From New Technologies to New Solutions: Exploiting FRAM Memories to Enhance Physical Security

Stéphanie Kerckhof, François-Xavier Standaert, Eric Peeters

CARDIS 2013 – November 2013

Context

Ferroelectric RAM (FRAM):

- non-volatile RAM using special dielectric material
- Integrated in Texas Instruments microcontrollers

Context

Ferroelectric RAM (FRAM):

- non-volatile RAM using special dielectric material
- Integrated in Texas Instruments microcontrollers

Flash	FRAM
Program memory only	Unified memory
10^5 reprogramming	10^{15} reprogramming
1 page (256 bytes) at a time	1 byte at a time
4,5 ms per page write or erase	a few clock cycles per byte

Context

- non-volatile memory useful for countermeasures needing secure precomputations

Context

- non-volatile memory useful for countermeasures needing secure precomputations
- Two questions:
  - FRAM as a more secure technology against side channel attacks?
  - FRAM as a more efficient way to implement existing countermeasures?

Context

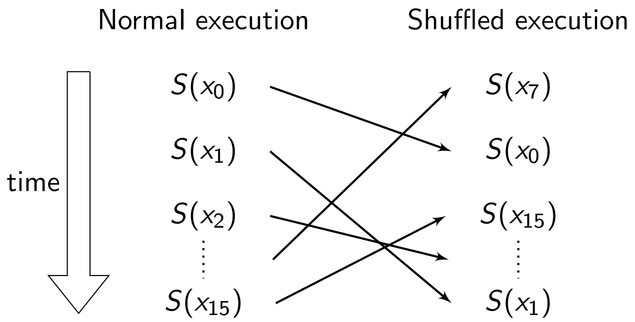
- non-volatile memory useful for countermeasures needing secure precomputations
- Two questions:
  - FRAM as a more secure technology against side channel attacks?
  - FRAM as a more efficient way to implement existing countermeasures?
- We follow the second approach:
  - Improving past results $\rightarrow$ Shuffling
  - Making new results possible $\rightarrow$ Masking with RLUT

Outline

What is Shuffling?

Shuffling: Modifies the order in which independent operations are performed

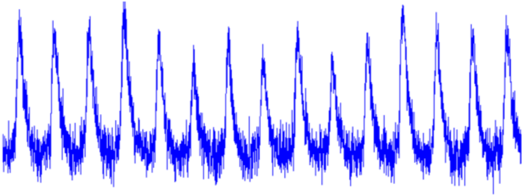
Example:



What is Shuffling?

Goal:

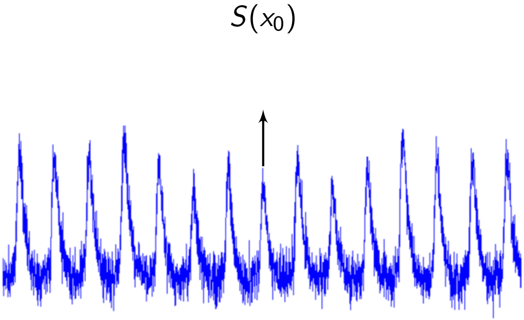
- Spread points of interest over t cycles
- Amplify physical noise by forcing the adversary to combine multiple points



What is Shuffling?

Goal:

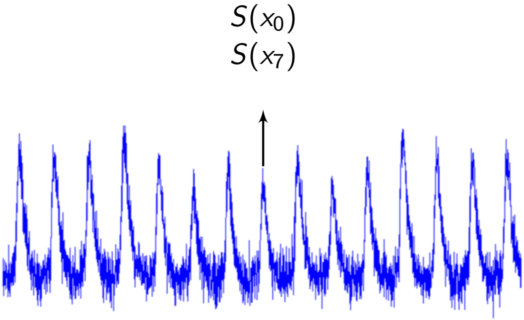
- Spread points of interest over t cycles
- Amplify physical noise by forcing the adversary to combine multiple points



What is Shuffling?

Goal:

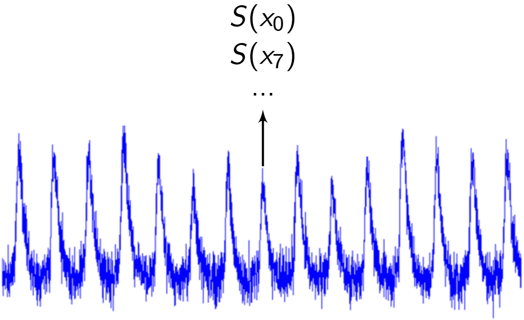
- Spread points of interest over t cycles
- Amplify physical noise by forcing the adversary to combine multiple points



What is Shuffling?

