Laboratorio di Crittografia

(Crypto Engineering)

Prof. Mario Di Raimondo

a.a. 2016–2017

C.d.L. in Informatica (magistrale)
Dipartimento di Matematica e Informatica — Catania

Contatti

- docente: Prof. Mario Di Raimondo
  - email: `diraimondo(at)dmi.unict.it`
  - home-page: `http://www.dmi.unict.it/diraimondo/`
  - telefono: 095 738 3038
  - ufficio: MI-32 — Blocco III — DMI
- pagina del corso:
  `http://www.dmi.unict.it/diraimondo/web/teaching/crypto/`
- Studium
- calendario (lezioni, ricevimento, esami)
- gruppo Facebook del DMI
- FAQ

Il Corso

durata 6 CFU (48 ore)

requisiti
- basi teoriche sulla moderna Crittografia
- buone capacità di programmazione
- infarinatura di networking

propedeuticità Crittografia

esame assegnamento di un articolo scientifico in lingua originale (no dimostrazioni):
- ~~mini-seminario (20 minuti circa)~~
- progetto implementativo
- colloquio orale (progetto e programma del corso)

Ciphertext Indistinguishability

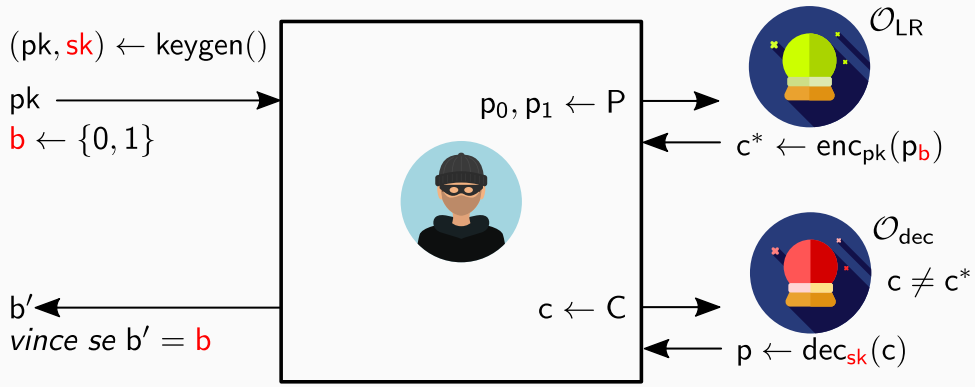
Il gioco **IND** prevede due fasi:

find alla fine di questa fase l'avversario sceglie due messaggi p_0, p_1 di uguale lunghezza;

guess all'avversario è dato c^*, la cifratura di p_b, dove b è un bit segreto a lui nascosto; il suo obiettivo è indovinare b con probabilità maggiore di $\frac{1}{2}$.

Combinato con i vari attacchi: **IND-PASS**, **IND-CPA** e **IND-CCA**.

Guardiamo alla versione asimmetrica di **IND-CCA**:



La versione simmetrica ha in più l'oracolo $\mathcal{O}_{\mathsf{enc}}$.

Curve Ellittiche

curva ellittica (forma normale di Weierstrass) su $\mathbb{Z}_p$

La figura geometrica descritta dai punti $(x, y) \in \mathbb{Z}_p^2$, con $p > 3$, che soddisfano la seguente congruenza:

$$y^2 \equiv x^3 + a \cdot x + b \bmod p$$

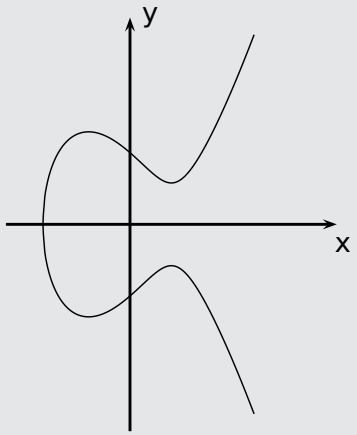
dove $a, b \in \mathbb{Z}_p$ e per cui vale la condizione $4 \cdot a^3 + 27 \cdot b^2 \not\equiv 0 \bmod p$ (la curva non è super-singolare).
Aggiungiamo anche un punto all'infinito denotato da 0 (zero).

esempio su $\mathbb{R}$

curva $y^2 = x^3 - 3 \cdot x + 3$ in $\mathbb{R}$

osservazioni:

- simmetrica rispetto all'asse x;
- tocca l'asse x in un punto;
- ogni retta tocca la curva in 3 punti (al limite 2 all'infinito).



Assembliamo il Gruppo

elementi le coppie di punti $(x, y) \in \mathbb{Z}_p$ che soddisfano l'equazione caratterizzante;

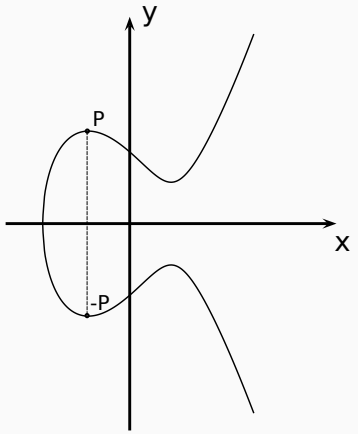
identità utilizziamo il punto all'infinito (zero);

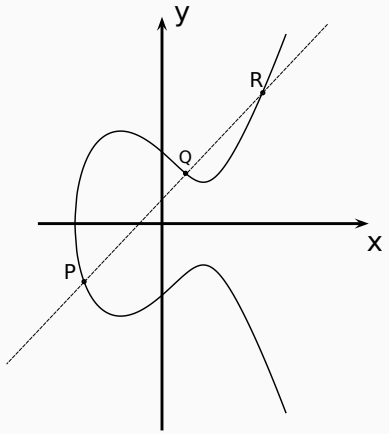
inversione dato un punto sulla curva, prendiamo il suo speculare rispetto all'asse x;

operazione definiamola in base alla seguente regola:

