

M2 GI (Univ. Grenoble Alpes)
Cours Cybersécurité
Panorama sur ECC et le Post-quantique.

v1.0, Mars 2017

Philippe Elbaz-Vincent



Content of the lecture

- ❑ What is the DLP ?
- ❑ Classical cryptographic methods based on the DLP
- ❑ Elliptic Curve Cryptography (ECC) and the DLP
- ❑ Protocols based on ECC
- ❑ Computational costs and keysize comparison with RSA
- ❑ What happens if we have real-life quantum computers ?

What is the DLP ?

Start with a simple example : assume q is a prime integer and look at integers modulo p . Let g be an integer (not divisible by q) and assume that h is an integer such that $h = g^r \pmod q$ for some natural number r . Can we recover r from the values of h and g ?

Experimental Fact : Knowing h and g , it is very difficult in general to find r if p is big enough. It is what we call the *Discrete Logarithm Problem modulo p* or *DLP* for short.

Idea : Use this difficulty as basis for a cryptosystem.

We will call the mathematical structure of the integers modulo p viewed only with the *multiplicative law* a group. Such a structure can be generalized to other type of objects. In practice we must also assume that $g^p = 1 \pmod q$ for a large prime p and $g^s \neq 1 \pmod q$ for all s with $0 < s < p$. This is called the order of g .

The Diffie/Hellman “Key Exchange”

The Diffie/Hellman (DH) protocol works as follows. Suppose that A and B wish to agree on a common secret key while communicating over an insecure channel. They can perform the procedure below :

- 1 They agree on a group G (i.e., integers modulo a large prime q only seen with the multiplicative law), a large prime number p and an element $g \in G$ of order p .
- 2 A chooses an integer $\alpha \in \{0, 1, \dots, p-2\}$ randomly, computes $a = g^\alpha$ and sends the result a to B (but **keep α secret**).
- 3 B chooses an integer $\beta \in \{0, 1, \dots, p-2\}$ randomly, computes $b = g^\beta$ and sends the result b to A (but **keep β secret**).
- 4 A computes b^α and B computes a^β .
- 5 The common secret key is $K = g^{\alpha\beta}$.



Whitfield Diffie

The fathers of public key cryptography...



Martin Hellman

The data for the DH Key exchange

Notice that the parameters G , g and p are public. The attacker has g^α and g^β , if he/she is able to solve the DLP easily in G (with basis g) then he/she will be able to retrieve K .

☞ The DHP or Diffie/Hellman Problem is to retrieve K from the public data. We do not know if solving the DHP is equivalent to solving the DLP.

Remark : the protocol does not provide authentication between A and B .

Main caveat of the DH protocol

As pointed out, the DH protocol does not provide authentication between the parties. As a result, the protocol is weak against a “Man/Woman-in-the-middle” attack. In the previous scheme, let C be a malicious third party. Then, he/she does the following :

- C intercepts both a and b and send instead $c = g^\gamma$ to A and B
- Then he/she can relay the messages from A to B (and see their contents).

The ElGamal method (Part I)

This public key protocol was described by Taher ElGamal (an Egyptian American cryptographer who pioneered the SSL protocol when he was at Netscape) in 1984.



Taher ElGamal (appointed CTO of Tumbleweed in October 2006)

Public Parameters : We fix a commutative group G and an element g of order p with p a large prime (strictly speaking we do not need this, we only need the order large with a large prime divisor).

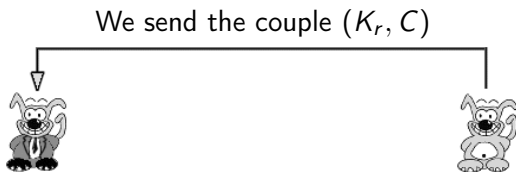
Private key : a random integer $s \in \{1, \dots, p-1\}$.

Public key : $pub = g^s$.