Goal:

- Spread points of interest over t cycles
- Amplify physical noise by forcing the adversary to combine multiple points



Outline

1. Improving Past Results: Shuffling
   - What is Shuffling?
   - Previous Implementation
   - FRAM Implementation
2. Making New Results Possible: Masking with RLUT
3. Conclusion

Shuffling - Randomized Program Memory

Program Memory

1. Shuffle program memory
2. Execute $S(x_0)$
3. Execute $S(x_1)$
4. Execute $S(x_2)$
5. Execute $S(x_3)$

...

Data Memory

Permutation

1. 7
2. 0
3. 15
4. 3

...

Proposed by Veyrat-Charvillon et al. at Asiacrypt 2012

Shuffling - Randomized Program Memory

Program Memory

1. Shuffle program memory
2. Execute $S(x_0)$
3. Execute $S(x_1)$
4. Execute $S(x_2)$
5. Execute $S(x_3)$

...

Data Memory

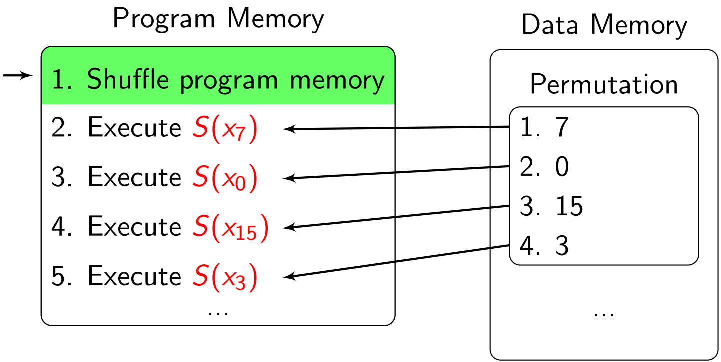
Permutation

1. 7
2. 0
3. 15
4. 3

...

Proposed by Veyrat-Charvillon et al. at Asiacrypt 2012

Shuffling - Randomized Program Memory



Proposed by Veyrat-Charvillon et al. at Asiacrypt 2012

Shuffling - Randomized Program Memory

Program Memory

1. Shuffle program memory
2. Execute $S(x_7)$ ←
3. Execute $S(x_0)$
4. Execute $S(x_{15})$
5. Execute $S(x_3)$

...

Data Memory

Permutation

1. 7
2. 0
3. 15
4. 3

...

Proposed by Veyrat-Charvillon et al. at Asiacrypt 2012

Shuffling - Randomized Program Memory

Program Memory

1. Shuffle program memory
2. Execute $S(x_7)$
3. Execute $S(x_0)$
4. Execute $S(x_{15})$
5. Execute $S(x_3)$

...

Data Memory

Permutation

1. 7
2. 0
3. 15
4. 3

...

Proposed by Veyrat-Charvillon et al. at Asiacrypt 2012

Outline

Shuffling with FRAM

Setup:

- MSP430FR5739 Texas Instrument microcontroller
- 16-bit RISC CPU
- 16 kB of FRAM

Implementation of the countermeasure:

- Definition of an AES having sets of 16 independent operations
- Addition of dummy key-schedule operations
- Access to FRAM memory between each operation

Security evaluation:

- Similar to the one presented at Asiacrypt 2012

Shuffling with FRAM

		Code Size	Data Size
Unprotected AES		1076	52
Shuffled AES	Perm. Generation	194	18
	Code Shuffling	418	0
	AES execution	2404	146
	Total	3016	164

- Unshuffled version of AES for reference
- Difference between unprotected and shuffled AES mainly due to dummy key schedule

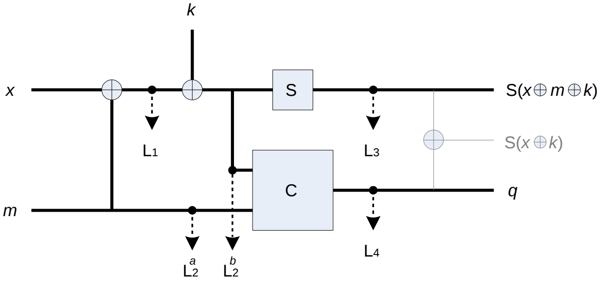
Shuffling with FRAM

		Cycle Count
Unprotected AES		5800
Shuffled AES	Perm. Generation	2240
	Code Shuffling	2751
	AES execution	8479
	Total	13470