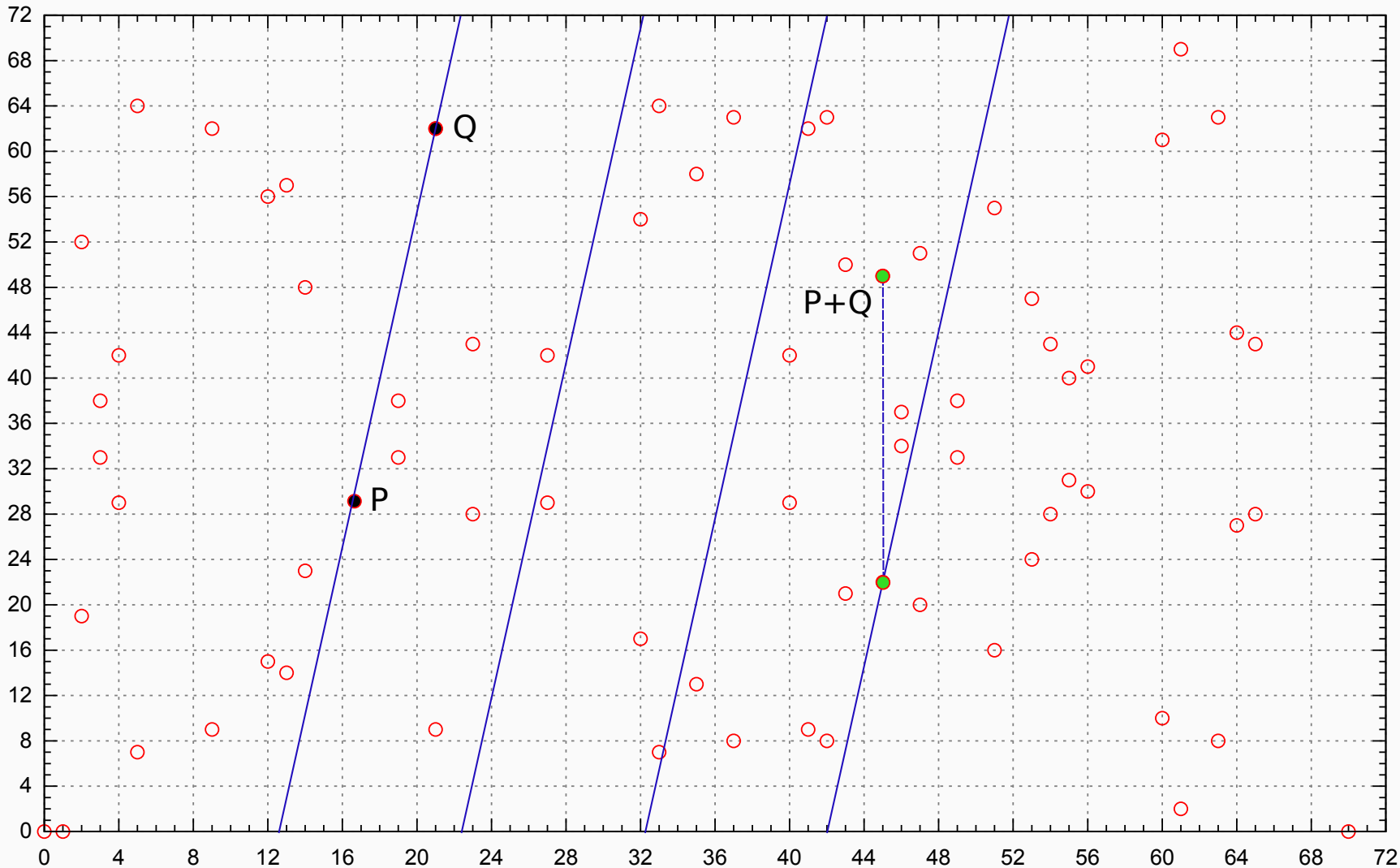
> dati 3 punti allineati non nulli P, Q, R, la loro somma deve dare zero:
> $$P + Q + R = 0$$

Si tratta di un gruppo abeliano.





Addizione in Versione Geometrica Discreta



Esponenziazione a Rango k

Generalizziamo elaborando k bit alla volta, lavorando su base $b = 2^k$

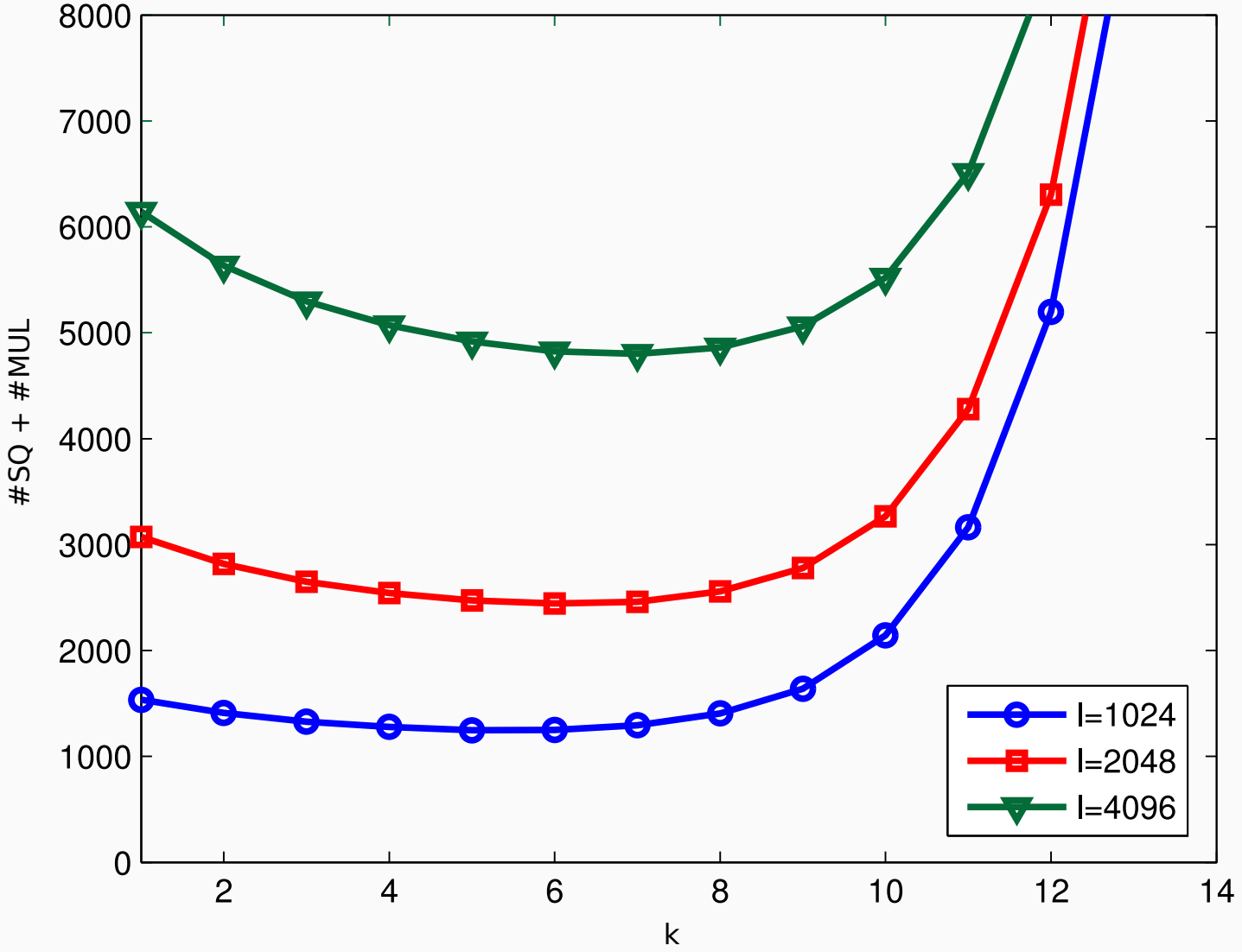
algoritmo di esponenziazione a rango k

Input: base g, modulo n, esponente $e = (E_t E_{t-1} \dots E_1 E_0)_b$ con $b = 2^k$ per qualche $k \geq 1$ e $E_i \in \{0, 1, \dots, 2^k - 1\}$

