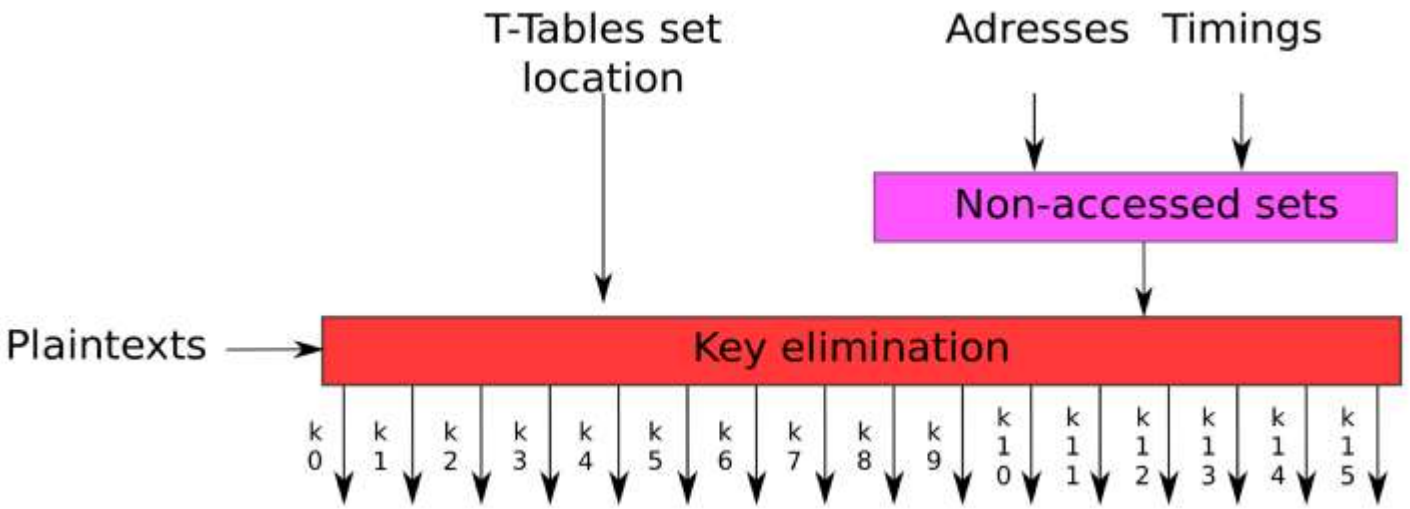
Bus-based Prime+Probe attack



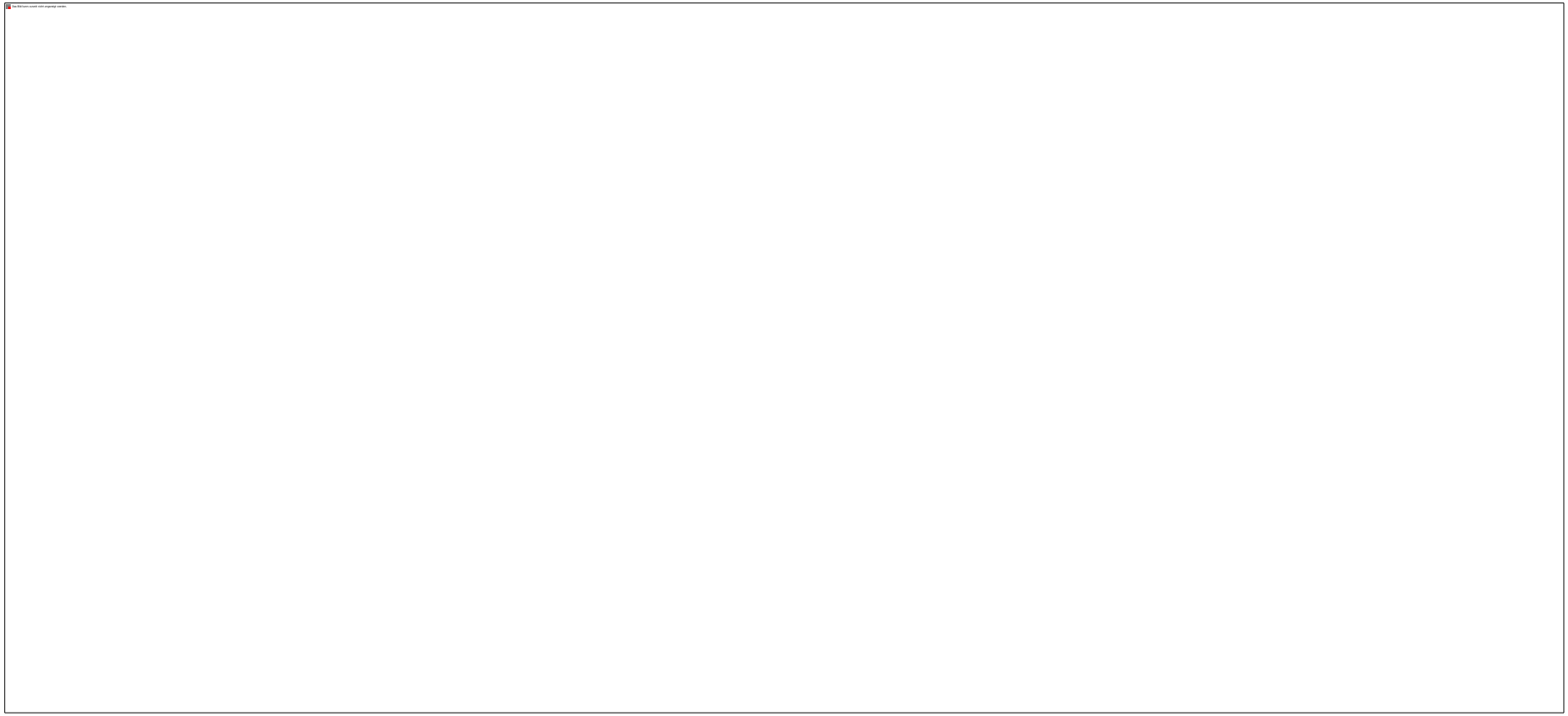
1. Trigger encryption
2. Perform continuously a read request and check completion time
3. T-Tables access from the cryptocore
4. Increase of attacker's request delay=> Detection of access

