Lecture 5: Public-Key Algorithms

Helger Lipmaa
Helsinki University of Technology
helger@tcs.hut.fi

Recap: what we have done

- First lecture: general overview
- Second lecture: secret-key cryptography
- Third lecture: Modes of operation
- Fourth lecture: Hash functions

  Lectures 2–4 are all about secret-key cryptography!
- Today: Public-key algorithms

Problems of symmetric model (1/3)

- Alice and Bob need to share a key
  - ⋆ distributed over a private channel
  - ⋆ say, when they meet in a pub
- Private channels are very expensive
  - ⋆ especially in Finland

Problems of symmetric model (2/3)

Huge number of keys when scaling:

- n parties need to communicate secretly with everybody else
- Every pair needs a secret key, there are $\binom{n}{2} = \frac{n^2-n}{2}$ pairs
- Thus, $\frac{n^2-n}{2}$ keys must be pre-distributed!
- Every participant needs to store n different keys
- Say, $n = 6 \cdot 10^9 \ldots$

Problems of symmetric model (2/3)

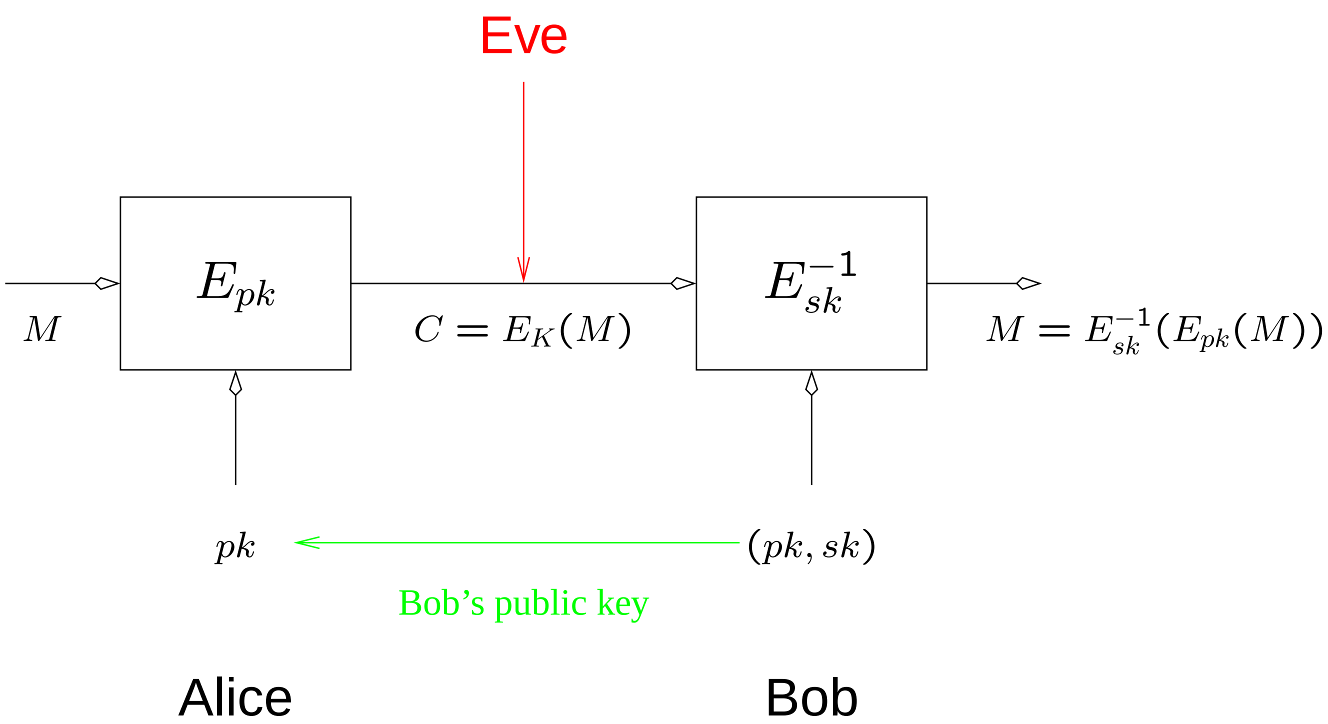
Non-repudiation:

- You can authenticate yourself and your messages to your friends by using MAC=s
- However, MAC-s use shared key
- Therefore, you cannot prove to third parties that messages were really sent by your friend and not by yourself!

