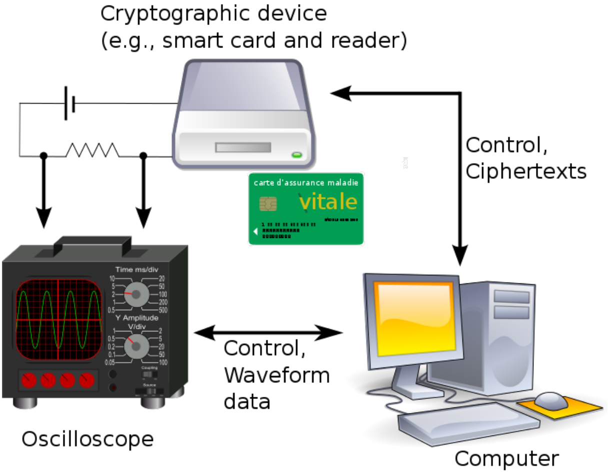
Side-Channel Attacks

Jean-Sébastien Coron

University of Luxembourg

May 20, 2014

Side-channel Attacks



Side-Channel Attack

- Consists in obtaining a side-channel information during the execution of a cryptographic algorithm
- Side-channel Attacks
  - Timing attack
  - Power attack
  - Fault attack
  - Differential Power Analysis
- Side-channel countermeasures
  - Countermeasures for RSA
  - The masking countermeasure
  - The Ishai-Sahai-Wagner transform

Implementation attacks

- The implementation of a cryptographic algorithm can reveal more information
- Passive attacks :
  - Timing attacks (Kocher, 1996): measure the execution time
  - Power attacks (Kocher et al., 1999): measure the power consumption
- Active attacks :
  - Fault attacks (Boneh et al., 1997): induce a fault during computation
  - Invasive attacks: probing.

Timing attacks

- Described on RSA by Kocher at Crypto 96.
  - Let $d = \sum_{i=0}^{n} 2^i d_i$.
  - Computing $m^d \mod N$ using square and multiply :
    - Let $z \leftarrow m$
      For $i = n-1$ downto 0 do
      Let $z \leftarrow z^2 \mod N$
      If $d_i = 1$ let $z \leftarrow z \cdot m \mod N$
- Attack
  - Let T_i be the total time needed to compute $m_i^d \mod N$
  - Let t_i be the time needed to compute $m_i^3 \mod N$
  - If $d_{n-1} = 1$, the variables t_i and T_i are correlated, otherwise they are independent. This gives d_{n-1}.

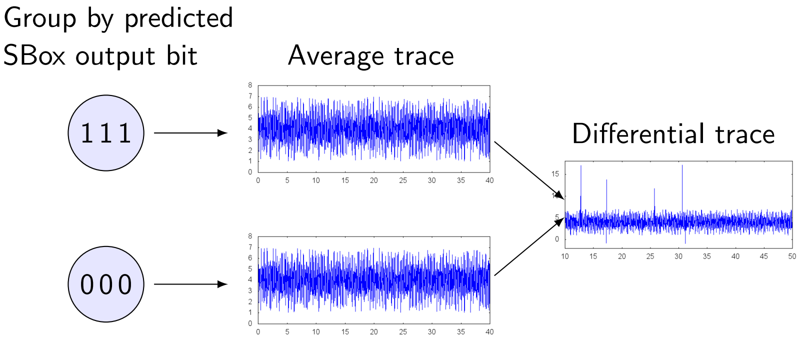
Power attacks

- Based on measuring power consumption
  - Introduced by Kocher *et al.* at Crypto 99.
  - Initially applied on DES, but any cryptographic algorithm is vulnerable.
- Attack against exponentiation $m^d \mod N$:
  - If power consumption correlated with some bits of $m^3 \mod N$, this means that $m^3 \mod N$ was effectively computed, and so $d_{n-1} = 1$.
  - Enables to recover d_{n-1} and by recursion the full d.

Fault attacks

- Induce a fault during computation
  - By modifying voltage input
- RSA with CRT: to compute $s = m^d \mod N$, compute :
  - $s_p = m^{d_p} \mod p$ where $d_p = d \mod p - 1$
  - $s_q = m^{d_q} \mod q$ where $d_q = d \mod q - 1$
  - and recombine s_p and s_q using CRT to get $s = m^d \mod N$
- Fault attack against RSA with CRT (Boneh *et al.*, 1996)
  - If s_p is incorrect, then $s^e \neq m \mod N$ while $s^e = m \mod q$
  - Therefore, $\gcd(N, s^e - m)$ gives the prime factor q.

