CIMPA-UNESCO-INDIA School

Security of Computer Systems and Networks

January 25 – February 5, 2005

Public key cryptosystems

F. Morain

ÉCOLE POLYTECHNIQUE CNRS INRIA

`http://www.lix.polytechnique.fr/Labo/Francois.Morain/`

General introduction

Fundamental questions concerning security:

Who are the bad guys? What power do they have?

Two approaches to cryptographic security:

- **Old approach:** my system is secure since **I**, nor anybody, found an attack (until one is found, etc.).
- **Modern approach:** a system is secure if and only if **I** can prove it, in some model, as close to the real world as possible.

The asymetric world

Cryptosystem: use one algorithm E to encrypt, a different one D to decrypt; E can be made public.

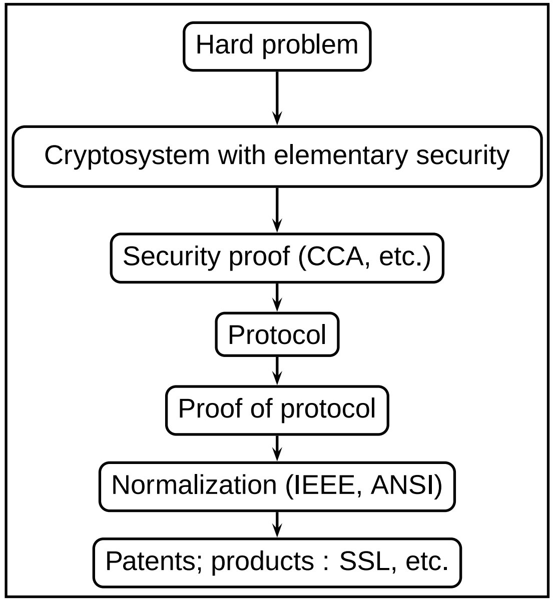
Signature: signing is done with algorithm S; everybody can verify using algorithm V.

Properties:

- Efficiency: **easy** to compute $E(M)$ (resp. $D(C)$).
- Elementary security: **difficult** to recover D from E .

How to find E and D? take a **hard problem** (complexity theory) and transform it into a **secure cryptosystem** using a secret **trapdoor**.

The ideal picture



General overview of the three lectures

1st lecture: a tour of hard problems.

2nd lecture: RSA.

3rd lecture: elliptic curve cryptography.

Bibliography

- *Prime numbers – A Computational Perspective* (Crandall & Pomerance);
- ***Handbook of applied cryptography* (A. Menezes & P. C. van Oorschot & S. A. Vanstone);**
- *Elliptic curve public key cryptosystems* (Menezes);
- *Elliptic curves in cryptography* (Blake, Seroussi, Smart);

CIMPA-UNESCO-INDIA School

Security of Computer Systems and Networks

I. A tour of hard problems

1. Miscellaneous hard problems.

2. Discrete logarithm.

3. Integer factorization.

Part 1: miscellaneous hard problems

I. Knapsack.

II. Error correcting codes.

III. Polynomial systems.

I. Knapsack

1st example of public key cryptosystem (Merkle, 1976).

Hard problem: Given $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ and $N \in \mathbb{N}$, find $(x_0, x_1, \ldots, x_{n-1})$ in $\{0,1\}^n$ s.t.

$$N = \sum_{i=0}^{n-1} \alpha_i x_i.$$

Thm. Decision problem is **NP**-complete.

Easy case: (superincreasing sequences) $\forall\, i, \alpha_i > \sum_{0 \leqslant j < i} \alpha_j$.

Ex. $\alpha_0 = 1, \alpha_1 = 3, \alpha_2 = 9, \alpha_3 = 15, N = 19$.

KEY GENERATION: Alice chooses an integer m, (α_i) a superincreasing sequence s.t. $\sum_{i=0}^{n-1} \alpha_i < m$, and w an integer prime to m; she computes $\alpha_i' = w\alpha_i \bmod m$.

PUBLIC KEY: (α_i').

PRIVATE KEY: w, m.

ENCRYPTION: to send $(x_0, x_1, \ldots, x_{n-1})$, Bob sends $N' = \sum_{i=0}^{n-1} \alpha_i' x_i$.

DECRYPTION: Alice computes $N \equiv w^{-1}N' \bmod m \equiv \sum_i (w^{-1}\alpha_i')x_i \bmod m = \sum_i \alpha_i x_i$ and solves the easy instance of the knapsack problem.

Rem. Broken by Shamir (1978); all generalizations also broken (using the famous LLL algorithm).

Rem. Idem for systems proposed following Ajtai's result.

II. Error correcting codes: the McEliece cryptosystem

KEY GENERATION:

- $\mathcal{C}$ linear code (n, k) correcting t errors and G' a $k \times n$ generating matrix;
- P permutation matrix ($n \times n$);
- S non singular matrix ($k \times k$).

PUBLIC KEY: $G = SG'P$ (matrix $k \times n$).

PRIVATE KEY: G'.

ENCRYPTION: Bob computes $c = mG + z$ with a random z of weight $\leqslant t$.

DECRYPTION: Alice computes $c' = cP^{-1}$, decodes c' to recover m'; finally $m = m'S^{-1}$.

Example: $\mathcal{C}$ is a Goppa code, $n = 1024$, $t = 50$, $k = 524$.

Advantages:

- old and resistant;
- faster than RSA;
- security not related to integer factorization;
- very short signatures (Courtois, Finiasz, Sendrier, ASIACRYPT'2001).

Drawbacks:

- huge public key (n^2);
- ciphertext twice as long as cleartext.

III. Polynomial systems

Hidden field equations (HFE)

(J. Patarin, EUROCRYPT'96)

KEY GENERATION: $K = \mathbb{F}_{p^m} = \mathbb{F}_q, [L_n : K] = n, \beta_{i,j}, \alpha_i \in L_n$, $\theta_{i,j}, \varphi_{i,j}, \xi_i$ integers, $s, t : L_n \to L_n$ affine bijections.

$$\begin{array}{rccl} f: & L_n & \to & L_n \\ & x & \mapsto & \displaystyle\sum_{i,j} \beta_{i,j} x^{q^{\theta_{i,j}} + q^{\varphi_{i,j}}} + \sum_i \alpha_i x^{q^{\xi_i}} + \mu_0. \end{array}$$

$$y = t(f(s(x))) \Longleftrightarrow \begin{cases} y_1 = p_1(x_1, x_2, \ldots, x_n) \\ y_2 = p_2(x_1, x_2, \ldots, x_n) \\ \cdots \\ y_n = p_n(x_1, x_2, \ldots, x_n) \end{cases}$$

Thm. the p_i are of degree 2 ($x \mapsto x^{q^k}$ is linear).

Rem. f must be invertible; typical example: $q = p = 2$, $d = 80$, $n = 80$.

SECRET KEY: (f, s, t).

PUBLIC KEY: (p_i).

ENCRYPTION: $y = (p_1(x), p_2(x), \ldots, p_n(x))$.

DECRYPTION: $x = s^{-1}(f^{-1}(t^{-1}(y)))$.

Security: MQ problem (solving a quadratic system) is NP-complete.

Advantages: ciphertext and signature are very short.

Drawbacks: really equivalent to MQ? Attacks by Shamir & Kipnis, Courtois, J.-C. Faugère, A. Joux (Buchberger algorithm is simply exponential over finite fields).

Partie 2: discrete logarithm

I. Cryptographic motivation: Diffie-Hellman

(**1st known example** of public key algorithm.)

PUBLIC PARAMETERS: p prime number, g generator of $\mathbb{F}_p^*$.

PROTOCOL:

$$A \overset{g^a \bmod p}{\longrightarrow} B$$

$$A \overset{g^b \bmod p}{\longleftarrow} B$$

$$A : K_{AB} = (g^b)^a \equiv g^{ab} \bmod p$$

$$B : K_{BA} = (g^a)^b \equiv g^{ab} \bmod p$$

DH problem: given (p, g, g^a, g^b), compute g^{ab}.

DL problem: given (p, g, g^a), find a.

Thm. DL $\Rightarrow$ DH; converse true for a large class of groups (Maurer & Wolf).

II. Generic algorithms

Pb: $G = \langle g \rangle$ of ordre n; one wants to solve $g^x = a$.

Pohlig-Hellman

Idea: reduce to n prime.

$$n = \prod_i p_i^{\alpha_i}$$

Solving $g^x = a$ is equivalent to knowing $x \bmod n$, i.e. $x \bmod p_i^{\alpha_i}$ for all i (chinese remainder theorem).

