Elliptic Curves and Public Key Cryptography
2. DLP in Elliptic Curve Cryptography

Adolfo Quirós
adolfo.quiros@uam.es
Universidad Autónoma de Madrid

3rd VDS Summer School
Techendorf am Weißensee, 17-21/09/2018

Cryptography based on the DLP

We are Given $G =< g >$, a finite cyclic group, $|G| = n$.

Encryption (for example Diffie-Hellman Key Exchange)

Based on calculating exponentials:

$$g^k, \qquad k \in \mathbb{Z}, 1 \leq k \leq n$$

Breaking the cypher

Based on calculating Discrete Logarithms:

$$\text{Given } h \in G, \text{ find } k \in \mathbb{Z}, 1 \leq k \leq n \text{ such that } h = g^k$$

- Exponentiation in G must be easy.
- Finding Discrete Logarithms in G must be hard.

How hard is exponentiation?

"Hardness" of g^k depends on:

1. Difficulty of doing one multiplication.
2. Number of multiplications we need to make.

1. Might depend on G. If $G = \mathbb{F}_p^*$, we never multiply numbers larger than p: 1 multiplication + 1 division to reduce the result mod p take, at most, $c(\log_2 p)^2$ bit-operations: multiplication mod p can be done in time $O(\log^2 p)$ (easy) In all groups we consider from now on, time to multiply will be bounded and will go into the O-constant.
2. But many easy things may amount to a very hard task. Fortunately, to calculate $g^k, 1 \leq k \leq |G|$ (any G) we need at most $O(\log |G|)$ multiplications.

Exponentiation is easy in $\mathbb{F}_p^*$

We can calculate g^k in time $O(\log^3 p)$.

Square and multiply

- How many products do I need to calculate g^{64}?
- And to calculate g^{73}?
- How many products to calculate g^k for arbitrary k?

Write k in binary ($\log_2 k$ elementary divisions, time $O(\log k)$).

$$k = \sum_{i=0}^{\log_2 k} b_i 2^i, \qquad b_i \in \{0, 1\},$$

and notice that

$$g^k = g^{\sum b_i 2^i} = \prod_{b_i=1} g^{2^i}.$$

- With $\log_2 k$ squarings we can calculate all g^{2^i}.
- (At most) $\log_2 k$ additional products give g^k.

Square-and-multiply algorithm (in any group G)

We can calculate g^k doing $O(\log k)$ multiplications.

1. BABY-STEP / GIANT-STEP (SHANKS, 1971)

$G =< g >$, $|G| = n$. Given $h \in G$, find k such that $h = g^k$.

Trying all g^k until one matches h takes time $O(n)$.

Let $t = \lceil \sqrt{n} \rceil$.

- Calculate Giant-Steps: $\{\alpha_0, \alpha_1, \ldots, \alpha_{t-1}\}$, where $\alpha_i = (g^t)^i$.
- Calculate Baby-Steps: $\{\beta_0, \beta_1, \ldots, \beta_{t-1}\}$, where $\beta_i = hg^{-i}$.
- When calculating each β_i, check if $\beta_i = \alpha_j$ for some j.
- $\beta_i = \alpha_j \Rightarrow hg^{-i} = g^{tj} \Rightarrow h = g^{tj+i} \Rightarrow k = tj + i$.
- The algorithm always outputs k because we know that $k = tj + i$ exists.

BABY-STEP / GIANT-STEP (IN ANY GROUP G)

We can find k such that $h = g^k$ in time $O(\sqrt{n} \log n)$.

2. Pohlig-Hellman (1978)

Write $|G| = n = \prod p_i^{e_i}$, with p_i prime.
$h = g^k$. We need to determine $k \mod n$.
Chinese R. Thm. $\Rightarrow$ Enough to determine $k \mod p_i^{e_i}$ for each i.