Output: $g^e \bmod n$

1 $G_0 \leftarrow 1$
2 **for** i **from** 1 **to** $2^k - 1$ **do**
3 $\quad G_i \leftarrow G_{i-1} \cdot g \bmod n$
4 $A \leftarrow G_{E_t}$
5 **for** i **from** $t - 1$ **downto** 0 **do**
6 $\quad A \leftarrow A^{2^k} \bmod n$
7 $\quad A \leftarrow A \cdot G_{E_i} \bmod n$
8 **return** A

Esponenziazione a Rango k: Analisi

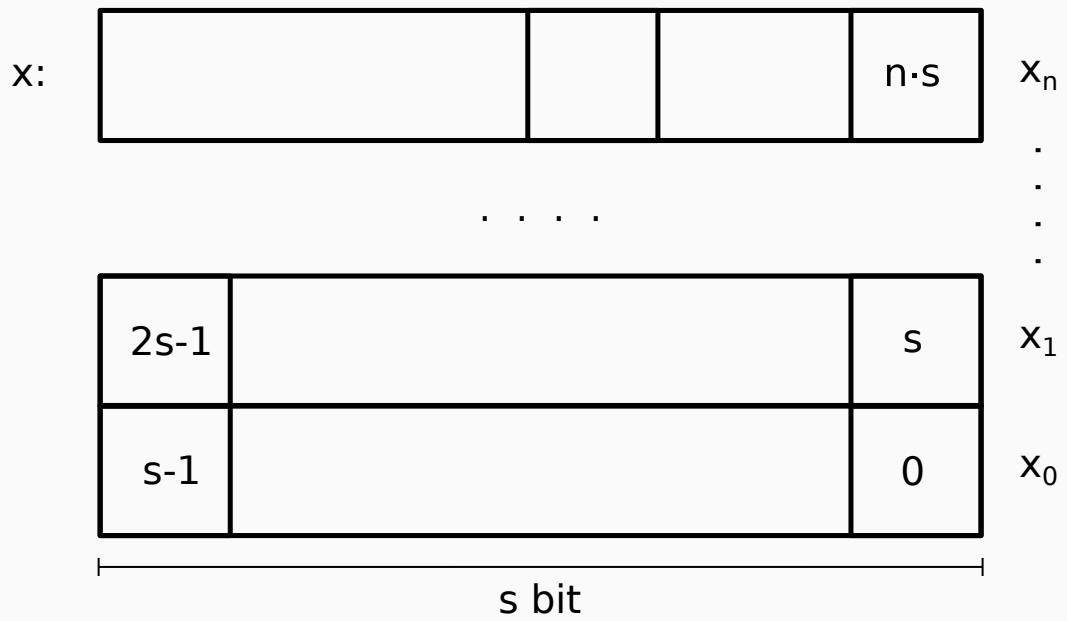


Multi-Precision Interger e Loro Rappresentazione

multi-precision integer (mpi)

Aritmetica su numeri più grandi della word di base del calcolatore

Rappresentazione tramite word da s bit di un MPI:



$$x = (x_n, x_{n-1}, \ldots, x_1, x_0) = x_n b^n + x_{n-1} b^{n-1} + \cdots + x_1 b^1 + x_0 b^0$$

Le operazioni sugli x_i sono operazioni atomiche a singola-precisione.

Misurare con Precisione il Tempo

Motivazioni:

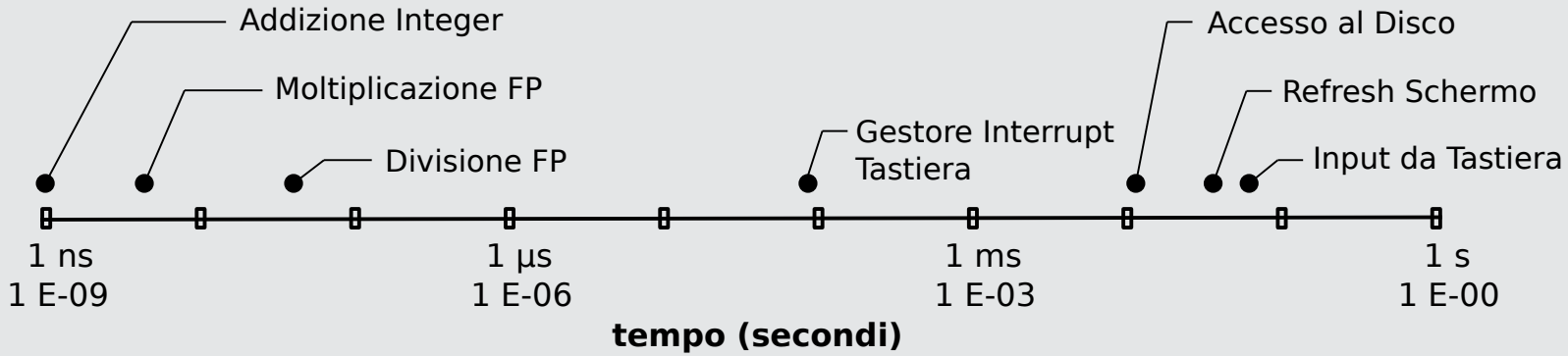
- comparare le prestazioni di due primitive;
- misurare il throughput di una primitiva;
- timing attack.

perchè?

L'analisi della complessità asintotica algoritmica ci da importanti indicazioni ma per disambiguare certi scenari bisogna misurare il livello di ingegnerizzazione delle implementazioni.

scala dei tempi su un elaboratore

riferimento: una macchina ad 1 GHz



Esponenziazione Modulare

esponenziazione modulare

```
void mpz_mod(mpz_t r, mpz_t n, mpz_t d);
int  mpz_invert(mpz_t rop, mpz_t op, mpz_t mod);
void mpz_powm(mpz_t rop, mpz_t base, mpz_t exp, mpz_t mod);
void mpz_powm_ui(mpz_t rop, mpz_t base, unsigned long int exp,
                 mpz_t mod);
```

algoritmo impiegato

Algoritmo a finestra fissa (a rango k) nel dominio di Montgomery:
dimensione che cresce con la taglia dell'esponente a scaglioni prestabiliti:

finestra (bit)	2	3	4	5	6	7	8	...
dim. max esp (bit)	25	81	241	673	1793	4609	11521	...