Idea: let $p^\alpha \mid\mid n$ and $m = n/p^\alpha$. Then $b = a^m$ is in the cyclic group of ordre p^α generated by g^m. We can find the log of b in this group, which yields $x \bmod p^\alpha$.

Cost: $O(\max(DL(p)))$.

Consequence: in DH, n must have at least one large prime factor.

Shanks

$$x = cu + d, 0 \leq d < u, \quad 0 \leq c < n/u$$
$$g^x = a \Leftrightarrow a(g^{-u})^c = g^d.$$

- Step 1 **(baby steps)**: $\mathcal{B} = \{g^d, 0 \leq d < u\}$;
- Step 2 **(giant steps)**: compute $f = g^{-u} = 1/g^u$; for $c = 0..n/u$, if $af^c \in \mathcal{B}$, then stop.
- End: $af^c = g^d$ hence x.

Analysis: $u + n/u$ group operations, minimal for $u = \sqrt{n} \Rightarrow$ (deterministic) time and space complexity $O(\sqrt{n})$.

Implementation: use hashing to test membership in $\mathcal{B}$.

Rem. Pollard (collisions), space $O(1)$, randomized time $O(\sqrt{n})$.

II. Index-calculus

(Western and Miller, Pollard, Adleman, etc.)

Rem. works over finite fields or in the cases where some notion of prime number exist.

- **Step 1:** compute the logs of $\mathcal{B} = \{p_1, p_2, \ldots, p_k\}$;
- **Step 2:** express ag^b over $\mathcal{B}$ and deduce the log of a.

Step 1: look for relations of the type

$$g^u \equiv \prod_i p_i^{\alpha_i} \bmod p$$

$$u \equiv \sum_i \alpha_i \log_g p_i \bmod (p-1).$$

Once k relations have been collected, solve the linear system and get $\log_g p_i$.

Step 2: look for b s.t.

$$ag^u \equiv \prod_i p_i^{\beta_i} \bmod p$$

which gives ($a = g^x$):

$$x + u \equiv \sum_i \beta_i \log_g p_i \bmod (p-1)$$

hence x.

Analysis

Notation: $L_N[\alpha, c] = \exp\left(c(\log N)^\alpha (\log\log N)^{1-\alpha}\right)$

$$L_N[0, c] = (\log N)^c, \quad L_N[1, c] = N^c$$

Prop. Step 1 costs $L_p[1/2, 2]$, step 2 $L_p[1/2, 3/2]$.

Improvements

- Coppersmith, Odlyzko, Schroeppel (sieve).
- $\mathbb{F}_{2^n}$: Coppersmith *et al.*.
- Number field sieve (Gordon, Schirokauer): $L_p[1/3, c]$.

Records: Joux & Lercier in april 2001, 120 decimal digits (10 weeks, on a unique 525MHz quadri-processors Digital Alpha Server 8400 computer); $\mathbb{F}_{2^{607}}$ by E. Thomé in february 2002 (7 month on one hundred 600 MHz-PC; sparse matrix $1\,033\,593 \times 766\,150$).

Let's do some theory: what about DL in general?

Generic weak instance: $n = \#G$ is smooth (Pohlig-Hellman) $\Rightarrow$ better to have n prime.

Upper-bound: Shanks $O(\sqrt{n})$. Hence, n at least $\approx 2^{200}$.

Lower-bound: (Nechaev, Shoup) any algorithm solving DL (resp. DH) using group operations only, must perform at least $O(\sqrt{\#G})$ operations.

Nechaev group: best algorithm is $O(\sqrt{\#G})$.

Do Nechaev group exist at all?

Which groups?

Group	$\#G$	LD
$\mathbb{F}_q^*$	$q-1$	$L_q[1/3]$
class groups	subexp	subexp
jacobian	$g=1$: poly	$\sqrt{\#G}$
	$g=2,3,4$: poly (?)	$\sqrt{\#G}$
	$g \to \infty$: poly (?)	$L_{q^g}[1/2]$

$$L_N[\alpha, c] = \exp((c + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}).$$

Security: 1024 **bits for** $\mathbb{F}_q^* = 200$ **bits for elliptic curves**.

Part 2: integer factorization

```
From: xxx@zzz (yyy)
Subject: Factoring public keys attack?
Newsgroups: sci.crypt
Date: 02 Oct 1999 22:12:54 GMT

Instead of trying to factor a prime based public key after
somebody has used it, why not have a lookup table of all
the keys. It is quicker to create the keys than to factor
a key.
[...]
The government could have just been making keys for the past
20 years to put on its lookup table. Then if you use one of
the keys of the standard lengths, they already know the prime
```

Answer: $\pi(2^{256}) > 6 \times 10^{74}$.

Which algorithms?

Methods that depend on p**:**

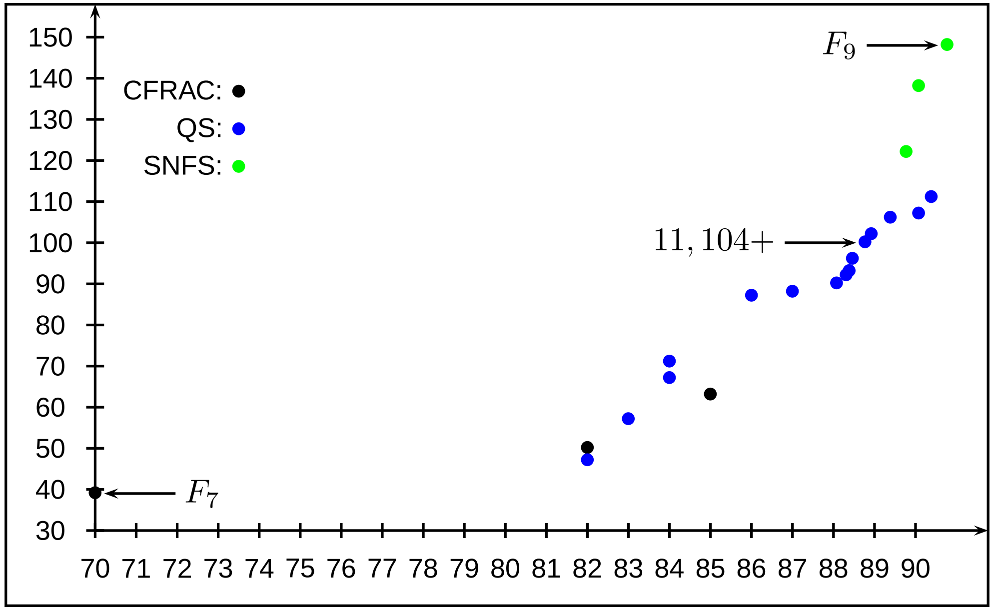
- **sieve**, ρ;
- $p-1$:

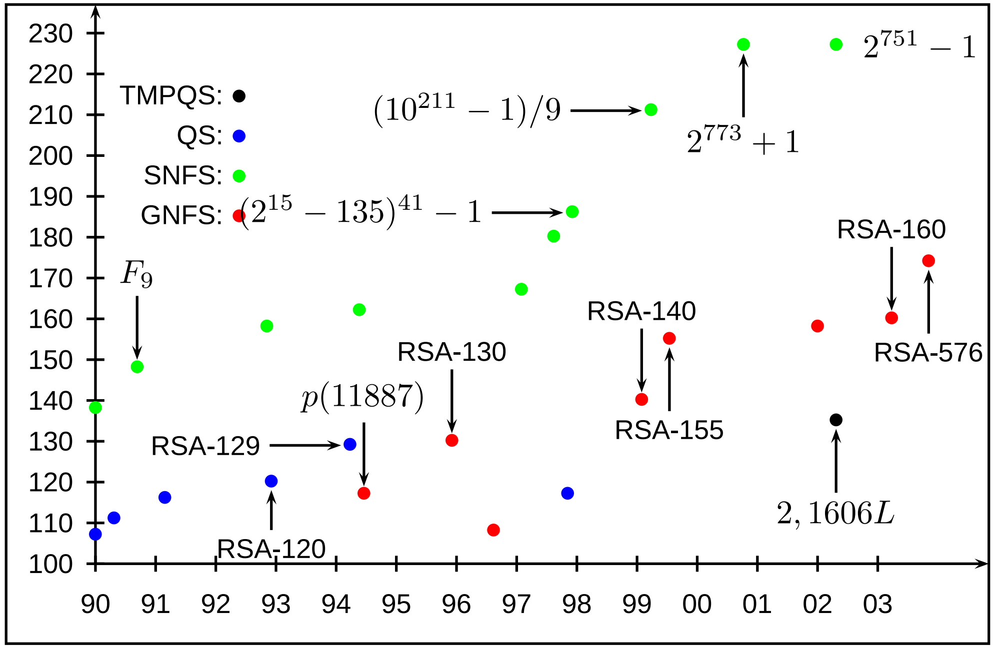
  a- compute $g = (a^{k!} - 1, N)$ for a prime to N. If $p \mid N$ and $p-1 \mid k!$, then $g > 1$.

  b- **other groups:** $p+1$ (Lucas sequences); quadratic forms; **ECM** (elliptic curves), etc.

size of p	N	who	when
55	$629^{59}-1$	Miyamoto	06/10/01
57	$6^{396}+1$	Zimmermann	31/10/03
58	$8 \cdot 10^{141}-1$	Backstrom	31/10/03

General purpose methods: quadratic sieve, algebraic sieve.