- $p^e := p_i^{e_i}$, $k = k_0 + k_1 p + k_2 p^2 + \cdots + k_{e-1} p^{e-1} \mod p^e$.
- Let $H < g^{\frac{n}{p}} >$ and notice that $|H| = p$ and $e = O(\log n)$.
- $h^{\frac{n}{p}} = (g^{\frac{n}{p}})^k = (g^{\frac{n}{p}})^{k_0} \in H \Rightarrow$ Find k_0 using BS-GS in H.
- $(hg^{-k_0})^{\frac{n}{p^2}} = (g^{k-k_0})^{\frac{n}{p^2}} = (g^{k_1 p + k_2 p^2 + \ldots})^{\frac{n}{p^2}} = (g^{\frac{n}{p}})^{k_1} \in H \Rightarrow$ Find k_1 using BS-GS in H.
- Find $k \mod p_i^{e_i}$ with $e_i = O(\log n)$ BS-GS.
- Use ChRT (easy) to get $k \mod n$.

Pohlig-Hellman (+BS-GS)

If $|G| = n = \prod p_i^{e_i}$, we can find k such that $h = g^k$ in time

$$O(\sum e_i(\log n + \sqrt{p_i} \log p_i)).$$

Bad, good, not so good groups for DLP-crypto

- $|G| = n = \prod p_i^{e_i} \rightsquigarrow$ can solve DLP in $O(\sum e_i(\log n + \sqrt{p_i}\log p_i))$.
- If n is B-smooth (all $p_i \leq B$) $\rightsquigarrow O(\log^2 n + \sqrt{B}\log n)$.
- DLP is easy if $B << n$, for example, $|G| = p^e$, small p (large e).
- Preferred situation: $|G| = p$ prime.
- For $\mathbb{F}_p^*$, best is $p = 2\ell + 1$ for ℓ prime. $p :=$ secure prime.
- Many: $\ell :=$ Germain prime and $\#\{GerPr \leq x\} \sim \frac{1.113}{\log x}\pi(x)$.

Shoup, 1997

There is no generic algorithm faster than BS-GS + Pohlig-Hellman:

$$n \text{ prime } \rightsquigarrow O(\sqrt{n}\log n), \qquad \text{exponential time.}$$

For some types of groups, there are faster specific algorithms.

- The DLP problem is trivial in $(\mathbb{Z}/n, +)$, even if n is prime.
- We can solve DLP in $\mathbb{F}_p^*$, any p, in subexponential time.

Index Calculus

$\mathbb{F}_p^* =< g >$. Given $h \in \mathbb{F}_p^*$, find $k \in [1, p-1]$ such that $h = g^k$.

1. $\mathcal{B} = \{2, 3, \ldots, p_r\}$, a base of "small" primes.
2. [Precomputation: sieving] Choose $s \in [1, p-1]$ at random and try to factor $g^s \in [1, p-1]$ using only primes in $\mathcal{B}$:

   $$g^s = 2^{e_1} 3^{e_3} \cdots p_r^{e_r} \rightsquigarrow s = e_1 \log_g 2 + e_2 \log_g 3 + \cdots + e_r \log_g p_r.$$
3. [Precomputation: linear algebra] Enough (r linearly independent) of these equations $\rightsquigarrow$ we find

   $$\log_g 2, \log_g 3, \ldots, \log_g p_r.$$
4. [Sieving] Choose $s \in [1, p-1]$ at random and try to factor $hg^s \in [1, p-1]$ using only primes in $\mathcal{B}$:

   $$hg^s = 2^{f_1} 3^{f_3} \cdots p_r^{f_r} \rightsquigarrow \log_g h + s = f_1 \log_g 2 + f_2 \log_g 3 + \cdots + f_r \log_g p_r.$$

What about elliptic curves?

- Index Calculus can be extended to any finite field $\mathbb{F}_q^*$
- It uses subexponential time $O(e^{c(\log n)^{1/3}(\log\log n)^{2/3}})$.
- The (asymptotically) best general purpose factoring method, the Number Field Sieve, needs the same time.
- J. H. Silverman, *Design, Codes and Cryptography*, vol. 20, (2000), 5-40, The Xedni Calculus and the elliptic curve discrete logarithm problem.
- M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein y E. Teske, *Design, Codes and Cryptography*, vol. 20, (2000), 41-64, Analysis of the Xedni Calculus attack.
- There is up to now no subexponential algorithm for the DLP on a well chosen elliptic curve.
- Hence, if we use elliptic curves to do public-key cryptography based on DLP , we will need shorter keys than for the DLP over $\mathbb{F}_q^*$ or for RSA .