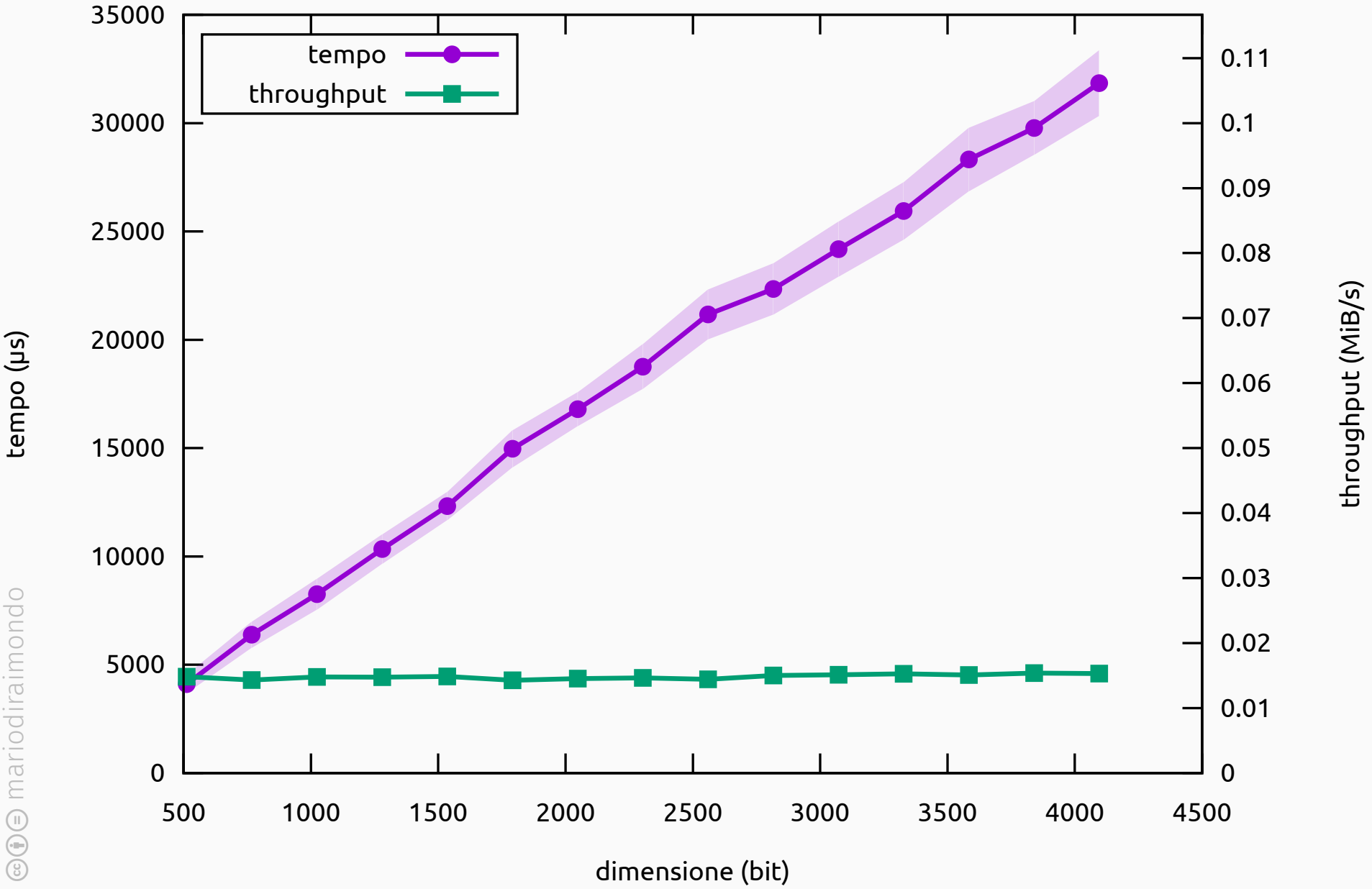
studio: `test-mpz-powm.c`

variante sicura

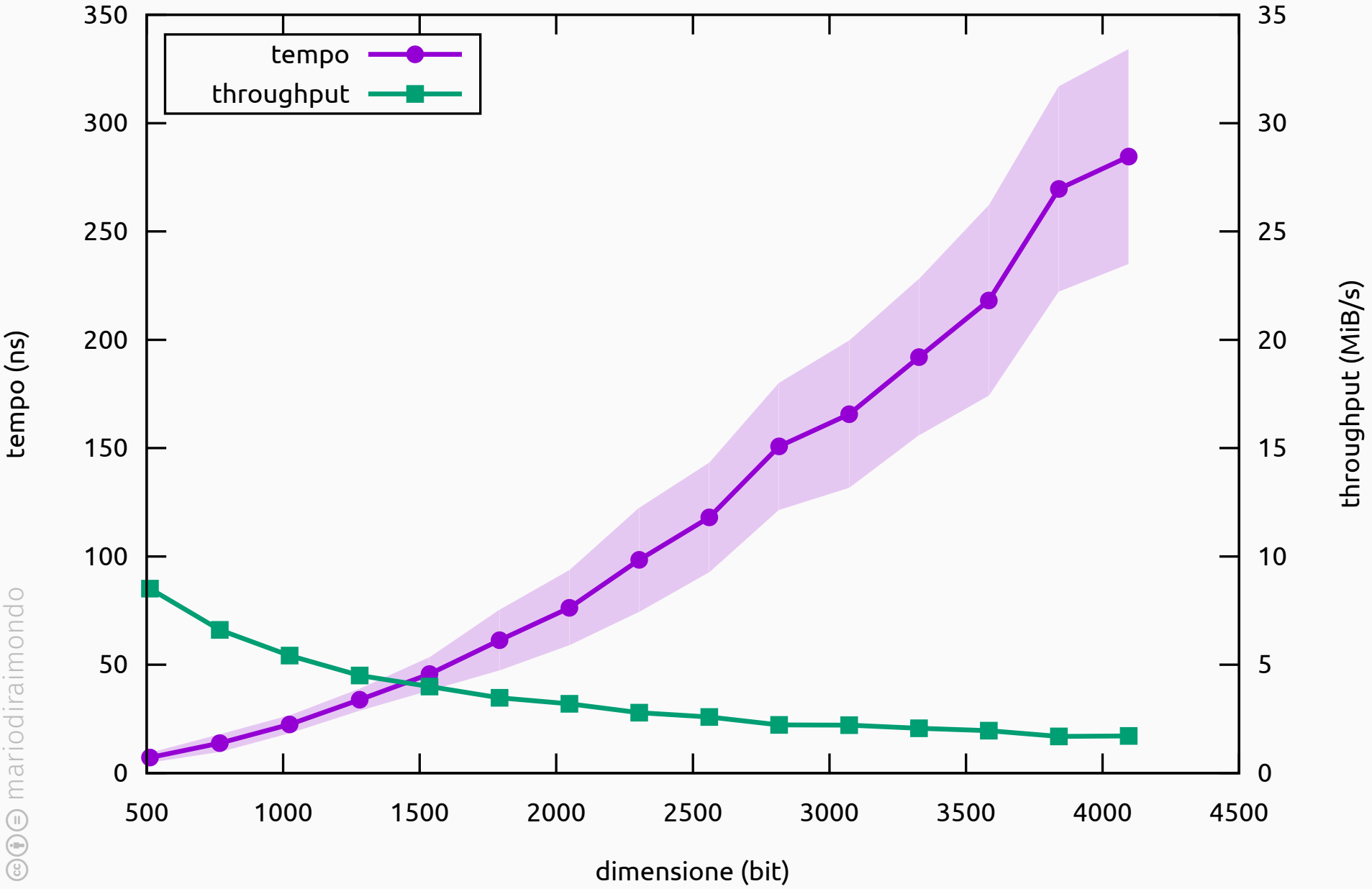
```
void mpz_powm_sec(mpz_t rop, mpz_t base, mpz_t exp, mpz_t mod);
```

studio: `test-mpz-powm-sec.c`

Esponenziazione Modulare GMP tipo MPI^{MPI}: Prestazioni



Esponenziazione Modulare GMP tipo MPI^{SPI}: Prestazioni



Esponenziazione Modulare con Precomputazione

La libreria GMP non offre alcuna implementazione che sfrutti la precomputazione off-line.