Suppose that we want to send a message $m \in G$ to a recipient. We get the public key say pub and we generate a random integer $r \in \{1, \dots, p-1\}$. We compute

$$K_r = g^r, \quad C = m \times pub^r.$$

The ElGamal method (Part II)



We then get $m = C(K_r^{-s})$. Indeed

$$pub^r = (g^s)^r = (g^r)^s = K_r^s.$$

Remark : Again, encryption is probabilistic. Two encipherings of the same message will (likely) get different ciphertexts.

Computation of the exponentiation is supposed to be fast (as in RSA). The *ElGamal method is used in the Digital Signature Algorithm (DSA)*.

The Digital Signature Standard (DSS)

This is the concrete realisation for the DSA. It was published in 1994 by the NIST. As function H , we will use SHA-1 (which gives a hash of 160 bits) but SHA-2 can also be used.

Public parameters : we choose a 160-bit prime number q and a large prime number $p = aq + 1$ (say p of length at least 1024 bits). We set $G = (\mathbb{Z}/p\mathbb{Z})^\times$. We choose g a generator of G and set $g_a = g^a$. We have the SHA-1 function H which take value in $\mathbb{Z}/q\mathbb{Z}$ and $f : G \rightarrow \mathbb{Z}/q\mathbb{Z}$ the reduction mod q (it makes sense in the below scheme because $q|(p-1)$, but mathematically speaking f is not a function).

Secret key : we choose a random $s \in \mathbb{Z}/q\mathbb{Z}$

Public key : $pub = g_a^s$

N.B. : $\mathbb{Z}/q\mathbb{Z}$ denotes integers modulo q and $(\mathbb{Z}/p\mathbb{Z})^\times$ the integers modulo p which are *invertibles* modulo p .

The Digital Signature Standard (DSS)

Suppose that we want to sign a message M (which could be unrelated to G).

- 1 We choose a random $r \in (\mathbb{Z}/q\mathbb{Z})^\times$ and compute $k_r = f(g_a^r)$ (i.e., $k_r = g_a^r \bmod q$).

- 2 We compute

$$k_M = (sk_r + H(M))r^{-1}.$$

- 3 We send to the recipient M and the pair (k_r, k_M) .

- 4 The recipient computes

$$u = H(M)k_M^{-1}, \quad v = k_r k_M^{-1},$$

and accepts the signature if the following equation holds

$$f(g_a^u \text{pub}^v) = k_r.$$

Complexity of DLP modulo prime numbers (a.k.a “prime finite fields”)

Algorithms	Complexity
Shanks (1971) / Rho method (Pollard/Brent, 1975-80)	$O(\sqrt{p})$
Index calculus (Adleman, 1972 for the initial idea)	$O(\exp(c + o(1)) \sqrt{\log(p) \log(\log(p))})$
FFS (NFS for DLP, 1993-2016)	$O(\exp((C + o(1)) \log(p)^{\frac{1}{3}} \log(\log(p))^{\frac{2}{3}})), C = (64/9)^{\frac{1}{3}}$

In **red** exponential complexity and in **blue** subexponential complexity.

Conclusion : The DLP over finite fields offer no specific advantages over RSA (both problems can be solved in subexponential time). It was hoped during the early days that the DLP (over finite fields) was harder to solve than RSA (at that time only exponential algorithms were known).

....Hence we have to look for new groups..... !

Elliptic Curve Cryptography (ECC)

This new DLP based method was proposed, independently, in 1985 by Neal Koblitz (U. of Washington) and Victor S. Miller (from IBM at the time)



Neal Koblitz

Cryptography based on elliptic curves is mainly commercialised by Certicom, co-founded in 1985 by Scott Vanstone and Gordon Agnew, with lot of consulting from Alfred Menezes (all of them were at U. Waterloo at the time).



The main company behind ECC...

