Practical Electromagnetic Template Attack on HMAC

Pierre Alain Fouque[1] Gaétan Leurent[1] Denis Réal [2,3]
Frédéric Valette[2]

1 ENS, 75 Paris, France.

2 CELAR, 35 Bruz, France.

3 INSA-IETR, 35 Rennes, France.

September 7, 2009

Motivations

Why HMAC ?

- Standardized and deployed in a lot of Internet protocols.
- Security proofs.

SCA Attacks on HMAC

- DPA attacks on the hash function :
  - on MD4 and MD5 familly, SHA familly.
  - The internal states are figured out but isn't the value of k.

Classical Countermeasures

- Randomization of the execution of the implementation.
- A new key for each new computation.

Problematics

Can Side Channel Analysis be used in order to recover the whole HMAC secret key k with only 1 measure?

Problematics

Can Side Channel Analysis be used in order to recover **the whole** HMAC **secret key** k with only 1 **measure**?

Outline

1. Introduction
   - The Side Channel Leakage
   - The cryptographic target : HMAC
   - Our attack features
2. The Cryptanalysis
   - The leakage on SHA-1
   - The leakage on HMAC
   - The sketch of the attack
3. Practical experiments
   - The whole leakage
   - The hamming distances involved
4. Conclusion

Outline

Outline

Outline

Prints of the internal cryptographic activity filter through

Cryptographic Devices

- microprocessor,
- FPGA,
- smart card ...

Side Channel

1. computational time,
2. power consumption,
3. electromagnetic radiations.

The side channel leakage can be used to mount attack

SCA Attacks

- Simple Power Analysis Attack (SPA),
- Template Analysis (TA),
- Differential Power Analysis Attack (DPA).

SCA Software Countermeasures

1. SPA : power balanced implementation such as Montgomery Ladder,
2. DPA-TA : secret randomization and masking.

HMAC features

The authentication code $\mathrm{HMAC}_k(M)$ is defined as:

$$\mathrm{HMAC}_k(M) = H(\bar{k} \oplus \mathrm{opad} \,||\, H(\bar{k} \oplus \mathrm{ipad} \,||\, M)),$$

- opad and ipad are constant paddings.
- The secret key is manipulated twice:
  - $\bar{k} \oplus \mathrm{opad}$,
  - $\bar{k} \oplus \mathrm{ipad}$.

The Template Strategy

Loading a value in a register

- The hamming distance leaks.

Template-like Attack

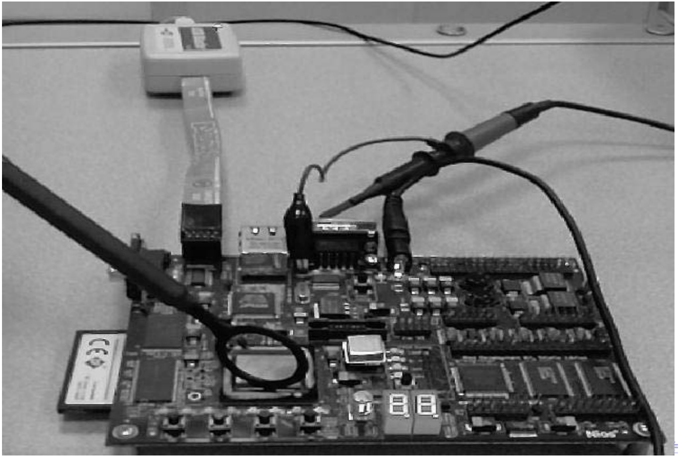
- Software implementation with known assembly code.
  - The secret key k is split in l words of 32 bits each.
  - Each word is treated one after each other.
- EM near field techniques : template on hamming distance
  - 33 template traces for 33 possible hamming distances.
  - 1 operational EM trace to figure the hamming distance out.

Secret recovery

- Load operations $\Rightarrow$ constraints on secret words.
- The whole secret k is recovered with $l \times 2^{32}$ computations.

Attack validation

HMAC SHA-1 on a STRATIX FPGA (ALTERA) with a software implementation on a NIOS II.



Goal of this section

Assumption

The attacker is able to figure out the hamming distance during a load operation.

Problematics

Is there enough loads in HMAC ?