Elliptic Curve Discrete Logarithm Problem

$E =$ elliptic curve over $\mathbb{F}_p$ or, more generally, $\mathbb{F}_q$, $q = p^r$.

- Hasse bound: $||E(\mathbb{F}_q)| - (q+1)| \leq 2\sqrt{q}$
- Groups have roughly the same size: $|E(\mathbb{F}_q)| \sim q \sim |\mathbb{F}_q^*|$.

If we take a point $P \in E(\mathbb{F}_q^*)$, such that the (sub)group $G =< P > \subset E(\mathbb{F}_q^*)$ has size $\sim q$, we can look at

Elliptic Curve Discrete Logarithm Problem (ECDLP):

Given $Q \in G =< P >$, find $k \in \mathbb{Z}$ tal que $kP = Q$.

Operations in $E/\mathbb{F}_q$ are "bounded time"

$E : y^2 = x^3 + Ax + B; \quad P_1 = (x_1, y_1), P_2 = (x_2, y_2), P_1 \neq \pm P_2.$

- $x(P_1 + P_2) = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2.$
- $x(2P_1) = \left(\frac{3x_1^2 + A}{2y_1}\right)^2 - 2x_1.$

Cryptography based on the ECDLP

$\sim$ 1985, Koblitz-Miller: Analog of Elgamal.

- $\mathbb{F}_q$ =finite field ; E=elliptic curve/$\mathbb{F}_q$.
- $P \in E(F_q)$; clear messages: $M \in E(F_q)$.
- Private keys: $d \in \mathbb{Z}$; public keys: $Q = dP$.
- Encrypt: $c = (rP, rQ + M) = (c_1, c_2) \in E(F_q) \times E(F_q)$.
- Decrypt: $c_2 - dc_1 = rQ + M - drP = rdP + M - drP = M$.

Key advantage: shorter keys

Security "equivalent to 128 bits in symmetric key" requires:

- Keys of lenght 3072 bits in DLP over finite field or RSA.
- Keys of lenght 256 bits in ECDLP.

Second advantage

- For the same p, many curves E, all with $|E(\mathbb{F}_p)| \sim p$.
- Can look for E where ECDLP is really hard.

A TOY EXAMPLE OF POINTS ON A CURVE

$E : y^2 = x^3 + x + 1$ OVER $\mathbb{F}_5$.

- $E(\mathbb{F}_5) = \{O, (0, \pm 1), (2, \pm 1), (3, \pm 1), (4, \pm 2)\}$
- Hasse bound holds

$$5 + 1 - 2\sqrt{5} = 1,52 \cdots \leq 9 \leq 5 + 1 + 2\sqrt{5} = 10,47 \ldots$$

- $P = (0, 1)$. We know $9P = O$.
- Does P generate all of $E(\mathbb{F}_5)$?
- Calculate $2P, 3P, \ldots$ Or just $3P$.

P	$2P$	$3P$	$4P$	$5P$	$6P$	$7P$	$8P$	$9P$
$(0, 1)$	$(4, 2)$	$(2, 1)$	$(3, -1)$	$(3, 1)$	$(2, -1)$	$(4, -2)$	$(0, -1)$	O

A SIDE REMARK ON FINDING GENERATORS

Let G be a cyclic group, $|G| = n$.

- Unless one knows the factorization of n, there is no easy way to find a g such that $< g >= G$.
- However, if $n = p$ prime or $n = bp$ with p prime and $b << p$ (so that $p \sim n$), it is easy to find g such that $| < g > | = p$.
  - Take $g \in G$ at random.
  - If $g^b \neq 1$, then, since $g^{bp} = 1$ and p is prime $| < g^b > | = p$.
- If $n = p$ even easier: any $1 \neq g \in G$ generates G.
- Example: generator for $\mathbb{F}_p^*$ with $p = 2\ell + 1$ a secure prime.
  - Take $g \in \mathbb{F}_p^*$ at random.
  - If $g^2 \neq 1$, then $| < g^2 > | = \ell$.
  - If $g^2 \neq 1$ and $g^\ell \neq 1$, then $< g >= \mathbb{F}_p^*$.

