Introduction to Cryptography

F. Koeune – O. Pereira

Slides 06

Key Agreement

How do A and B get their keys?

Key agreement:

- Pre-Internet: meet, run Gen, store k, leave
- Internet: no meeting, just public conversation

Can we create a shared secret from a public conversation?

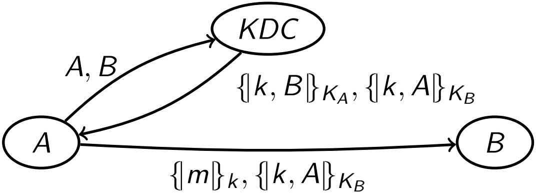
Key distribution

Consider large networks, with n users

How do we share the keys?

- every new user brings n new keys, to be setup with all other users, or
- key distribution centers (KDC) can be used

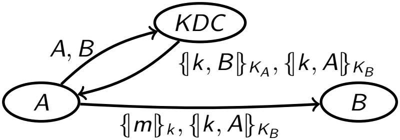
Suppose A and B share K_A and K_B (resp.) with KDC



($\{\!|x|\!\}_y$ stands for the symmetric encryption of x with key y)

Key distribution

Suppose A and B share K_A and K_B (resp.) with KDC



Advantages:

- Only one long-term key to store per user
- Only one key to create when adding a user

Challenges:

- Can we trust KDC?
- What if KDC fails? (robustness)

The Public Key Revolution

Merkle – Hellman – Diffie

The Diffie-Hellman Protocol (Outline)

The Diffie-Hellman protocol [DH76]

$$\begin{array}{ccc} A & \xrightarrow{\quad g,\, g^x \quad} & B \\ \Downarrow & & \Downarrow \\ \bullet & \xleftarrow{\quad g^y \quad} & \bullet \end{array}$$

- A and B compute $k := g^{x \cdot y}$ as $(g^y)^x$ or $(g^x)^y$
- Computing $g^{x \cdot y}$ without x or y seems to require a logarithm extraction

Challenges:

- Can we make logarithm extraction difficult?
- Can we make sure that $g^{x \cdot y}$ does not become too big?

Asymmetric cryptography

Merkle – Hellman – Diffie

Groups

We can do DH with any set $\mathbb{G}$ equiped with a multiplication operation that "works well": we need $(g^x)^y = (g^y)^x$

We use:

Prime Order Cyclic Groups

That is:

- $\mathbb{G}$ is a group, i.e., a set equipped with operator "$\cdot$":
  - $\exists 1_{\mathbb{G}}$ (or just "1") s.t.: $\forall g \in \mathbb{G} : 1 \cdot g = g \cdot 1 = g$
  - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - $\forall g \in \mathbb{G}, \exists h \in \mathbb{G} : g \cdot h = 1$ (h noted g^{-1})
- Cyclic: $\exists g \in \mathbb{G} : \mathbb{G} = \{g, g^2, g^3, \ldots, g^{|\mathbb{G}|}\}$
  Such a g is called a *generator*
- Prime Order: $|\mathbb{G}|$ is prime

Examples

- $\mathbb{Z}$ equipped with usual "$+$" is a group
- $\mathbb{Z}_n = \{0, \ldots, n-1\}$ with "add $\bmod n$" is a cyclic group
  $a + b \bmod n$ is the remainder of division of $a +_{\mathbb{Z}} b$ by n
- $\mathbb{Z}_p$ is a cyclic group of prime order if p is prime

Logarithm extraction

Can DH be secure in $\mathbb{Z}_p$?

Example: take $p = 17$, $g = 3$ and $x = 11$.

- Given $(g = 3, g^x = 16)$, can you compute x?
  (Remember: group operation is addition mod 17, but noted multiplicatively)

This is finding x s.t. $3x = 16 \bmod 17$

- x is $16 \cdot$ the **multiplicative inverse** of 3 mod 17
- Can we compute it efficiently (i.e., in PPT)?
  Does it exist for any p and g?

$\mathbb{Z}_p^*$

$\mathbb{Z}_p^* = \{1, \dots, p-1\}$ with "mult mod p" is a group when p is prime

- *Ex:* $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$ and $3 \cdot 6 = 4 \in \mathbb{Z}_7^*$

Properties:

- $\exists 1_{\mathbb{G}}$ (or just "1") s.t.: $\forall g \in \mathbb{G} : 1 \cdot g = g \cdot 1 = g$
- $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- $\forall g \in \mathbb{G}, \exists h \in \mathbb{G} : g \cdot h = 1$ (h noted g^{-1})

Inversion in $\mathbb{Z}_p^*$

Let p be prime and $a \in \mathbb{Z}_p^*$

- Is there $b : a \cdot b = 1$?
- Can we compute this b?