Combining congruences

Kraitchik: find x tq $x^2 \equiv 1 \bmod N$, $x \neq \pm 1 \bmod N$.

Step 1: find pairs $\{(u_i, v_i)\}_{i \in I}$ s.t.

$$\mathbf{u_i^2 \equiv v_i \bmod N}, \quad \mathbf{u_i^2 \neq \pm v_i}.$$

Step 2: find $J \subset I$,

$$\prod_{j \in J} v_j = V_J^2$$

Step 3:

$$U_J = \prod_{j \in J} u_j, \quad U_J^2 \equiv V_J^2 \bmod N.$$

Step 4: $x = U_J / V_J \bmod N$ is a squareroot of 1 and with probability $\geq 1/2$, it is non-trivial.

How to test a square

$$v_i = \prod_{p \in \mathbb{P}} p^{\alpha(i,p)}$$

$$Z = \prod_{j \in J} v_j = \prod_{p \in \mathbb{P}} p^{\sum_J \alpha(j,p)} = \square \Leftrightarrow \forall\ \mathbf{p}, \sum_{\mathbf{J}} \alpha(\mathbf{j}, \mathbf{p}) \equiv \mathbf{0} \bmod \mathbf{2}$$

$\Rightarrow$ **linear algebra problem**: find dependance relations in the matrix $\mathcal{M} = (\alpha(i,p) \bmod 2)$.

Pb. $\#\mathbb{P}$ is quite huge.

Idea: replace $\mathbb{P}$ by a factor base $\mathcal{B} = \{p_1, p_2, \ldots, p_k\}$:

$$v_i = \prod_{r=1}^{k} p_r^{\alpha(i,r)} \quad \Rightarrow \quad Z = \prod_{j \in J} v_j = \prod_{r=1}^{k} p_r^{\sum_J \alpha(j,r)}$$

Dixon's algorithm

Take $u_i = i$ and $v_i \equiv i^2 \bmod N$.

Ex. $N = 2117$, $\mathcal{B} = \{-1, 2, 3, 5, 7, 11\}$:

rel	i	v_i	rel	i	v_i
1	65	-1×3^2	5	81	$2 \times 3 \times 5 \times 7$
2	74	$-1 \times 5^3 \times 7$	6	92	-1×2^2
3	75	$-1 \times 2 \times 3 \times 11^2$	7	99	$-1 \times 2^4 \times 7^2$
4	79	$-1 \times 2 \times 5 \times 11$			

$R_2 \times R_3 \times R_5$ yields:

$$(74 \times 75 \times 81)^2 \equiv (-5^3 \times 7)(-1 \times 2 \times 3 \times 11^2)(2 \times 3 \times 5 \times 7)$$

$$\equiv (2 \times 3 \times 5^2 \times 7 \times 11)^2 \bmod N$$

$746^2 \equiv 11550^2$, $\mathrm{pgcd}(746 - 11550, N) = 73$.

Variants

- **CFRAC:** (Morrison & Brillhart, 1970) $\alpha = 1/2$
- **QS, etc.:** (Pomerance, Montgomery, Lenstra & Manasse) $\alpha = 1/2$.
- **NFS:** (Pollard, Lenstra, Buhler) $\alpha = 1/d$ with d as a function of $N \Rightarrow$ change in complexity.

Notation: $L_N[\alpha, c] = \exp\left(c(\log N)^\alpha (\log\log N)^{1-\alpha}\right)$

$$L_N[0, c] = (\log N)^c, \quad L_N[1, c] = N^c$$

Prop. Dixon, CFRAC, QS have complexity $L_N[1/2, c]$; NFS has complexity $L_N[1/3, c]$.

N	$\sqrt{N}$	$L_N[1/2, 1]$	$L_N[1/3, 1]$
2^{512}	1.16×10^{77}	6.69×10^{19}	1.02×10^{10}

The quadratic sieve

Basic version (Pomerance, 1981):

$$u_i = i + \left\lfloor \sqrt{N} \right\rfloor, v_i = \left(i + \left\lfloor \sqrt{N} \right\rfloor\right)^2 - N.$$

Advantages:

○ $v_i \approx 2i\sqrt{N} \ll N$;

○ **crible:**

$$p \mid v_i \Leftrightarrow \left(i + \left\lfloor \sqrt{N} \right\rfloor\right)^2 \equiv N \bmod p$$

implies N square modulo p and

$$p \mid v_i \Leftrightarrow i \equiv i_- \text{ ou } i \equiv i_+ \bmod p$$

Thm. QS runs in time $O(L_N[1/2, 3/\sqrt{8}])$, and space $O(k = L_N[1/\sqrt{8}])$.

Programming the sieve

procedure sieve(L) (* sieve $[0, L[$ *)

1. $S[i] \leftarrow v_i$ for $i \in [0, L[$;

2. **for** $p \in \mathcal{B}$

 for $i_0 = i_\pm(p)$

 $i \leftarrow i_0$;

 while $i < L$

 $S[i] \leftarrow S[i]/p$; $i \leftarrow i + p$;

3. **if** $S[i] = 1$, v_i is completely factored.

Rem. ∞ of tricks to speed up.

MPQS: (Montgomery, 1985) use a lot of polynomials $\Rightarrow$ QS can be **massively distributed**: email (A. K. Lenstra & M. S. Manasse, 1990), INTERNET (RSA-129).

B) Number Field Sieve (NFS)

- Combination of congruences method invented by Pollard in 1988.
- Use $f(X) = a_d X^d + a_{d-1} X^{d-1} + \cdots + a_0$ irreducible over $\mathbb{Q}$ s.t. $f(m) \equiv 0 \bmod N$.
- Operations in the field $\mathbb{Q}[X]/(f(X)) = \left\{\sum_{i=0}^{d-1} b_i X^i, b_i \in \mathbb{Q}\right\}$.

  Ex. In $\mathbb{Q}[X]/(X^2+1)$

  $$(b_1 X + b_0)(c_1 X + c_0) \equiv (b_1 c_0 + b_0 c_1) X + b_0 c_0 - b_1 c_1.$$
- One can sieve (in fact two in parallel).
- The size of the coefficients of f has a great impact on the algorithm: **SNFS**: factorizes $b^n \pm 1$; **GNFS**: **all** numbers.
- Non-trivial implementation. **Faster than PPMPQS** for 120dd–130dd.

IV. Some records

dd	who	when	timings
100	Manasse & A. K. Lenstra	1991	7 MIPS-years
110	AKL	1992	one month on $5/8$ of a 16K MasPar
120	AKL, Dodson, Denny, Manasse Lioen, te Riele	1993	835 MIPS-years
129	Atkins, Graff, AKL, Leyland + INTERNET	1994	5000 MIPS-years
130	Dodson, Montgomery, AKL, WWW, Elkenbracht-Huizing, Fante, Leyland, Weber, Zayer	1996	500 MIPS-years
140	te Riele, Cavallar, Lioen, Montgomery, Dodson, AKL, Leyland, Murphy, Zimmermann	1999	1500 MIPS-years
155	CABAL	1999	8000 MIPS-years
160	Franke et al.	04/2003	??
174	Franke et al.	12/2003	??

V. Linear algebra

Rem.: $\mathcal{M}$ is **very sparse** ($\Omega(N) \le \log_2 N$).