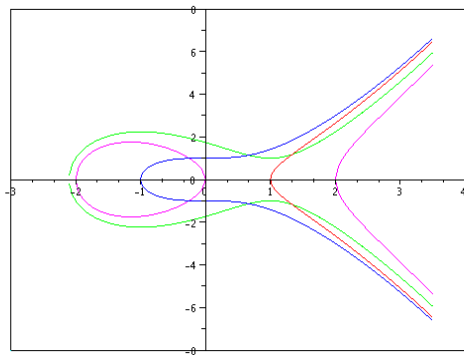


Scott Vanstone

👉 In 2003, NSA bought for \$25 millions of Certicom licensing rights for ECC.

In 2009, Certicom was bought by RIM (owner of BlackBerry cellphones).

What on earth are Elliptic Curves?

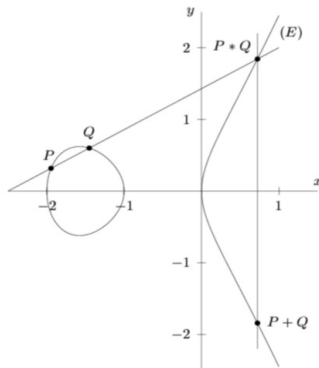


Examples of “elliptic curves” (as we can see them) in the real affine plane

They are particular curves “in the plane” enjoying some symmetry properties and endowed with an abelian group structure...

Viewing the group law on Elliptic Curves

Over \mathbb{Q} or \mathbb{R} , you can geometrically visualize this group law.

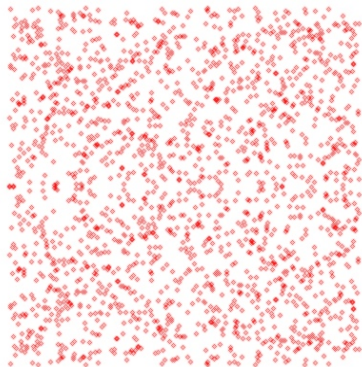


$$y^2 = x(x+1)(x+2).$$

If P and Q are two points of the “elliptic curve” and if $P * Q$ designates the intersection of the straight line passing through P and Q with the curve E , then the point $P + Q$ is obtained by taking the symmetrical opposite (relatively to the x axis) of the point $P * Q$.

Elliptic curves over “integers modulo a prime”

But on a finite fields, the group law is less easy to see...



Some affine points of a curve over $\mathbb{Z}/2003$.

From the cryptographic viewpoint, one will look at curves over \mathbb{Z}/p with p of (binary) size ν with $\nu > 160$.

Elliptic Curves in Mathematics?



Pierre de Fermat (1607/8(?) - 1665)

The famous Fermat's conjecture states that : if u, v, w are rational numbers, $p > 2$ prime and $u^p + v^p + w^p = 0$ holds, then $uvw = 0$. This conjecture was solved by A. Wiles (and R. Taylor) in 1995. One key ingredient in the prove was the study of the elliptic curve $y^2 = x(x + u^p)(x - v^p)$ and showing that this curve *cannot be modular*. The Fermat's conjecture has been instrumental in the developpment of the arithmetic of elliptic curves in Mathematics.

Elliptic Curves over “integers modulo a prime p ”

Assuming $p > 3$, the general equation of an elliptic curve in the plane is of the form $Y^2 = X^3 + aX + b$. This is known as Weierstrass forms in affine coordinate. We have an explicit group law on an elliptic curve, the “point at infinity” plays the role of a neutral element, and there is a finite number of points on the curve with coordinate in $\mathbb{Z}/p\mathbb{Z}$.

Main fact : DLP is hard to solve on “random elliptic curves over $\mathbb{F}_q := \mathbb{Z}/q\mathbb{Z}$ ” and the best generic algorithm (ρ -Pollard) is in $O(\sqrt{s})$ where s is the largest prime in the order of $E(\mathbb{F}_q)$ (the set of points on the curve with coordinates in \mathbb{F}_q).