We need $b, k \in \mathbb{Z} : ab + kp = 1 \in \mathbb{Z}$

Multiplicative inverse mod N

More general statement:

Proposition: a is invertible mod N iff $\gcd(a, N) = 1$

$\Rightarrow$ Suppose b is the inverse of a mod N

- $ab = 1 \pmod{N} \Rightarrow \exists c : ab - 1 = cN$
- Then $ab - cN = 1$.
  Suppose $d|a$ and $d|N$. Then $d|1 \Rightarrow d = 1$.
  $\Rightarrow \gcd(a, N) = 1$

Multiplicative inverse mod N

Proposition: a is invertible mod N iff $\gcd(a, N) = 1$

$\Leftarrow$ Suppose $\gcd(a, N) = 1$

- We show $\exists X, Y : Xa = YN + 1$ (then $X = a^{-1}$)
- Let $d = Ua + VN$ be the smallest integer of $\mathcal{S} := \{\hat{U}a + \hat{V}N : \hat{U}, \hat{V} \in \mathbb{Z}\} \cap \mathbb{N}^{\geq 1}$
- Fix any $c := U'a + V'N$, $c \in \mathcal{S}$
  Express $c = qd + r$ with $0 \leq r < d$.
- We have $r = U'a + V'N - q(Ua + VN) \Rightarrow r = 0$
- So, $d|c$, and
  $d|a \in \mathcal{S}$ and $d|N \in \mathcal{S}$ (take $(U', V') = (0, 1)$ or $(1, 0)$)
  $\Rightarrow d = 1$

Multiplicative inverse mod N

Corollary: if a is invertible mod N then it has one only inverse in $[0, N)$

- Suppose $a \cdot b = a \cdot b' = 1 \pmod{N}$
  $\Rightarrow a \cdot (b - b') = 0 \pmod{N}$
  $\Rightarrow b - b' = 0 \pmod{N}$, since $\gcd(a, N) = 1$

Corollary: if p is prime, then every element of $\mathbb{Z}_p^*$ has an inverse

Can we compute the inverse?

Let $\gcd(a, p) = 1$, we need $b, k \in \mathbb{Z} : ab + kp = 1 \in \mathbb{Z}$

Trick of the **Extended Euclidian Algorithm**:

$$\text{If } a < p, \text{ write } p = qa + r \text{ with } 0 \leq r < q$$
$$\text{So, solve equation } ab' + kr = 1$$

Observations:

- We have $b' = b + kq$, so solution of 2nd equation gives solution to 1st equation
- 2nd equation looks for inverse of r mod a, and $a < p$ so recursion can work
- If $a \leq p/2$ then modulus lost one bit
  If $a > p/2$ then $r < p/2$, and modulus will lose one bit next
  So efficient recursion of depth at most $2|p|$

Extended Euclidian Algorithm

Extended Euclidian algorithm eEucl (twisted):

in: a, b, with $a \geq b > 0$
out: (X, Y) where $Xa + Yb = 1$

if $b = 1$ **then return** $(0, 1)$
else $(q, r) := (\lfloor a/b \rfloor, [a \bmod b])$
(X', Y'):=eEucl(b, r)
return $(Y', X' - Y'q)$

Observe, if $(q, r) := (\lfloor a/b \rfloor, [a \bmod b])$

- if $X'b + Y'r = 1$ then
  $X'b + Y'(a - bq) = 1$
  $Y'a + (X' - Y'q)b = 1$

Extended Euclidian Algorithm

Compute $(X, Y) = \text{eEucl}(a, b)$
if $b = 1$ **then return** $(0, 1)$
else $(q, r) := (\lfloor a/b \rfloor, [a \bmod b])$
(X', Y'):=eEucl(b, r)
return $(Y', X' - Y'q)$

Example:

			q	r
Call	57	53	1	4
Call	53	4	13	1
Call	4	1		

- Last recursive call outputs $(0, 1)$
- $1st$ unwinding outputs $(1, -13[= 0 - 1 \cdot 13])$
- $2nd$ unwinding outputs $(-13, 14[= 1 - (-13) \cdot 1])$
- Correctness: $1 = -13 \cdot 57 + 14 \cdot 53 = -741 + 742$
  So $53^{-1} = 14 \in \mathbb{Z}^*_{57}$

Logarithm extraction

Can DH be secure in $\mathbb{Z}_p$?

Example: take $p = 17$, $g = 3$ and $x = 11$.

