Cryptologie asymétrique 1/2

Damien Vergnaud

Sorbonne Université

CRYPTO 1

Contents

Limitations of Secret Key (Symmetric) Cryptography

- Secret key cryptography
  - symmetric encryption $\rightsquigarrow$ confidentiality
  - MAC $\rightsquigarrow$ authentication (integrity)
- Sender and receiver must share the same key
  - **needs** secure channel for key distribution
- Other limitation of authentication scheme
  - cannot authenticate to **multiple** receivers
  - does not have non-repudiation

Limitations of Secret Key (Symmetric) Cryptography

- Secret key cryptography
  - symmetric encryption $\rightsquigarrow$ confidentiality
  - MAC $\rightsquigarrow$ authentication (integrity)
- Sender and receiver must share the same key
  - **needs** secure channel for key distribution
- Other limitation of authentication scheme
  - cannot authenticate to **multiple** receivers
  - does not have non-repudiation

Limitations of Secret Key (Symmetric) Cryptography

- Secret key cryptography
  - symmetric encryption $\rightsquigarrow$ confidentiality
  - MAC $\rightsquigarrow$ authentication (integrity)
- Sender and receiver must share the same key
  - **needs** secure channel for key distribution
- Other limitation of authentication scheme
  - cannot authenticate to **multiple** receivers
  - does not have non-repudiation

How to distribute the cryptographic keys?

- If the users can meet in person beforehand - it's simple.
- But what to do if they cannot meet? (e.g. on-line shopping)

A Naive solution

- give to every user P_i a separate key K_{ij} to communicate with every P_j
- $\rightsquigarrow$ quadratic number of keys is needed
- $\rightsquigarrow$ someone needs to "give the keys"
- $\rightsquigarrow$ the users need to store large numbers of keys in a secure way

How to distribute the cryptographic keys?

- If the users can meet in person beforehand - it's simple.
- But what to do if they cannot meet? (e.g. on-line shopping)

A Naive solution

- give to every user P_i a separate key K_{ij} to communicate with every P_j
- $\rightsquigarrow$ quadratic number of keys is needed
- $\rightsquigarrow$ someone needs to "give the keys"
- $\rightsquigarrow$ the users need to store large numbers of keys in a secure way

The solution: Public-Key Cryptography

- first proposed by Diffie and Hellman:

W.Diffie and M.E.Hellman,
New directions in cryptography
IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp. 644-654.

- similar idea by Merkle:
  - **1974**: a project proposal for a Computer Security course at UC Berkeley
    (it was rejected)
  - **1975**: submitted to the CACM journal (it was rejected)
    (see `http://www.merkle.com/1974/`)

- 2015 Turing Award

- It 1997 the GCHQ revealed that they new it already in 1970 (James Ellis).

The idea

- instead of using one key K: use 2 keys (e, d)
  - $e \rightsquigarrow$ encryption,
  - $d \rightsquigarrow$ decryption,
- e can be public and only d has to be kept secret!

- **Public Key Encryption**
  - Message + Bob's Public Key = Ciphertext
  - Ciphertext + Bob's Private Key = Message
- anyone with Bob's public key can send Bob a secret message.
- only Bob can decrypt the message, since only Bob has the private key.

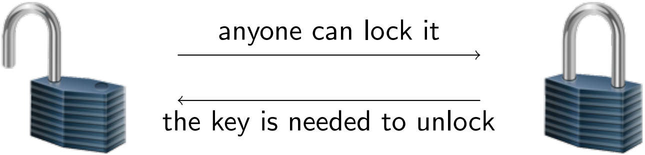
The idea

- instead of using one key K: use 2 keys (e, d)
  - $e \rightsquigarrow$ encryption,
  - $d \rightsquigarrow$ decryption,
- e can be public and only d has to be kept secret!

