Cryptographic Algorithms Implemented on FPGAs

Francisco Rodríguez Henríquez

Why Secure Hardware?

- …bedded systems now common in the industry
  - …Hardware tokens, smartcards, crypto accelerators, internet …pliances
- … analysis & reverse engineering …s available to all
- …fficulty of attack
- … exist

Francisco Rodríguez Henríquez

Attacker resources and methods vary greatly

	Teenager	Academic	Org. Crime	Gov't
	Limited	Moderate	Large	Large
	<$1000	$10K-$100K	$100K+	Unknown
	…aries	High	Varies	Varies
	…gh	High	Low	Low
	…allenge	Publicity	Money	Varies
	…y	Moderate	Few	Unknown
		No	Yes	Yes
		Yes	Varies	No

Source: Cryptography Research, Inc. 1999, "Crypto Due Diligence"

Minimal key lengths for symmetric ciphers

Source: Blaze/Diffie/Rivest/Schneier/Shimoura/Thompson/Wiener: www.bsa.org/policy/encryption

…e …er	Budget	Tool	Time and cost per key recovered		Length needed for protection in late 1995
			40 bits	56 bits	
	tiny	scavenged computer time	1 week	infeasible	45
	$400	FPGA	5 hours ($0.08)	38 years ($5,000)	50
	…,000	FPGA	12 min ($0.08)	556 days ($5,000)	55
	…K	FPGA	24 sec ($0.08)	19 days ($5,000)	60
		ASIC	18 sec ($0.001)	3 hours ($38)	
		FPGA	7 sec ($0.08)	13 hours ($5,000)	70
		ASIC	0.005 sec ($0.001)	6 min ($38)	
		ASIC	0.0002 sec ($0.001)	12 sec ($38)	75

Reconfigurable Hardware

nfigurable Hardware (RCHW) means in
mmercial applications mostly:

Programmable Gate Arrays (FPGAs)

le Programmable Logic Devices (EPLD).

Francisco Rodríguez Henríquez

Field Programmable Gate Arrays

realize a variety of circuits:

be reprogrammed in-system,

st of boolean and storage elements,

alize fairly large circuits > 100; 000 gates.

Francisco Rodríguez Henríquez

Reconfigurable Computing - Characteristics

is the middle ground between ASICs and processors. ASICs are the ultimate in speed but xibility while processors have the ultimate in y but lack speed.

ture is the ability to perform computations in increase performance, while retaining much ility of a software solution.

Francisco Rodríguez Henríquez

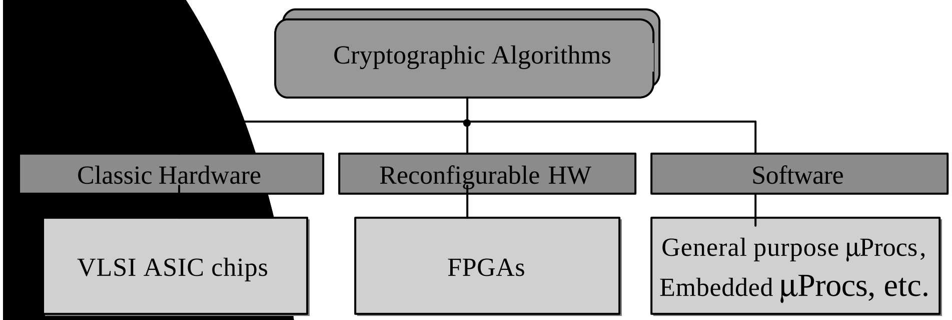
Choosing a Platform

of implementation is driven by:

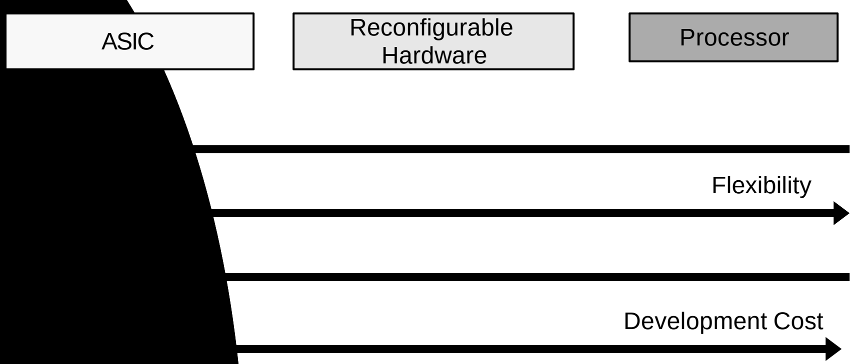
- rithm performance
- Per-unit cost, Development cost]
- onsumption (wireless devices!)
- y
- er change
- ity
- n agility
- urity

Francisco Rodríguez Henríquez

Platform Implementation for Cryptographic Algorithms



Reconfigurable Computing - defined



Why Crypto-algorithms in Hardware

[redacted] main reasons:

[redacted] tware implementations are too slow for some
[redacted] cations (symmetric alg: encryption rates
[redacted] bit/sec public-key alg: > 10 msec)

[redacted] are implementations are intrinsically more
[redacted] lly secure: Key access and algorithm
[redacted] on is considerably harder.