Probe after the detection of 16th access.

Analysis script



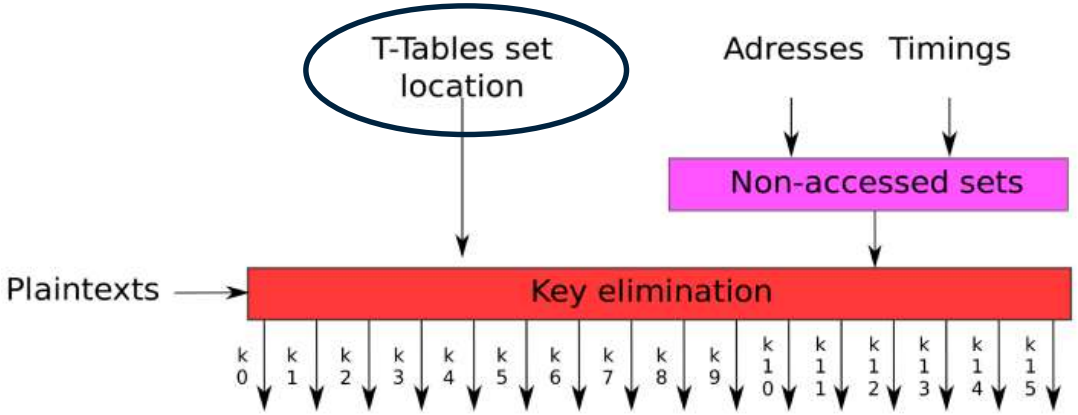
Key elimination



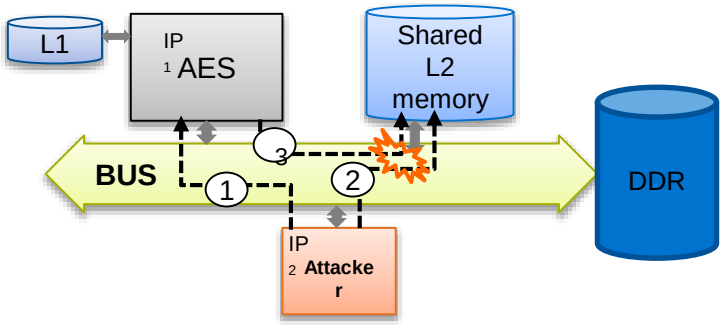
Recovering the non used indices =>
Eliminating wrong key candidates

Challenge of the attack

- Discovery of the cache-set location of the T-Tables: Make the analysis with a varying offset