Differential Power Analysis [KJJ99]



DPA

- The attack was originally described on DES, but any block-cipher is vulnerable.
- Given $x \in \{0,1\}^n$ for some small n (say $n = 8$), consider the computation:

  $$y = S(x \oplus k)$$

  where $y, k \in \{0,1\}^n$ and k is a subkey.
- Let E be the power consumption when $S(x \oplus k)$ is computed
  - We assume that E is correlated to $S(x \oplus k)$
  - For example

    $$E = H(S(x \oplus k)) + B$$

    where $H()$ is the Hamming weight and B is some noise.

DPA

- We assume that we get:

$$E_i = H(S(x_i \oplus k)) + B_i$$

for known x_i's, but unknown k.

- Let $k' \in \{0,1\}^n$. Let b_i be least significant bit of $S(x_i \oplus k)$. Compute:

$$d(k') =< E_i >_{b_i=1} - < E_i >_{b_i=0}$$

- If $k = k'$, then we get a peak:

$$d(k') =< S(x_i \oplus k) >_{b_i=1} - < S(x_i \oplus k) >_{b_i=0} = 1$$

- If $k \neq k'$, then $d(k') \simeq 0$.
- This enables to recover k from the power consumption E_i

Countermeasures

- Hardware countermeasures
  - Constant power consumption; dual rail logic.
  - Random delays to desynchronise signals.
- Countermeasures for public-key
  - Randomization based on the existing mathematical structure
- Countermeasure for block-ciphers
  - Randomization based on masking intermediate variable

Countermeasures for RSA

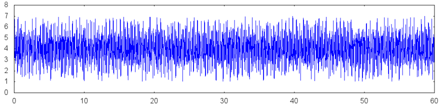
- Implement in constant time
  - Not always possible with hardware crypto-processors.
- Exponent blinding:
  - Compute $m^{d+k\cdot\phi(N)} = m^d \mod N$ for random k.
- Message blinding
  - Compute $(m\cdot r)^d/r^d = m^d \mod N$ for random r.
- Modulus randomization
  - Compute $m^d \mod (N\cdot r)$ and reduce modulo N.
- or a combination of the three.

Masking Countermeasure

- Let x be some variable in a block-cipher.
- Masking countermeasure: generate a random r, and manipulate the masked value x'

  $$x' = x \oplus r$$

  instead of x.
- r is random $\Rightarrow$ x' is random
  $\Rightarrow$ power consumption of x' is random



$\Rightarrow$ no information about x is leaked

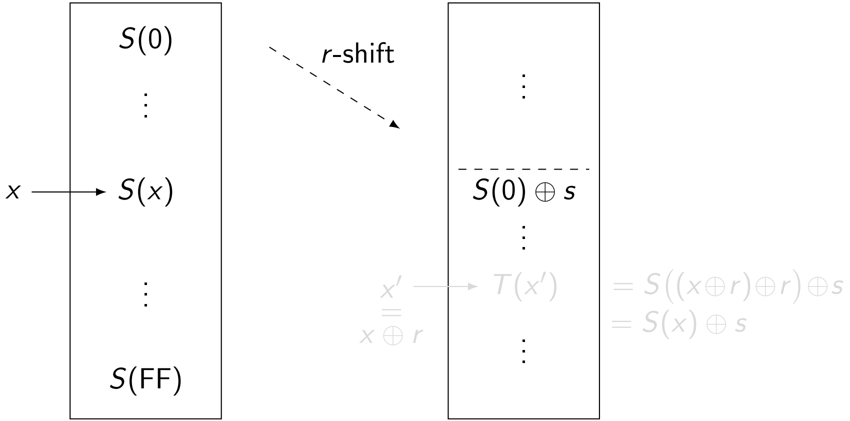
Masking Countermeasure

- How do we compute with $x' = x \oplus r$ instead of x ?
- Linear operation $f(x)$ (*e.g.* `MixColumns` in AES): easy

$$f(x') = f(x) \oplus f(r)$$

  - We compute $f(x')$ and $f(r)$ separately.
  - $f(x)$ is now masked with $f(r)$ instead of r.
- Non-linear operations (SBOX): randomized table [CJRR99]

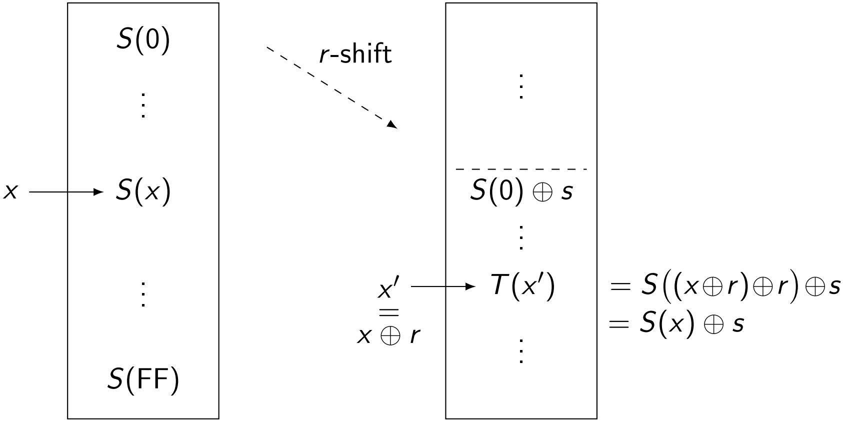
Randomized Table Countermeasure [CJRR99]