Public key cryptography: mysterious helper

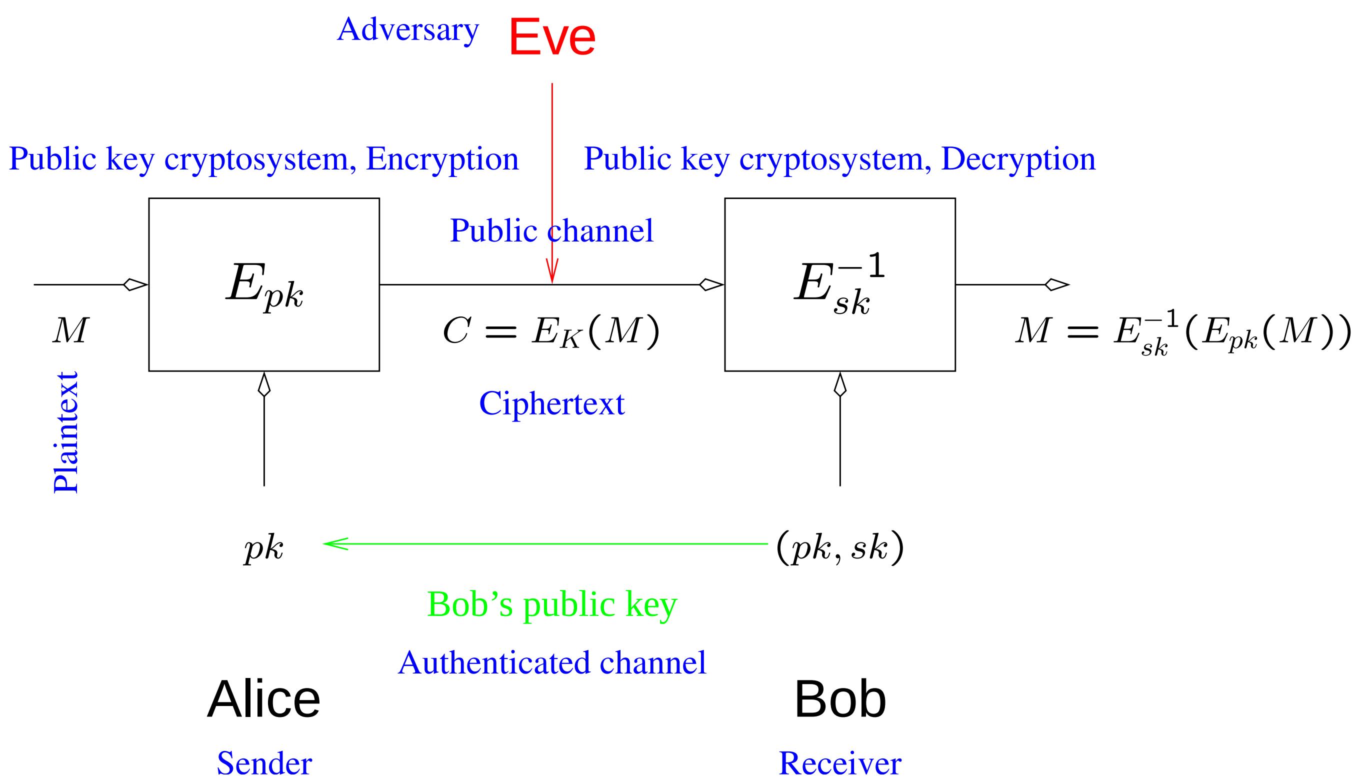
- All mentioned problems can be solved by using PKC
- Basic idea: everybody has a pair (pk, sk) of public and secret keys
- If you want to send to me a message, you
  - ⋆ a) fetch my pk from a directory, b) encrypt a message by pk and c) send the result to me
- I will decrypt the ciphertext by using my secret key

PKC: model



Eve
E_{pk}
M
$C = E_K(M)$
E_{sk}^{-1}
$M = E_{sk}^{-1}(E_{pk}(M))$
pk
(pk, sk)
Bob's public key
Alice
Bob

PKC: model



Alice obtains public key from an *authenticated* channel, no privacy during this is necessary!