- Given $(g = 3, g^x = 16)$, can you compute x?
  (Remember: group operation is addition mod 17, but noted multiplicatively)

This is finding x s.t. $3x = 16 \bmod 17$

And computing multiplicative inverses can be done in PPT
So, no, DH protocol is not secure in $\mathbb{Z}_p$

$\mathbb{Z}_p^*$

$\mathbb{Z}_p^* = \{1, \ldots, n-1\}$ with "mult mod p" is a group

- *Ex:* $\mathbb{Z}_{17}^* = \{1, 2, \ldots, 16\}$ and $3 \cdot 7 = 4 \in \mathbb{Z}_{17}^*$

What about computing logarithms in $\mathbb{Z}_p^*$?

Example: take $p = 17$, $g = 3$ and $x = 11$.

- Given $(g = 3, g^x = 7)$, can you compute x?
  (Remember: group operation is multiplication mod 17)
- No obvious way apart from exhaustive search
  But how long would it be?
- Depends of **order** of g, i.e., smallest $i : g^i = 1$
  Powers of
  $\langle 3 \rangle = \{3, 9, 10, 13, 5, 15, 11, 16, 14, 8, 7, 4, 12, 2, 6, 1\} = \mathbb{Z}_{17}^*$
  But powers of $\langle 16 \rangle = \{16, 1\}$

Fermat's little theorem

Let $\mathbb{G}$ be a commutative group with $m := |\mathbb{G}|$
Then, $\forall g \in \mathbb{G}, g^m = 1$

Proof:

- Let $g_1, \ldots, g_m$ be the elements of $\mathbb{G}$
- Observe $g_1 \cdots g_m = (gg_1) \cdots (gg_m)$
  (all terms of the right-hand product must be distinct)
- Multiply both sides by $(g_1 \cdots g_m)^{-1}$

Corollaries:

- $g^i = g^{[i \bmod m]}$
- $\forall g \in \mathbb{G}$, if $\mathrm{ord}(g) = i$, then $i|m$
  (Otherwise, $g^{[m \bmod i]} = 1$)

$\mathbb{Z}_p^*$

Can we find g s.t. $\mathrm{ord}(g) = p - 1$?

Theorem (Gauss, 1801):

$$\exists g \in \mathbb{Z}_p^* \text{ s.t. } \mathrm{ord}(g) = p - 1 \text{ for every prime } p$$

And these *primitive roots* are quite common:

$$\Pr[g \text{ is a primitive root}] = \frac{1}{\log \log p}$$

when g and p are picked uniformly at random

The Discrete Logarithm Problem

Consider experiment $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$ where g generates the group $\mathbb{G}$ of order q, with $|q| = n$
2. Choose $h \overset{R}{\leftarrow} \mathbb{G}$
3. Set $x \leftarrow \mathcal{A}(\mathbb{G}, q, g, h)$
4. $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1 \Leftrightarrow g^x = h$

The *discrete logarithm problem* is hard relative to $\mathcal{G}$ if $\forall$ PPT $\mathcal{A}$, there is a negligible ϵ s.t.:

$$\Pr[\mathsf{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \epsilon(n)$$

The Discrete Logarithm Problem in $\mathbb{Z}_p^*$

1. How to choose p?
2. How to choose g?

Generic Algorithms for the DL

Given an instance $(\mathbb{G}, q, g, h)$, how can we find x?

In a *generic* way, that is, assuming that the only available operation is the multiplication in $\mathbb{G}$

1. Try all x until $g^x = h$
   Takes up to q attempts, so $|q| = 128$ looks good
2. Baby-step, giant-step algorithm, with $s = \sqrt{q}$:
   (i) Compute g^s (takes $\mathcal{O}(\sqrt{q})$ steps)
   (ii) Compute $S = \{g^s, g^{2s}, \ldots, g^{s^2}\}$ (takes $\mathcal{O}(\sqrt{q})$ steps)
   (iii) Search $i \in \{0, \ldots, q\}$ s.t., if $g^i h = g^{js} \in S$ (takes $\mathcal{O}(\sqrt{q})$ steps)
   (iv) Return $x = js - i$

   Takes $\mathcal{O}(\sqrt{q})$ work, so $|q| = 256$ looks good

No asymptotically better solution is known.

Non Generic Algorithms for the DL

Given an instance $(\mathbb{Z}_p^*, q, g, h)$, how can we find x?
Can we do better than generic algorithms?
E.g., exploit the fact that addition $\bmod p$ works too?

Yes:

- Index calculus: $\approx \mathcal{O}(2^{|p|^{1/2}(\log |p|)^{1/2}})$ [Kraitchik, 1922]
- NFS: $\approx \mathcal{O}(2^{|p|^{1/3}(\log |p|)^{2/3}})$ [Lenstra, Lenstra, 1993]

So, today:

- $|p| = 3072$
- $|q| = 256$: we can work in subgroups!