Detailed implementation

The message m is expanded in 80 words W_i with $0 < i \leq 80$

- W_i is a 32-bit word.
- $W_i = m_i$ for $0 < i \leq l$

the compression function : repeat 80 times

- $P(A, B, C, D, E, W[i])$
- switch (A,B,C,D,E)

Values manipulated during the first rounds of SHA-1 only on the first message words

- the internal value A_1 depend on m_0:
- A_1 leaks each time it is manipulated :
  - B_2 (copy), C_3, D_4 and E_5 (after a rotation)

2 Conclusions about SHA-1

1^{st} conclusion : recovering k is a recurrence problem

- The internal values for the 1^{st} call of P only depend on m_0.
- The internal values for the 2^{nd} call of P only depend on m_1 and m_0.
- The internal values for the i^{th} call of P only depend on m_j with $j < i$.

From now, we just focus on m_0.

Function P causes 8 load operations with m_0-dependent values

- 3.5 bits of information for each load (hamming weight on 32 bits).
- $8 \times 3.5 = 28$ bits of constraint on $W[0] = m_0$.

2 Conclusions about SHA-1

1^{st} conclusion : recovering k is a recurrence problem

- The internal values for the 1^{st} call of P only depend on m_0.
- The internal values for the 2^{nd} call of P only depend on m_1 and m_0.
- The internal values for the i^{th} call of P only depend on m_j with $j < i$.

From now, we just focus on m_0.

Function P causes 8 load operations with m_0-dependent values

- 3.5 bits of information for each load (hamming weight on 32 bits).
- $8 \times 3.5 = 28$ bits of constraint on $W[0] = m_0$.

2 Conclusions about SHA-1

1^{st} conclusion : recovering k is a recurrence problem

- The internal values for the 1^{st} call of P only depend on m_0.
- The internal values for the 2^{nd} call of P only depend on m_1 and m_0.
- The internal values for the i^{th} call of P only depend on m_j with $j < i$.

From now, we just focus on m_0.

Function P causes 8 load operations with m_0-dependent values

- 3.5 bits of information for each load (hamming weight on 32 bits).
- $8 \times 3.5 = 28$ bits of constraint on $W[0] = m_0$.

But the key is manipulated twice in HMAC

- $m = \bar{k} \oplus \text{ipad}$ or $m = \bar{k} \oplus \text{opad}$.
- The secret key is used twice in the HMAC construction: it is used in the inner hash function as $H(\bar{k} \oplus \text{ipad})$ and in the outer hash function as $H(\bar{k} \oplus \text{opad})$.
- $28 \times 2 > 32$ bits of constraints on k_0.

SCA and the resulting Cryptanalysis

The Side Channel Analysis

- Spy the register load operation with EM techniques.
- Find out the hamming distance between 2 consecutive register values.
- Deduce the hamming weight of internal values computed by the compression function.

Cryptanalysis

- SCA gives constraints on hamming weights of words of $\bar{k} \oplus \text{ipad}$ and $\bar{k} \oplus \text{opad}$.
- Constraints on $(\bar{k} \oplus \text{ipad})_i$ and $(\bar{k} \oplus \text{opad})_i$ with $0 \leq i < l$ depend on k_j with $0 \leq j < i$
- Find every k_i, one after each other.

SCA and the resulting Cryptanalysis

The Side Channel Analysis

- Spy the register load operation with EM techniques.
- Find out the hamming distance between 2 consecutive register values.
- Deduce the hamming weight of internal values computed by the compression function.

Cryptanalysis

- SCA gives constraints on hamming weights of words of $\bar{k} \oplus \text{ipad}$ and $\bar{k} \oplus \text{opad}$.
- Constraints on $(\bar{k} \oplus \text{ipad})_i$ and $(\bar{k} \oplus \text{opad})_i$ with $0 \leq i < l$ depend on k_j with $0 \leq j < i$
- Find every k_i, one after each other.

Summary of the attack

- Each call to the function P induces 8 load operations of secret dependent values.
- Measuring the hamming distance gives 3.5 bits of information.
- If the values are sufficiently independent : $8 \times 3.5 = 28$ bits of constraint on $W[i]$.
- Information on $\bar{k} \oplus$ ipad and on $\bar{k} \oplus$ opad
- Enough information to guess k recursively by 32 bits and check if the guess is compliant with the hamming weight measured on each call of P.
- A key of size $32 \times l$ bits will be figured out with $l \times 2^{32}$ tries
- The algorithm still works if there is some errors in the measurement (less than 1.5 bits).