Francisco Rodríguez Henríquez

But why reconfigurable hardware?

[redacted] tial advantages of crypto algorithms implemented on
[redacted] onfigurable platforms:

[redacted] rithm Agility
[redacted] thm Upgrade
[redacted] cture Efficiency
[redacted] Efficient
[redacted] n Modification
[redacted] put relative to software)
[redacted] ciency relative to ASICs)

Francisco Rodríguez Henríquez

Crypto and FPGAs: Algorithm Agility

rvation: Modern security protocols are defined to
rithm independent:

ryption algorithm is negotiated on a per-session

variety of ciphers can be required. Ex: IPsec-
algorithms: DES, 3DES, Blow-Fish, CAST,
C4 and RC6, & future extensions!

olds for public-key algorithms, e.g., Diffie-
nd ECDH.

SIC solutions can provide algorithm agility
sts.

Francisco Rodríguez Henríquez

Crypto and FPGAs: Algorithm Upgrade

ons may need upgrade to a new algorithm because:

nt algorithms was broken (DES)

rd expired (again DES)

ndard was created (AES)

list of algorithm independent protocol was

-implemented algorithm is practically
devices are affected or in applications
ommunications.

Francisco Rodríguez Henríquez

Crypto and FPGAs: Architecture Efficiency

In certain cases a hardware architecture can be much
ore efficient if it is designed for a specific set of
rameters. Parameters for cryptographic algorithms
be for example the key, the underlying finite
the coefficient used (e.g., the specific curve of
C system), and so on. Generally speaking, the
pecific an algorithm is implemented the more
t it can become.

Francisco Rodríguez Henríquez

Crypto and FPGAs: Resource Efficiency

vation: The majority of security protocols uses
key as well as public-key algorithms during
ssion, but not simultaneous.

A device can be used for both through run
uration.

Francisco Rodríguez Henríquez

Crypto and FPGAs: Algorithm Modification

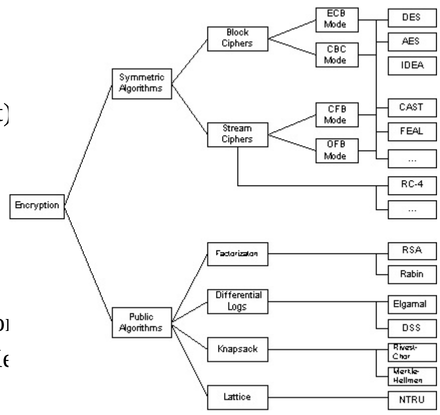
- ne applications require Public algorithms (such as candidates) with proprietary modules, e.g., tary S-boxes or permutations.
- of modes of operations (feedback modes, ode, etc.)
- alytical implementation, such as key-search may use slightly altered version of the

e changes can readily be implemented.

Francisco Rodríguez Henríquez

Cryptography

- dentiality is provided by on primitives
- the process of an readable (plaintext) unintelligible rtext)
- e inverse
- ion primitives
- phers (Conventio
- iphers (Public Ke



Cryptographic Primitives for Encryption

Francisco Rodríguez Henríquez

Case of Study 1: GF(2^m) Squaring

Francisco Rodríguez Henríquez

GF(2^m) Squarer

- …n most algorithms the modular product is computed in two steps: …ynomial multiplication followed by modular reduction. Let *A(x)* Î …m) be an arbitrary element in the field and *P(x)* be the irreducible …nerator polynomial.
- …o compute the modular square of the element *A*(*x*) we first … polynomial product *C(x)*, of degree at most *2m-2*, as

…omial product …oordinates

$$C(x) = A(x)A(x) = \left(\sum_{i=0}^{m-1} a_i \alpha^i\right)\left(\sum_{i=0}^{m-1} a_i \alpha^i\right)$$

- …ond step, a reduction operation is performed in order …-1 degree polynomial *C'(x)* defined as

…ction step …rdinates

$$C'(x) = C(x) \bmod P(x)$$

Francisco Rodríguez Henríquez

Squaring: Example

[redacted] be an element of the finite field $F=GF(2^5)$. Then, the square of A [redacted] as,

$$a_4a_3a_2a_1a_0 \;*\; a_4a_3a_2a_1a_0$$

$a_4a_0\ \ a_3a_0\ \ a_2a_0\ \ a_1a_0\ \ a_0a_0$ $+$

$a_4a_1\ \ a_3a_1\ \ a_2a_1\ \ a_1a_1\ \ a_0a_1$

[redacted] $a_3a_2\ \ a_2a_2\ \ a_1a_2\ \ a_0a_2$

[redacted] $a_2a_3\ \ a_1a_3\ \ a_0a_3$

[redacted] a_0a_4 $=$

[redacted] $a_2\quad 0\quad a_1\quad 0\quad a_0$

[redacted] element A in the field $F=GF(2^5)$, we have,

[redacted] $= A^2(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} a_i x^i\right) = \sum_{i=0}^{m-1} a_i x^{2i}$