$S(u)$
Original table in ROM

$T(u) = S(u \oplus r) \oplus s$
Randomized table in RAM

Randomized Table Countermeasure [CJRR99]

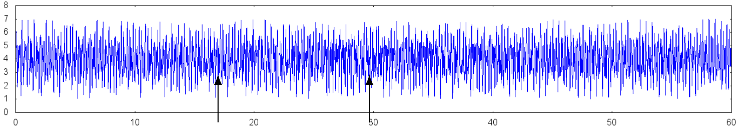


$S(u)$
Original table in ROM

$T(u) = S(u \oplus r) \oplus s$
Randomized table in RAM

Second-order Attack

- Second-order attack:



$E(x')$ $\qquad$ $E(r)$

$f(E(x'), E(r))$ correlated with $x = x' \oplus r$

- Requires more curves but can be practical

Higher-order masking

- Solution: n shares instead of 2:

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

- Any subset of $n-1$ shares is uniformly and independently distributed
  - If we probe at most $n-1$ shares x_i, we learn nothing about $x \Rightarrow$ secure against a DPA attack of order $n-1$.
- Linear operations: still easy
  - Compute the $f(x_i)$ separately

$$f(x) = f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_n)$$

Higher-order computation of SBoxes

- SBox computation ?
  - We have input shares $x_1, \ldots, x_n$, with

$$x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

  - We must output shares $y_1, \ldots, y_n$, such that

$$S(x) = y_1 \oplus y_2 \oplus \cdots \oplus y_n$$

  - without leaking information about x.

Existing Higher Order Countermeasure

- Ishai-Sahai-Wagner private circuit [ISW03]
  - Shows how to transform any boolean circuit C into a circuit of size $\mathcal{O}(|C| \cdot t^2)$ perfectly secure against t probes.
- Rivain-Prouff (CHES 2010) countermeasure for AES:

$$S(x) = x^{254} \in \mathbb{F}_{2^8}$$

  - Secure multiplication based on [ISW03]:

$$z = xy = \left(\bigoplus_{i=1}^{n} x_i \right) \cdot \left(\bigoplus_{i=1}^{n} y_i \right) = \bigoplus_{1 \le i,j \le n} x_i y_j$$

  - Provably secure against t-th order DPA with $n \ge 2t+1$ shares.

Existing Higher Order Countermeasures

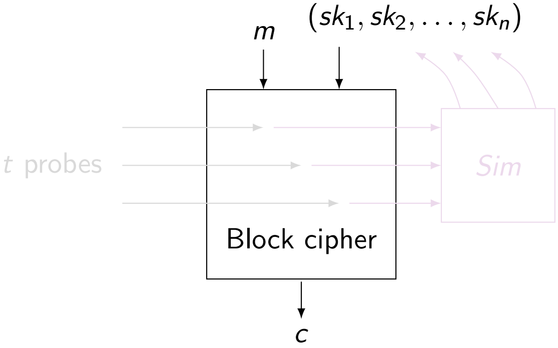
- Carlet *et al.* (FSE 2012) countermeasure for any Sbox.
  - Lagrange interpolation

$$S(x) = \sum_{i=0}^{2^k-1} \alpha_i \cdot x^i$$

over $\mathbb{F}_{2^k}$, for constant coefficients $\alpha_i \in \mathbb{F}_{2^k}$.

ISW security model

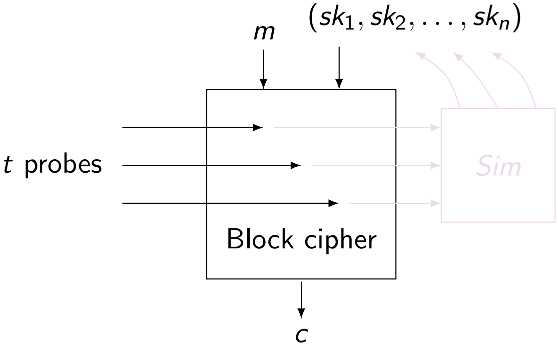
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

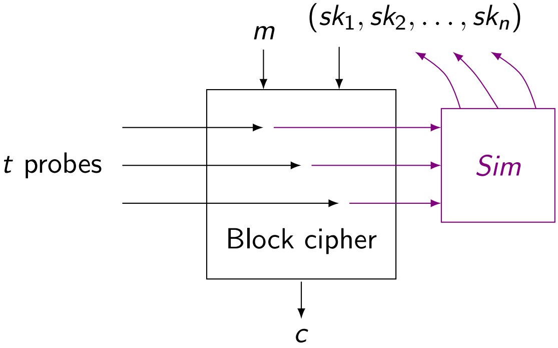
- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

ISW security model

- Simulation framework of [ISW03]:



- Show that any t probes can be perfectly simulated from at most $n-1$ of the sk_i's.
- Those $n-1$ shares sk_i are initially uniformly and independently distributed.
- $\Rightarrow$ the adversary learns nothing from the t probes, since he could perfectly simulate those t probes by himself.