ECC from Microsoft viewpoint

In their WMA (i.e., the DRM in it), Microsoft is using an elliptic curve over \mathbb{F}_p , where p is a 160 bit prime number (given below). The equation of the curve is given in standard Weierstrass form (as required by most of the standards) $y^2 = x^3 + ax + b$, a and b are coefficients (in \mathbb{F}_p) given below.

All values are represented as packed binary values : in other words, a single value over \mathbb{F}_p is encoded simply as 20 bytes (stored in little endian order). A point on the elliptic curve is therefore a 40 bytes block, which consists of two 20 byte words representing the coordinates (x, y) .

Here are the parameters for the elliptic curve used in MS-DRM :

- $p = 89abcdef012345672718281831415926141424f7$
- $a = 37a5abccd277bce87632ff3d4780c009ebe41497$
- $b = 0dd8dabf725e2f3228e85f1ad78fdedf9328239e$
- $P_x = 8723947fd6a3a1e53510c07dba38daf0109fa120$
- $P_y = 445744911075522d8c3c5856d4ed7acda379936f$
- Order of the curve :
 $89abcdef012345672716b26eec14904428c2a675$

But does it bring interesting performance ?

ECC from Apple viewpoint

Their focus is on mobile device (i.e., small memory footprint) and use what we call “the Montgomery normal form”

$$y^2 = x^3 + cx^2 + x, \quad c \neq \pm 2,$$

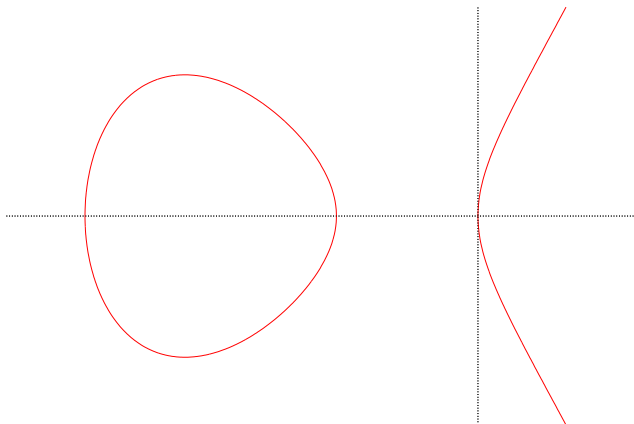
over \mathbb{F}_p with $p = w^s - k$, $w > 1$ and $1 \leq k \leq w - 1$, with $k \equiv 1 \pmod{4}$. The size of w is related to the CPU architecture and s can be seen as the “security parameter”. For instance, on a 32bit CPU with fast 16x16 bits operations, we can take $w = 2^{16}$ (and $s = 10$ as a minimal security level).

ECC from Apple viewpoint (continued)

The curves are usually chosen such that their orders over \mathbb{F}_p are equal to $w^s - j$ (for some $j \leq w^{\frac{1+s}{2}}$), with $c = 4$. The goal is to have “fast encryption” (it is what they call “Small Memory Fast Elliptic Encryption”). In practice, the order is divided by 4 and most of the curves could be put into either Edwards or Jacobi quartic forms. The base point will be chosen of the type $(x, 1)$.

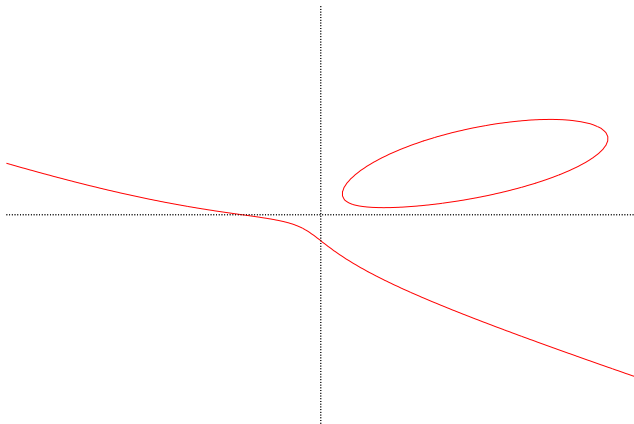
Example : as above, with $k = 57$, $x = 30$, and $j = 134739906578290596453580$.