Squaring: Software Solution

```
[redacted]_low[256] = {
[redacted] 5, 16, 17, 20, 21, 64, 65, 68, 69, 80, 81, 84, 85,
[redacted] 272, 273, 276, 277, 320, 321, 324, 325, 336, 337, 340, 341,
[redacted] 1040, 1041, 1044, 1045, 1088, 1089, 1092, 1093, 1104, 1105, 1108, 1109,
[redacted] 1296, 1297, 1300, 1301, 1344, 1345, 1348, 1349, 1360, 1361, 1364, 1365,
[redacted] 4112, 4113, 4116, 4117, 4160, 4161, 4164, 4165, 4176, 4177, 4180, 4181,
[redacted] 4368, 4369, 4372, 4373, 4416, 4417, 4420, 4421, 4432, 4433, 4436, 4437,
[redacted] 5136, 5137, 5140, 5141, 5184, 5185, 5188, 5189, 5200, 5201, 5204, 5205,
[redacted] 5393, 5396, 5397, 5440, 5441, 5444, 5445, 5456, 5457, 5460, 5461,
[redacted] 16400, 16401, 16404, 16405, 16448, 16449, 16452, 16453, 16464, 16465, 16468
[redacted] 16656, 16657, 16660, 16661, 16704, 16705, 16708, 16709, 16720, 16721
[redacted] 17413, 17424, 17425, 17428, 17429, 17472, 17473, 17476, 17477, 17488
[redacted] 17668, 17669, 17680, 17681, 17684, 17685, 17728, 17729, 17732, 17733
[redacted] 20481, 20484, 20485, 20496, 20497, 20500, 20501, 20544, 20545, 20548
[redacted] 20736, 20737, 20740, 20741, 20752, 20753, 20756, 20757, 20800, 20801
[redacted] 20821, 21504, 21505, 21508, 21509, 21520, 21521, 21524, 21525, 21568
[redacted] 21588, 21589, 21760, 21761, 21764, 21765, 21776, 21777, 21780, 21781
[redacted] 21841, 21844, 21845
```

Squaring: Software Implementation

```
FieldSqr2k_Random(rct_word   *ax,  rct_word   *tx,  rce_context *cntxt,
            rct_octet   *offsetptr)

index i;
ord C, S;
ex wlen, blen_p;
d *tmp;

txt->ecp->wlen;
ntxt->ecp->blen_p;

vord*) offsetptr;

p[1]=0;

n; i++) {
  sqr_table_low[(ax[i]&0xff)];
  (sqr_table_low[(ax[i]>>8)&0xff]<<16);
  sqr_table_low[(ax[i]>>16)&0xff];
  (sqr_table_low[(ax[i]>>24)&0xff]<<16);

2] = S; tmp[i*2+1] = C;

UC2K(cntxt) (tmp, blen_p, cntxt->ecp->poly);

p, blen_p, cntxt->ecp->poly);

+) tx[i] = tmp[i];
```


Second step: reduction

blem: Given the polynomial product *C(x)* with at most, *2m-1*, obtain dular product *C'* with *m* coordinates, using the generating e polynomial *P(x)*.

$$C'(x) = C(x) \bmod P(x)$$

we are interested in the polynomial reminder of the e can safely add any multiple of $P(x)$ to $C(x)$ without d result. This simple observation suggest the following reduce k bits of the polynomial product *C* at once.

Second step: reduction

...s assume that the *m+1* and *2m-1* coordinates of $P(x)$ and $C(x)$, ...ely, are distributed as follows:

$$C = [c_{2m-2} \quad c_{2m-3} \quad \dots \quad c_{2m-1-k} \quad c_{2m-2-k} \quad \dots \quad c_1 \quad c_0]$$

$$= [p_m \quad p_{m-1} \quad \cdots \quad p_1 \quad p_0]$$

...ays exists a *k*-bit constant scalar *S*, such that

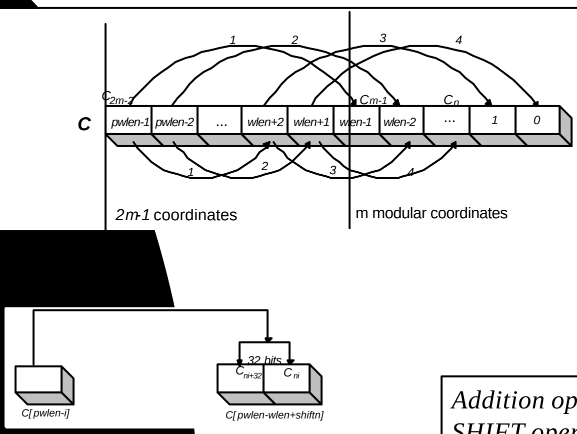
$$[p_m \quad p_{m-1} \quad \dots \quad p_{m-k+1} \quad p_{m-k} \quad \dots \quad p_1 \quad p_0]$$

$$\dots_{2m-2} \quad c_{2m-3} \quad \dots \quad c_{2m-k-1} \quad p'_{m-k} \quad \dots \quad p'_1 \quad p'_0]$$