libreria: `lib-powm.c/.h`

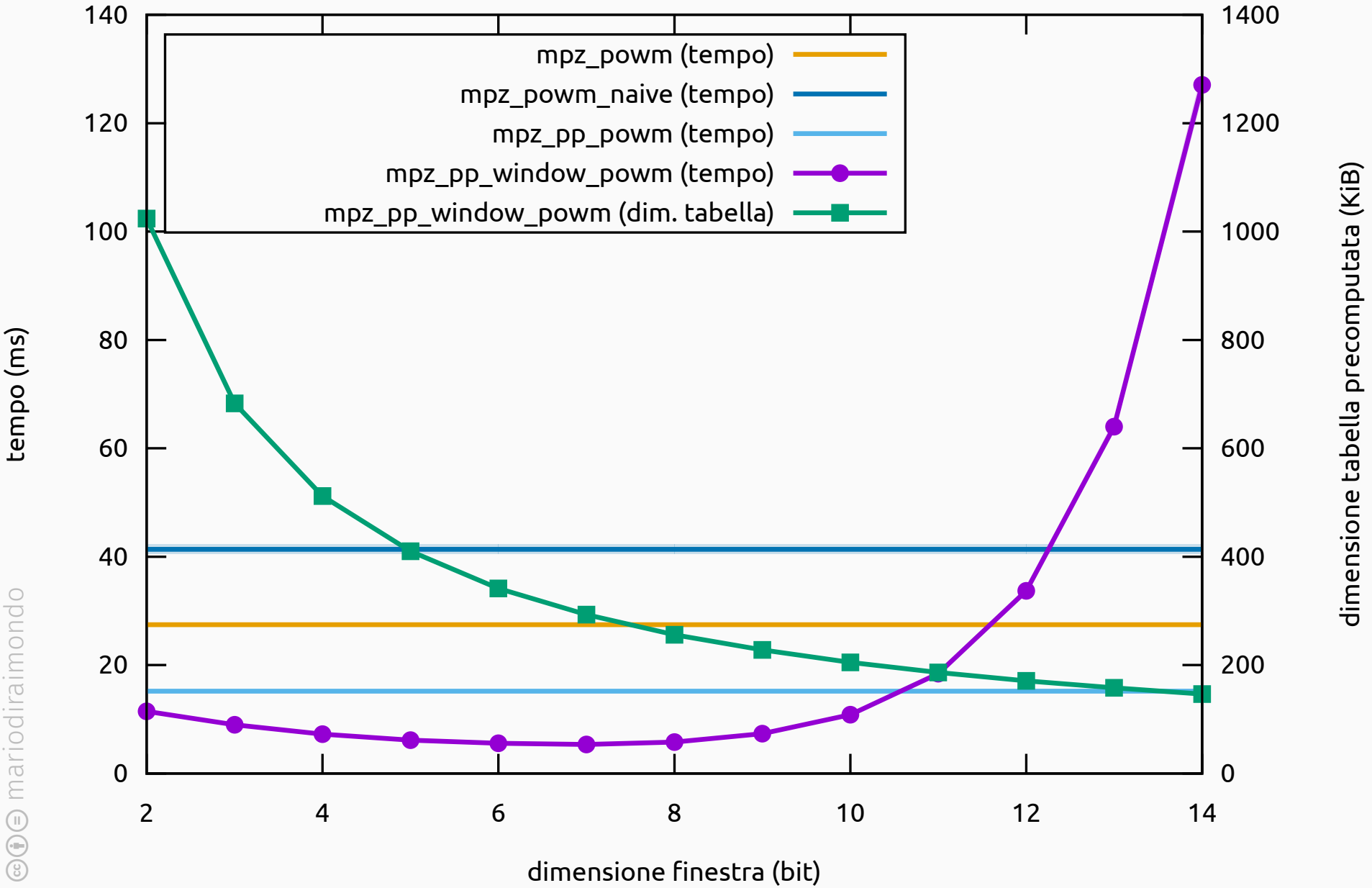
metodi aggiuntivi

```
void mpz_pp_powm_init(mpz_pp_powm_t pp, size_t exp_bits,
                      mpz_t base, mpz_t mod);
void mpz_pp_powm(mpz_t res, mpz_t exp, mpz_pp_powm_t pp);
void mpz_pp_powm_clear(mpz_pp_powm_t pp);

void mpz_pp_window_powm_init(mpz_pp_window_powm_t pp, size_t
                             exp_bits, size_t window_size,
                             mpz_t base, mpz_t mod);
void mpz_pp_window_powm(mpz_t res, mpz_t exp,
                        mpz_pp_window_powm_t pp);
void mpz_pp_window_powm_clear(mpz_pp_window_powm_t pp);
```

studio: `test-mpz-pp-w-powm.c`

Esponenziazione Mod. con Precomputazione: Prestazioni



Seed Random

Qualunque PRNG ha bisogno di seed veramente random e non predicibili. Possibi fonti:

- clock di sistema: poco sicuro;
- collezionare entropia:
  - a livello di applicazione: poco pratico ma plausibile;
  - a livello di sistema: gestito direttamente dal Sistema Operativo;
  - a livello di servizio: ad es. EGD (Entropy Gathering Daemon).

Interfacce offerte per il pool di sistema:

Linux dispositivi virtuali `/dev/random` e `/dev/urandom`;

> **quanta entropia ho?**

```
cat /proc/sys/kernel/random/entropy_avail
```

Mac OS X come Linux ma con una semantica unificata (non bloccante);

Windows esiste l'API WIN32 `CryptGenRandom`.

libreria: `lib-misc.c/.h` esempio: `test-random.c`

Attacco Basato sull'Oracolo di Padding (PKCS#1 v1.5)

idea di base (Bleichenbacher)

Avendo un oracolo $\mathcal{O}_{PKCS}(c)$ che mi dice se il messaggio cifrato $c = \mathbf{enc}_{pk}(m)$ è *PKCS#1 v 1.5 conforming*, potrei dedurre alcune informazioni sul messaggio m.

Altro attacco teorico? Tutt'altro: SSL 3 espone un oracolo simile!

- se $\mathcal{O}_{PKCS}(c) = \checkmark$, allora $m \in [2B, 3B - 1]$, dove $B = 2^{8(k-2)}$ e $k = |n|$ in byte;
- se troviamo un s intero tale che $\mathcal{O}_{PKCS}(c \cdot s^e) = \checkmark$, allora anche $m \cdot s \in [2B, 3B - 1]$: questo mi da ulteriori informazioni su m!