- **Digital signatures**
  - Message + Alice's Private Key = Signature
  - Message + Signature + Alice's Public Key = 0 or 1
- anyone with Alice's public key can verify that the message comes from Alice.
- only Alice can produce the signature, since only Alice has the private key.

But is it possible?

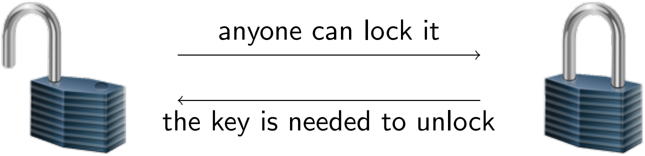
- In “physical world”: yes!
  ⇝ Example: padlock



- Diffie and Hellman proposed the public key cryptography in 1976.
  - They just proposed the concept, not the implementation.
  - But they have shown a protocol for key-exchange

But is it possible?

- In "physical world": yes!
  $\rightsquigarrow$ Example: padlock

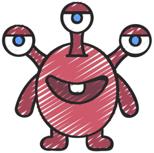


- Diffie and Hellman proposed the public key cryptography in 1976.
  - They just proposed the concept, not the implementation.
  - But they have shown a protocol for key-exchange

Key Exchange

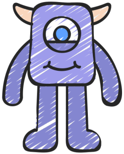
Diffie-Hellman Key Exchange

$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



$\xrightarrow{y_a = g^a}$

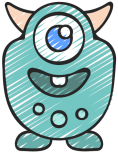
$\xleftarrow[y_b = g^b]{}$



Alice

Bob

$K_a = y_b^a$



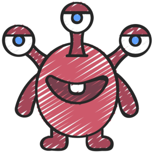
$K_b = y_a^b$

Eve

$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange

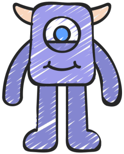
$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



Alice

$$y_a = g^a \longrightarrow$$

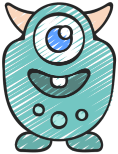
$$\longleftarrow y_b = g^b$$



Bob

$\downarrow$ $K_a = y_b^a$

$\downarrow$ $K_b = y_a^b$

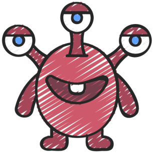


Eve

$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

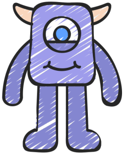
Diffie-Hellman Key Exchange

$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



$$y_a = g^a \longrightarrow$$

$$\longleftarrow y_b = g^b$$

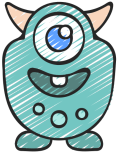


Alice

Bob

$$\downarrow$$

$$K_a = y_b^a$$



Eve

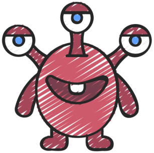
$$\downarrow$$

$$K_b = y_a^b$$

$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange

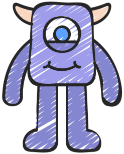
$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



Alice

$$\xrightarrow{\;y_a = g^a\;}$$

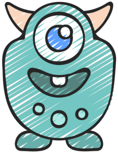
$$\xleftarrow[\;y_b = g^b\;]{}$$



Bob

Alice $\downarrow$ $K_a = y_b^a$

Bob $\downarrow$ $K_b = y_a^b$

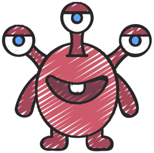


Eve

$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange

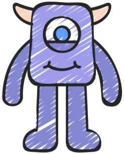
$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



Alice

$$\xrightarrow{y_a = g^a}$$

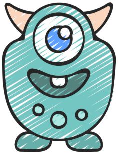
$$\xleftarrow[y_b = g^b]{}$$



Bob

$$\downarrow$$

$$K_a = y_b^a$$



Eve

$$\downarrow$$

$$K_b = y_a^b$$

$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange: Security

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have no information on $K = g^{ab}$**.**

- If finding a from y_a is easy then the DH key exchange is not secure.
- Even if it is hard, then

  …the scheme may also not be completely secure

- How to choose $\mathbb{G}$?