How to find a big prime p?

How to select a large prime p?

- Select random integer, test primality, retry if failure

Can this work?

- Is primality testing easier than factoring?
  Can we test primality using a PPT algorithm?
- How many failures should we expect?
  What is the prime numbers' density?

Primality testing

Is primality testing easier than factoring?

> *"In August 2002 one of the most ancient computational problems was finally solved."* [1]

[Agrawal-Kayal-Saxena 2002]:

Prime is in **P**!

[1] 2006 Gödel Prize citation

Primality testing

Prime is in **P**!

How can we test for primality without factoring?

Reminder: Let $\mathbb{G}$ be an commutative group with $m := |\mathbb{G}|$
Then, $\forall g \in \mathbb{G}, g^m = 1$

- N prime $\Rightarrow |\mathbb{Z}_N^*| = N - 1$ and $a^{N-1} = 1 \pmod{N}$
- if $a^{N-1} \neq 1 \pmod{N} \Rightarrow N$ is composite

Primality testing

Test of primality for N:

Repeat t times:

- $a \leftarrow [1, N-1]$
- **If** $\gcd(a, N) \neq 1$ **then return** "composite"
- **If** $a^{N-1} \neq 1$ **then return** "composite"

return "prime"

Expectation: Some a will be a compositeness witness

How many a's do we need to test, on average?

Primality testing

Theorem: Suppose N is composite and
$\mathrm{Good} := \{a : a^{N-1} \neq 1 \pmod{N}\}$, and $b \in \mathrm{Good} \cap \mathbb{Z}_N^*$.
Then $|\mathrm{Good}| \geq \frac{|\mathbb{Z}_N^*|}{2}$

Define

- $\mathrm{Bad} := \mathbb{Z}_N^* - \mathrm{Good} = \{a : a^{N-1} = 1 \pmod{N}\}$

Observe:

- $\forall a \in \mathrm{Bad},\ (ab)^{N-1} = a^{N-1} b^{N-1} = b^{N-1} \neq 1 \pmod{N}$

$\Rightarrow$ $ab \in \mathrm{Good}$

Besides:

- If $a_1, a_2 \in \mathrm{Bad}$ then $a_1 \neq a_2 \Rightarrow a_1 b \neq a_2 b$

$\Rightarrow$ $|\mathrm{Good}| \geq |\mathrm{Bad}|$, and $|\mathrm{Good}| \geq \frac{|\mathbb{Z}_N^*|}{2}$

Primality testing

Theorem: Suppose N is composite and
$\mathrm{Good} := \{a : a^{N-1} \neq 1 \pmod N\}$, and $b \in \mathrm{Good} \cap \mathbb{Z}_N^*$.
Then $|\mathrm{Good}| \geq \frac{|\mathbb{Z}_N^*|}{2}$

What does it mean?

- If there is a compositeness witness, then our primality test algorithm succeeds with proba $\approx \frac{1}{2}$.
- t repetitions ensure $\Pr[\mathrm{error}] \approx \frac{1}{2^t}$
- Our witness test succeeds in PPT – if there is a witness
- But no guarantee that a witness exists!
- Carmichael numbers do not have witnesses
  561, 1105, 1729, 2465, 2821, 6601, ...

Primality testing

- Carmichael numbers do not have witnesses
  561, 1105, 1729, 2465, 2821, 6601, . . .

Can we do anything better?

- Rather than considering $a^{N-1} \pmod{N}$, also look at $a^{\frac{N-1}{2}}$, $a^{\frac{N-1}{4}}$ . . .
- . . .
- This provides the Miller-Rabin test, which has no "Carmichael"-type exception

How to find a big prime p?

How to select a large prime p?

- Select random integer, test primality, retry if failure

Can this work?

- Is primality testing easier than factoring?
  Can we test primality using a PPT algorithm?
- How many failures should we expect?
  What is the prime numbers' density?

Density of Primes

Prime number theorem [de la Vallée-Poussin 1896], [Hadamard 1896]

$$\Pr[N \text{ is prime}] \approx \frac{constant}{|N|}$$

$|N|^2$ attempts makes failure probability negligible...

Exponentiation in Groups

How to compute g^x?

