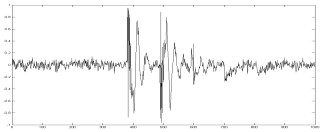
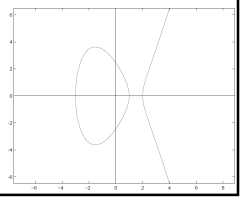


Power Analysis on Curve-based Cryptography

Benedikt Gierlichs
Lejla Batina
K.U.Leuven/Cosic



BCRYPT ECC DAY, Leuven, 20/03/08

Outline

- A very brief intro to simple and differential power analysis
- Security hierarchy of curved-based crypto
- Protocols and adversary's options
- Optimization vs. Vulnerability
- Countermeasures

Simple Power Analysis

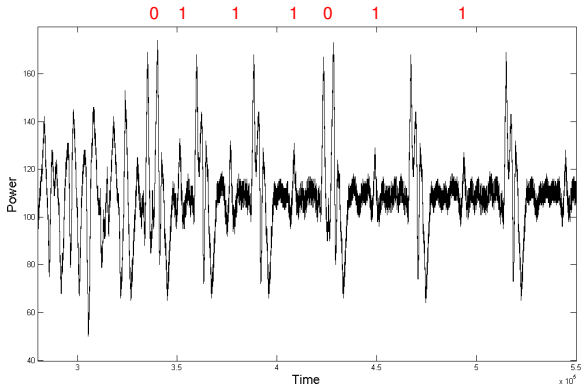
- Based on one or few measurements
- Mostly discovery of features that depend on the sequence of instructions
- Threats for asymmetric crypto:
  - Key recovery (if badly implemented, e.g. RSA / ECC)
  - Detection of keylength
  - Implementation details: for example RSA with CRT
- Search for repetitive patterns

Toy example: Double and Add

Require: P, $\mathcal{O}$, $k = (k_{l-1}k_{l-2}\ldots k_0k_0)_2$
Ensure: kP
1: $Q \leftarrow \mathcal{O}$
2: **for** i from $l-1$ to 0 **do**
3: $Q \leftarrow 2Q$ point doubling
4: **if** $k_i = 1$ **then**
5: $Q \leftarrow Q + P$ point addition
6: **end if**
7: **end for**
8: **Return** Q

Conditional operation: Side Channel

Simple Power Analysis

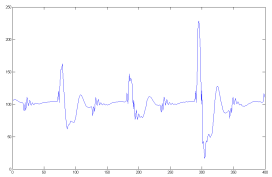


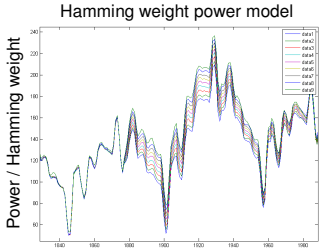
Differential Power Analysis

- Based on many measurements
  - < 100 for unprotected software implementations
  - Up to several 100 000 for protected hardware
- Exploit deterministic variations in the power consumption that are caused by processing varying data
- Power consumption allows to confirm/reject a guess about a vector of intermediate results
- Statistics, hypothesis tests, lots of data
- Choose intermediate result that depends on only a few key bits, exhaustive search is simple, divide et impera
- Applies to symmetric and to asymmetric crypto

Toy example: one AES Sbox

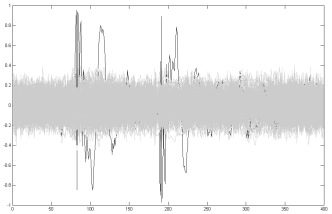
- Key E (0..255) stored in device, 1 byte input X_i, 1 byte output Y_i
- For i = 1..100
  - Read input X_i
  - Compute $Y_i = S(X_i + K)$
- EndFor

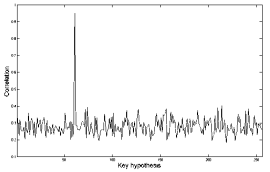




Toy example (cont'd)

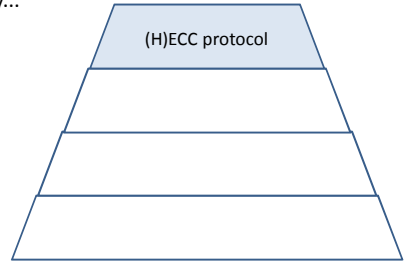
- For K = 0..255
  - For i = 1..100
    - Read input X_i
    - Compute $Y_i = S(X_i + K)$
  - EndFor
- EndFor
- Apply hypothesis test