  …First choice (**bad**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}, +)$ for some integer n.

  …Second choice (**good**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}^*, \cdot)$ for some integer n.

Diffie-Hellman Key Exchange: Security

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have no information on $K = g^{ab}$.

- If finding a from y_a is easy then the DH key exchange is not secure.
- Even if it is hard, then

  ...the scheme may also not be completely secure

- How to choose $\mathbb{G}$?

  ...First choice (**bad**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}, +)$ for some integer n.

  ...Second choice (**good**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}^*, \cdot)$ for some integer n.

Diffie-Hellman Key Exchange: Security

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have no information on $K = g^{ab}$.

- If finding a from y_a is easy then the DH key exchange is not secure.
- Even if it is hard, then

  …the scheme may also not be completely secure

- How to choose $\mathbb{G}$?

  …First choice (**bad**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}, +)$ for some integer n.

  …Second choice (**good**): $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z}^*, \cdot)$ for some integer n.

The Fundamental Equation

$$Z = Y^X \mod N$$

When Z is unknown, it can be efficiently computed

Exponentiation by squaring – *square-and-multiply*

$$y^x = \begin{cases} 1 & \text{if } x = 0 \\ y \cdot y^{x-1} & \text{if } x \text{odd} \\ (y^2)^{x/2} & \text{if } x \text{even} \end{cases}$$

```
long pow(long y, long x)
{
    long result = 1;
    while ( x ) {
        if ( x & 1 ) {
            result *= y;
        }
        y *= y;
        x /= 2;
    }
    return result;
}
```

Efficiency of computation modulo n

Suppose that n is a k-bit number, and $0 \leq x, y \leq n$

- $(x \pm y) \mod n \rightsquigarrow O(k)$
- $(xy) \mod n \rightsquigarrow O(k^2)$ (ou $\tilde{O}(k)$)
- $(x)^c \mod n \rightsquigarrow O((logc)k^2)$ ou $\tilde{O}((\log c)k)$
- $(x^{-1}) \mod n \rightsquigarrow O(k^3)$ (ou $\tilde{O}(k^2)$) ou $O(k^2)$ (ou $\tilde{O}(k)$)

The Fundamental Equation

$$Z = Y^{X} \mod N$$

When X *is unknown, the problem is known as the* discrete logarithm *and is generally believed to be hard to solve*

We will see that later. . .

The Fundamental Equation

$$Z = Y^{X} \mod N$$

When X *is unknown, the problem is known as the* discrete logarithm *and is generally believed to be hard to solve*

We will see that later. . .

The Fundamental Equation

$$Z = Y^X \mod N$$

When Y is unknown, the problem is known as the discrete root extraction *and is generally believed to be hard to solve unless the factorisation of N is known.*

We will see that later...

The Fundamental Equation

$$Z = {\color{red}Y}^X \mod N$$

When ${\color{red}Y}$ *is unknown, the problem is known as the* discrete root extraction *and is generally believed to be hard to solve unless the factorisation of* N *is known.*

We will see that later...

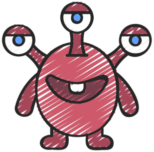
The Fundamental Equation

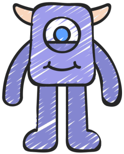
$$Z = Y^X \mod N$$

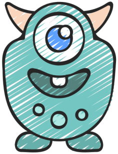
When Y is unknown, the problem is known as the discrete root extraction *and is generally believed to be hard to solve unless the factorisation of N is known.*

We will see that later...