- $g \cdot g \cdot g \cdots g$ (i.e., $x - 1$ multiplications)
  Problem: complexity is exponential in $|x|$!
- Better idea: $g^8 = ((g^2)^2)^2)$
  So: g^{2^i} can be computed with $i - 1$ multiplications
- General version: **Square and Multiply** algorithm
  $$g^x = \begin{cases} (g^{\frac{x}{2}})^2 & x \text{ is even} \\ g \cdot (g^{\frac{x-1}{2}})^2 & x \text{ is odd} \end{cases}$$
  Requires only $|x|$ iterations...

What about other groups?

Is there any useful group besides $\mathbb{Z}_p^*$?

Yes!
Groups defined on the points of elliptic curves

[Koblitz 1985],
[Miller 1985]

Neal Koblitz

Elliptic Curve Cryptography (ECC)

Groups defined on the points of elliptic curves...

Is this useful?

- Harder to break
  Best DL extraction algorithms are generic
- We can use shorter keys
  $\mathbb{Z}_p^*$ 3072 bits often compared to ECC 256 bits
- Faster algorithms
  Specially useful in constrained environments

Elliptic Curve Cryptography (ECC)

Groups defined on the points of elliptic curves...

Why don't we all use ECC?

- $>$ 100 patents held by US companies
  - efficient multiplication on EC
  - efficient point representation on EC
  - ...

  Considerably slowed down adoption!

Elliptic Curve Cryptography (ECC)

Groups defined on the points of elliptic curves. . .

Why don't we all use ECC?

- NSA reported to have paid $25.000.000 to Certicom for 26 patents licences

SUITE B includes:

Encryption:	Advanced Encryption Standard (AES) - FIPS 197 (with keys sizes of 128 and 256 bits) http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*
Digital Signature:	Elliptic Curve Digital Signature Algorithm - FIPS 186-2 (using the curves with 256 and 384-bit prime moduli) http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf*
Key Exchange:	Elliptic Curve Diffie-Hellman Draft NIST Special Publication 800-56 (using the curves with 256 and 384-bit prime moduli) http://csrc.nist.gov/CryptoToolkit/kms/keyschemes-Jan03.pdf*
Hashing:	Secure Hash Algorithm - FIPS 180-2 (using SHA-256 and SHA-384) http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf*

Elliptic Curves

Groups defined on the points of elliptic curves...

How does it work?

Elliptic curves are defined by the equation:

$$y^2 = x^3 + Ax + B$$

Examples:

$y^2 = x^3 - 2x$

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

Elliptic Curves

Elliptic curves are defined by the equation:

$$y^2 = x^3 + Ax + B$$

We want to work with finite groups/fields!

- Consider:

  $$y^2 = x^3 + Ax + B \pmod{p}$$

  instead, with the constraints
  - $p > 3$ is prime (simplifies treatment)
  - $4A^3 + 27B^2 \neq 0 \pmod{p}$ (guarantees distinct roots)

Elliptic Curves

Consider:

$$y^2 = x^3 + Ax + B \pmod{p}$$

with the constraints

- $p > 3$ is prime (simplifies treatment)
- $4A^3 + 27B^2 \neq 0 \pmod{p}$ (guarantees distinct roots)

Define:

- $E(\mathbb{Z}_p) := \{(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p \text{ on the curve } \} \cup \mathcal{O}$
- $\mathcal{O}$ is the *point at infinity*
  $\mathcal{O}$ can be imagined at (x, ∞) $(\forall x)$

Elliptic Curves

Example: $y^2 = x^3 - 2x + 2 \mod 7$

Only 4 elements of $\mathbb{Z}_7$ have square roots

- $0 = 0^2$; $1 = 1^2 = 6^2$; $2 = 3^2 = 4^2$; $4 = 2^2 = 5^2$

Consider $f(x) := x^3 - 2x + 2 \mod 7$

- $f(0) = 2 \Rightarrow (0,3)$ and $(0,4) \in E(\mathbb{Z}_7)$
- $f(1) = 1 \Rightarrow (1,1)$ and $(1,6) \in E(\mathbb{Z}_7)$
- $f(2) = 6$, but 6 has no square roots $\mod 7$
- $f(3) = 2 \Rightarrow (3,3)$ and $(3,4) \in E(\mathbb{Z}_7)$
- $f(4) = 2 \Rightarrow (4,3)$ and $(4,4) \in E(\mathbb{Z}_7)$
- $f(5) = 5$, but 5 has no square roots $\mod 7$
- $f(6) = 3$, but 3 has no square roots $\mod 7$

$\Rightarrow |E(\mathbb{Z}_7)| = 9$ (including $\mathcal{O}$)

Elliptic Curves

Equation $y^2 = x^3 + Ax + B \pmod{p}$ provides a set of points.
We need an operator to get a group!