Nb	size	$\#$coeffs $\neq 0$ per row
RSA-100	$50,000 \times 50,000$	
RSA-110	$80,000 \times 80,000$	
RSA-120	$252,222 \times 245,810$ $(89,304 \times 89,088)$	
RSA-129	$569,466 \times 524,338$ $(188,614 \times 188,160)$	47
RSA-130	$3,504,823 \times 3,516,502$	39
RSA-140	$4,671,181 \times 4,704,451$	32
RSA-155	$6,699,191 \times 6,711,336$	62
RSA-160	$5,037,191 \times 5,037,191$	$??$

A) Gaussian elimination

$O(k^3)$ but with a very low constant (32 bits into an `int`, vector processors);

```
    do i=2, ni
        i1 = (piv-1)*nblocs
        i2 = (tabi(i)-1)*nblocs
CDEC$ INIT_DEP_FWD
        do k=1, nblocs
          M(i2+k) = M(i2+k).xor.M(i1+k)
        enddo
    enddo
```

Variants taking sparsity into account (**structured Gaussian elimination**).

B) Sparse methods

- **Wiedemann:** look for the minimal polynomial of $\mathcal{M}$ via the minimal polynomial of the sequence of bits $e_i = u \cdot (M^i b)$ with the Berlekamp-Massey algorithm in time $O(k^{2+\varepsilon})$; **bloc** method due to Coppersmith.

- **Lanczos:** adapted from **numerical analysis**, used over a finite field (!), $O(k^{2+\varepsilon})$; better constant than Wiedemann; **bloc** variant by P. L. Montgomery finds 64 **dependance relations in the same time**.

Predictions?

It is unwise to make predictions about the difficulty of factoring

Back to complexity:

$T(N)$	$N \mapsto N^2$
$\sqrt{N}$	T^2
$L_N[1/2]$	$T^{\sqrt{2}}$
$L_N[1/3]$	$T^{\sqrt[3]{2}}$

Ex. $N = 2^{512}$, $T(N) = 8000$ MIPSY, $T(2^{1024}) = 82715$ **MIPSY**, but with a matrix of **size** $(3 \times 10^8)^2$ (feasable in 2018 (Brent)?)

Moore's law? get 32 bits each time.

CIMPA-UNESCO-INDIA School

Security of Computer Systems and Networks

II. RSA

F. Morain

Plan

I. Introduction.

II. Theory.

III. Implementation.

IV. Advanced security.

V. Signing.

VI. RSA in TLS.

I. Introduction

Cryptosystem: use one algorithm E to encrypt, a different one D to decrypt; E can be made public.

Signature: signing is done with algorithm S; everybody can verify using algorithm V.

Properties:

- Efficiency: **easy** to compute $E(M)$ (resp. $D(C)$).
- Elementary security: **difficult** to recover D from E .

How to find E and D? take a **hard problem** (complexity theory) and transform it into a **secure cryptosystem** using a secret **trapdoor**.

II. Theory

KEY GENERATION: Alice chooses two random primes p and q, $p \neq q$, $N = pq$, e s.t. $\mathrm{pgcd}(e, \lambda(N)) = 1$, $d \equiv 1/e \bmod \lambda(N) = \mathrm{lcm}(p-1, q-1)$.

PUBLIC KEY: (N, e).

PRIVATE KEY: d.

ENCRYPTION:

- Bob retrieves the **authenticated** public key of Alice.
- Bob computes $y = x^e \bmod N$ and sends it to Alice.

DECRYPTION: Alice computes $y^d \bmod N \equiv x$.

Justification

Prop. Let N be an odd integer > 2. Then N is squarefree iff $\forall\, a \in \mathbb{Z}/N\mathbb{Z}$, $a^{\lambda(N)+1} \equiv a \bmod N$.

Proof.

$\Rightarrow$ if $a \equiv 0 \bmod N$: clear;

$a \equiv 0 \bmod p : a^{1+\lambda(N)} \equiv 0^{1+\lambda(N)} \bmod p \equiv a \bmod p$;

$(a, p) = 1 : a^{1+\lambda(N)} \equiv a^{1+K\lambda(p)} \bmod p \equiv a \bmod p$,

$\Leftarrow$ write $N = p^e N'$, $(p, N') = 1$: choose $a = N'p$:

$$a^{p-1} \equiv 0 \bmod p^2 \not\equiv a \bmod p^2. \square$$

Back to RSA:

$$a^{1+k\lambda(N)} \equiv a^{1+\lambda(N)} a^{(k-1)\lambda(N)} \equiv a \times a^{(k-1)\lambda(N)} \bmod N. \square$$

Elementary security of RSA

RSA pb: given (N, e, y), find x s.t. $x^e \equiv y \bmod N$.

Thm. Breaking RSA $\Leftarrow$ factor N; converse may be false (Boneh and Venkatesan).

Prop. Knowing $(N, \lambda(N))$ is equivalent to knowing (p, q).

Proof. Enough to compute $\varphi(N) = (p-1)(q-1) = N - (p+q) + 1$. $\varphi(N) = \gcd(p-1, q-1)\lambda(N) = g\lambda(N)$.

Claim: $g = \gcd(N-1, L)$.

$$g = \gcd(p-1, q-1), \quad p-1 = gp', \quad q-1 = gq',$$

$$L = \lambda(N) = (p-1)(q-1)/g = gp'q'$$

Now:

$$\gcd(N-1, L) = g \gcd(gp'q' + p' + q', p'q') = 1 \square$$

Prop. Knowing (e, d) is equivalent to knowing (p, q) via a randomized algorithm.

Proof. $k = ed - 1 = 2^s \ell \equiv 0 \bmod \lambda(N)$, hence

$$\forall a \in (\mathbb{Z}/N\mathbb{Z})^*, a^k \equiv 1 \bmod N.$$

Lem. 1 has four squareroots modulo N. Two of them break N.

Proof. If $r \equiv 1 \bmod p, r \equiv -1 \bmod q$, then $(r-1, N) = p$. □

Back to the thm. $ed - 1 = 2^s \ell$, ℓ odd; for some $u < s$, $b = a^{2^u \ell}$ is a squareroot of 1. With probability $1/2$, $b \neq \pm 1$. □

A. May, CRYPTO'2004: the same result is true via a **deterministic** algorithm (using LLL).

III. Implemention

Choosing prime numbers:

- $p \neq q, \log_2 p \approx \log_2 q \approx 512$ (NFS);
- $(p-1, q-1) = 2$ (maximize $\lambda(N)$); $p/q \neq$ small rational; $p - q$ big (de Weger).
- $p \pm 1$ with a large prime factor $p - 1 = 2kp'$ (Pollard) s.t. $p' - 1$ has a large prime factor to prevent the **cycling attack**: find n s.t.

$$y \equiv x^e, \mathbf{y^{e^n}} \equiv \mathbf{y} \bmod \mathbf{N} \quad (*)$$

which gives $x \equiv y^{e^{n-1}} \bmod N$. Then

$$(*) \Leftrightarrow e^n \equiv 1 \bmod \lambda(N).$$

Possible prime generating algorithm:

- build r_0 (probably) prime s.t. $r_0 - 1$ has a large (probable) prime factor found by the Artjuhov-Miller-Rabin algorithm;
- build r_1 (probably) prime;
- find p prime s.t. $p \equiv 1 \bmod r_0, p \equiv -1 \bmod r_1$ using CRT.

ARTJUHOV-MILLER-RABIN: $N - 1 = 2^s t$, t odd:

$$a^{N-1} - 1 = (a^t - 1)(a^t + 1)(a^{2t} + 1) \cdots (a^{2^{s-1} t} + 1).$$

If N is prime, it must divide one of the factors.

Thm. The number of false witnesses is $\leq N/4$.

Coro. Proba$(N$ passes k runs$|N$ is composite$) \leq 1/4$.

Rem. We can deduce from that: Proba$(N$ is prime$|N$ passes k runs$)$.

Choosing e: minimize the number of fixed points of $x \mapsto x^e$, which amount to $(1 + \gcd(p-1, e-1))(1 + \gcd(q-1, e-1))$.

Rem. $e = 3$ or e small is possible, but see below.

"Choosing" d:

- d big: if $d < N^{0.292}$, attacks of Wiener; Boneh et al.;
- if using CRT to decrypt: $d \equiv d_p \bmod (p-1)$, otherwise $\gcd(N, x - y^\delta) = p$ for small δ.
- **A. May, CRYPTO'2002:** if $q < N^\beta$, $d_p \leqslant N^\delta$ and if $3\beta + 2\delta \leqslant 1 - \log_N(4)$, then one can factor N in polynomial time. (cf. also J.Blömer & A.May, **CRYPTO'2003**).