Public-Key Cryptography: Assumptions

- PKC bases on clear mathematics
  - ⋆ Existence of one-way functions, and related primitives
- “Crazy” solutions (AES-like or DES-like) are not accepted
- IMPORTANT: PKC bases on the assumption that there is *one* OWF

  Caveat: Real assumptions are slightly more complicated
- If this OWF gets “broken”, it can be substituted with another one — assuming that *OWFs exist*

Etude: Elementary mathematics (1/2)

- For any integer n, $\mathbb{Z}_n = \{0, \ldots, n-1\}$
- $\mathbb{Z}_n$ is an additive group: $a + b = c \mod n$. E.g., $7 + 12 = 19 \equiv 6 \mod 13$, thus $7 + 12 = 6$ in $\mathbb{Z}_{13}$
- Analogously, modular multiplication: $7 \cdot 12 = 84 \equiv 6 \mod 13$
- $\mathbb{Z}_n$ is not a multiplicative group:
  - ⋆ not all elements of $\mathbb{Z}_n$ have inverses

(Known from the discrete mathematics course)

Etude: Elementary mathematics (2/2)

- y is inverse of x modulo n iff $xy = 1 \mod n$
- Elementary: x has an inverse iff $\gcd(x, n) = 1$
- E.g., $4^{-1} \equiv 10 \mod 13$ since $4 \cdot 10 = 40 \equiv 1 \mod 13$, but 4 does not have an inverse modulo 12, since $\gcd(4, 12) = 4 \neq 1$
- For any integer n,

$$\begin{aligned} \mathbb{Z}_n^* &= \{x \in \mathbb{Z}_n : x \text{ has an inverse modulo } n\} \\ &= \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\} \end{aligned}$$

- Euler's totient function $\varphi(n) := \sharp\mathbb{Z}_n^* = \sharp\{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$

RSA Cryptosystem

- The first PKC (Rivest, Adleman, Shamir, 1977)
- Still the most used public-key cryptosystem but
  - Slow key generation
  - Sub-exponential attacks known, thus long keys
  - Not readily generalizable to other algebraic structures
  - No “semantic security”

RSA Key Generation

- Generate two random large primes p, q
- Set $n = pq$
- Choose an e, s.t. $\gcd(e, \varphi(n)) = 1$
- Compute $d := e^{-1} \mod \varphi(n)$
- (n, e) is the public key, (p, q, d) is the secret key.

RSA Encryption and Decryption

- To encrypt an $x \in \mathbb{Z}_n^*$, compute $y = x^e \mod n$
- To decrypt $y \in \mathbb{Z}_n^*$, compute $y^d \mod n$
- Clearly, $x^{ed \mod \varphi(n)} \equiv x \mod n$
  - ⋆ Since $\sharp\mathbb{Z}_n^* = \varphi(n)$ then $x^{\varphi(n)} = x$.

RSA Efficiency: Key generation and decryption

- Key generation:
  - ⋆ Generating primes p and q can be done efficiently by using randomized algorithms (Rabin-Williams, . . .)

- Decryption:
  - ⋆ In average $k/2$ multiplications modulo n when a k-bit modulus is used
  - ⋆ Can be sped up by using the Chinese Remainder Theorem

RSA efficiency: Encryption

- Usually, $e = 3$ or $e = 2^{16} + 1$ is used
  - ⋆ This speeds up exponentiation:
    $$x^3 \equiv x^2 \cdot x \mod n$$
    can be computed in two multiplications,
    $$x^{2^{16}+1} = (((x^2)^2)^{\cdots 2})^2 \cdot x \mod n$$
    in 17 multiplications. Thus, encryption is fast

See algorithms from the textbook

RSA: Basic Security

- If n can be factorized then one can recompute $\varphi(n) = (p-1)(q-1)$, and hence also $d = e^{-1} \mod \varphi(n)$
  - ⋆ Factoring is easy $\Rightarrow$ RSA is broken
- Best factorization algorithms: quadratic field sieve, generalized number field sieve, elliptic curve factorization method
- Modulus must be at least 1024-bit long to resist factoring
- It is *not* known whether breaking RSA is equivalent to factoring, it is believed that it is actually easier