It can be shown:

- Any line cutting an EC twice cuts it in a 3rd point
  - a point is counted twice if line is tangent to EC
  - $\mathcal{O}$ counts too

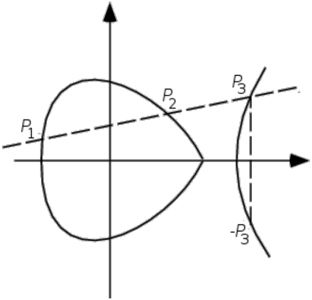
Elliptic Curves

This is used to build "+" as follows:

- $\mathcal{O}$ is the identity, that is: $P + \mathcal{O} = \mathcal{O} + P = P$
- If P_1, P_2 and P_3 are colinear, then $P_1 + P_2 + P_3 = \mathcal{O}$

- A vertical line through $P_3 := (x, y)$ also has $\mathcal{O} \Rightarrow -P_3 := (x, -y)$

$\Rightarrow$ $P_1 + P_2 = -P_3$

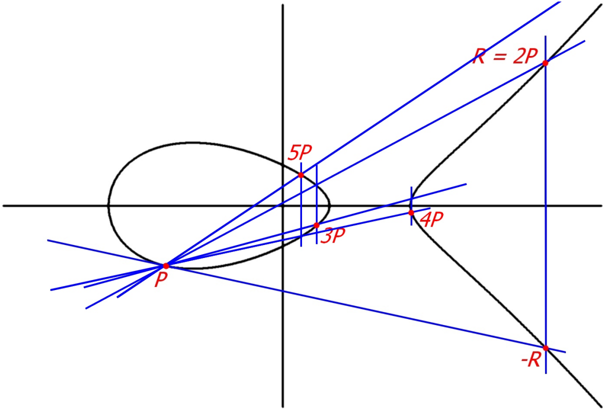


Surprisingly, this "+" is internal, associative, commutative
$\Rightarrow$ We have a commutative group!

Elliptic Curve Cryptography

Transposing DL on EC:

- ECDLP: Given points P and aP, find a



Pics by P. Bulens.

Elliptic Curve Cryptography

Schemes based on generic group operations can be used in the EC setting...

- How to choose a curve?
- What is the order of the group obtained?
- How to translate a message into a point?
- How to represent this point efficiently?
- How to compute sums, multiplications efficiently?

Many standards exist[2]

[2] See, e.g., `https://safecurves.cr.yp.to/`

Diffie-Hellman Key Agreement

Usage in TLS: [3]

	Key exchange/agreement and authentication					
Algorithm	**SSL 2.0**	**SSL 3.0**	**TLS 1.0**	**TLS 1.1**	**TLS 1.2**	**TLS 1.3**
RSA	Yes	Yes	Yes	Yes	Yes	No
DH-RSA	No	Yes	Yes	Yes	Yes	No
DHE-RSA (forward secrecy)	No	Yes	Yes	Yes	Yes	Yes
ECDH-RSA	No	No	Yes	Yes	Yes	No
ECDHE-RSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes
DH-DSS	No	Yes	Yes	Yes	Yes	No
DHE-DSS (forward secrecy)	No	Yes	Yes	Yes	Yes	No[43]
ECDH-ECDSA	No	No	Yes	Yes	Yes	No
ECDHE-ECDSA (forward secrecy)	No	No	Yes	Yes	Yes	Yes

[3] Credit: Wikipedia

Public Key Encryption

Diffie-Hellman offers a secret key, which we can then use to encrypt messages.

Can we extend it into a full encryption scheme?

- Recipient announces a *public* encryption key
  Everyone can use it to encrypt as many messages as needed
- Recipent keeps a *secret* decryption key
  He is the only one who can decrypt.

Public Key Encryption

What are we looking for, exactly?

A triple $\langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ of PPT algos:

- Gen probabilistically selects $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
  pk/sk are the public/private key
- Enc provides $c \leftarrow \mathrm{Enc}_{pk}(m)$
- Dec provides $m := \mathrm{Dec}_{sk}(c)$

s.t., $\exists$ negl. $\epsilon : \forall n$, $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$, and $\forall m$:

$$\Pr[\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m)) \neq m] < \epsilon(n)$$

Assumption: $|pk| \geq n$, $|sk| \geq n$

Public Key Encryption

Security against eavesdropper

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and PPT adversary $\mathcal{A}$, define the following experiment $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. Generate $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}(pk)$ outputs m_0, m_1 of same length
3. Choose $b \leftarrow \{0, 1\}$, and send $\mathrm{Enc}_{pk}(m_b)$ to $\mathcal{A}$
4. $\mathcal{A}$ outputs b'
5. Define $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) := 1$ iff $b = b'$