Elgamal on the curve $E : y^2 = x^3 + x + 1$ over $\mathbb{F}_5$

P	$2P$	$3P$	$4P$	$5P$	$6P$	$7P$	$8P$	$9P$
$(0,1)$	$(4,2)$	$(2,1)$	$(3,-1)$	$(3,1)$	$(2,-1)$	$(4,-2)$	$(0,-1)$	O

- We use the Elgamal-Koblitz-Miller cryptosystem on $< P >= E(\mathbb{F}_5)$.
- The curve E and the point P are public system data.
- We assume that messages have been translated into $M \in E(\mathbb{F}_5)$.
- Private keys are $d \in \mathbb{Z}$, and the corresponding public keys $Q = dP \in E(\mathbb{F}_5)$.
- If we want to send message M to a user whose public key is Q, we choose $r \in \mathbb{Z}$ at random and send $(rP, rQ + M) \in E(\mathbb{F}_5) \times E(\mathbb{F}_5)$.

1. If Alice's private key is $d = 5$, what's her public key?
2. If Alice recieves $((3,4),(0,4))$, what message was transmited?
   - Remember: messages are elements of $E(\mathbb{F}_5)$, not words.
   - Remember: She is not supposed to be able to find r.

Diffie-Helman on Elliptic Curves(ECDH)

- We are given $E, < P > \subset E(\mathbb{F}_q)$.
- A and B choose respectively n_A, n_B, secret integers.
- They exchange (using a public channel) $n_A P$ and $n_B P$.
- The shared secret key is (something extracted from the coordinates of) $n_A n_B P$.

Example

- Consider the curve $E/\mathbb{F}_{7211}$ with equation

$$y^2 = x^3 + x + 7206$$

  and the point $P = (3, 5)$.
- Alice and Bob choose $n_A = 12$ and $n_B = 23$ as 'secrets'.
- What is their common Diffie-Hellman key?
- Common key is $276P = (?, ?)$.

Digital signature algorithm (ECDSA)

Infrastructure

- Everybody agrees on a curve E/F_q^* and a point $P \in E(F_q^*)$ of large prime order n.
- Alice (Bob) chooses, ... secret d_A and publishes $Q_A = d_A P$.

Alice wants to signe a message M she sends to Bob

1. She calculates "using a *hash*" $f = HASH(M)$, an integer of adequate size.
2. Chooses at random a *nonce*: a cryptographycally secure integer $k \in [1, n-1]$ that she will use only once.
3. Calculates $(x_0, y_0) = kP$.
4. Takes $r = x_0 \mod n$. If $r = 0$, go back to step 2.
5. Calculates $s = k^{-1}(f + rd_A) \mod n$. If $s = 0$, go back to step 2.
6. Alice's digital signature for message M is (r, s).

Bob verifies the ECDSA signature

- $f = HASH(M), \quad (x_0, y_0) = kP,$
- $0 \neq r = x_0 \mod n, \quad 0 \neq s = k^{-1}(f + rd_A) \mod n,$
- Alice's signature (r, s).

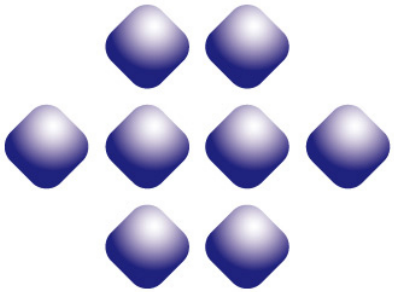
Verification. Bob:

1. Calculates $f = HASH(M)$.
2. Calculates $u_1 = s^{-1}f \mod n$ and $u_2 = s^{-1}r \mod n$.
3. Calculates the point $(x_1, y_1) = u_1P + u_2Q_A$. If this point is O, signature is not valid.
4. Signature is accepted as valid $\iff x_1 = r(= x_0) \mod n$. $[u_1P + u_2Q_A = kP.]$

NOTE: $d_A = \frac{sk-f}{r}$. I know $k \iff$ I know d_A.