RSA: Security Requirements

- RSA security (in the sense of message recovery) bases on the difficulty of computing roots (*the RSA problem*):

  ⋆ Given (x, e) and modulus n, it is difficult to compute $x^{e^{-1}} \mod n$

- *Semantic security*:

  ⋆ Attacker chooses m_1 and m_2, and handles both of them to the black box. The black box picks a random $b \leftarrow \{1, 2\}$ and encrypts the corresponding m_b. Attacker sees the ciphertext $y = E_K(m_b)$. He must guess the value of b

- Example: you know that Napoleon is either encrypting "Attack" or "Wait". Clearly the cryptosystem must be semantically secure!

RSA and Semantic Security (1/2)

- RSA is not semantically secure, since it is deterministic:
  - ⋆ You can encrypt both "Attack" and "Wait" yourself, and compare the outcomes with the received ciphertext

- Various methods exist for making RSA semantically secure
  - ⋆ Many ad hoc methods have been broken (including PKCS as described in the textbook)

RSA and Semantic Security (2/2)

- RSA together with OAEP (Optimal Asymmetric Encryption Padding)
- Proposed and proved to be secure by Bellare and Rogaway, 1994
- A flaw in proof found by Shoup in 2001
- Proof corrected by others in 2001
- Result: OAEP is *provably* semantically secure, but the resulting scheme is quite complex
- (Even the proof that it is secure is complex!)

Alternative: Discrete logarithm problem

- Take *any* “good” group G
  - $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$
  - Elliptic curves
  - Class groups, ...
- In such groups:
  - Exponentiation g^x is easy
  - Given (g, g^x), it is (conjectured to be) difficult to find x
  - This is the *discrete logarithm problem*: $(g, g^x) \to x$

Elliptic curve

Fix a field $\mathbb{F}$ of characteristic $c \neq 2, 3$ (for those cases, formulas are slightly different). Elliptic curve is a nonsingular cubic curve,

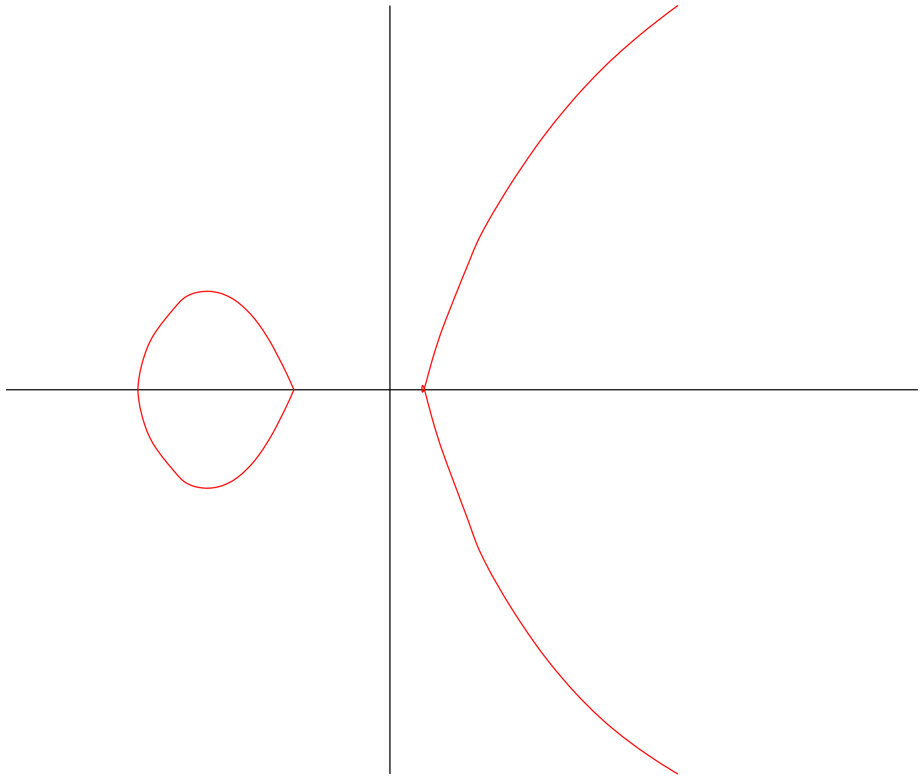
$$C : y^2 = x^3 + ax + b$$

over $\mathbb{F}$.