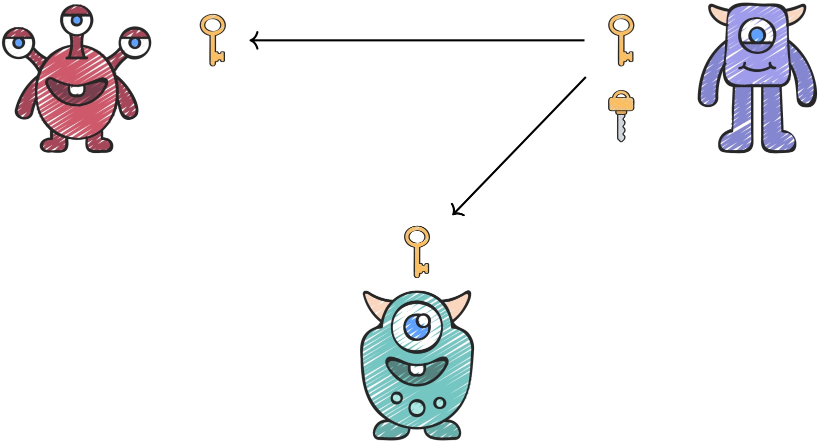
Public Key Encryption



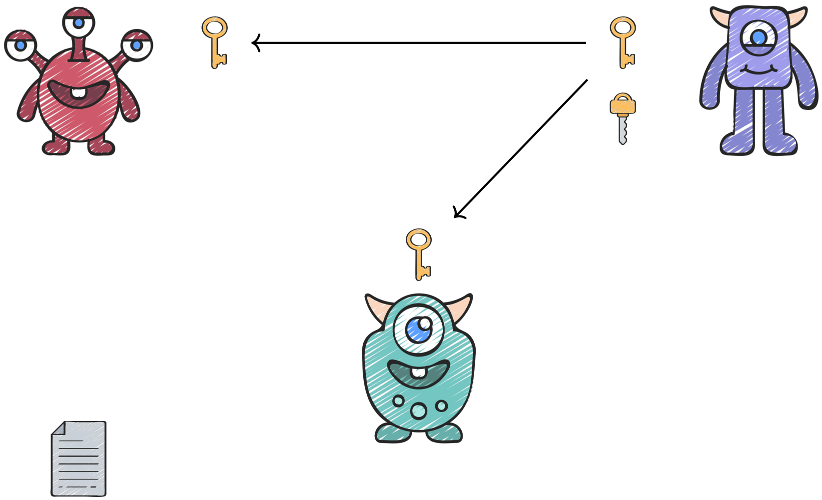




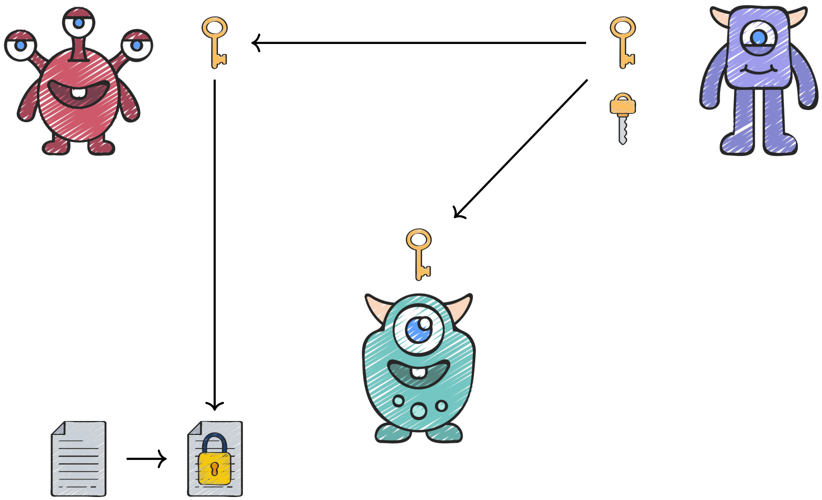
Public Key Encryption



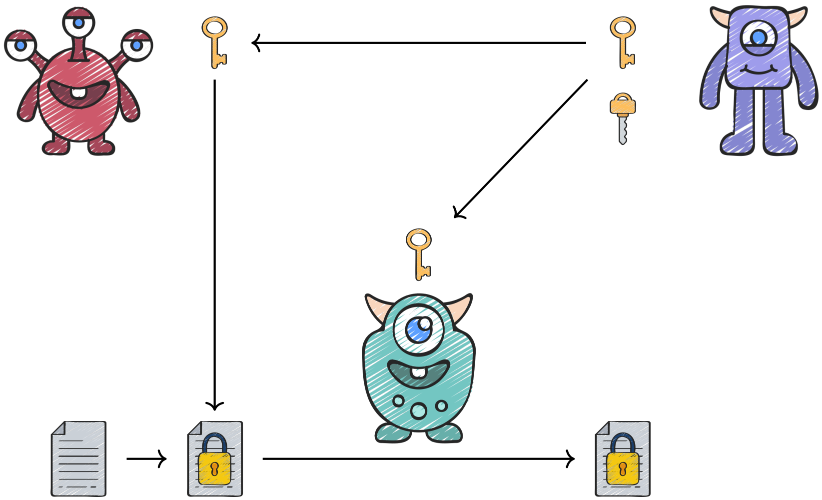
Public Key Encryption



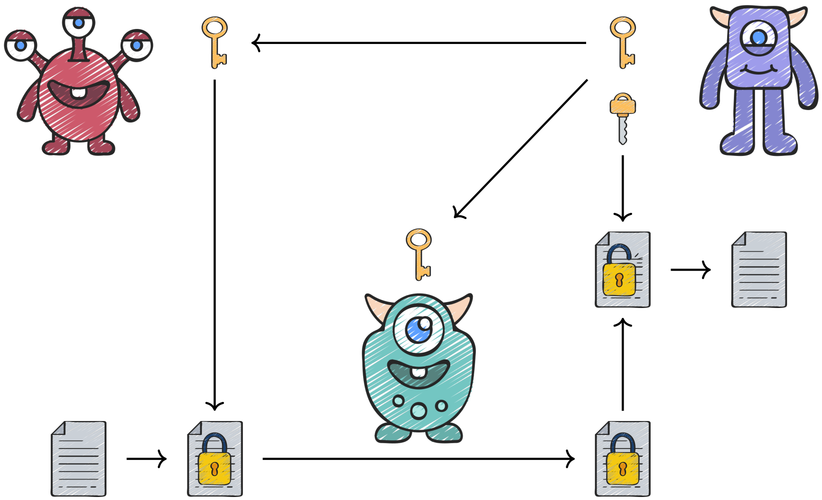
Public Key Encryption



Public Key Encryption



Public Key Encryption



Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic **encryption algorithm** which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic **decryption algorithm** which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic **encryption algorithm** which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic **decryption algorithm** which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic **encryption algorithm** which takes on input a public key pk and a plaintext $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a ciphertext c,
- $\mathcal{D}$ is a deterministic **decryption algorithm** which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m, u)) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Public-Key Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But ...
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext
etc	*etc*

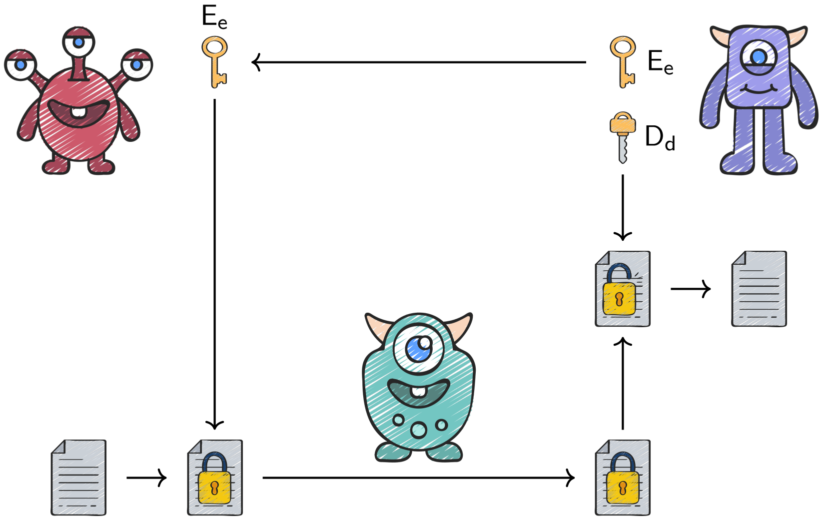
So what is a **secure** encryption scheme ?