Weierstrass



$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

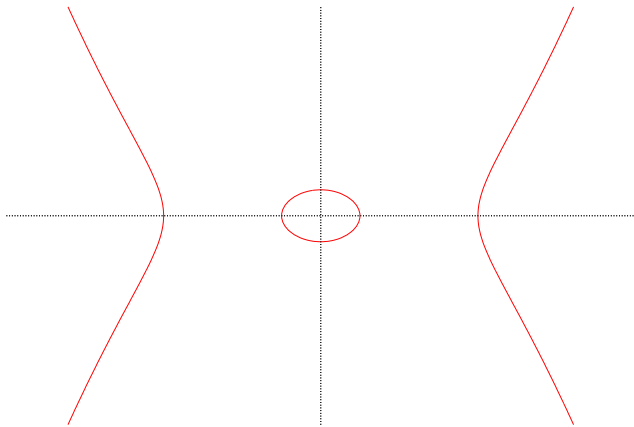
Hesse's cubic



$$x^3 + y^3 + 1 = 3cxy$$

Chudnovsky & Chudnovsky, 1986 ; Joye & Quisquater, 2001 ; Smart, 2001

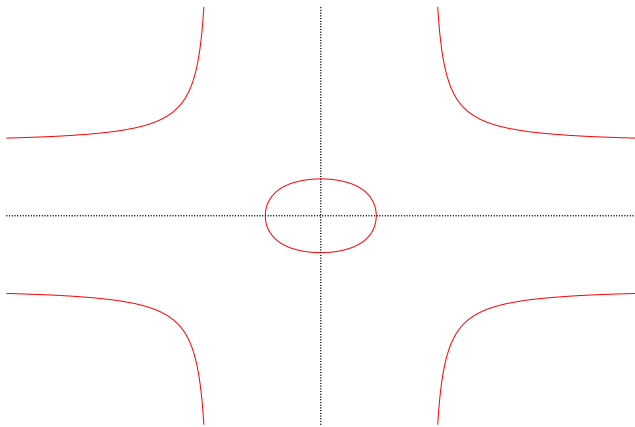
Jacobi's quartic



$$y^2 = x^4 + 2ax^2 + 1$$

Chudnovsky & Chudnovsky, 1986 ; Liardet & Smart, 2001

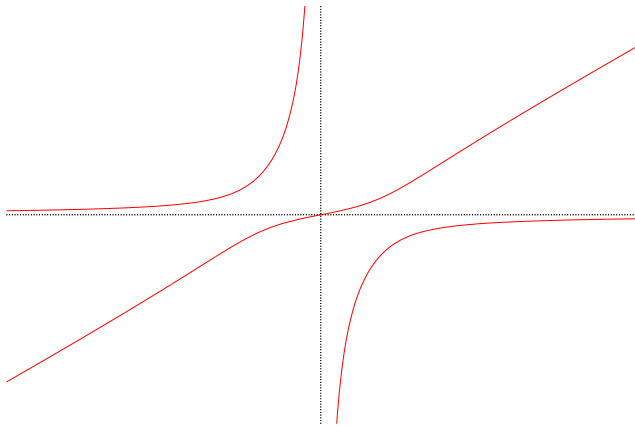
Edwards



$$x^2 + y^2 = a + b(xy)^2$$

Bernstein & Lange, 2007

Huff's cubic



$$ax(y^2 - 1) = by(x^2 - 1)$$

Joye, Tibouchi & Vergnaud, 2010

Influence of coordinate systems on the performances

The use of primes of specific shapes for NIST curves is for efficiency purpose (fast reduction on computers with word of 32bits or 64bits, fast multiplication, fast squaring). The coordinates systems of the curves have also a huge impact on the computational cost. We can summarise some comparison in the following table (for non-binary curves) :