Why k can only be used once?

DOES ANYBODY USE THIS?



certicom™

securing innovation

Sells ECC: `https://www.certicom.com/`

Certicom Device Certification Authority For Zigbee Smart Energy

Home > Products > Certicom's Managed PKI Service

Elliptic Curve Cryptography Offers Trusted and Reliable Security for Utility Companies and Meter Manufacturers Worldwide.

Certicom's ZigBee Smart Energy certificate service issues ZigBee Smart Energy certificates to manufacturers whose products are certified by the ZigBee Alliance. The ZigBee Smart Energy PKI uses Elliptic Curve Qu Vanstone (ECQV) *implicit* certificates, which serve as an identity certificate for each ZigBee Smart Energy device as it gets enrolled on the network. Each certificate binds a device MAC address and manufacturing identifier to an ECC key pair, allowing the device to uniquely authenticate itself to the network using its private key. Elliptic Curve Cryptography enables the device to perform efficient and secure authenticated key agreement and perform digital signature operations.

U.S. Department of Commerce / National Institute of Standards and Technology (NIST)

Proposed ECDSA as one of three standards for digital signatures to be used starting 27/07/2001.

FIPS PUB 186-3

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

Digital Signature Standard (DSS)

CATEGORY: COMPUTER SECURITY **SUBCATEGORY: CRYPTOGRAPHY**

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

Issued June, 2009

U.S. Department of Commerce
Gary Locke, Secretary
National Institute of Standards and Technology
Patrick Gallagher, Deputy Director

Appendix D: Recommended Elliptic Curves for Federal Government Use

This collection of elliptic curves is recommended for Federal government use and contains choices for the private key length and underlying fields. These curves were generated using SHA-1 and the method given in the ANS X9.62 and IEEE Standard 1363-2000 standards. This appendix describes the process that was used. Note that these curves are the same as those included in the previous version of this Standard.

D.1.2 Curves over Prime Fields

For each prime p, a pseudo-random curve

$$E: y^2 \equiv x^3 - 3x + b \pmod{p}$$

of prime order n is listed[4]. (Thus, for these curves, the cofactor is always $h = 1$.) The following parameters are given:

D.1.2.1 Curve P-192

$p =$ 6277101735386680763835789423207666416083908700390324961279

$n =$ 6277101735386680763835789423176059013767194773182842284081

$SEED =$ 3045ae6f c8422f64 ed579528 d38120ea e12196d5

$c =$ 3099d2bb bfcb2538 542dcd5f b078b6ef 5f3d6fe2 c745de65

$b =$ 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1

$G_x =$ 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012

$G_y =$ 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811

Uses ECDSA on the curve ***secp256k1*** defined by the equation

$$y^2 = x^3 + 7$$

over the prime field $\mathbb{F}_p$ with

$$\begin{aligned} p =& 115792089237316195423570985008687907853\backslash\backslash \\ & 269984665640564039457584007908834671663 \\ =& 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \end{aligned}$$

p is a 78-digit, 256-bits prime.
It has very few signed bits (speeds-up kP).

$$\begin{aligned} n =& |secp256k1(\mathbb{F}_p)| = \\ & 115792089237316195423570985008687907853\backslash\backslash \\ & 283756427907490438260516314151816149433 7 \end{aligned}$$

n is also a 78-digit, 256-bits prime.
Good for cryptography!

EGIZ E-Government Innovationszentrum

http://www.egiz.gv.at
email: post@egiz.gv.at
tel.: +43 (316) 873 – 5514
fax: +43 (316) 873 – 5520
Inffeldgasse 16a / 8010 Graz / Austria

Integrationsmöglichkeit e-card in Windows

Windows Logon mit e-card

Wolfgang Bauer

bescheinigt sind. Daher wird in weiterer Folge nur mehr von der e-card gesprochen.