- TI microcontrollers only have 12 available registers
  - Intermediate state must be stored in memory
  - TI implementation slower than AVR one (3546 cycles)
- Precomputation time divided by 100 compared to AVR:
  - 0,19 ms (at 16 MHz) vs 18 ms
- Difference between unprotected and shuffled AES mainly due to dummy key schedule

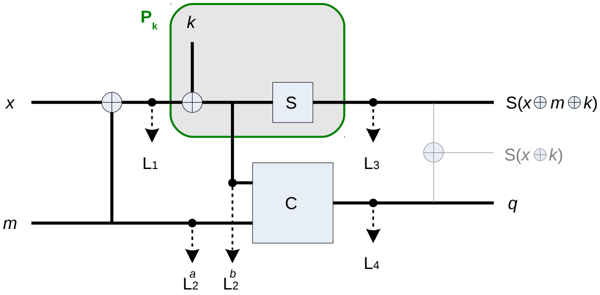
Outline

Masking with Randomized Look Up Tables (RLUT)



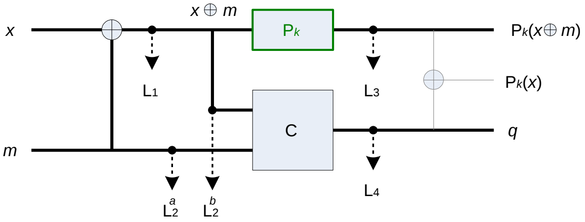
Typical boolean masking of order 1

Masking with Randomized Look Up Tables (RLUT)



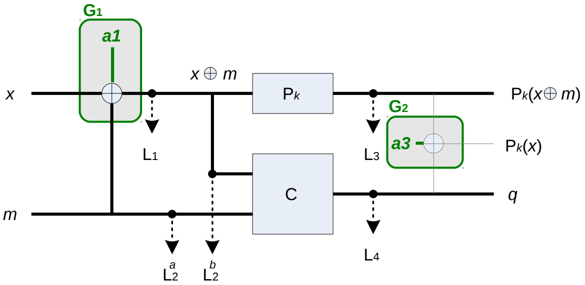
Key addition included in precomputed table P_k

Masking with Randomized Look Up Tables (RLUT)



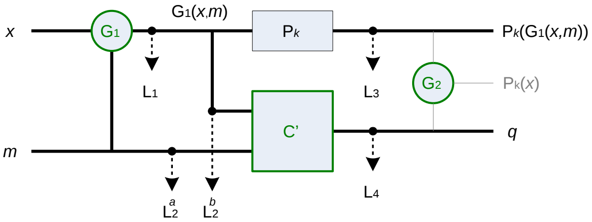
Key addition included in precomputed table P_k

Masking with Randomized Look Up Tables (RLUT)



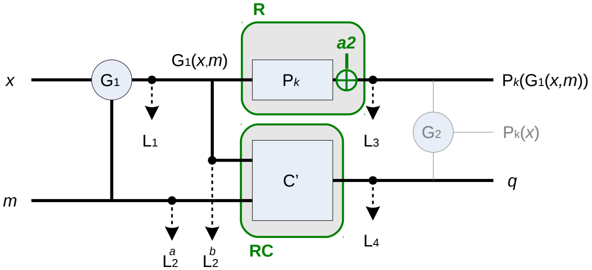
- Replace of $x \oplus m$ operations by $G_i = x \oplus m \oplus a_i$
- a_i = precomputed random mask

Masking with Randomized Look Up Tables (RLUT)



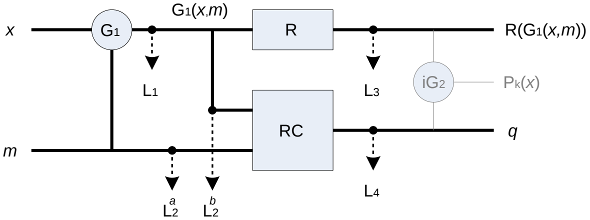
- Replace of $x \oplus m$ operations by $G_i = x \oplus m \oplus a_i$
- a_i = precomputed random mask

Masking with Randomized Look Up Tables (RLUT)

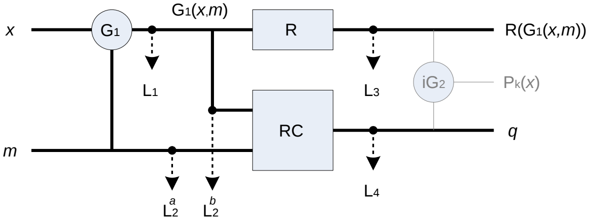


Randomization of P_k (and C) using a random variable

Masking with Randomized Look Up Tables (RLUT)