Coordinates system	Doubling	General addition
Affine	1I, 2M, 2S	1I, 2M, 1S
Projective	7M, 3S	12M, 2S
Jacobian	4M, 4S	12M, 4S
Chudnovsky	5M, 4S	11M, 3S
Edwards (2008)	3M, 4S	9M, 1S, 1D
Jacobi quartic (2009)	2M, 5S, 1D	7M, 3S, 1D
Huff model (Joye/Tibouchi/Vergnaud 2010)		12M
Mixed coordinates		
$J + A \rightarrow J$		8M, 3S
$J + C \rightarrow J$		11M, 3S
$C + A \rightarrow C$		8M, 3S

Influence of coordinate systems on the performances

It should be pointed out that some models need constraints on the shape of the curve or equivalently on the group structure of $E(\mathbb{F}_q)$. In particular, for some coordinates systems, we need to have very specific symmetries on the curve. It is for instance the case for Edwards, Jacobi and Huff coordinate systems (but this is usually transparent for the user!).

ECC versions of the classical DLP protocols

As usual we will need to precise the domain parameters. In the case of ECC, there will be the base field F (which could be \mathbb{F}_{2^ν} , or \mathbb{F}_p for a large prime p or eventually \mathbb{F}_{p^ν}), an elliptic curve E (usually given in affine Weierstrass form) over F and a point $P \in E$ of large order.

The DLP for ECC (that we will call ECDLP), will be to recover the integer d from the public data $Q = dP$ and P .

As group G , we will consider, supposing that P is of order n , the cyclic group

$$\langle P \rangle = \{\infty, P, 2P, \dots, (n-1)P\}.$$

The private key is an integer d that is selected uniformly at random from the interval $[1, n-1]$ and the corresponding public key is $Q = dP$.

Domain parameters for ECC

Domain parameters for an elliptic curve scheme

Definition : Domain parameters $D = (q, FR, S, a, b, P, n, h)$ are comprised of :

- 1 The field order q .
- 2 An indication FR (field representation) of the representation used for the elements of \mathbb{F}_q .
- 3 A seed S if the curve was randomly generated.
- 4 Two coefficients $a, b \in \mathbb{F}_q$ that define the equations of the elliptic curve E over \mathbb{F}_q (i.e., $y^2 = x^3 + ax + b$ for a prime field of $\text{char} \neq 2, 3$ and $y^2 + xy = x^3 + ax^2 + b$ for a binary field).
- 5 Two field elements x_P and y_P in \mathbb{F}_q that define a finite point $P = (x_P, y_P) \in E(\mathbb{F}_q)$ in affine coordinates. P has prime order and is called the *base point*.
- 6 The order n of P .
- 7 The cofactor $h = \#E(\mathbb{F}_q)/n$.

Key size comparisons between ECC and RSA

bits of security	Passwd size	RSA key size	ECC key size
80	13	1024	160
112	18	2048	224
128	21	3072	256
256	41	15360	512

Recall that for ECDLP, the best known general attack (assuming well chosen parameters) is in $O(\sqrt{n})$ where n is the order of the base point.

The above table should be understood as follows : given a line with k bits of security, an RSA key size of m bits and an ECC key size of n bits, it means that we need 2^k operations in order to recover the RSA key of m bits or the ECC key of n bits. Passwd sizes (in characters) assume that cost of passwd cipher is neglectible.

ElGamal for ECC

We can adapt the basic ElGamal scheme for EC. With the above parameters, we have the following encryption scheme.

- 1 Represent the message m as a point M in $E(\mathbb{F}_q)$.
- 2 Choose *randomly* $r \in [1, n - 1]$.
- 3 Compute $k_r = rP$.
- 4 Compute $C = M + rQ$.
- 5 Return (k_r, C)

and its decryption scheme, assuming we get the pair (k_r, C) .