Experimental setup



- Microblaze as crypto core

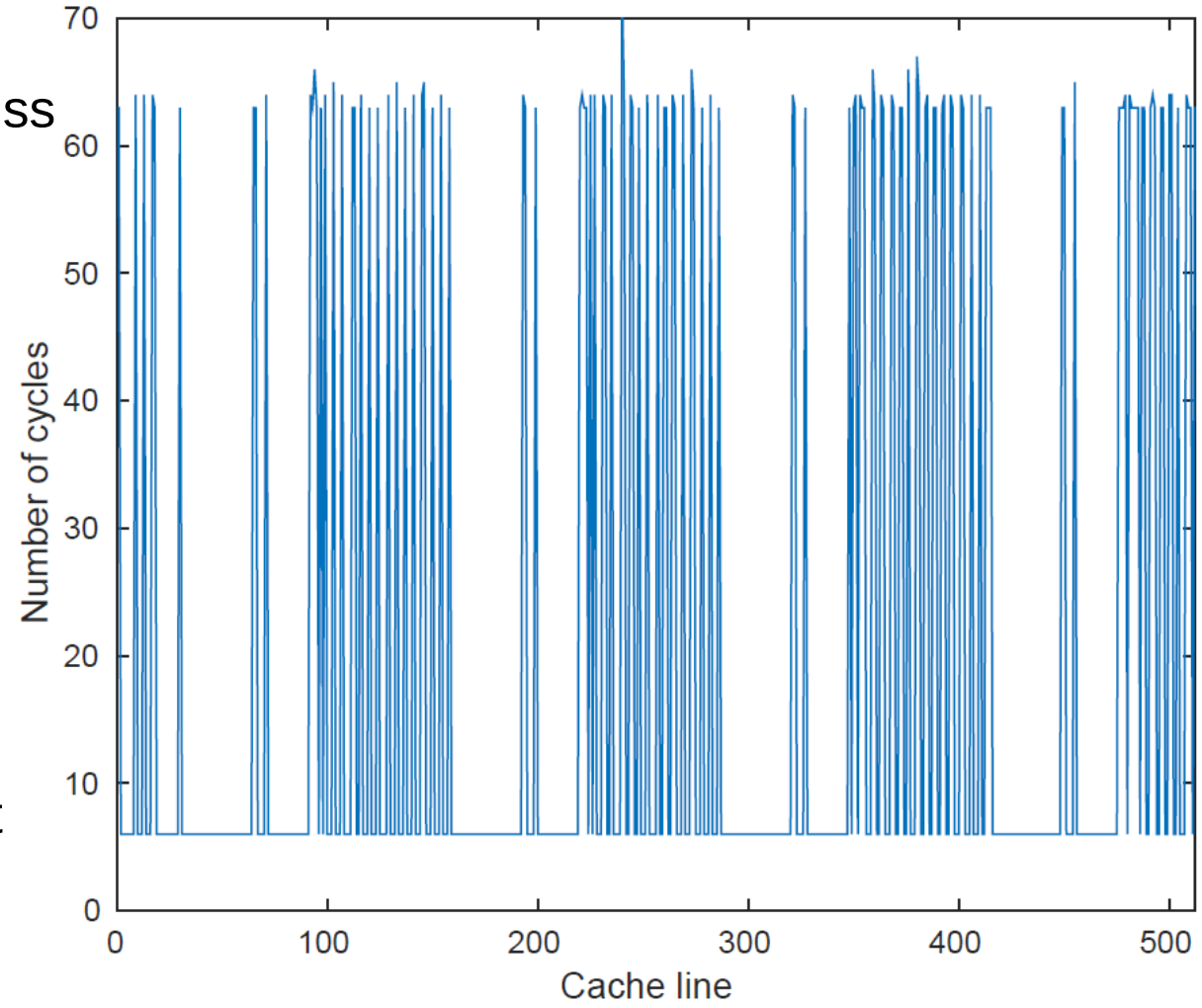
	Private L1 Cache	Shared L2 Cache
Configuration	Direct-mapped	4 ways set-associative
Size	8 kB	32 kB
Replacement policy	LRU	LRU
Inclusiveness	-	Inclusive

- Two Experiments performed:
  - Priming the cache at the end of the encryption
  - Detect the end of the first round through the communication observation and probe the L2 earlier