ENCRYPTION:

- Primitive: $m \mapsto m^e \bmod N$ with $0 \leqslant m < N$; takes time $O(\log e)$.
- Conversion `uchar t[0..n-1]` to `mpz_t z`:

$$z = t[0]256^{n-1} + t[1]256^{n-2} + \cdots + t[n-1]$$

  called **OS2IP** in PKCS #1 v2.1; inverse function **I2OSP**.
- Put the length of the **useful** message at the beginning:

$$M = l_U || M_U ||\text{MD5}(l_U || M_U)$$

  with $l_U = a_3 256^3 + a_2 256^2 + a_1 256 + a_0 \mapsto$ `a3 a2 a1 a0`.
- Cut M into blocks and add noise:

N	n_{k-1}	n_{k-2}	$\cdots$	n_0
m	0	m_{k-2}	$\cdots$	m_0

Side channel attacks

Timing attacks: (Kocher) monitor the time taken when exponentiating to recover the secret bits one at a time.

$\Rightarrow$ new algorithmics where computations must be concealed.

Error attacks: (Boneh et al.) Simplest example when using CRT for decrypting $y = x^e \bmod N$. One computes $z = y^d \bmod N$ in the following way:

$z_p = y^{d \bmod (p-1)} \bmod p, \quad z_q = y^{d \bmod (q-1)} \bmod q$ + CRT.

If z_p is correct, but not z_q, then recover p as $\gcd(z - x, N)$.

IV. Advanced security

Textbook RSA does not obey Shannon

Common modulus: (Simmons) N common to all users: if M is sent to two users with $(e_1, e_2) = 1$, then using $ue_1 + ve_2 = 1$, one gets:

$$(M^{e_1})^u (M^{e_2})^v \equiv M \bmod N.$$

Common exponent: $C_i = M^3 \bmod N_i$ for $i = 1, 2, 3$; one builds $C = M^3 \bmod N_1 N_2 N_3$; since $M < N_i$, we deduce $C = M^3$, hence M. Generalization to more general polynomials $g_i(M)$ by J. Håstad.

Timestamp attacks

If $M^e \bmod N$ and $(M+c)^e \bmod N$ are sent with known c, M can be recovered.

Ex. (Franklin-Reiter) $C_1 \equiv M^3 \bmod N$, $C_2 \equiv (M+1)^3 \bmod N$; then:

$$\begin{cases} C_2 + 2C_1 - 1 & = & 3M^3 + 3M^2 + 3M \\ C_2 - C_1 + 2 & = & 3M^2 + 3M + 3 \end{cases}$$

hence $M = (C_2 + 2C_1 - 1)/(C_2 - C_1 + 2) \bmod N$.

More generally: $\gcd(M^e - C_1, (M+c)^e - C_2)$ even if $\mathbb{Z}/N\mathbb{Z}$ has zero divisors.

Thm. (Coppersmith) **if $f(X)$ has degree d, one can find all solutions $< N^{1/d-\varepsilon}$ of $f(X) \equiv 0 \bmod N$ in polynomial time in $\min(1/\varepsilon, \log N)$.**

Beyond elementary security

Goals:

- **IND:** indistinguishability (Goldwasser & Micali). One cannot distinguish $E(\text{"yes"})$ from $E(\text{"no"})$.
- **NM:** non-malleability (Dolev, Dwork, Naor). Given $E(m)$ and $E(m')$, one cannot build $E(m \otimes m')$ (say).

Attacks:

- **CPA:** *chosen-plaintext attack* (in asymetric crypto, **everybody can encrypt!**).
- **CCA1:** *non-adaptative chosen-ciphertext attack* (Naor & Yung), decryption oracle before the attack.
- **CCA2:** *adaptative chosen-ciphertext attack* (Rackoff & Simon), decryption oracle available except on the target message.

The fundamental theorem

Thm. (Bellare, Desai, Pointcheval, Rogaway)

$$
\begin{tikzcd}
\text{NM-CPA} \arrow[d] \arrow[drr, shift left, "\times" marking] & & \text{NM-CCA1} \arrow[ll] \arrow[d] \arrow[rr, shift left, "\times" marking] & & \text{NM-CCA2} \arrow[ll, shift left] \arrow[d, shift left] \\
\text{IND-CPA} & & \text{IND-CCA1} \arrow[ll] \arrow[ull, shift left, "\times" marking] & & \text{IND-CCA2} \arrow[ll] \arrow[u, shift left]
\end{tikzcd}
$$

Examples with text book RSA

Text book RSA is not IND-CPA: easy to distinguish TB-RSA("yes") from TB-RSA("no").

TB-RSA is not NM-CPA: $x^e \times y^e = (xy)^e$.

Ex. if $M < 2^m$ and $M = M_1 M_2$, $M_i < 2^{m/2}$, then

$$M_1^e M_2^e \equiv C \bmod N \Longleftrightarrow C / M_2^e \equiv M_1^e \bmod N.$$

TB-RSA does not resist a CCA2:

- Charlie intercepts $C = M^e \bmod N$;
- Charlie chooses r at random and asks the oracle to decrypt $y = r^e C$;
- the oracle sends back $y^d = r^{ed} C^d = r C^d$ from which M is recovered s.t. $C^d = M$.

Counterattack: OAEP, etc.

Idea: take a CPA cryptosystem and transform it into a IND-CCA one.

OAEP: (Bellare & Rogaway)

INPUT:

- Public algorithm f, private algorithm g operating on strings $\in \{0,1\}^k$; $k_0 + k_1 < k$;
- Two hash functions $G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$, $H : \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$.
- The algorithm encrypts $M \in \{0,1\}^n$, with $n = k - k_0 - k_1$.

Encryption	Decryption
	$x = z[0..n-1], c = z[n..n+k_1-1]$
$\mathbf{s} = \mathbf{G}(\mathbf{r}) \oplus (\mathbf{M}\Vert\mathbf{0}^{\mathbf{k_1}}) \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{n}+\mathbf{k_1}}$	$z = G(r) \oplus s$
$t = H(s) \oplus r \in \{0,1\}^{k_0}$	$r = H(s) \oplus t$
$w = s\Vert t \in \{0,1\}^k$	$s\Vert t = w[0..n+k_1-1\Vert n+k_1..k]$
$C = f(w)$	$w = g(C)$

If $c = 0^{k_1}$, then $M = x$, otherwise reject C and **do not send** x **back**.

Thm. In the random oracle model, OAEP is IND-CCA2.

Rem. In practice, take G and H as variants of MD5 à la Full Domain Hash.

Rem. Shoup discovered a breach in the proof and proposed with

$$\mathbf{s} = (\mathbf{G}(\mathbf{r}) \oplus \mathbf{M})||\mathbf{H}'(\mathbf{r}||\mathbf{M}).$$

Rem. RSA-OAEP is sure anyway (Fujisaki, Okamoto, Pointcheval and Stern).

Boneh: (CRYPTO 2001)

SAEP:

$$((M||0^{s_0}) \oplus H(r))||r$$

SAEP+:

$$((M||G(M||r)) \oplus H(r))||r$$

V. Signing

A) Signature with appendix

PREREQUISITE: each user has a pair (S, V) where S is the private signature algorithm and V the public verification algorithm, s.t. $V(m, S(m)) =$`true`.

SIGNATURE: Alice signs m and sends $(m, S_A(m))$.

VERIFICATION: Bob gets the authenticated algorithm V_A of Alice and tests whether $V_A(m, s) ==$`true`.

Rem.

- must use m to verify;
- if m is too long, use $S(m) = S'(\mathcal{H}(m))$.

Ex. Alice has RSA parameters (N_A, e_A, d_A); $S_A(m) = m^{d_A} \bmod N_A$; $V_A(m, s) = (s^{e_A} \bmod N == m)$.

But: $(E(x), x)$ is a valid pair, since $V(\mathbf{E}(\mathbf{x}), x) = E(x) == \mathbf{E}(\mathbf{x})$. **One should not accept everything!**

Application to RSA: $S(m) = \mathcal{H}(m)^d \bmod N$ with $\mathcal{H} = MD5$; $V(m, s) = ((s^e \bmod N) == \mathcal{H}(m))$?.

Desmedt-Odlyzko; Coron-Naccache-Stern: if $\mathcal{H}(x)$ is too small, use a smooth-number attack.

$\Rightarrow$ *Full Domain Hash*: (Bellare & Rogaway; Coron) $S(m) = \mathcal{H}(m)^d \bmod N$ with $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}/N\mathbb{Z}$.

PSS

Probabilistic signature scheme (Bellare, Rogaway) with security proof.