...otice that all the *k* MSB of *SP* become identical to the ... of the number *C*. By left shifting the number *SP* ...-*k-1* positions, we effectively reduce the number in

Francisco Rodríguez Henríquez

Software reduction implementation

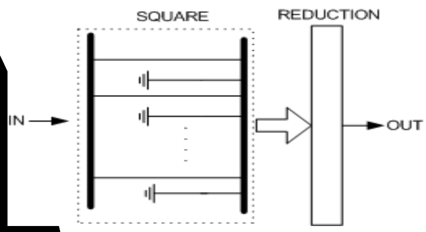


$$wlen := \frac{m}{w}$$

Addition operations < 4wlen;
SHIFT operations < 4wlen;
Comparisons = 2wlen.

Francisco Rodríguez Henríquez

Squaring: Polynomial Multiplication Step FPGA Implementation [by Nazar Saqib]

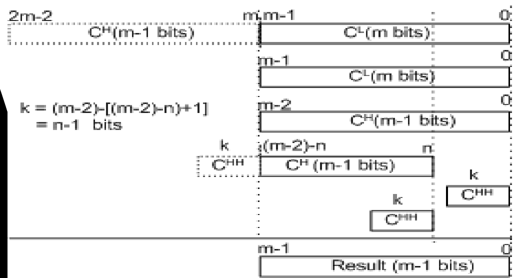


$$A = a_3x^3 + a_2x^2 + a_1x + a_0$$
$$A^2 = a_6x^6 + a_4x^4 + a_2x^2 + a_0$$

A = 1111
A² = 1010101

Francisco Rodríguez Henríquez

Squaring: Reduction Step FPGA Implementation [by Nazar Saqib]



Francisco Rodríguez Henríquez

Case of Study: Modular Exponentiation

Francisco Rodríguez Henríquez

But why are we interested in modular exponentiation in the first place?

Francisco Rodríguez Henríquez

RSA cryptosystem by layers

Protocols and Applications: SSL, TLS, WTLS, WAP, etc.

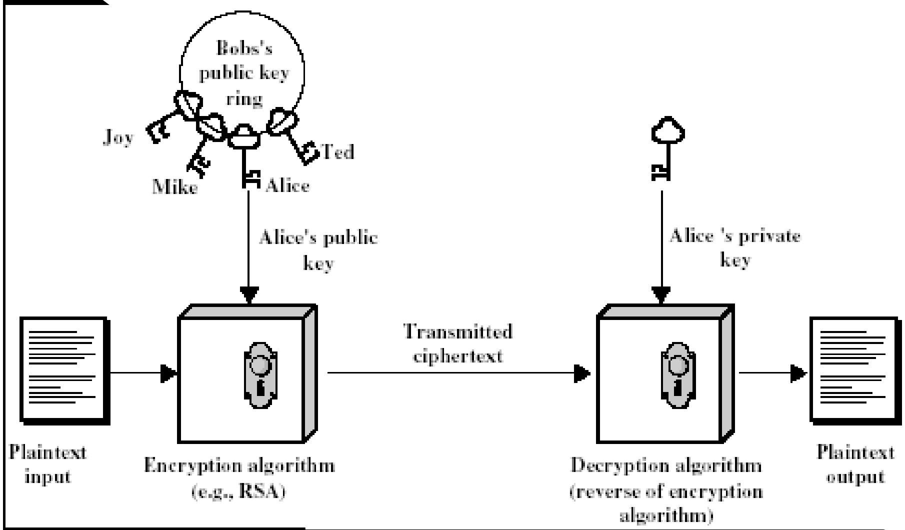
PKCS User Functions:PKCS1_OAEP_Encrypt, PKCS1_OAEP_Decrypt, PKCS1_v15_Sign,