Die e-card verwendet den Elliptic Curve Digital Signature Algorithm (ECDSA) als Signaturalgorithmus. Dabei kommt die vom National Institute of Standards and Technology (NIST) empfohlene elliptische Kurve P-192 [3] zum Einsatz. Dies ist eine 192-Bit Kurve über Primzahlenkörper. Unter Windows wird Kryptographie basierend auf elliptischen Kurven erst seit Windows Vista unterstützt. Die Unterstützung beschränkt sich leider auf elliptische Kurven über Primzahlenkörpern mit Schlüssellängen von 256 Bit und darüber [1]. Das bedeutet, dass derzeit Signaturen und Zertifikate der e-card nicht von Windows Vista unterstützt werden.

How to choose E and $< P >\subset E$ (1)

Fix a prime p and a finite field $\mathbb{F}_q$ with $q = p^r$. (often $r = 1$).

1. Choose a random elliptic curve $E/\mathbb{F}_q$. To do that [when $p \neq 2$]:
   - Choose at random $A, B \in \mathbb{F}_q$
   - If $4A^3 + 27B^2 \neq 0$, the curve is $y^2 = x^3 + Ax + B$.
   - If $4A^3 + 27B^2 = 0$, try again.

   Remark 1: For $p \neq 2$, NIST fixes $q = p$ and $A = -3$.
   Remark 2: For $p = 2$, adapt the algorithm so that the output curve is $y^2 + xy = x^3 + Ax^2 + B$.
2. Find $N := |E(\mathbb{F}_q)|$.
   Using some optimized version of the *Schoof Algorithm*.

How to choose E and $< P >\subset E$ (2)

3. Check that the curve is not cryptographically weak.
   - It is not anomalous.
     That is, I do not have $q = p = N$, in which case there is an attack based on p-adic points.
   - It is not weak against a MOV attack.
     MOV= Menezes-Okamoto-Vanstone. An attack using pairings. In particular, E should not be supersingular (we should not have $N \equiv 1 \mod p$).
   - Check it can not be attacked in any other known way.

If it turns out that E is weak, go back to step 1 .

How to choose E and $< P > \subset E$ (3)

4. Look for a large prime n such that $E(\mathbb{F}_q)$ has a subgroup (necessarily cyclic) of order n.
   - Can try to"easily factor" N.
   - Want to arrive at $N = s \cdot n$, with n prime and $s \leq B$ [B a "small" bound fixed a priori].
   - Checking if this is the case is easy (How can we do it?).
   - If N can not be easily factored, go back to step 1 .
5. Look for a random point $R \in E(\mathbb{F}_q)$, $R \neq O$.
   - Choose a random $x \in \mathbb{F}_q$ and check if $f(x) = x^3 + Ax + B$ is a square in $\mathbb{F}_q$.
   - In case it is, $R = (x, \sqrt{f(x)})$; if not, choose another x. [Modify the method when $p = 2$.]
   - (Why should this work fast?)
6. Look for a random point $P \in E(\mathbb{F}_q)$ such that $| < P > | = n$.
   - Calculate $P := sR$.
   - If $P \neq O$, this P works.
   - If $P = O$, go back to step 5 .

We can also look for the curve $E/\mathbb{F}_q$ and the point $R \in E(\mathbb{F}_q)$ simultaneously

Fix a prime p and a finite field $\mathbb{F}_q$ with $q = p^r$. (often $r = 1$).

1. Choose at random $x_0, y_0, A \in \mathbb{F}_q$.
2. Define $B := y_0^2 - x_0^3 - Ax_0$.
   - If $4A^3 + 27B^2 \neq 0$, the curve is $y^2 = x^3 + Ax + B$ y $R = (x_0, y_0)$.
   - If $4A^3 + 27B^2 = 0$, repeat.

- These play the role of steps 1 and 5 in the previous algorithm.
- The other steps are the same.