Nonsingular: $x^3 + ax + b$ has no repeated factors

Elliptic curve points: all pairs $(x, y) \in \mathbb{F}^2$ that belong to C together with a special point $\mathcal{O}$ at the infinity.

Elliptic curve: illustration

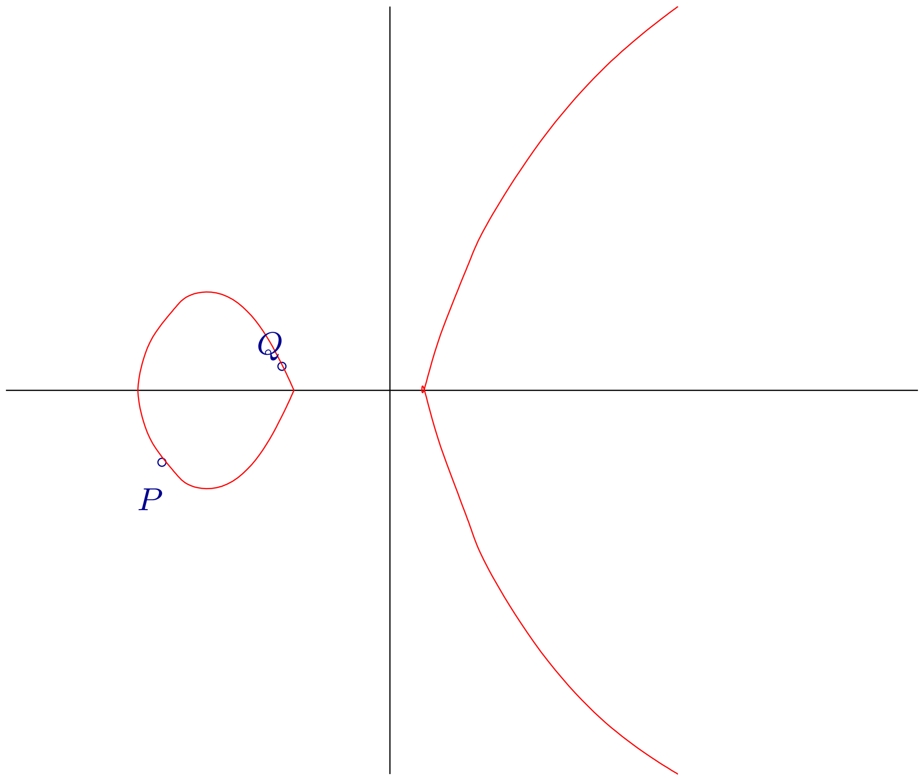


Here, $\mathbb{F} = \mathbb{R}$!