- 1 Compute $M = C - dk_r$ and extract m from M .
- 2 Return m .

ECDSA : the new standard for signature

This is analogous of DSA for ECC standardised in PKCS #11, ANSI X9.62, FIPS 186-2, IEEE 1363-2000, ISO/IEC 15946-2, TLS 1.1 and others. Let see the ECDSA signature generation for a message m (assuming a cryptographic hash function H with bitlength outputs smaller than n and also a conversion function from coordinates of the curves to integers).

- 1 Choose *randomly* $r \in [1, n - 1]$.
- 2 Compute $rP = (x, y)$ and convert x to an integer \tilde{x} .
- 3 Compute $k_r = \tilde{x} \bmod n$. If $k_r == 0$ then go to step (1).
- 4 Compute $h_m = H(m)$.
- 5 Compute $k_m = r^{-1}(h_m + dk_r) \bmod n$. If $k_m == 0$ then go to step (1).
- 6 Return (k_r, k_m) .

ECDSA : acceptance of signature

Given a ECDSA signature generation we proceed as follows.

- 1 Verify that k_r and k_m are integers in the interval $[1, n - 1]$. If it fails, we reject the signature.
- 2 Compute $h_m = H(m)$.
- 3 Compute $w = k_m^{-1} \bmod n$.
- 4 Compute $u = h_m w \bmod n$ and $v = k_r w \bmod n$.
- 5 Compute $X = uP + vQ$.
- 6 If $X = \infty$ then reject the signature.
- 7 Convert the x -coordinate of X to an integer \tilde{x} and compute $k' = \tilde{x} \bmod n$.
- 8 If $k' = k_r$ then we accept the signature else we reject the signature.

What happens if the RNG goes havoc in ECDSA ?

Recall first ECDSA for a message m , assuming a domain parameter.

- 1 Choose *randomly* $r \in [1, n - 1]$.
- 2 Compute $rP = (x, y)$ and convert x to an integer \tilde{x} .
- 3 Compute $k_r = \tilde{x} \bmod n$. If $k_r == 0$ then go to step (1).
- 4 Compute $h_m = H(m)$.
- 5 Compute $k_m = r^{-1}(h_m + dk_r) \bmod n$. If $k_m == 0$ then go to step (1).
- 6 Return (k_r, k_m) .

Now suppose that for two distinct messages m_1 and m_2 we got the same r (without *a priori* knowing r).


What happens if the RNG goes havoc in ECDSA ? (continued)

We end up (most likely) with two distinct signatures (k_r, k_{m_1}) and (k_r, k_{m_2}) . Thus

$$k_{m_1} - k_{m_2} = r^{-1}(h_{m_1} - h_{m_2}) \mod n.$$

But, most likely $h_{m_1} - h_{m_2} \neq 0$, hence we can recover r . Then we can recover d as

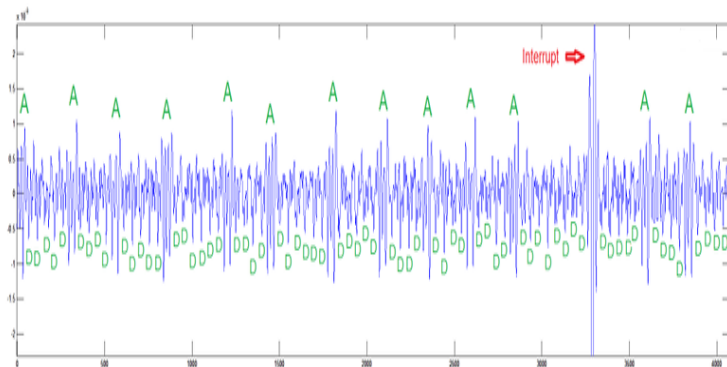
$$d = (k_r)^{-1}(rk_{m_1} - h_{m_1}) \mod n.$$

 it is how the secret key of the PS3 have been recovered due to an error in the RNG! (this is for real)

Conclusion : attack the RNG not the scheme...