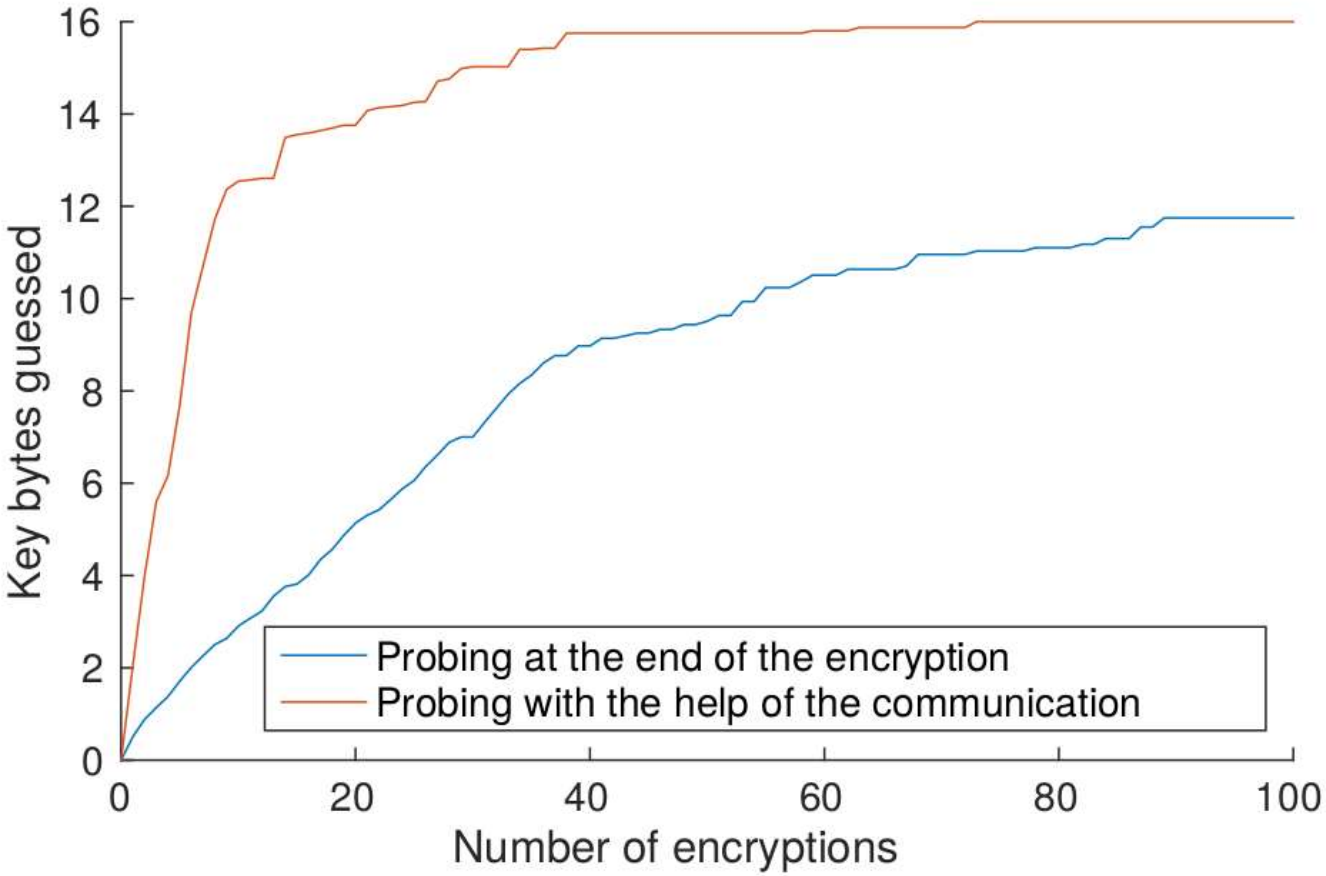
Results: One trace

Cache miss

Cache hit



Efficiency of the attack



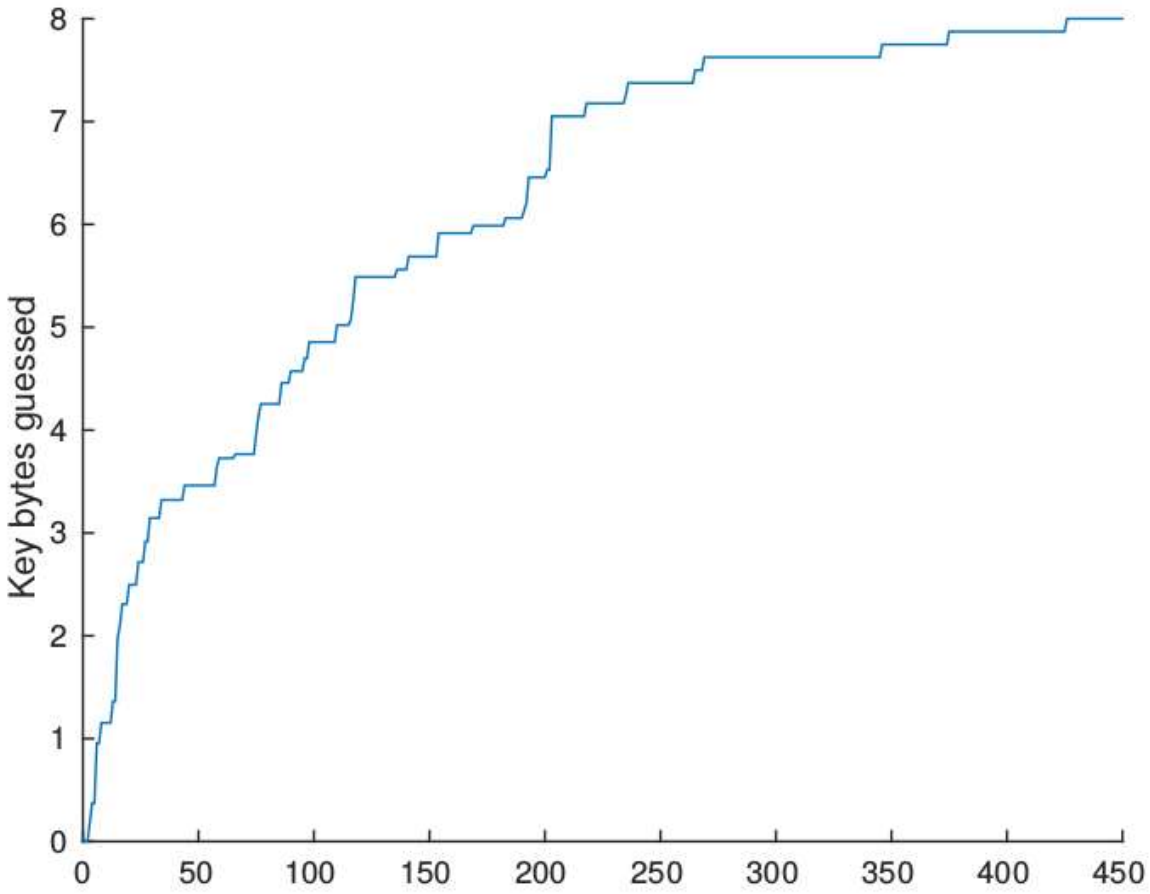
Benefits of the improved attack for alternative AES implementations

- AES can also be implemented with the SBOX (4 cache lines) instead of the T-Tables (16 cache lines per table).
- The T-tables lookups are replaced by SBOX lookups.