Security hierarchy

- Protocol defines the context
- Context provides many things to attack!
  But what is of interest?
- Usually the private key…

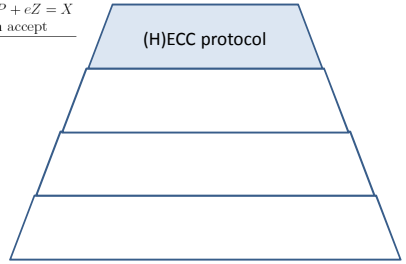


Security hierarchy: example Schnorr

Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

- Possible targets:
  - Scalar mult. rP
  - Mod. mult. ae+r



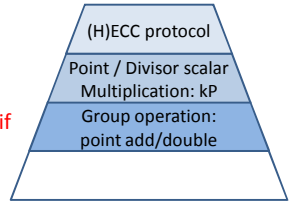
Security hierarchy: example Schnorr

- SPA on rP might reveal r
  (depending on the implementation of the group ops)
- Is knowing r useful?
- Yes, if r is known, compute $a = (y-r)e^{-1}$

Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

- If group ops are SPA resistant, try DPA on points and recover key bit-by-bit
- Toy example: 3P is only computed if second key bit = 1



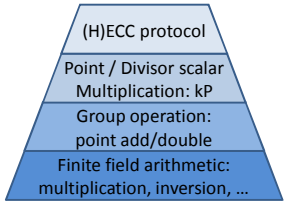
Example Schnorr (cont'd)

- But why all that effort?
- We want a!
- Y = ae + r, simple mod mult
- DPA might reveal a immediately
  (if mod multiplication not DPA resistant)

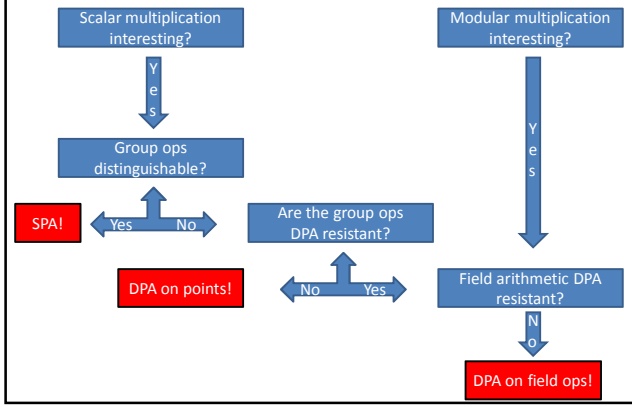
Table 1. Schnorr Identification Protocol

Prover		Verifier
$r \in_R \mathbb{Z}_n$		
$X \leftarrow rP$	$\xrightarrow{X}$	
	$\xleftarrow{e}$	$e \in_R Z_{2^t}$
$y = ae + r$	$\xrightarrow{y}$	
		if $yP + eZ = X$ then accept

- For a = 0..p-1
  - For i = 1..100
    - Read input e_i
    - Compute $a \cdot e_i$
  - EndFor
- EndFor
- Apply hypothesis test



Adversary‘s options (not exhaustive)

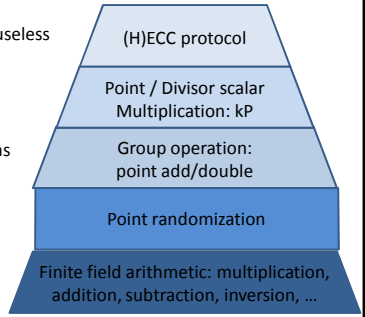


Countermeasures

(not exhaustive and not provably secure)

- Can be applied on all levels of the hierarchy

- Make sure leaked information is useless
- Special multiplication algorithms
- Indistinguishable group operations
- Randomize intermediate results
- Secure hardware, randomization



Countermeasures – protocol level

- Leakage aware protocol design
- If leakage can be tolerated, we are done!
- Scalar randomization [Cor00] $(k + r \cdot l)P = k'P$
  (assuming the randomization operation is secure, SPA recovers a randomized scalar; DPA on points recovers randomized scalar; DPA on field arithmetic recovers randomized scalar)
- Point randomization [Cor00], DPA resistance
  ($kP = k(R+P) - kR$), since P is randomized, adversary cannot predict the value of e.g. 3P

Countermeasures – scalar mult.