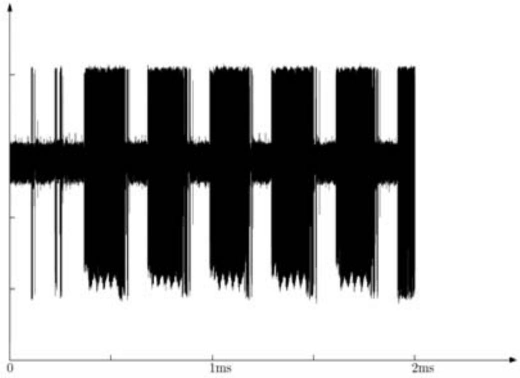
Goal of this section

Problematics

Is the attacker able to figure out the hamming distance during a load operation?

HMAC implementation

Each call to the compression function is followed by a sleep of $100\mu s$.



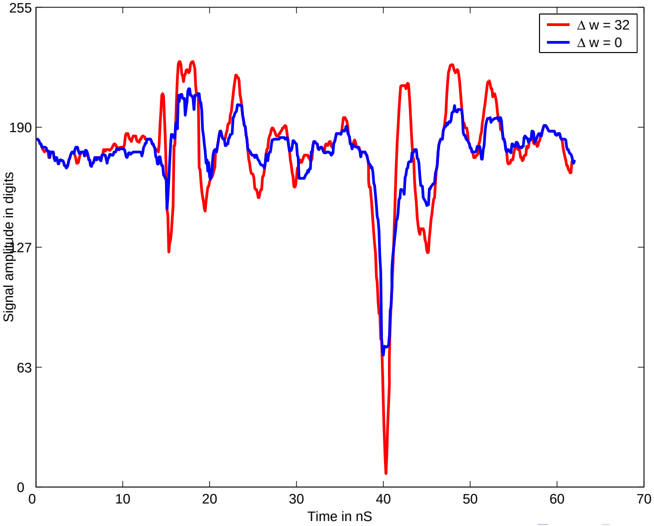
When one's should look at?

- The instruction

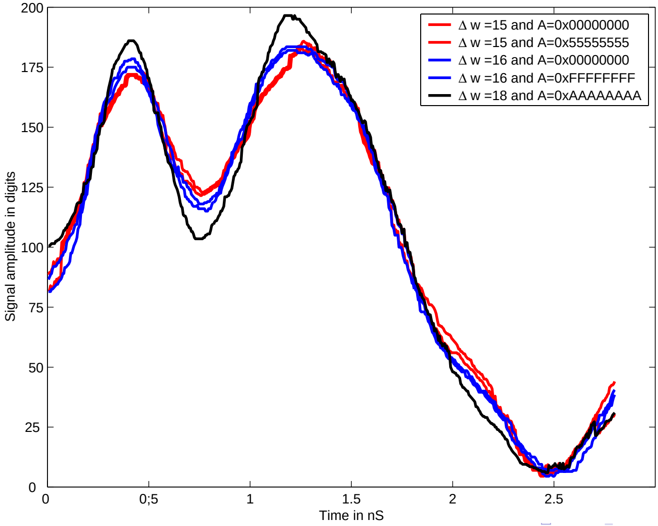
  ldw R2 (80) fp

  corresponds to the loading of W[0] in the register R2.
- CEMA on this instruction in order to localize the proper clock cycle.

Extremal hamming distances



Near hamming distances



Cost of the attack

Template profiling stage

- Around 20000 traces.
- A few CEMAs to time localize instructions.

Template operational stage

- 1 trace.
- 4×2^{32} on SHA-1 guesses to retrieve the whole 128-bit key .
- Tolerate an error of 1.5 bit.

Conclusion

Very efficient attack

- First attack which allow to retrieve the key on HMAC.
- Practical experiments have been done for validation.
- Only one curve is needed on a non-protected implementation.

Logic countermeasure

- Precharging the target register with a random value.

Thanks for your attention

Do you have any questions ?