PKCS Primitives: PKCS1_OAEP_Encode, PKCS1_OAEP_Decode, etc

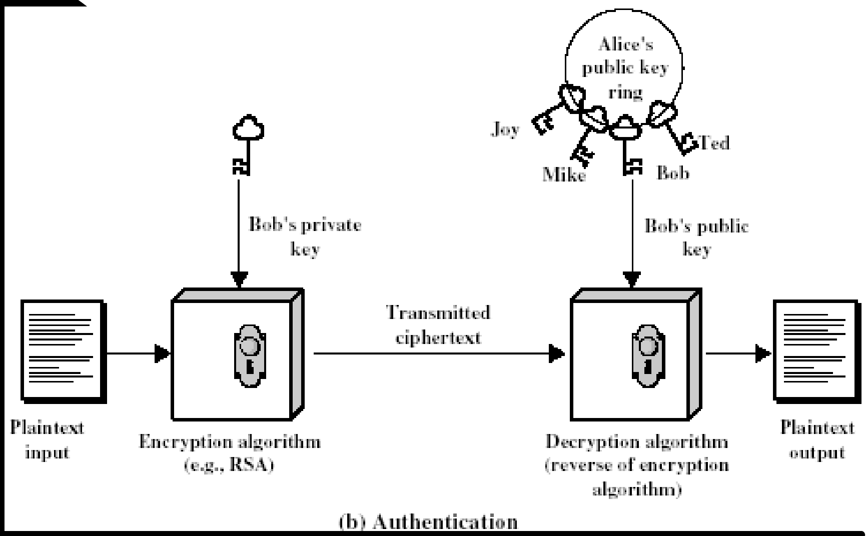
RSA primitive Operations: Encryption: $C = M^e \bmod n$, Decryption $M = C^d \bmod n$.

F_P finite field operations : Addition, Squaring, multiplication, inversion and exponentiation

Public-Key Cryptography



Public-Key Cryptography



Francisco Rodríguez Henríquez

RSA: Key Generation

Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d = e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Francisco Rodríguez Henríquez

RSA: Encryption, Decryption

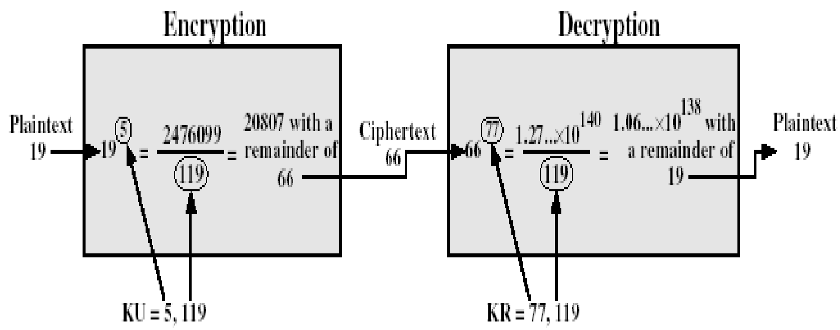
Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

RSA: An Example



Example of RSA Algorithm

Modern Cryptosystems: A Top-Down Model

Applications: e-commerce, smart cards, digital money, secure communications, etc.

Crypto-protocols: Diffie-Hellman, authentication protocols, etc.

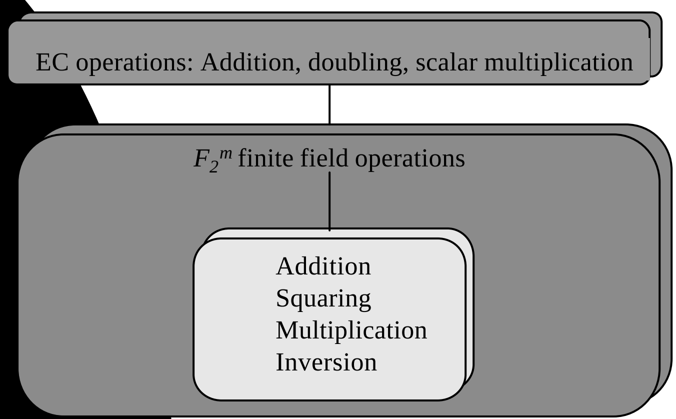
Top level Crypto-primitives: Key-pair generation, Signing and Verification

Low-level crypto-primitives: addition, doubling, scalar multiplication

F_{2^m} finite field operations : Addition, Squaring, multiplication and inversion

Francisco Rodríguez Henríquez

Elliptic curves over finite fields



Francisco Rodríguez Henríquez

Arithmetic on Elliptic Curves

…dition and Doubling

- $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, then $P + Q = (x_3, y_3)$
- $x_3 = \lambda^2 - x_1 - x_2$
- $y_3 = \lambda(x_1 - x_3) - y_1$

$(y_2 - y_1) / (x_2 - x_1)$ for $P \neq Q$

$(3x_1^2 + a) / 2y_1$ for $P = Q$ (doubling)

…lication

$P+P+\dots+P$ ——— k times

…ormed in the finite field $F=GF(2^m)$ over $K=GF(2)$.

…lliptic curves requires addition, squaring, …nd inversion in finite fields.

Francisco Rodríguez Henríquez

Case of Study: Modular Exponentiation

Francisco Rodríguez Henríquez

Modular Exponentiation

Ve do ***NOT*** compute $C := M^e \bmod n$

first computing M^e

then computing $C := (M^e) \bmod n$

rary results must be reduced modulo

each step of the exponentiation.