- Very few leakage observable in the cache access pattern if probing is done at the end of the encryption.
- Only the first, second and last round need to be protected => Interesting for performance.

- **Efficient countermeasure against classical access-driven attack.**

- **With the communication+cache attack, the traces become exploitable. An attack is then possible !**

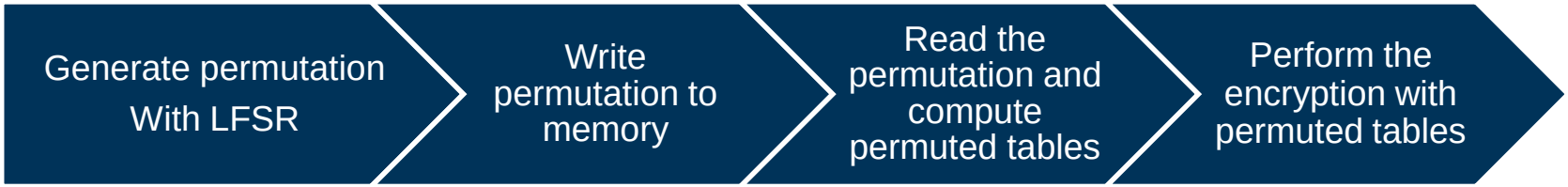
Combined attack on a AES-SBOX implementation