L'algoritmo itera continuamente 2 passi di base:

passo 1 ricerca possibili s per cui $\mathcal{O}_{PKCS}(c \cdot s^e) = \checkmark$;
passo 2 dato un tale s, restringe l'intervallo di ricerca per m.

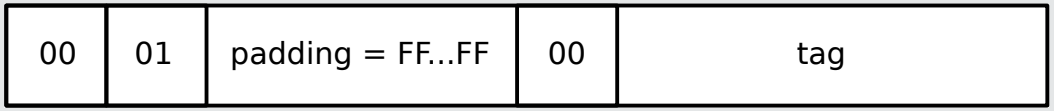
Passi extra se il messaggio cifrato originale non è già conforme.

esempio: `attack-rsa-padding-oracle.c`

Padding per Firme RSA

padding disponibili

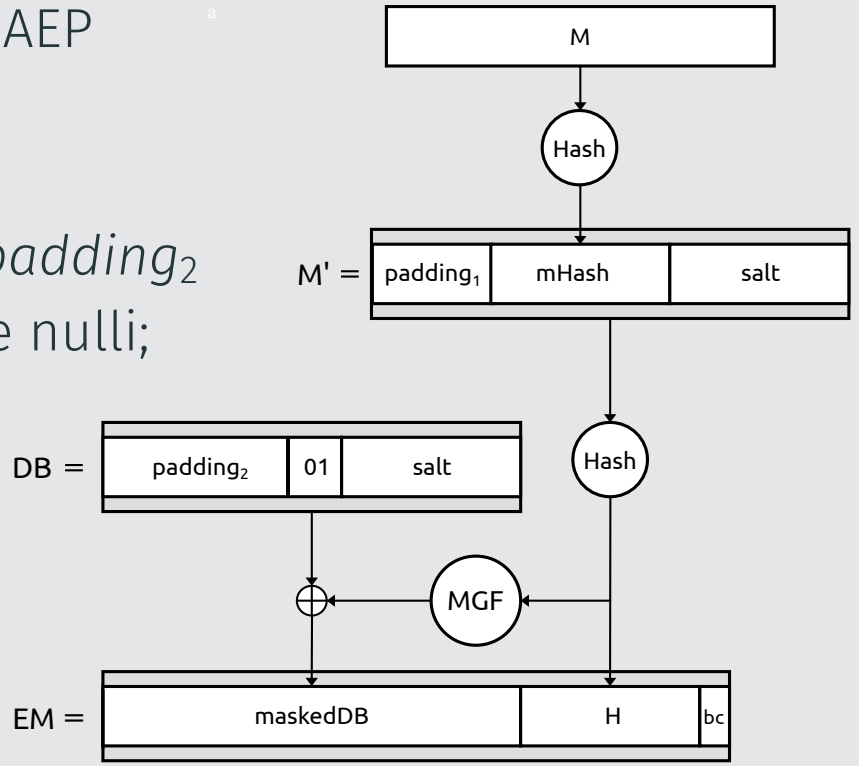
PKCS#1 v1.5 similare a quello visto per la cifratura con



$|padding| \geq 8$ e *tag* un encoding particolare del digest del messaggio e del tipo di funzione utilizzata;

EMSA-PSS similare a EME-OAEP in PKCS# v2.0

con $padding_1$ e $padding_2$ composti da byte nulli;



Firma DSA

Variante dello schema precedente facente parte dello standard Digital Signature Standard (DSS) del NIST.

schema

$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathbf{keygen}()$ dato p primo (L bit) con q primo (N bit) divisore di $p-1$,
sia g un generatore del sottogruppo G_q ordine q,
si scelga a caso $d \in [2, \ldots, q-1]$,
si calcoli $y = g^d \bmod p$
$\Rightarrow \mathrm{pk} = (p, q, g, y), \quad \mathrm{sk} = (d)$

$(r, s) \leftarrow \mathbf{sig}_{\mathrm{sk}}(m)$ si scelga a caso $k \in [1, \ldots, q-1]$,
si calcoli $r = (g^k \bmod p) \bmod q$,
$s = (\mathbf{hash}(m) + dr) \cdot k^{-1} \bmod q$

$\mathbf{ver}_{\mathrm{pk}}(m, (r, s))$ si calcoli $w = s^{-1} \bmod q$ e $\mathbf{hash}(m)$,
si calcoli $u_1 = w \cdot \mathbf{hash}(m) \bmod q$, $u_2 = w \cdot r \bmod q$,
si verifichi che $(g^{u_1} \cdot y^{u_2} \bmod p) \stackrel{?}{\equiv} r \bmod q$

Firma DSA — Criticità

L'utilizzo di un buon PRNG con le firme DSA è un requisito fondamentale.

attacco

Avendo due messaggi diversi m_1, m_2 firmati riutilizzando lo stesso segreto effimero k, vediamo come è possibile recuperare la chiave di firma d!

Ma figurati se può succedere in pratica... Sony PS3 hack! 🔍🔍

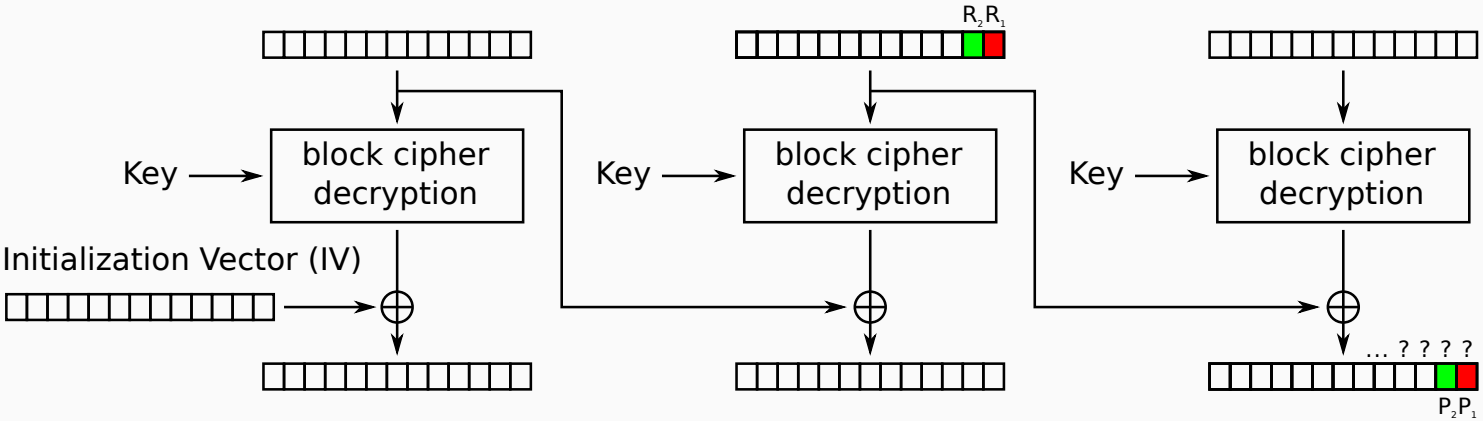
contromisure

- essere sicuri di usare un buon PRNG;
- rendere deterministico lo schema: $k \leftarrow \mathbf{hash}(sk||m)$; 🔍
- un approccio ibrido nel caso in cui lo PRNG facesse cilecca: $k \leftarrow \mathbf{hash}(sk||m||\$)$.