Difference with $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$

- $\mathcal{A}$ is given pk before choosing m_0, m_1

Security of Encryption

$\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ has *indistinguishable encryptions in the presence of eavesdroppers* if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ :

$$\Pr[\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] = \frac{1}{2} + \epsilon(n)$$

Security of Encryption

Chosen-plaintext security

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and adversary $\mathcal{A}$, define the following experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$:

1. Select $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}(pk)$ is given oracle access to $\mathrm{Enc}_{pk}(\cdot)$
3. $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}$
4. Choose $b \leftarrow \{0, 1\}$ and send $\mathrm{Enc}_{pk}(m_b)$ to $\mathcal{A}$
5. $\mathcal{A}$ is again given oracle access to $\mathrm{Enc}_{pk}(\cdot)$
6. $\mathcal{A}$ outputs b'
7. Define $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) := 1$ iff $b = b'$

Security against CPA

$\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ has *indistinguishable encryption under a chosen-plaintext attack* if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ :

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

Quiz 1

What is the relationship between $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ and $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$?

1. It is **harder** to win the $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ game than to win the $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ game.
2. It is **easier** to win the $\mathsf{PubK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$ game than to win the $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ game.
3. These two games are just as easy/difficult to win

Answer: They are **equivalent**!

Quiz 2

Could we build a perfectly secure public key encryption scheme?

1. Maybe, this is an open research problem
2. No: since the private key has a finite length, an adversary can take a challenge ciphertext and try to decrypt it with all possible private keys until he gets a meaningful plaintext
3. No: since the adversary has the public key, he can take a challenge ciphertext c and encrypt the two challenge messages with all possible random coins until it gets c
4. No: since the adversary has the public key, he can encrypt messages of his choice and search for the correct decryption key

Secure multiple encryption

Define the multiple message eavesdropping experiment $\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n)$:

1. Select $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}(pk)$ outputs $M_0 = (m_0^1, \ldots, m_0^t), M_1 = (m_1^1, \ldots, m_1^t)$
3. Choose $b \leftarrow \{0, 1\}$, send $(\mathrm{Enc}_{pk}(m_b^1), \ldots, \mathrm{Enc}_k(m_b^t))$ to $\mathcal{A}$
4. $\mathcal{A}$ outputs b'
5. Define $\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) := 1$ iff $b = b'$

Secure multiple encryption

$\Pi := \langle \text{Gen}, \text{Enc}, \text{Dec} \rangle$ has *indistinguishable multiple encryption in the presence of eavesdroppers* if

$\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ :

$$\Pr[\mathsf{PubK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

This property is equivalent to the first two. . .

$\rightarrow$ K&L, 1st edition, theorem 10.10, p.344

$\rightarrow$ K&L, 2nd edition, theorem 11.10, p.381

El Gamal Encryption

A CPA-secure encryption scheme: *ElGamal* (1985)

- Idea: use session key to encrypt $m \in \mathbb{G}$:

$$\begin{tikzcd} A \arrow[rr, "{\mathbb{G}, q, g, g^x}"] \arrow[d, Rightarrow] & & B \arrow[d, Rightarrow] \\ \bullet & & \bullet \arrow[ll, "{g^y, m \cdot g^{xy}}"'] \end{tikzcd}$$

ElGamal

ElGamal Encryption

A CPA-secure encryption scheme: *ElGamal* (1985)

- $\mathrm{Gen}(1^n)$
  - runs $\mathcal{G}$ to obtain $(\mathbb{G}, q, g)$
  - selects $x \leftarrow \mathbb{Z}_q$, computes $h := g^x$
  - $\langle pk, sk\rangle := \langle(\mathbb{G}, q, g, h), (\mathbb{G}, q, g, x)\rangle$
- Message $m \in \mathbb{G}$ is encrypted as
  - $y \leftarrow \mathbb{Z}_q$
  - $\mathrm{Enc}_{pk}(m) := \langle g^y, m \cdot h^y\rangle$
- $\mathrm{Dec}_{sk}(\langle c_1, c_2\rangle) := \frac{c_2}{c_1^x}$

Observe:

- $\frac{c_2}{c_1^x} = \frac{m \cdot h^y}{(g^y)^x} = \frac{m \cdot (g^x)^y}{g^{x \cdot y}} = m$

ElGamal Security

Can we prove that ElGamal is CPA-secure based on the hardness of the DL problem w.r.t. $\mathcal{G}$?

1. Yes: if I can solve the DL problem, I can compute the secret key from the public key and decrypt
2. No: if I can compute the least significant bit of h^y given (g, h, g^y), then it is enough to break ElGamal, but not necessarily to solve DL

ElGamal Security

We need more than the DL assumption!