Countermeasures

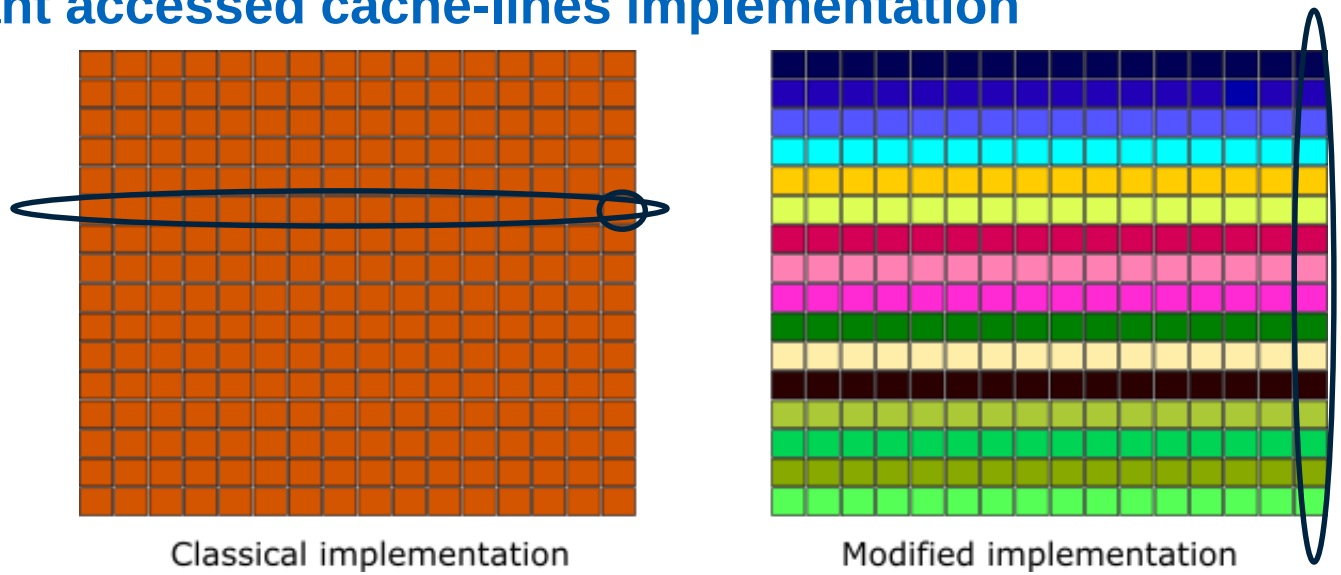
Shuffling of T-Tables

Idea: changing the memory layout of the tables for each encryption makes the attack more difficult



Countermeasures

Constant accessed cache-lines implementation



Classical implementation

Modified implementation

- Modified implementation:
  - Every line contains 2 bits of the 256 lookups
  - 16 T_i-tables for one T-table
  - $T[x] = \sum_{i=0}^{15}(Ti[x] \ll 2 * i)$

- The cache lines that are accessed are constant

Comparison of the countermeasures

Implementation	Speed degradation	Additional features required	Strength
T-Tables	1	-	Broken with 40 traces
SBOX for the outer rounds	1.98	Definition of the SBOX as precomputed table	Breakable with combined attack
Shuffled Tables	4,1	Compute permutation and new tables	No attack tested
Constant accesed cache lines for the outer rounds	3,42	Integrate 80 tables of 64 bytes (only first, second and last round are protected)	Good. Need an attack capable of distigushing a word access within a cache line.

Conclusion

- The communication leakage improves the classical Prime+Probe attack
- T-Tables implementation of AES are not secured
- SBOX only implementation are also threatened by the combined attack
- 2 Possible countermeasures consist in shuffling the T-Tables and using an alternative constant accessed cache lines implementation

Thank you for your attention !

Questions ?

Backup slide

$$T_0(z) = \begin{bmatrix} 02 * S(z) \\ S(z) \\ S(z) \\ 03 * S(z) \end{bmatrix}$$

$$T_1(z) = \begin{bmatrix} 03 * S(z) \\ 02 * S(z) \\ S(z) \\ S(z) \end{bmatrix}$$

$$T_2(z) = \begin{bmatrix} S(z) \\ 03 * S(z) \\ 02 * S(z) \\ S(z) \end{bmatrix}$$

$$T_3(z) = \begin{bmatrix} S(z) \\ S(z) \\ 03 * S(z) \\ 02 * S(z) \end{bmatrix}$$