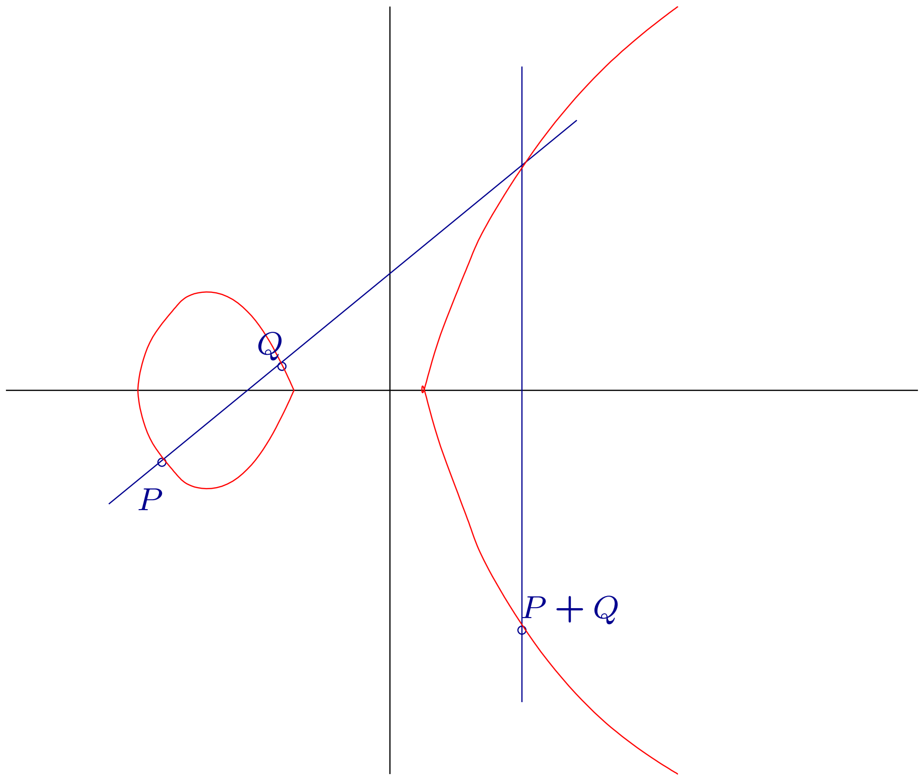
Elliptic curve group

- Take $E(C)$ be the set of all EC points
- For two points P, Q on the curve, define $P + Q$ as follows:
- $\ldots$ Draw a line that crosses P and Q
- $\ldots$ Find the third intersection point of this line and the curve
- Mirror this point w.r.t. y-axis

Elliptic curve group: illustration



Elliptic curve group: illustration



EC addition: formulas

Curve: $y^2 = x^3 + ax + b$, $\mathbb{F} = \mathbb{R}$. Define group $E_{\mathbb{F}}(C)$ as follows.

Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$. If $Q = (x_1, -y_1)$, define $P + Q = \mathcal{O}$. Otherwise, define the slope of line connecting P and Q: $\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q, \\ \frac{3x_1^2 + a}{2y_1}, & P = Q. \end{cases}$

Then $P + Q = (x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1)$.

Special cases when one of the two addends is $\mathcal{O}$: $P + \mathcal{O} = \mathcal{O} + P = P$.

EC group

Theorem Let $\mathbb{F}$ be an *arbitrary* field of characteristic $c \neq 2, 3$. Let $C : y^2 = x^3 + ax + b$. Then $(E_{\mathbb{F}}(C), +)$ is a group w.r.t. addition defined in previous slide.

Unit element: $\mathcal{O}$

Inverse: $-\mathcal{O} = \mathcal{O}$, $-(x, y) = (x, -y)$

Commutativity: easy

Associativity: harder to prove

Discrete logarithm problem in EC group

- Fix the field $\mathbb{F} = \mathrm{GF}(q)$, usually $q = 2^p$ or $q = p$ for a prime p, and $q \geq 2^{160}$

- *DL problem in EC group*: Given $g \in E_{\mathbb{F}}(C)$ of large order, and a random $x \in \mathbb{Z}_{\mathrm{ord}\, g}$, compute x from (g, xg)

  ⋆ Note: here we use the additive notation. (xg is exponentiation!)

- Believed to be hard: the best *known* algorithm to solve the discrete logarithm problem on a random curve takes $\approx \sqrt{q}$ steps

Algorithms for discrete logarithm problem

Generic algorithms (work for all groups, do not use the structure of group):

- Exhaustive search
- Shanks's baby-step giant-step
- Pollard's rho algorithm
- Pohlig-Hellman algorithm

Algorithms for discrete logarithm problem

Tailored algorithm (for specific groups):

- Index calculus for DL problem in $\mathbb{Z}_p^*$
- DL in $(\mathbb{Z}_p, +)$ can be solved trivially!
  - ⋆ Given $g, xg \in \mathbb{Z}_p$: $x = (xg)/g \mod p$
- No tailored algorithms are known for *randomly chosen* elliptic curves!