- It may be possible to compute h^y without solving the DL (We do not know how, but we cannot exclude it)
- We need h^y to be *completely* unpredictable, else parts of m may be leaked

This is actually true even for DH key agreement: the key g^{xy} must be "as" a random key, not just a hard to compute value

Computational Diffie-Hellman

Consider experiment $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$ where g generates the group $\mathbb{G}$ of order q, with $|q| = n$
2. Choose $(x, y) \xleftarrow{R} \mathbb{Z}_q^2$
3. Set $h \leftarrow \mathcal{A}(\mathbb{G}, q, g, g^x, g^y)$
4. $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1 \Leftrightarrow h = g^{x \cdot y}$

Computational Diffie-Hellman

The *computational Diffie-Hellman problem* is hard relative to $\mathcal{G}$ if $\forall$ PPT $\mathcal{A}$, there is a negl. ϵ s.t.:

$$\Pr[\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \epsilon(n)$$

Observe:

- If $\mathsf{DLog}_{\mathcal{A},\mathcal{G}}$ is easy then $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}$ is easy
- The converse is not necessarily true!

Decisional Diffie-Hellman

Consider experiment $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain $(\mathbb{G}, q, g)$ where g generates the group $\mathbb{G}$ of order q, with $|q| = n$
2. Choose $(x, y, z) \xleftarrow{R} \mathbb{Z}_q^3$
3. Flip a coin $b \xleftarrow{R} \{0, 1\}$, and set $h_1 := g^{x \cdot y}$ and $h_0 := g^z$
4. Set $b' \leftarrow \mathcal{A}\big(\mathbb{G}, q, g, (g^x, g^y, h_b)\big)$
5. $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n) = 1 \Leftrightarrow b = b'$

Observe:

- No need to *compute* g^{xy}: just *decide* whether you see g^{xy} or g^z

Decisional Diffie-Hellman

The *decisional Diffie-Hellman problem* is hard relative to $\mathcal{G}$ if $\forall$ PPT $\mathcal{A}$, there is a negl. $\epsilon(n)$ s.t.:

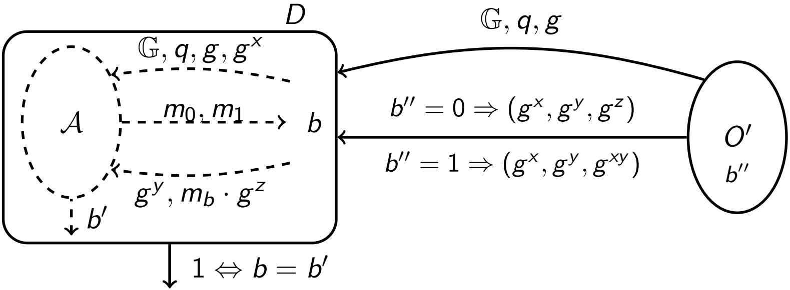
$$\Pr[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \frac{1}{2} + \epsilon(n),$$

Observe:

- If $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}$ is easy then $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}$ is easy
- The converse is believed to be false

El Gamal Security

Indistinguishability of encryptions ($\Leftrightarrow$ CPA security)



Observe:

- If $b'' = 0$, $\Pr[D \text{ outputs } 1] = \frac{1}{2}$ (as $m_b \cdot g^z$ is random in $\mathbb{G}$)
- If $b'' = 1$, $\Pr[D \text{ outputs } 1] = \frac{1}{2} + \eta(n)$ (normal game)
- D distinguishes with same Pr as $\mathcal{A}$ $\Rightarrow$ safe under DDH

DHIES – ISO/IEC 18033-2

ElGamal can only encrypt short messages: one group element
How to proceed for long messages?

1. Make many ElGamal ciphertexts? Expensive!
2. Derive a symmetric key from g^{xy}, and use it to encrypt the long message!

DHIES/ECIES ($\approx$ ISE/IEC 18033-2):

- Gen as in ElGamal, gives $\langle pk, sk\rangle := \langle(\mathbb{G}, q, g, h), x)\rangle$
- $\mathrm{ENC}_{pk}(m)$: pick $y \leftarrow \mathbb{Z}_q$, derive $k_e = H(h^y)$, return $(c_1, c_2) = \langle g^y, \mathrm{Enc}_{k_e}(m)\rangle$ with Enc part of an AE scheme
- $\mathrm{DEC}_{sk}(c_1, c_2)$: compute $k_e = H(c_1^x)$
  return: $\mathrm{Dec}_{k_e}(c_2)\rangle$

Proven to be CCA secure under reasonable assumptions