Francisco Rodríguez Henríquez

Modular Exponentiation

M^{15}

ow many multiplications are needed??

wer (requires 14 multiplications):

$M^3 \rightarrow M^4 \rightarrow M^5 \rightarrow \ldots \rightarrow M^{15}$

od (requires 6 multiplications):

$^3 \rightarrow M^6 \rightarrow M^7 \rightarrow M^{14} \rightarrow M^{15}$

Francisco Rodríguez Henríquez

Modular Exponentiation: Binary Method

e ***binary method*** requires:

quarings: *k*-1

ltiplications: The number of 1s in the binary nsion of *e*, excluding the MSB.

umber of multiplications:

(*k*-1) + (*k*-1) = 2(*k*-1)

(*k*-1) + 0 = *k*-1

k-1) + 1/2 (*k*-1) = 1.5(*k*-1)

Francisco Rodríguez Henríquez

Modular Exponentiation

scanning the bits of *e*

time: **quaternary method**

me: **octal method**

: ***m*-ary method**.

e quaternary method: 250 = <u>11</u> <u>11</u> <u>10</u> <u>10</u>

ocessing required.

2 squaring performed.

Francisco Rodríguez Henríquez

Modular Exponentiation: Quaternary Method

[redacted]ample:

[redacted]	bits	j	M^j
[redacted]	00	0	1
[redacted]	01	1	M
[redacted]	10	2	$M \times M = M^2$
[redacted]	11	3	$M^2 \times M = M^3$

Modular Exponentiation: Quaternary Method

[redacted]ample: e = 250 = 11 11 10 10

[redacted]	bits	*Step 2a*	*Step 2b*
[redacted]	11	M^3	M^3
[redacted]	[redacted]	$(M^3)^4 = M^{12}$	$M^{12} \times M^3 = M^{15}$
[redacted]	[redacted]	$(M^{15})^4 = M^{60}$	$M^{60} \times M^2 = M^{62}$
[redacted]	[redacted]	$(M^{62})^4 = M^{248}$	$M^{248} \times M^2 = M^{250}$

[redacted] of multiplications: 2+6+3 = 11

Modular Exponentiation: Average Number of Multiplications

[redacted]	BM	MM	d	Savings %
[redacted]	11	10	2	9.1
[redacted]	23	21	2	8.6
[redacted]	47	43	2, 3	8.5
[redacted]	95	85	3	10.5
[redacted]	191	167	3, 4	12.6
[redacted]	383	325	4	15.1
[redacted]	767	635	5	17.2
[redacted]	[redacted]535	1246	5	18.8
[redacted]	[redacted]071	2439	6	20.6

Addition Chains

[redacted] a sequence of integers $a_0, a_1, a_2, \ldots, a_r$ [redacted] and $a_r = e$. The sequence is constructed in such a way [redacted] k there exist indices $i, j = k$ such that, $a_k = a_i + a_j$.

[redacted] chain is r. A short chain for a given e implies an [redacted] rithm for computing M^e.

[redacted] BM: 1 2 3 6 12 13 26 27 54 55

[redacted] QM: 1 2 3 6 12 13 26 52 55

[redacted] M: 1 2 4 5 10 20 40 50 55

[redacted] M: 1 2 3 5 10 11 22 44 55

Addition Chains

ding the shortest addition chain is NP-complete.

bound is given by binary method:

$$\lfloor \log_2 e \rfloor + H(e) - 1$$

he *Hamming* weight of *e*.

given by Schönhage:

$$\lfloor \log_2 e \rfloor + H(e) - 2.13$$

ary, m-ary, adaptive m-ary, sliding windows,

or.

Francisco Rodríguez Henríquez

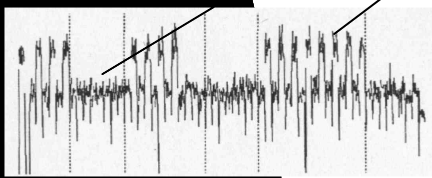
Modular Exponentiation: Binary Method Variations

Francisco Rodríguez Henríquez

Side Channel Attacks

[redacted]inary exponentiation
[redacted] exponent d = (d_k,d_{k-1},…,d_0)
[redacted]s the most significant bit)
[redacted]G
[redacted]n to 0;
[redacted]= c*a;

The time or the power to execute c^2 and c*a are different (side channel information).



Algorithm Coron's exponentiation
Input: a in G, exponent d = (d_k,d_{k-1},…,d_{l0})
Output: c = a^d in G
1. c[0] = 1;
2. For i = k-1 down to 0;
3. c[0] = c[0]^2;
4. c[1] = c[0]*a;
5. c[0] = c[d_i];
6. Return c[0];

Francisco Rodríguez Henríquez

Mod. Exponentiation: LSB-First Binary

[redacted] be the number of bits of *e*, i.e.,

$$k = 1 + \lfloor \log_2 e \rfloor$$

[redacted]*e*, *n*.

$$e = (e_{k-1}e_{k-2}\ldots e_1 e_0) = \sum_{i=0}^{k-1} e_i 2^i$$

[redacted]= M^e mod *n*

for $e_i \in \{0,1\}$

[redacted]:= *M;*
[redacted]to *n*-1
[redacted]1 then *R* := *R*·*C* mod *n*
[redacted]mod *n*