Attacco OW-CCA basato sull'Oracolo di Padding CBC (Vaudenay)

scenario

- stiamo usando la strategia MAC-then-Encrypt con padding PKCS#7;
- abbiamo un oracolo $\mathcal{O}^{CBC}_{MAC}(c)$ che riporta: *padding-error* o *mac-error*; errore generico? ci si può basare sul timing!

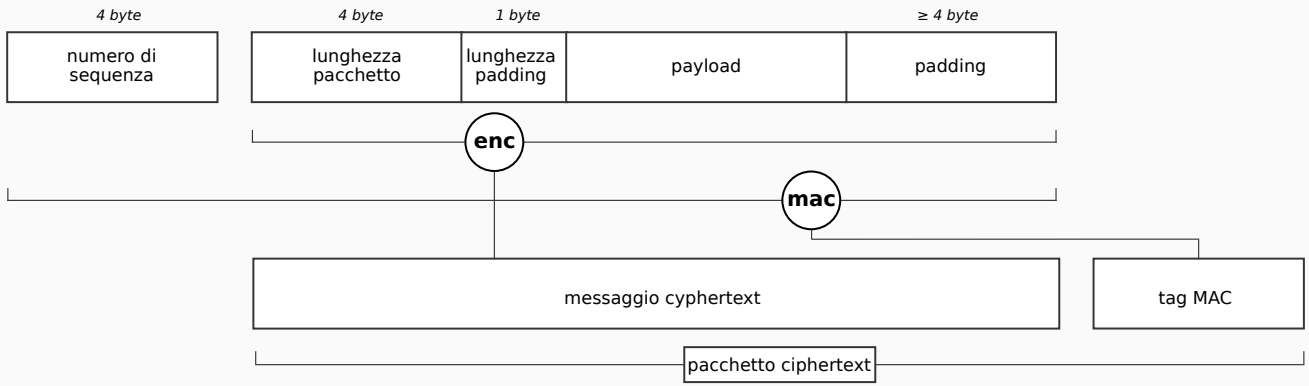


strategia

1. alteriamo $R_1 \in [00, \ldots, FF]$ finché $\mathcal{O}^{CBC}_{MAC}(c_{R_1}) = \textit{mac-error}$:
   $\Rightarrow$ abbiamo trovato $P_1 = R_1 \oplus 1$!
2. fissato $\bar{R}_1 = P_1 \oplus 2$, alteriamo anche R_2 finché $\mathcal{O}^{CBC}_{MAC}(c_{\bar{R}_1, R_2}) = \textit{mac-error}$:
   $\Rightarrow$ abbiamo trovato anche $P_2 = R_2 \oplus 2$!
3. e così via, troviamo tutti gli altri blocchi!

Attacco OW-CCA ad SSH

SSH usa una strategia MAC-and-Encrypt con cifratura CBC.



strategia

1. il target ciphertext c^* (preso dal flusso precedente) viene inviato come primo blocco di un nuovo pacchetto;
2. viene inviato, un byte alla volta, un contenuto spurio;
3. non appena arriva il *mac-error*, ho determinato la lunghezza attesa l^*;
4. ma allora $\lceil p^* \rceil_{32} = l^* \oplus \lceil IV \rceil_{32}$, dove IV corrisponde al penultimo ciphertext passato (catena CBC) e $p^* = \mathbf{dec}_k(c^*)$ è il plaintext parzialmente recuperato (primi 32 bit)!

Pratica: controlli sulla lunghezza ($5 \leq l^* < 2^{18}$) riducono la probabilità a 2^{-18}.

Modalità di Cifratura Autenticate — AEX e GCM

EAX *AEAD*

nato come rimpiazzo di CCM, aggiunge:

- supporta i dati associati esterni
- supporta la modalità on-line
- facile da implementare

GCM – galois counter mode *AEAD*

- fornisce autenticazione
- supporta i dati associati esterni
- parallelizzabile
- supporta la modalità on-line
- può sfruttare un supporto hardware AES-NI e PCLMULQDQ
- supporta accesso diretto (ma senza autentica)
- usa una nounce random ⚠

Modalità di Cifratura Autenticate — GCM in Nettle

supporto GCM

```
#include <nettle/gcm.h>

struct gcm_key;
struct gcm_ctx;
#define GCM_BLOCK_SIZE 16
#define GCM_DIGEST_SIZE 16
#define GCM_IV_SIZE 12

void gcm_set_key(struct gcm_key *key, void *cipher,nettle_cipher_func *f);
void gcm_set_iv(struct gcm_ctx *ctx, struct gcm_key *key,
                size_t length, uint8_t *iv);
void gcm_update(struct gcm_ctx *ctx, struct gcm_key *key,
                size_t length, uint8_t *data);
void gcm_encrypt(struct gcm_ctx *ctx, struct gcm_key *key,
                 void *cipher, nettle_cipher_func *f,
                 size_t length, uint8_t *dst, uint8_t *src);
void gcm_decrypt(struct gcm_ctx *ctx, struct gcm_key *key,
                 void *cipher, nettle_cipher_func *f,
                 size_t length, uint8_t *dst, uint8_t *src);
void gcm_digest(struct gcm_ctx *ctx, struct gcm_key *key,
                void *cipher, nettle_cipher_func *f,
                size_t length, uint8_t *digest);
```

studio: `test-cipher-modes.c`

Mappe Bilineari

definizione

Siano G_1, G_2 due gruppi ciclici additivi di ordine primo r e G_T un altro gruppo di ordine r ma moltiplicativo. Una mappa bilineare (detta anche pairing) è una mappa $e : G_1 \times G_2 \to G_T$ che soddisfa le seguenti proprietà:

1. bilineare: $\forall a, b \in F_r^*, \forall P \in G_1, Q \in G_2 : e(aP, bQ) = e(P, Q)^{ab}$;
2. non-degenere: $\exists P \in G_1, Q \in G_2 : e(P, G) \neq 1$;
3. computabile: e è computabile in modo efficiente.

classificazione

Una mappa bilinere può essere simmetrica ($G_1 = G_2$) o asimmetrica ($G_1 \neq G_2$); in particolare si distinguono mappe di:

tipo 1 $G_1 = G_2$;

tipo 2 $G_1 \neq G_2$ ed esiste un omomorfismo efficientemente computabile da G_2 a G_1;

tipo 3 $G_1 \neq G_2$ ed non esiste un omomorfismo efficientemente computabile tra G_2 e G_1.