Not an easy question to answer ...

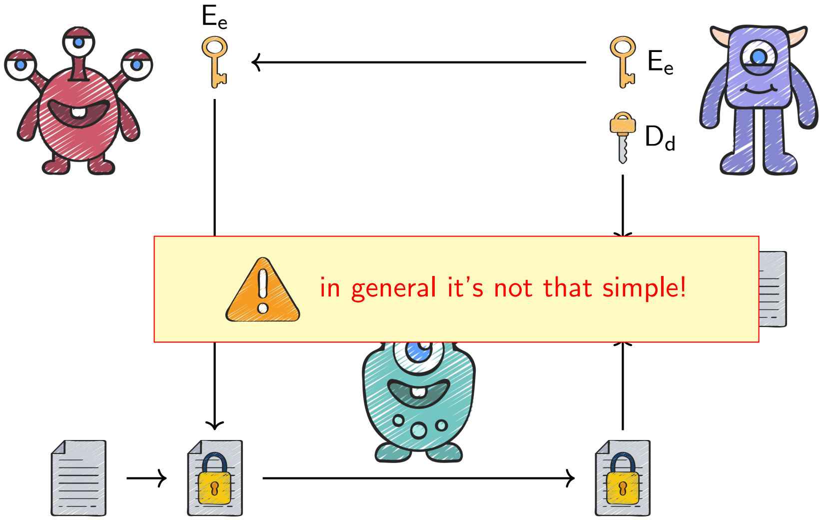
Trapdoor permutations

- A **trapdoor function** is a function that
  - is easy to compute in one direction,
  - yet believed to be difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor".
- A trapdoor permutation family $\{E : X \longrightarrow X\}_{(e,d)}$
  - easy to compute $y = E_e(x)$ for any $x \in X$,
  - (believed to be) difficult to compute $E_e^{-1}(y)$ for any $y \in X$,
  - except if one knows d : $E_e^{-1}(y) = D_d(y) = x$.
- Do such functions exist?

How to encrypt a message *m*



How to encrypt a message m



in general it's not that simple!

RSA - Key Generation

Rivest, Shamir, Adleman (1978)
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21 (2): pp.120-126.

2002 Turing Award

- **Key generation:**
  - Generate two large primes p and q ($p \neq q$). How ?
  - Compute $N = p \cdot q$ and $\varphi(N) = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \varphi(N)$, such that $\gcd(e, (p-1)(q-1)) = 1$.
  - Compute the unique integer d, $1 < d < \varphi(N)$ with $e \cdot d \equiv 1 \mod \varphi(N)$.

Public key $= (N, e)$ which can be published.
Private key $= (d, p, q)$ which needs to be kept secret

RSA - Key Generation

Rivest, Shamir, Adleman (1978)
A method for obtaining digital signatures and public key cryptosystems.
Communications of the ACM 21 (2): pp.120-126.

2002 Turing Award

- **Key generation:**
  - Generate two large primes p and q ($p \neq q$). How ?
  - Compute $N = p \cdot q$ and $\varphi(N) = (p-1)(q-1)$.
  - Select a random integer e, $1 < e < \varphi(N)$, such that $\gcd(e, (p-1)(q-1)) = 1$.
  - Compute the unique integer d, $1 < d < \varphi(N)$ with $e \cdot d \equiv 1 \mod \varphi(N)$.