PREREQUISITE: $k_0 + k_1 < k$; $H : \{0,1\}^* \rightarrow \{0,1\}^{k_1}$,
$G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{k-k_1-1}$; $G(w) = \underbrace{G_1(w)}_{k_0 \text{ bits}} || G_2(w)$.

Signature			Verification		
choose $r \in_R \{0,1\}^{k_0}$					
w	$=$	$H(m\|\|r)$	$H(m\|\|r) == w$ **and** $G_2(w) == \gamma$ **and** $b == 0$		
r^*	$=$	$G_1(w) \oplus r$	r	$=$	$r^* \oplus G_1(w)$
y	$=$	$0\|\|w\|\|r^*\|\|G_2(w)$	z	$=$	$b\|\|\underbrace{w}_{k_1}\|\|\underbrace{r^*}_{k_0}\|\|\gamma$
x	$=$	$y^d \bmod N$	z	$=$	$y^e \bmod N$

B) Signatures with message recovery

Idea: $S(m)$ enables one to recover m, which increases the band-width.

Ex. $S_A(m) = m^{d_A} \bmod N_A$, $V_A(s) = s^{e_A} \bmod N_A$.

But: x is a valid signature for $E(x)$, since $V(\mathbf{E}(\mathbf{x}), x) = (E(x) == \mathbf{E}(\mathbf{x}))$;
$\Rightarrow$ one must be able to recognize a valid message, using some redundancy R.

Ex. $R(m) = m||m$: one m' at random is valid with probability 2^{-n}.

SIGNATURE: Alice compute $m' = R(m)$, and sends $s = S_A(m')$.

VERIFICATION:

- Bob gets the authenticated verification algorithm of Alice;
- Bob computes $m'' = V_A(s)$ and checks whether m'' presents the desired redundancy: if yes, he gets back $m = R^{-1}(m'')$; otherwise, he rejects the signature.

Simple idea: $R(m) = mw = m||\underbrace{0\ldots0}_{t \text{ bits}}$; $k = \lfloor \log_2 N + 1 \rfloor$, $t < k/2$,
$w = 2^t$ et $0 \leqslant m < n/w - 1$.

But... **existential forgery** on given m (De Jonge & Chaum):

- Euclid's algorithm applied to $(N, m' = mw)$: at each step $xN + ym' = r$ and at some point $|y|, r < N/w$;
- compute $(m_2, m_3) = (rw, |y|w)$;
- if $s_2 = m_2^d$ and $s_3 = m_3^d$ are known, then $s_2/s_3 = (m_2/m_3)^d = {m'}^d$.

Other choices: $00\cdots00||m||11\cdots11$ or $m||\mathcal{H}(m)$ are not enough (cf. Girault, Misarsky, Bleichenbacher, etc.), nor ISO/IEC 9796 (1999-2000: Coron-Naccache-Stern, Coppersmith-Halevi-Jutla, Grieu; broken again by Girault-Misarsky).

PSS with message recovery

Signature			Verification		
choose $r \in_R \{0,1\}^{k_0}$					
w	$=$	$H(m\|\|r)$	$\mathbf{H}(\mathbf{m}\|\|\mathbf{r}) == \mathbf{w}$ **and** $\mathbf{b} == \mathbf{0}$		
r^*	$=$	$G_1(w) \oplus r$	$\mathbf{m}$	$=$	$\gamma \oplus \mathbf{G_2}(\mathbf{w})$
$\mathbf{m}^*$	$=$	$\mathbf{G_2}(\mathbf{w}) \oplus \mathbf{m}$	r	$=$	$r^* \oplus G_1(w)$
y	$=$	$0\|\|w\|\|r^*\|\|\mathbf{m}^*$	z	$=$	$b\|\|\underbrace{w}_{k_1}\|\|\underbrace{r^*}_{k_0}\|\|\gamma$
x	$=$	$y^d \bmod N$	z	$=$	$y^e \bmod N$

From primitives to protocols: SignCryption

Goal : Bob ($\{E, D, S, V\}_B$) wants to be sure that the cleartext corresponding to the ciphertext he just received was actually written by Alice ($\{E, D, S, V\}_A$).

1) **send** $(E_B(m), S_A(m))$**:** Carole intercepts $(E_B(m_b), \sigma)$ and can compute for herself $V_A(m_0, \sigma)$ and $V_A(m_1, \sigma)$.

2) **send** $(E_B(m), S_A(E_B(m)))$**:** one knows that Alice signed $E_B(m)$ **and not** m. Carole can sign it too.

3) **send** $S_A(E_B(m))$ **:** beware of **Anderson & Needham** : Alice sends $\{M^{e_B} \bmod N_B\}^{d_A} \bmod N_A$. If Bob wants a signature on M', he can solve $[M']^x = M \bmod N_B$ and register the key (xe_B, N_B) as (another) public key of his own.

4) $E_B(m||S_A(m))$ **:** Carole cannot deduce anything.

VI. RSA in TLS – RFC 2246, january 1999

Client		Server
ClientHello	$\longrightarrow$	
	$\longleftarrow$	ServerHello
	$\longleftarrow$	Certificate* (X509)
	$\longleftarrow$	ServerKeyExchange*
	$\longleftarrow$	CertificateRequest*
	$\longleftarrow$	ServerHelloDone
Certificate*	$\longrightarrow$	
ClientKeyExchange	$\longrightarrow$	
CertificateVerify*	$\longrightarrow$	
[ChangeCipherSpec]	$\longrightarrow$	
Finished	$\longrightarrow$	
	$\longleftarrow$	[ChangeCipherSpec]
	$\longleftarrow$	Finished

With RSA

Client		Server
	$\longleftarrow$	Certificate* (X509) : $\ni$ Pub$_S$
	$\longleftarrow$	~~ServerKeyExchange*~~
	$\longleftarrow$	CertificateRequest*
	$\longleftarrow$	ServerHelloDone
Certificate* : $\ni$ Pub$_C$	$\longrightarrow$	
ClientKeyExchange: Pub$_S(pms)$	$\longrightarrow$	
CertificateVerify* : S_C(handshake msgs)	$\longrightarrow$	

Bleichenbacher (CRYPTO'98)

Client		Server
	$\longleftarrow$	Certificate* (X509) : $\ni$ Pub$_S$
	$\longleftarrow$	~~ServerKeyExchange*~~
	$\longleftarrow$	CertificateRequest*
	$\longleftarrow$	ServerHelloDone
Certificate* : $\ni$ Pub$_C$	$\longrightarrow$	
ClientKeyExchange: Pub$_S(pms)$	$\longrightarrow$	
	$\longleftarrow$	Error

Attack: by using the server as an oracle, can decrypt a message m with a large number of trials, if formatted using PKCS # 1 v1.5.

Conclusion: replace

```
if(! goodFormatForMessage(m))
    send_error("bad format");
```

by

```
ok = goodFormatForMessage(m);
if(ok){
    {remaining code}
}
if(! ok)
    kill_connection();
```


Manger's attack – CRYPTO'01

Timing attack on the preceding scheme. Replace it with:

```
ok = goodFormatForMessage(m);
{remaining code}
if(!ok) kill_connection();
```

$\Rightarrow$ **Do not turn a program into an oracle!**

Conclusions on RSA

- **Good cryptography is orthogonal to good software engineering!!** For instance, modularity is at stakes.
- RSA is the king, it generated much enthousiasm, anger, theorems, etc. over 30 years. But resisted. Still more to come?
- However, important drawbacks: **implementing a safe RSA is like crossing a mine field by night**; bandwidth has reduced a lot (768 bits over 1024).
- Isolated point in crypto space ($E(D(m)) = D(E(m))$ for instance).
- Replace with new stuff (elliptic curves?).

CIMPA-UNESCO-INDIA School

Security of Computer Systems and Networks

III. Algebraic curve cryptography

F. Morain

ÉCOLE POLYTECHNIQUE CNRS INRIA

Plan

I. ElGamal cryptosystem and signature.

II. Building AC-systems.

III. Attacking AC-systems.

IV. Pairings and applications.

V. Other algebraic curves; tori.

I. ElGamal cryptosystem and signature

A) ElGamal encryption

KEY GENERATION: Alice chooses a prime p, $(\mathbb{Z}/p\mathbb{Z})^* = \langle g \rangle, 0 < a < p-1$.

PUBLIC KEY: $(p, g, h = g^a \bmod p)$.

PRIVATE KEY: a.

ENCRYPTION: Bob chooses $r \in_R (\mathbb{Z}/(p-1)\mathbb{Z})^*$, sends $(u, v) = (g^r, h^r M)$.