Physical attack on ECC software implementation

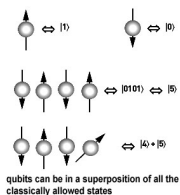
Genkin, Pachmanov, Pipman, Tromer and Yarom have shown (2016) that modern cryptographic software on mobile phones, implementing the ECDSA digital signature algorithm, may inadvertently expose its secret keys through physical side channels (with even a low cost attack) :



In Practical work

- We will compare speed performance between RSA and ECC for signature using OpenSSL.
- We will generate several GnuPG keys using ECC and compare their fingerprints with corresponding RSA keys.
- We will compare signatures time using GnuPG.

RSA, Classical DLP, ECC and Quantum computers



In 1994, Peter Shor (<https://arxiv.org/abs/quant-ph/9508027>) gave algorithms for the factorization of integers and for solving the DLP running in quantum polynomial time and requiring a “big enough” quantum computer. Those algorithms have been improved during the past years in both space requirement and running time.

In 2003, it has been shown by John Proos and Christof Zalka (<https://arxiv.org/abs/quant-ph/0301141>) that we can solve ECDLP in Quantum polynomial time. Furthermore, they have shown that we can solve it with less qubits and faster than the (quantum) factorization of an equivalent RSA modulus.

🔔 **Conclusion :** A good public key cryptosystem for the postquantum era is (still ?) to be discovered...

A Quantum Computer Not universal...but Real !

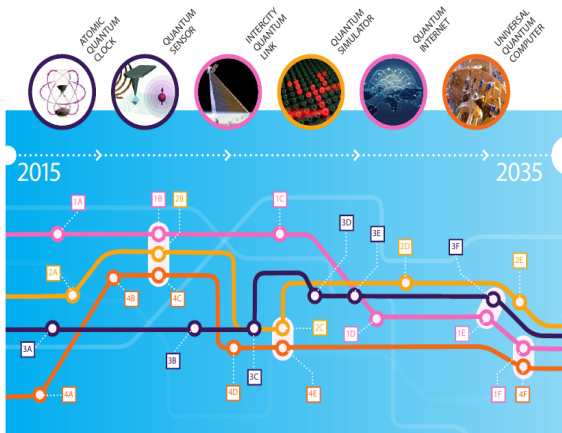


(C) 2016 D-WAVE.

☞ It is not a “Quantum Turing machine”, but it can perform certain type of combinatorial optimizations 100 millions of time faster than a conventinal supercomputer !

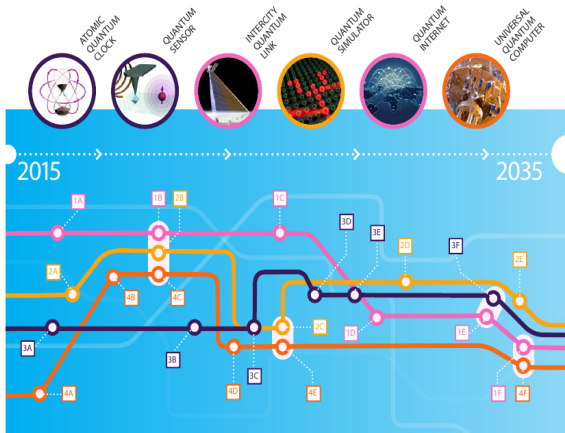
We still don't know if the D-WAVE One X² can speed up some cryptanalysis techniques...

Quantum computer, hash functions and AES ?



(C)

The Post-Quantum Timeline (?)



(C) 2016 «Quantum Manifesto»

👉 In December 2016, the NIST has published an international call for Post-quantum cryptosystems (submission deadline : september 2017)...