Real messages $\rightsquigarrow$ Points in an elliptic curve

Say we are using $E : y^2 = x^3 + Ax + B$ over $\mathbb{F}_p$. [Easily adapted to $\mathbb{F}_q$]

- We may asume messages are integers n, $0 \leq n < N \leq |E(\mathbb{F}_p)|$.
  - If we are using an alphabet $\mathcal{A}$ with a characters, take as messages blocks $l_0 l_1 \ldots l_{m-1}$ of m letters, where $a^m \leq N$.
  - The message is represented by the integer $n = l_0 + l_1\, a + l_2\, a^2 + \cdots + l_{m-1}\, a^{m-1} < a^m \leq N$.
  - To recover the message, write n in base a.
- We must turn these into points of $E(\mathbb{F}_p)$ in such a way that we can recover n from the point.

- Can $n \rightsquigarrow x$?
- Can $n \rightsquigarrow y$?
- Can $n \rightsquigarrow \ldots$?

Nobody knows how to make the assignment $n \rightsquigarrow P$ in a deterministic way, but we can make it work with probability as large as we want.

- We want to assign $n \rightsquigarrow P \in E(\mathbb{F}_p)$.
- We do not want to fail more than once in 2^K tries.

1. We use a prime such that $p > NK$.
2. We write integers between 1 and NK as $nK + j$, $j = 1, \ldots, K$, and put $\{1, \ldots, NK\} \subset \mathbb{F}_p$.
3. Given n (our message!), we obtain for each $j = 1, \ldots, K$ an element $x \in \mathbb{F}_p$.

   If $x^3 + Ax + B$ is a square in $\mathbb{F}_p$, we find y such that $y^2 = x^3 + Ax + B \rightsquigarrow P_n = (x, y)$ is the point attached to n.
4. We can recover n from P_n as $n = [(x - 1)/K]$.
5. $x^3 + Ax + B$ is a square roughly 50% of the time $\Rightarrow$ the algorithm fails with probability $\sim 2^{-K}$.

3-PARTY KEY EXCHANGE

A, B and C want to agree on a key k (a number $0 < k < p$, that is, an element of $\mathbb{F}_p*$).

First solution: D.-H. with two rounds of communication.

0. $\mathbb{F}_p^* =< g >$.
1. A chooses (secret) exponent a; B chooses b; C chooses c.
2. First round:
   - A sends g^a to B (and to C).
   - B sends g^b to C (and to A).
   - C sends g^c to A (and to B).
3. Second round:
   - A sends g^{ca} to B (and to C).
   - B sends g^{ab} to C (and to A).
   - C sends g^{bc} to A (and to B).
4. A calculates g^{bca}; B calculates g^{cab}; C calculates g^{abc}.
   AIll three of them have $k = g^{abc}$.

We can do the same using D-H on an Elliptic Curve, but Elliptic Curves provide other tools

$E/\mathbb{F}_q$, $P \in E(\mathbb{F}_q)$, $m = |< P >|$, $\mu_m := \{x \in \overline{\mathbb{F}}_p^* : x^m = 1\}$

(Modified) Weil pairing

$$e :< P > \times < P > \longrightarrow \mu_m$$

- Bilinear: $e(aP, bP) = e(P, P)^{ab}$.
- Defined: $e(P, P) \neq 1$.

One round 3-party key exchange [A. Joux]

1. A chooses (secret) integer a; B chooses b; C chooses c.
2. - A sends aP to B and C.
   - B sends bP to C and A.
   - C sends cP to A and B.
3. A calculates $e(bP, cP)^a = [e(P, P)^{bc}]^a$; B and C ....
4. The common key is $k = e(P, P)^{abc}$.

We could go on...

- Elliptic curves can be used to find prime numbers, and also to factor composite numbers.
- The Weil pairing can be used to do "Identity Based Cryptography".
- ...
- I. Semaev, *New algorithm for the discrete logarithm problem on elliptic curves* (2015). Uses *summation polynomials* to give an algorithm that could be subexponential in $E(\mathbb{F}_{2^r})$. Will elliptic curves loose their "competitive advantage"? Not for the moment if r is prime.
- We have to get ready for big enough quantum computers. Postquantum cryptography

More about that tomorrow