DLP: Exhaustive search

Given (g, h), $h = g^x$ for unknown x:

- Successively compute g^0, g^1, g^2, $\ldots$, until h is obtained
- Requires 1 multiplication per step, hence x multiplications in total
- Asymptotically: $O(\mathrm{ord}\, g)$ multiplications, $\mathrm{ord}\, g$ is the order of g

For function f, $g = O(f)$ if for some constant c, $g(x) \le cf(x)$ for all x

Recommendations for a good group

For the best algorithm for DL to take $\geq 2^k$ steps:

- To dwarf the rho algorithm, choose $n \geq 2k$
- To dwarf the Pohlig-Hellman algorithm, make sure that the greatest divisor p of $\operatorname{ord} g$ is big, $p \geq 2k$. Usually, g is chosen to generate a subgroup of prime order
- Choose a group without any tailored algorithms for DL

A randomly chosen EC group over $\mathsf{GF}(q)$, $q = 2^p$ or $q = p$, with $q \geq 2^{160}$ seems to be secure

Diffie-Hellman key exchange

Assume we have a fixed group G and an $g \in G$ with large order

Alice **Bob**

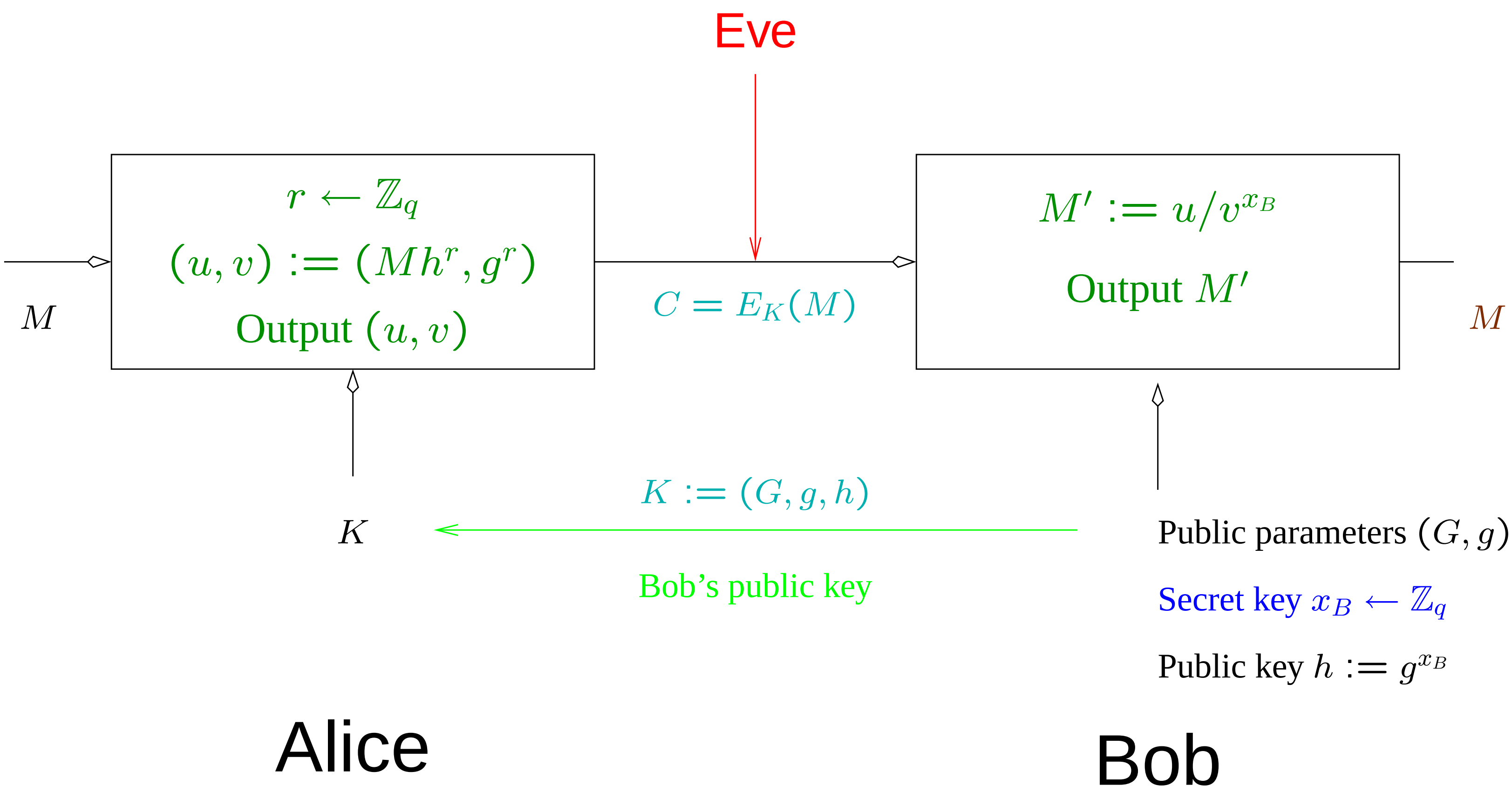
Alice		Bob
Private input: x_A (secret key)		Private input: x_B (secret key)
Compute $y_A := g^{x_A}$	$\xrightarrow{y_A}$	
	$\xleftarrow{y_B}$	Compute $y_B := g^{x_B}$
Output: $y_B^{x_A} = g^{x_A x_B}$		Output: $y_A^{x_B} = g^{x_A x_B}$