Protecting a XOR gate

- We wish to protect a XOR gate $c = a \oplus b$
  - Input: a_i such that $a_1 \oplus a_2 \oplus \cdots \oplus a_n = x$, and b_i such that $b_1 \oplus b_2 \oplus \cdots \oplus b_n = b$
  - Output: c_i such that $c_1 \oplus c_2 \oplus \cdots \oplus c_n = c$
  - Algorithm: let $c_i \leftarrow a_i \oplus b_i$ for all $1 \leq i \leq n$
- Proof of security
  - Let $I \leftarrow \emptyset$
  - If there is a probe for a_i or b_i or c_i, add i to I.
  - We get $|I| \leq t$
  - Any probe can be simulated from the knowledge of $a_{|I}$ and $b_{|I}$, where $a_{|I} = (a_i)_{i \in I}$.
  - If $t \leq n - 1$, the t probes can be perfectly simulated without the knowledge of a and b.

Protecting a AND gate

- We wish to protect a AND gate $c = ab$
  - Input: a_i such that $a_1 \oplus a_2 \oplus \cdots \oplus a_n = a$, and b_i such that $b_1 \oplus b_2 \oplus \cdots \oplus b_n = b$
  - Output: c_i such that $c_1 \oplus c_2 \oplus \cdots \oplus c_n = c$
  - For each $1 \leq i < j \leq n$, generate a random r_{ij}, and let $z_{ij} \leftarrow r_{ij}$ and $z_{ji} \leftarrow (z_{ij} \oplus a_i b_j) \oplus a_j b_i$
  - Let $c_i = a_i b_i \oplus \bigoplus_{j \neq i} z_{ij}$
- Every AND gate is expanded into a “gadget” of $\mathcal{O}(n^2)$ gates.

Proof of security

- Generation of the set I. Initially $I \leftarrow \emptyset$
  - If a wire a_i, b_i, $a_i b_i$, z_{ij} (for $i \neq j$) is probed, add i to I.
  - Same for a sum of values of the above form, including c_i.
  - For the wires $a_i b_j$ or $z_{ij} \oplus a_i b_j$ for $i \neq j$, add both i and j to I
  - We have $|I| \leq 2t$
- Simulation of the wires using only $a_{|I}$ and $b_{|I}$
  - Simulation of a_i, b_i, $a_i b_i$ for $i \in I$: obvious
  - Simulation of z_{ij} when $i \in I$ but $j \notin I$
    - If $i < j$, generate a random z_{ij}, as in the real circuit
    - If $i > j$, then in the real circuit $z_{ij} = (z_{ji} \oplus a_j b_i) \oplus a_i b_j$, where $z_{ji} = r$ where $r \leftarrow \{0,1\}$. Instead we can let $z_{ji} \leftarrow (r \oplus a_j b_i) \oplus a_i b_j$, which gives $z_{ij} = r$. Since $j \notin I$, z_{ji} is not used is the computation of any probe, so no need to know a_j and b_j. Summary: in both cases let $z_{ij} \leftarrow \{0,1\}$
  - Simulation of z_{ij} when both $i, j \in I$: obvious
  - Simulation of a sum of the above terms: obvious
  - Simulation of $a_i b_j$ or $z_{ij} \oplus a_i b_j$: obvious since in that case $i, j \in I$

Proof of security

- Simulation for a single gate
  - Since $|I| \leq 2t$, with a number of shares $n \geq 2t+1$, we can perfectly simulate all the probes in the circuit.
  - Namely since $|I| \leq n-1$, the sets $a_{|I}$ and $b_{|I}$ can be perfectly simulated by generating
- Simulation of a general circuit
  - Any circuit C can be written with XOR and AND gates only
  - We examine every gadget g in the expanded circuit C' as previously, building the set I
  - We still have $|I| \leq 2t$
  - We perform the simulation as previously. Inductively for each gadget g, the shares of the inputs to g belonging to I are perfectly simulated. Hence we can perfectly simulate all probes.
- Conclusion
  - Any boolean circuit C can be transformed into a circuit of size $\mathcal{O}(|C| \cdot t^2)$ perfectly secure against t probes.