Uno Studio Comparativo tra Curve/Pairing

dimensioni con sicurezza a 80 bit

nome PBC	tipo	emb. deg.	dimensione (bit) ordine	F_q	G_1	G_2	G_t	Z_r
A	1 (s)	2	160	512	512+1	512+1	1040	160
A1	1 (s)	2	512	520	520+1	520+1	1040	512
D	3 (a)	6	168	176	176+1	528+1	1056	168
E	1 (s)	1	160	1024	1024+1	1024+1	1024	160
F	3 (a)	12	160	160	160+1	320+1	1920	160
G	3 (a)	10	152	152	152+1	760+1	1520	152

prestazioni con sicurezza a 80 bit

n.	G_1 (μs)			G_2 (μs)			G_t (μs)			pairing (μs)	
	+	×	pp	+	×	pp	×	exp	pp	e()	pp
A	3.68	1738	230	3.62	1738	230	0.87	144	28	1074	476
A1	4.07	5680	842	4.08	5688	848	1.03	504	108	5117	1420
D	1.81	684	69	18.94	5215	708	5.41	1111	180	3937	2923
E	8.34	3626	501	8.38	3620	502	0.68	94	21	3729	~~3726~~
F	1.73	613	58	3.71	1155	127	22.71	4592	698	20253	~~20276~~
G	1.55	583	46	38.15	9519	1215	15.99	2963	451	10770	~~10606~~

Curve Standard del NIST in Nettle

operazioni sui punti

```
void ecc_point_mul(struct ecc_point *r, struct ecc_scalar *n,
                   struct ecc_point *p);
void ecc_point_mul_g(struct ecc_point *r, struct ecc_scalar *n);
```

La moltiplicazione (addizione reiterata di punti) del generatore g utilizza precomputazione.

studio: `test-nettle-ecc.c`

Side Channel

side channel

L'analisi di alcune caratteristiche delle implementazioni crittografiche che permettono di carpire informazioni (anche parziali) sui segreti celati (messaggi, chiavi, ...).

vita reale

Articolo "And Bomb The Anchovies" sul Times Magazine del 1990. 🔍

esempi

- tempo
- dimensione dei pacchetti di rete
- consumo energetico 🔍🔍
- utilizzo della cache 🔍🔍🔍
- residui di segreti in memoria RAM 🔍🔍
- suono 🔍🔍🔍

Attacchi ad RSA

storia

1. Il pioniere è stato Kocher nel 1996: limitato allo Square-and-Multiply ma applicabile anche a DH e DSA;
2. nel 2000 Schindler estese l'attacco anche al caso in cui si impiega il Teorema Cinese del Resto (TCdR) e le moltiplicazioni di Montgomery;
3. nel 2003 Brumley e Boneh mostrarono il primo attacco remoto su un server reale (OpenSSL) supportando anche l'esponenziazione a Sliding Window e l'uso Karatsuba.

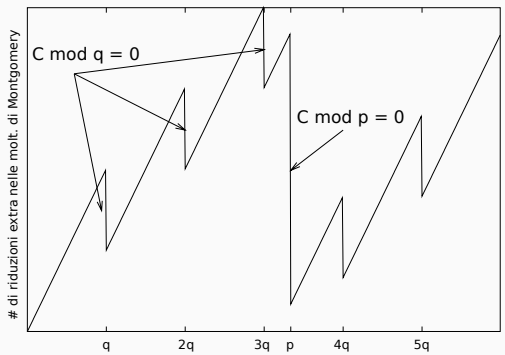
Attacco di Brumley e Boneh — Scenario

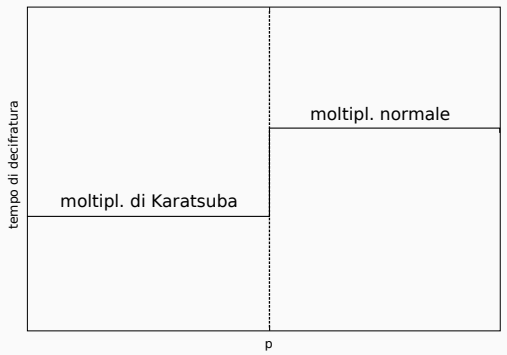
Attacco remoto ad un server web che usava OpenSSL per cifrature/decifrature RSA, impiegando S-a-M con TCdR, moltiplicazioni di Montgomery, a sliding window (rango k) e di Karatsuba.

differenze di tempo sfruttate

1. il numero di riduzioni extra nelle moltiplicazioni di Montgomery (Schindler): se C è vicino a p, con $C < p$, allora le riduzioni extra aumentano (e così $T(C)$); se invece $C > p$, queste diminuiscono;
2. utilizzo di Karatsuba solo nel caso di operandi simmetrici: se C è vicino a p, con $C < p$ gli operandi sono spesso di dimensioni simili e viene usato l'algoritmo ricorsivo (diminuisce $T(C)$), se $C > p$ viene invece usato quello generico (a causa della riduzione mod$p$).

Ma sono effetti contrari...





Bibliografia i

- A.J. Menezes, P.C. van Oorschot, S.A. Vanstone
  Handbook of Applied Cryptography
  ISBN 978-0-8493-8523-0
- C. Paar, J. Pelzl
  Understanding cryptography: a textbook for students and practitioners
  ISBN 978-3-642-04101-3 / bib. DMI coll. AC-11/12
- N.P. Smart
  Cryptography Made Simple
  ISBN 978-3-319-21935-6 / bib. DMI coll. AC-11/11
- R. Bryant, D. O'Hallaron
  Computer Systems: A Programmer's Perspective (prima edizione)
  ISBN 978-0-130-34074-0

Bibliografia ii

- The Linux man-pages project
  Linux Programmer's Manual (man pages)
- Free Software Foundation
  The GNU Multiple Precision Arithmetic Library
- Niels Möller
  Nettle: a low-level cryptographic library
- Ben Lynn
  PBC Library Manual
- G. Paoloni
  How to Benchmark Code Execution Times on Intel® IA-32 and IA-64 Instruction Set Architectures

Bibliografia iii

- D. Bleichenbacher
  Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1
- RSA Laboratories
  PKCS #1 v2.2: RSA Cryptography Standard
- D. Boneh, A. Joux, P.Q. Nguyen
  Why textbook ElGamal and RSA encryption are insecure
- National Institute of Standards and Technology (NIST)
  Digital Signature Standard (DSS)
- S. Vaudenay
  Security Flaws Induced by CBC Padding – Applications to SSL, IPSEC, WTLS...

Bibliografia iv


M. Albrecht, K. Paterson, G. Watson
Plaintext Recovery Attacks Against SSH


S. Galbraith, K. Paterson, N. Smart
Pairings for cryptographers


D. Boneh, B. Lynn, H. Shacham
Short Signatures from the Weil Pairing


D. Bernstein
Curve25519: new Diffie-Hellman speed records


P. Kocher
Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Bibliografia v


W. Schindler
A Timing Attack against RSA with the Chinese Remainder Theorem


D. Brumley, D. Boneh
Remote Timing Attacks are Practical


T. Duong, J. Rizzo
The CRIME Attack


T. Be'ery, A. Shulman
A Perfect CRIME? TIME Will Tell


Y. Gluck, N. Harris, A. Prado
BREACH: SSL, gone in 30 seconds


T. Duong, J. Rizzo
Here Come the ⊕ Ninjas