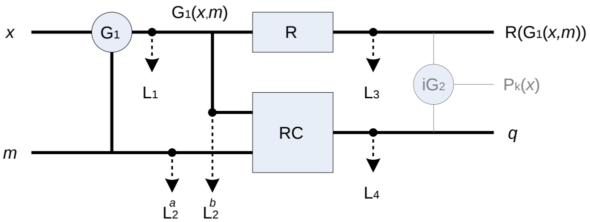


Masking with Randomized Look Up Tables (RLUT)



- G_1, G_2, R and RC are precomputed

Masking with Randomized Look Up Tables (RLUT)



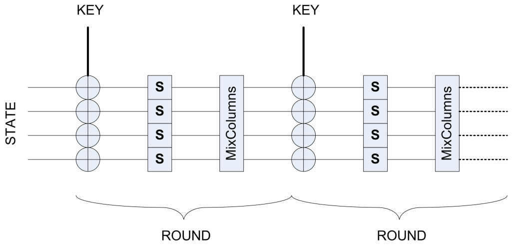
- G_1, G_2, R and RC are precomputed
- Unconditional security if secure precomputations

Outline

Application to Reduced LED

Reduced version of LED:

- 16-bit state
- 1 to 4 rounds



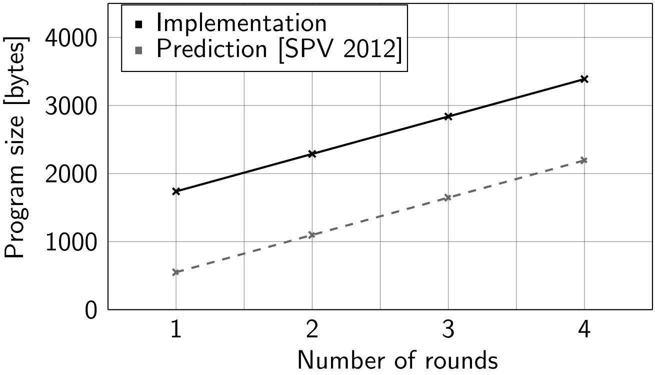
Application to Reduced LED

Implementation details:

- 16 kB TI FRAM microcontroller
- LFSR with CRC-32 polynomial used to generate random variables a_i
- Efficient arrangement of the 4-bit precomputed tables in memory
- MixColumn layer applied on each of the shares

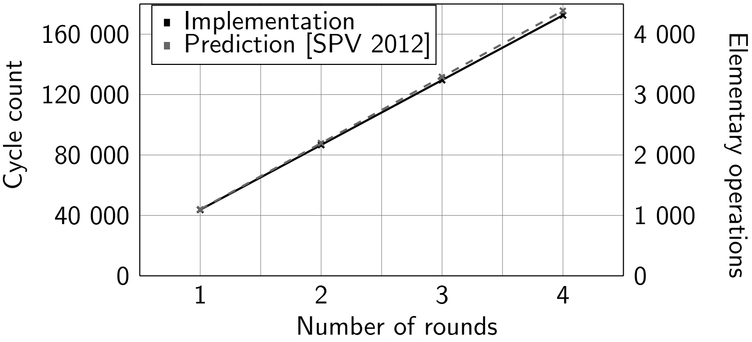
Outline

Results - Program Size



- Prediction in terms of number of rounds, number of S-Boxes and S-Box size
- Offset between curves = LED program size

Results - Precomputation Time



- Prediction in terms of elementary operations
- Here, 1 elementary operation ≈ 40 clock cycles

Results - Observations

- Memory and time requirements can be predicted for the parameters of any cipher
- Full LED implementation requires:
  - 70 kB of memory (128kB FRAM microcontroller soon available)
  - A precomputation time of 35 ms at 16 MHz
- Possible performances vs security tradeoffs:
  - Partial masking with RLUT
  - Partial refreshing of the precomputed tables (e.g.: refreshing 10% of the table takes as much cycles as order 3 masking scheme)

Conclusion

- FRAM enables efficient implementation of countermeasures needing precomputations
  - Improvement for the shuffling countermeasure
  - Makes the RLUT masking possible
- If secure precomputation is possible, RLUT provides unconditional security against side-channel attacks
- Future scope of research:
  - Impact of partial recomputation in leaking environment
  - Design of block ciphers suited to implementation with RLUT

Thank you!