DECRYPTION: Alice computes $M \equiv v/u^a$.

Justification: $v/u^a \equiv h^r M/g^{ra} \bmod p$.

Rem. ElGamal generalizes trivially to any cyclic group $G = \langle g \rangle$ of order n.

Drawback: ciphertext twice as long as the cleartext.

Rem. Encryption **must** be randomised, otherwise $h^r M_1/(h^r M_2) = M_1/M_2$. Choosing r must be done with great care (Phong Nguyen *et al.*).

Discrete logarithm and security

Three problems:

- discrete logarithm (LD): given g^x, compute x;
- computational Diffie-Hellman problem (CDH): given (g^x, g^y), compute g^{xy};
- decisionnal Diffie-Hellman problem (DDH): given (g^x, g^y, g^z), do we have $z \equiv xy \bmod n$?

Prop. LD $\Rightarrow$ CDH $\Rightarrow$ DDH.

Thm. (Maurer & Wolf) For a lot of groups LD $\Leftrightarrow$ CDH.

Thm. (Joux & Nguyen) There exist groups for which DDH is easier than CDH.

Security of ElGamal's cryptosystem

Pb ElGamal: given (p, g), for all $(h = g^a, u, v)$, one can compute v/u^a.

Prop. ElGamal $\Longleftrightarrow$ CDH**.**

Proof. If CDH is solvable: target message $(g^r, h^r M)$; from $h = g^a$ and g^r, one gets $g^{ar} = h^r$, hence M.

If ElGamal can be solved: send $(g^{-x}, g^y, 1)$, get $M = 1/(g^y)^{-x} = g^{xy}$. $\square$

Prop. ElGamal is not NM-CPA.

Proof. Given $(g^r, h^r m)$, one can compute $(g^{2r}, h^{2r} m^2)$. $\square$

Prop. ElGamal does not resist to a CCA2.

Proof. given (u, v), one asks the oracle to decrypt (gu, v) and we get back M/h, hence M. $\square$

Thm. ElGamal is IND-CPA iff DDH **is difficult.**

Proof. give m_0, m_1 to the encrypting oracle that sends back $(u, v) = (g^r, h^r m_b)$, $b \in \{0, 1\}$. The attacked must find out which of $(u, h, v/m_0)$ or $(u, h, v/m_1)$ is a valid DH triplet. $\square$

Rem. When $G = (\mathbb{Z}/p\mathbb{Z})^*$, this is not true, since (m/p) is available.

Variant: $(g^r, m \oplus H(h^r))$; **but** $m \oplus H(h^r) \oplus 1_n = (m \oplus 1_n) \oplus H(h^r)$.

Baek, Lee, Kim (ACISP2000): variant of Fujisaki-Okamoto, CRYPTO'99 that turns ElGamal into an IND-CCA2 scheme.

B) Signing with ElGamal

KEY GENERATION: Alice chooses a prime p, $(\mathbb{Z}/p\mathbb{Z})^* = \langle g \rangle, 0 < a < p - 1$.

PUBLIC KEY: $(p, g, h_A = g^a \bmod p)$.

PRIVATE KEY: a.

SIGNATURE OF m: Alice chooses a secret $k \in_R (\mathbb{Z}/(p-1)\mathbb{Z})^*$; signature is (r, s) with $r = g^k \bmod p$, $s = (m - ar)/k \bmod (p - 1)$.

VERIFICATION:

- Bob gets the **authenticated** key of Alice: h_A;
- Bob checks whether $1 \leq r < p \quad (*)$;
- Bob checks whether $h_A^r r^s = g^m \bmod p$.

Justification: $h_A^r r^s = g^{ar+ks} = g^m$.

Elementary security:

- $\Leftarrow$ DL: one gets a.
- If one knows s, one has to solve $h_A^r r^s = g^m$??
- If one knows r, one must solve DL on $r^s = g^m / h_A^r$;
- Take care to k.

Why Bob must check $(*)$**:** let (r, s) be a signature on some known m; m_2 is the target message. Write $u \equiv m_2/m \bmod (p - 1)$;

$$g^{m_2} \equiv g^{hu} \equiv (h_A)^{ru} r^{su} \bmod p.$$

Choose $s_2 \equiv su \bmod (p - 1)$ and $r_2 \equiv ru \bmod (p - 1)$, $r_2 \equiv r \bmod p$ using CRT. Then (r_2, s_2) is a valid signature on m_2.

Existential forgery: if b and c are prime to $p - 1$, then

$(r' = g^b h_A^c, s' = -r'/c \bmod (p - 1))$ is a valid signature for $m' = -r'b/c \bmod (p - 1)$.

C) DSA

KEY GENERATION: prime p of 512 to 1024 bits, q prime factor of $p-1$ with 160 bits; $g \equiv h^{(p-1)/q} \bmod p \not\equiv 1$.

PUBLIC KEY: $y = g^x \bmod p$.

PRIVATE KEY: $x < q$.

SIGNATURE: Alice chooses $k < q$ at random; signature is (r, s) with

$$r = (g^k \bmod p) \bmod q, \quad s = (k^{-1}(\mathcal{H}(m) + xr)) \bmod q.$$

VERIFICATION:

$w \equiv 1/s \bmod q, \quad u_1 \equiv (\mathcal{H}(m)w) \bmod q, \quad u_2 \equiv rw \bmod q,$

$$(g^{u_1} y^{u_2} \bmod p) \stackrel{?}{=} r \bmod q.$$

Advantage: short signature. **Drawback:** slow verification.

II. Building AC-cryptosystems

Why ACC? best candidates to be **Nechaev** groups.

Best groups so far: hyperelliptic curves of genus g, with size $\approx q^g$ over some finite field $\mathbb{F}_q$. Typical size $q^g \approx 2^{160--200} \approx 10^{50--60}$.

- Miller, Koblitz (1986): elliptic curves are suggested for use, following the breakthrough of Lenstra in integer factorization (1985).
- Koblitz (1988): hyperelliptic cryptosystems.
- See: *Algebraic curves and cryptography*, S. Galbraith & A. Menezes, January 12, 2005.

General definitions

Let C be a plane smooth projective curve of genus g with equation $F(X, Y) = 0$ with coefficients in $\mathbb{K}$, $\mathrm{char}(\mathbb{K}) = p$.

Conic: (genus 0) $x^2 + y^2 = 1$.

Elliptic curve: (genus 1) $y^2 = x^3 + x + 1$.

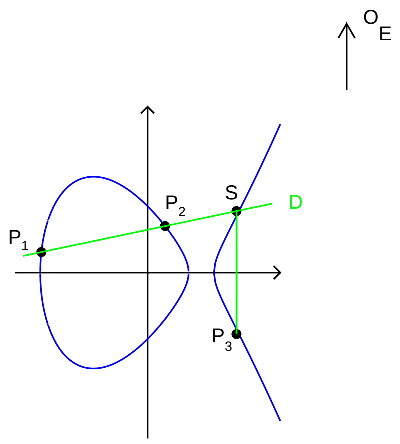
Hyperelliptic curve: (genus g) $y^2 = x^{2g+1} + \cdots$ (or in some cases $y^2 = x^{2g+2} + \cdots$).

Def. $C(\mathbb{K}) = \{P = (x, y) \in \mathbb{K}^2, F(x, y) = 0\}$.

Thm. When $g \leq 1$, there is a group law on $C(\mathbb{K})$. When $g > 1$, there is a group law on the **jacobian** of the curve.