Francisco Rodríguez Henríquez

Modular Exponentiation: LSB First Binary

[redacted] = 250 = (11111010), thus k = 8

[redacted]	e_i	Step 3 (R)	Step 4 (C)
[redacted]	0	1	M^2
[redacted]	1	$1*(M)^2 = M^2$	$(M^2)^2 = M^4$
[redacted]	0	M^2	$(M^4)^2 = M^8$
[redacted]	1	$M^2 * M^8 = M^{10}$	$(M^8)^2 = M^{16}$
[redacted]	1	$M^{10} * M^{16} = M^{26}$	$(M^{16})^2 = M^{32}$
[redacted]	1	$M^{26} * M^{32} = M^{58}$	$(M^{32})^2 = M6^4$
[redacted]	1	$M^{58} * M^{64} = M^{122}$	$(M^{64})^2 = M^{128}$
[redacted]	1	$M^{122} * M^{128} = M^{250}$	$(M^{128})^2 = M^{256}$

Francisco Rodríguez Henríquez

Modular Exponentiation: LSB First Binary

[redacted]he LSB-First ***binary method*** requires:

[redacted]quarings: k-1

[redacted]ltiplications: The number of 1s in the binary [redacted]nsion of e, excluding the MSB.

[redacted] number of multiplications:

[redacted]: (k-1) + (k-1) = 2(k-1)

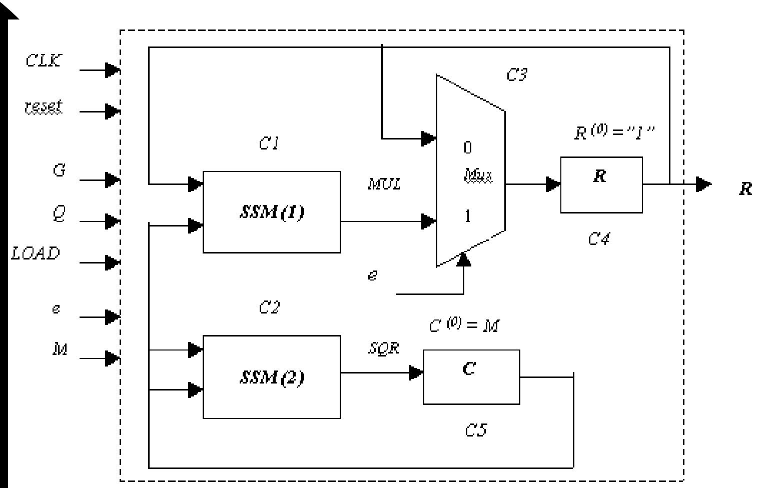
[redacted] (k-1) + 0 = k-1

[redacted] (k-1) + 1/2 (k-1) = 1.5(k-1)

[redacted]efore, but here we can compute the [redacted]ation operation in parallel with the [redacted]!!

Francisco Rodríguez Henríquez

Arquitectura del Multiplicador [Mario García et al ENC03]



Desarrollo (Método q-ario)

Input: $\mathbf{x} \in GF(2^m)$: Element in the Galois field
e: Exponent
$r > 1$: Integer

Output: $\mathbf{z} = \mathbf{x}^e \in GF(2^m)$

{ $N := e$; $\mathbf{x_2} := \mathbf{x} \cdot \mathbf{x}$; $\mathbf{w}[1] := \mathbf{x}$;
  for $i = 2$ **to** $\lfloor \frac{r}{2} \rfloor$ **do** $\mathbf{w}[i] := \mathbf{w}[i-1] \cdot \mathbf{x_2}$;
  $\mathbf{z} := 1$;
  while $N > 0$ **do**
  { $d :=$ Mod$[N, r]$;
    $N := \lfloor \frac{N}{r} \rfloor$;
    if $d \neq 0$ **then**
    { express $d = 2^p q$ with p greatest possible integer, q odd ;
      $i := (q+1)/2$;
      $\mathbf{z_1} := \mathbf{w}[i]$;
      $\mathbf{z} := \mathbf{z} \cdot \mathbf{z}_1^{2^p}$
    } ;
    for $i = 1$ **to** $\lfloor \frac{r}{2} \rfloor$ **do** $\mathbf{w}[i] := \mathbf{w}[i]^r$
  } ;
  output z
}

Desarrollo (Método q-ario)

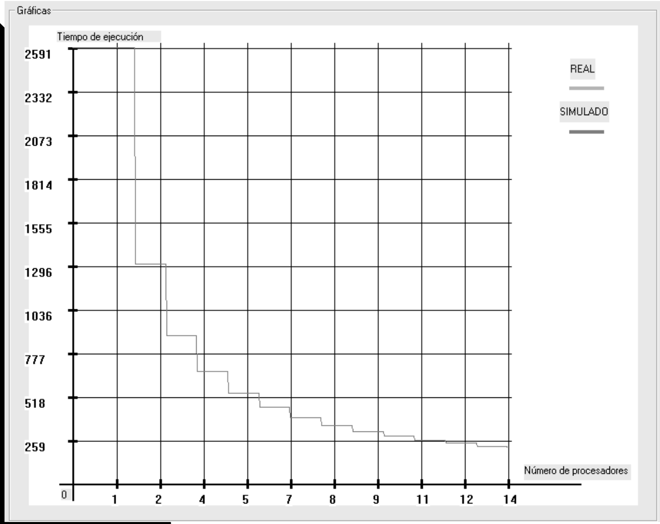
- [redacted]ecálculo de W.
- [redacted]ño de q.
- [redacted] de d = 2^p * q