- Double and always add [Cor00], SPA resistance
  (sequence of instructions is constant, vulnerable to DPA)
- Width w-NAF encoding [OT04], SPA
  (encoded scalar 00…X, vulnerable to DPA)
- Montgomery powering ladder [JY02], SPA resistance
  (sequence of instructions is constant, vulnerable to DPA)
- Highly regular right-to-left algorithms [J07]
  (sequence of instructions is constant, vulnerable to DPA)

Montgomery Powering Ladder

Algorithm 1 Algorithm for point multiplication

Require: an integer $k > 0$ and a point P
Ensure: $x(kP)$

1: $k \leftarrow k_{l-1}, ..., k_1, k_0$
2: $P_1 \leftarrow P, \quad P_2 \leftarrow 2P.$
3: **for** i from $l-2$ downto 0 **do**
4: If $k_i = 1$ then
5: $x(P_1) \leftarrow x(P_1 + P_2), x(P_2) \leftarrow x(2P_2)$
6: Else
7: $x(P_2) \leftarrow x(P_1 + P_2), x(P_1) \leftarrow x(2P_1)$
8: **end for**
9: Return $x(P_1)$

Countermeasures – group ops.

- Side channel atomicity [CCJ03], SPA
  (make "double" and "add" look the same, vulnerable to DPA)
- Unified addition and doubling [BJ02], SPA
  (one formula for both operations, vulnerable to DPA)
- Edward's coordinates [BL07], Hessian curves [JQ01], SPA
  (one formula for both operations, vulnerable to DPA)

Balanced ECC point operations

Algorithm 2 EC point addition and doubling

Require: X_i, Z_i	Require: $c \in \mathrm{GF}(2^n)$, $c = b^{2^{n-1}}$,
for $i = 1, ..., 4$; $x_i = \frac{X_i}{Z_i}$, $x_4 = x(P_1 - P_2)$	X_1, Z_1 where $x_1 = \frac{X_1}{Z_1}$
Ensure: $X(P_1 + P_2) = X(P_3) = X_3, Z_3$	Ensure: $X(2P_1) = X(P_5) = X_5, Z_5$
1. $X_3 \leftarrow X_1 + X_2, Z_3 \leftarrow Z_1 + Z_2$	1. $T_1 \leftarrow c + Z_1, T_1 \leftarrow T_1 + Z_1$
2. $Z_3 \leftarrow X_3 \cdot Z_3$	2. $Z_5 \leftarrow {Z_1}^2$
3. $T_2 \leftarrow X_1 Z_1$	3. $X_5 \leftarrow {X_1}^2$
4. $X_3 \leftarrow X_2 Z_2$	4. $T_1 \leftarrow Z_5 T_1$
5. $Z_3 \leftarrow Z_3 + T_2 + X_3$	5. $T_1 \leftarrow T_1 + X_1 + X_1$
6. $Z_3 \leftarrow {Z_3}^2$	6. $Z_5 \leftarrow X_5 Z_5$
7. $T_1 \leftarrow x_4 Z_3$	7. $T_1 \leftarrow {T_1}^2$
8. $X_3 \leftarrow T_2 X_3$	8. $X_5 \leftarrow {X_5}^2$
9. $X_3 \leftarrow X_3 + T_1$	9. $X_5 \leftarrow X_5 + T_1$

Countermeasures – point randomization

- Random isomorphism [JT01], DPA
  (adversary cannot predict the value of e.g. 3P, vulnerable to SPA)

ECC countermeasures and field arithmetic

- However, unified group operation formulae might be vulnerable to SPA, if the underlying field arithmetic is not secure
- Montgomery multiplication with conditional subtraction
- Multiplication and squaring distinguishable

Wrap up

- Power analysis is fairly cheap to set-up and a real threat for embedded cryptographic systems
- Protocol defines the context and thus attack targets
- What leakage can be tolerated? What needs to be secured?
- Adversary will go for the weakest link!
- Security is hard to add-on
- Think about it when designing your protocol and your implementation!

Thank you for your attention! Questions?