Group law

$$E : Y^2 = X^3 + aX + b$$

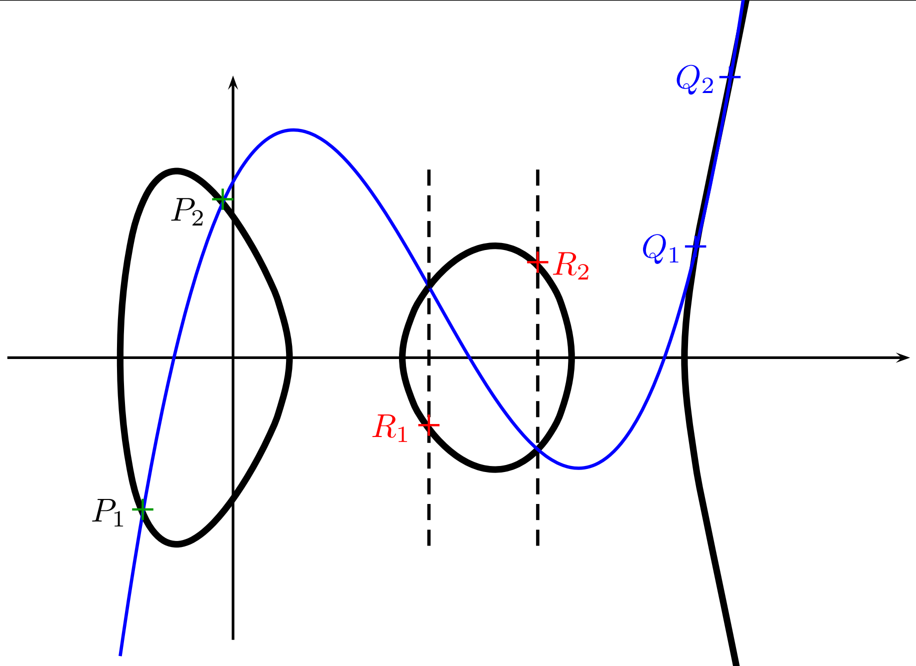


$$P_3 = P_1 \oplus P_2, [k]P = \underbrace{P \oplus \cdots \oplus P}_{k \text{ times}}$$

$$\lambda = \begin{cases} (y_1 - y_2)/(x_1 - x_2) \\ (3x_1^2 + a)/(2y_1) \end{cases}$$

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$



(Courtesy from PGaudry)

Cardinality

Thm. (Hasse-Weil) $(\sqrt{q}-1)^{2g} \leq \#\mathrm{Jac}(C) \leq (\sqrt{q}+1)^{2g}$.

$g = 1$: $\#E = q + 1 - t$, $|t| \leq 2\sqrt{q}$. Explains why so much success in integer factorization (ECM) or primality proving (ECPP).

Pb: compute this cardinality as quickly as possible (polynomial time?). No general formulae except in special cases that might be dangerous (CM curves, supersingular curves).

Cryptographic needs: $\mathbb{F}_p$ with large p or $\mathbb{F}_{2^n}$ with n prime (Weil descent, see below); subgroups of large prime order.

Algorithms:

- $g = 1$, p **large**: **Schoof** (1985), Pila, etc. Completely practical after improvements by Elkies, Atkin, and implementations by M., Lercier, etc. New recent record M. for $p = 10^{999} + 7$.

- $p = 2$: p-adic methods (**Satoh**, Fouquet/Gaudry/Harley; Mestre; Lercier-Lubicz, etc.; Kedlaya; Lauder-Wan). Completely solved.

$g \backslash p$	2	small	medium	large
1	MF & PG & Harley Mestre, etc.	Satoh Kohel	Couveignes RL & FM	SEA FM
2	Mestre, etc.	Kedlaya PG & NG	PG & NG & Bostan+Schost	PG & Schost
3-hyper	RL & Lubicz	idem	idem	tbd
3-super	Ritzenthaler	idem	idem	tbd

```
#define RL "R.~Lercier"
#define PG "P.~Gaudry"
#define MF "M.~Fouquet"
#define NG "N.~Gürel"
```

III. Attacking AC-systems

- **No (known) subexponential method for small** g (including $g = 1$); recover a subexp method when g increases.
- **Reduction** $\mathrm{Jac}(C)/\mathbb{F}_q \hookrightarrow \mathbb{F}_{q^k}$ with k small:
  - Supersingular curves: **MOV** (Menezes, Okamoto, Vanstone using the Weil pairing); Frey & Rück (using the Tate pairing); Galbraith.
  - other cases: elliptic curves with $t = 2$ with the Tate pairing.
- Discrete logs in subgroups of order p^e of $\mathrm{Jac}(C)/\mathbb{F}_{p^r}$ can be found in **polynomial time**: $g = 1$ (anomalous curves) done by Satoh-Araki, Semaev, Smart; $g > 1$ by Rück.
- **Elliptic curves:** largest example done: ECC2-109 in april 2004 (**1200 years of Athlon XP 3200+**, `http://www.certicom.com/chal/`).

Discrete log on hyperelliptic curves

- Algorithm ADH from **Adleman, DeMarrais, Huang** (ANTS I):

$$\mathbf{L_{p^{2g+1}}[1/2, c]}$$

  with $c \leq 2.181$ if $\log p \leq (2g+1)^{0.98}$ (heuristic using Lovorn's theorem on smooth polynomials); SNF.
- **Flassenberg & Paulus:** using sieving techniques; experiments with $y^2 = x^{2g+1} + 2x + 1$, faster than Shanks for $g \geq 6$.
- $y^2 = x^{2g+2} + \cdots$ (Müller-Stein-Thiel): proved $L_{p^{2g+2}}[1/2, 1.44]$.
- Extensions, proved analysis and optimizations by **Enge**: if $\theta \log q \leq g$

$$\mathbf{L_{q^g}[1/2, c(\theta)]},$$

  with $\lim_{\theta \to 0} c(\theta) = +\infty$; easier SNF. Smaller $c = \sqrt{2}$ by Enge and Gaudry.

Gaudry's variant

Idea: use a $O(q)$ factor basis $+$ random walk to generate relations.

Time $O(q^2 \log^c q)$ for fixed g. Provably (and practically) better than Pollard's ρ for $g > 4$.

Thériault (2003): use one large prime, leads to $O(q^{2-2/(g+0.5)})$, so $g = 3$ and $g = 4$ are in danger (assuming q is large).

Gaudry/Thériault/Thomé (2004): use double large primes leads to a method in $O(q^{2-2/g})$.

Weil descent

(Frey, 1998; Gaudry-Hess-Smart, 2002)

Rough idea: to attack DLP in $\mathrm{Jac}(C/\mathbb{F}_{q^n})$, find another curve $X/\mathbb{F}_q$ and a non-constant rational map $f : X \to C$ s.t. DLP is easier on X.

Typical example. $\mathbb{F}_q = \mathbb{F}_{2^{21}}$, $E/\mathbb{F}_{q^4}$, leads to a curve $X/\mathbb{F}_q$ of genus $g = 4$ (therefore $O(q^{3/2})$ using GTT).

Rem. m further analyzed by Menezes & Wu, $\mathbb{F}_{2^p}$ **not breakable**; see also Menezes, Maurer, Teske for the composite case.

Rem. Recent computations of Smart: can break $E/\mathbb{F}_{q^4}$, $g = 8$, faster than ρ for $q > 2^{17}$.

Recent results: Semaev; Gaudry; Diem: **Subexponential $L_{p^n}[3/4]$ attack for $E/\mathbb{F}_{p^n}$ when $n \sim \log p$.**

IV. Pairings and applications

Setup: ℓ prime, $\ell \mid \#E$ and $\ell \mid q^k - 1$, $\Rightarrow \exists P \in E(\mathbb{F}_q), Q \in E(\mathbb{F}_{q^k})$ that generate $E[\ell]$.

Weil and Tate pairings: $e : \langle P \rangle \times \langle Q \rangle \rightarrow \mu_\ell \subseteq \mathbb{F}_{q^k}^\times$

- bilinear: $e(aP, bQ) = e(P, Q)^{ab}$;
- non-degenerate;
- efficiently computable **if** k **is small** (in $O(\log(\ell) M(q^k))$).

Immediate application: MOV reduction when k is small, reduction of DL to $\mathbb{F}_{q^k}$.

More recent applications: identity based cryptosystems, short signatures (Boneh, Lynn, Sacham), etc.

Non interactive key exchange (Sakai–Ohgishi–Kasahara)

Private Key Generator
Master key $s \in [0, \ell - 1]$

sP_A, sQ_A $\qquad$ sP_B, sQ_B

Alice $\qquad$ **Bob**

$P_A = H(\text{“ALICE”})$ $\qquad$ $P_B = H(\text{“BOB”})$

$Q_A = H'(\text{“ALICE”})$ $\qquad$ $Q_B = H'(\text{“BOB”})$

$$e(\mathbf{P_B}, sQ_A) = e(P_B, Q_A)^s = e(sP_B, \mathbf{Q_A})$$

Conclusions on algebraic curves

- Recent, but resist to many attacks, especially in genus 1 or 2.
- Many advantages: **short keys**, short signatures, new tools (pairing), etc.
- Many systems can be interpreted in terms of curves (e.g., **torus based cryptography** of Rubin and Silverberg reinterpreted by Kohel as generalized jacobians of curves).

General conclusions for the three talks

- A lot of systems were designed; new must be added/tested (**biodiversity**).
- **Theory of security** emerged, though not completely satisfactory. Algebra of composition still needed (possible at all?).
- More and more MATHEMATICS involved, but used in a **computer science game**.