Desarrollo (Análisis)

- [redacted]maño de memoria y tiempo de [redacted]ución del precómputo W.
- [redacted]o de multiplicaciones y [redacted]ones al cuadrado para método q-[redacted]

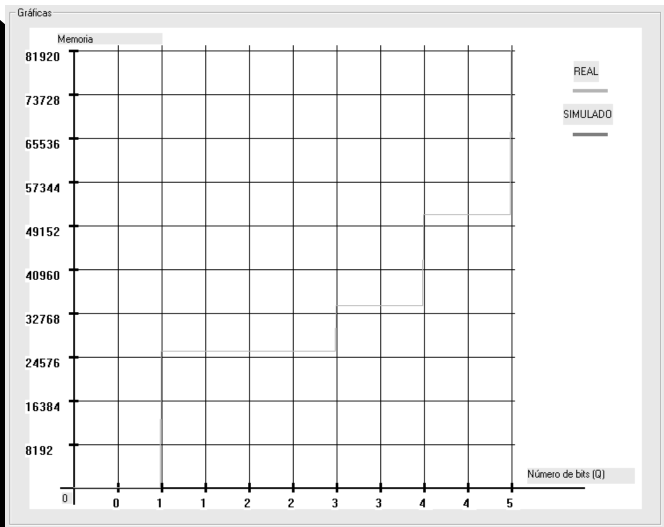
Francisco Rodríguez Henríquez

Tiempo de Ejecución Vs. Número de Procs.



Francisco Rodríguez Henríquez

Tamaño de Memoria

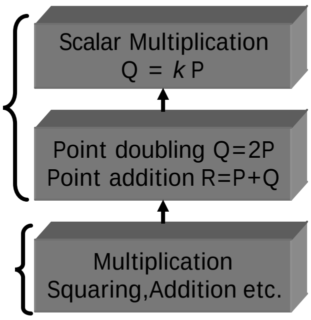


Francisco Rodríguez Henríquez

Elliptic Curve Point Multiplication Revisited

Francisco Rodríguez Henríquez

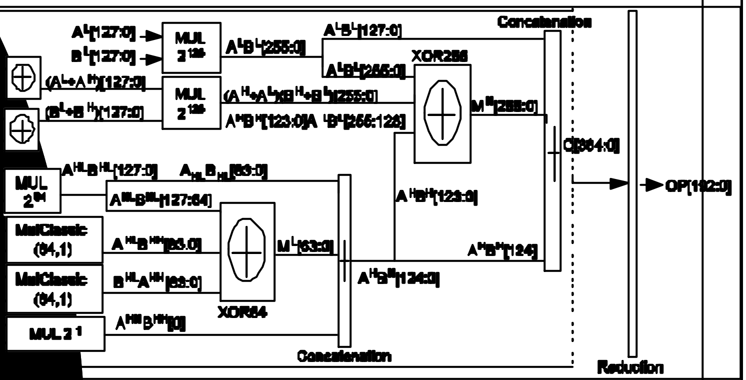
Elliptic Curve Cryptography



Francisco Rodríguez Henríquez

First Layer: Field Multiplication

[illegible]reliminary results yield a time delay of 50-70 ηSec [illegible] and ≈9K Slices of hardware resources utilization.



EC Point Addition and Doubling: A model

[illegible]tic Curve [illegible]rm	Point Addition	Point Doubling	*# of Multipliers*
[illegible]e [illegible]s	9M	4M	1
[illegible]	12M	6M	1
[illegible]	6M	3M	2
[illegible]	4M	2M	3
[illegible]	4M	2M	1
[illegible]	2M	1M	2

EC Point Multiplication: A model

…orithm	Number of Point Addition	Number of Point Doubling	PA and PD Boxes
[illegible]	1/2m	m	Sequential
[illegible]	1/2m	1/2m	Parallel
[illegible]	1/3m	m	Sequential
[illegible]	1/3m	2/3m	Parallel
[illegible]	m	m	Sequential
[illegible]	m	0	Parallel

Francisco Rodríguez Henríquez

EC Point Multiplication: A model

Method	Total Number of Field Multiplications	Number of Multipliers
…roj	9/2mM+4mM = 8.5mM	1
…n2	3mM+3mM=6mM	2
…1	6mM+3mM=9mM	2
…2	2mM+3mM=5mM	2
…ar	2mM+1mM=3mM	2

Francisco Rodríguez Henríquez