Alternatively, y_A is Alice's public key, y_B is Bob's public key, and both can be fetched from a directory

Security of the DH key exchange

- *Diffie-Hellman (DH) problem*:
  - ⋆ Given (g, g^{x_A}, g^{x_B}), compute $g^{x_A x_B}$.
- If DL problem is tractable, then so is the DH problem:
  - ⋆ Compute x_A from (g, g^{x_A}) and then compute $g^{x_A x_B}$ from (g, x_A, g^{x_B})
- It is *not* known, if the opposite reduction holds, but the best known algorithms for the DH problem need solving the DL problem

ElGamal cryptosystem



Basic Security of the ElGamal cryptosystem

- Message recovery from (mh^r, g^r) and public key $h = g^x$ can be done if DH is tractable
  - ⋆ Compute $h^r = g^{xr}$ from g^r and $h = g^x$.
- Is the opposite true?
  - ⋆ I.e., can one solve DH, if it is feasible to recover m from (mh^r, g^r) and $h = g^x$?
  - ⋆ Yes, since then one can also recover $h^r = g^{rx}$.
- Thus: one can use any group where the DH problem is hard

Semantic Security, Again

- *Semantic security*: given m_0 and m_1, distinguish random encryptions of m_0 from m_1
  - ⋆ E.g., was the plaintext “yes” or “now”?
- Equivalent (informal) definition: given an encryption of unknown plaintext m, decide where $P(m)$ is true for some predicate P
  - ⋆ E.g., decide whether plaintext contains the word “attack”

Semantic Security of ElGamal

- **Theorem (Jakobsson, Tsiounis, Yung, 1998).** ElGamal is semantically secure if the following *Decisional Diffie-Hellman* (DDH) problem is hard: Given (g, g^x, g^y, h), decide whether $h = g^{xy}$ or $h = g^z$ for random z.

- ElGamal is not secure against the chosen ciphertext attack. Why? (Try to solve)

  ⋆ (Hint: use the *homomorphic* property $E_K(m_1 + m_2) = E_K(m_1)E_K(m_2)$.)

  ⋆ (Why does this property hold?)

PKC: brief overview

- *ECC*: ElGamal over EC. Short keys (≥ 160 bits), fast key generation. Semantically secure. Can be made secure against the CCA. Security bases on the DDH assumption in elliptic curves

- *RSA*. Long keys (≥ 1024 bits), slow key generation, fast encryption. Can be made semantically secure by using the OAEP. Security bases on the RSA assumption

- Other systems: *NTRU* (long keys, ≥ 1700 bits, $100 \ldots 300$ times faster than RSA, less known and studied), *XTR* (a variant of ElGamal in $\mathrm{GF}(p^6)$), key ≥ 340 bits, approximately as fast as ECC, security bases on the DDH assumption in $\mathbb{Z}_p^*$), ...

Next time

- Lecture given by Markku-Juhani Saarinen
- Public-key cryptanalysis
- Algorithms for factoring
- Algorithms for discrete logarithm
- Etc