Public key $= (N, e)$ which can be published.
Private key $= (d, p, q)$ which needs to be kept secret

RSA - Encryption / Decryption

- **Encryption:** if Alice wants to encrypt a message for Bob, she does the following:
  - Obtain Bob's authentic public key (N, e).
  - Represent the message as a number $0 < m < N$.
  - Compute $c = m^e \mod N$.
  - Send the ciphertext c to Bob.

- **Decryption:** to recover m from c, Bob does the following:
  - Use the private key d to recover $m = c^d \mod N$.

RSA - Encryption / Decryption

- **Encryption:** if Alice wants to encrypt a message for Bob, she does the following:
  - Obtain Bob's authentic public key (N, e).
  - Represent the message as a number $0 < m < N$.
  - Compute $c = m^e \mod N$.
  - Send the ciphertext c to Bob.

- **Decryption:** to recover m from c, Bob does the following:
  - Use the private key d to recover $m = c^d \mod N$.

RSA - Proof That Decryption Works

Recall that $e \cdot d \equiv 1 \mod \varphi(N)$, so there exists an integer k such that

$$e \cdot d = 1 + k \cdot \varphi(N).$$

- If $\gcd(m, p) = 1$:
  - By **Fermat's Little Theorem** we have $m^{p-1} \equiv 1 \mod p$.
  - Taking $k(q-1)$-th power and multiplying with m yields

$$m^{1+k(p-1)(q-1)} \equiv m \mod p$$

- If $\gcd(m, p) = p$, then $m \equiv 0 \mod p$ and the previous equality is valid again.

Hence, in all cases $m^{e \cdot d} \equiv m \mod p$ and by a similar argument we have $m^{e \cdot d} \equiv m \mod q$.
Since p and q are distinct primes, the **CRT** leads to

$$c^d = (m^e)^d = m^{ed} = m^{k(p-1)(q-1)+1} = m \mod N.$$

Outline

Prime Numbers

- prime numbers are needed for RSA

Theorem (Prime number theorem)

The number of primes less than x is about $x/\log x$.

- $\rightsquigarrow$ primes are quite common ($\simeq 2^{1014}$ primes $\leq 2^{1024}$).
- testing primes can be done very fast!
- generating primes can be done very fast!
  (on average, one need to test 708 numbers before one find a 1024-bit prime)

Fermat's test

Theorem (Fermat's little theorem)

For $a \in (\mathbb{Z}/n\mathbb{Z})^*$, $a^{\varphi(n)} \equiv 1 \bmod n$.

- if n is prime we have $a^{n-1} \equiv 1 \bmod n$ always
- if n is not prime we have $a^{n-1} \equiv 1 \bmod n$ is unlikely

Fermat's test

For $i = 1$ to k do

- Pick a randomly from $(\mathbb{Z}/n\mathbb{Z})^*$
- Compute $b = a^{n-1} \bmod n$
- If $b \not\equiv 1$ output (Composite,a)

output Possibly Prime

Fermat's test

Theorem (Fermat's little theorem)

For $a \in (\mathbb{Z}/n\mathbb{Z})^*$, $a^{\varphi(n)} \equiv 1 \bmod n$.

- if n is prime we have $a^{n-1} \equiv 1 \bmod n$ always
- if n is not prime we have $a^{n-1} \equiv 1 \bmod n$ is unlikely

Fermat's test

For $i = 1$ to k do

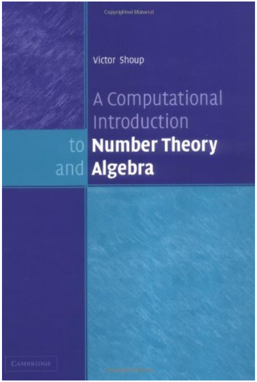
- Pick a randomly from $(\mathbb{Z}/n\mathbb{Z})^*$
- Compute $b = a^{n-1} \bmod n$
- If $b \not\equiv 1$ output (Composite,a)

output Possibly Prime

Carmichael numbers

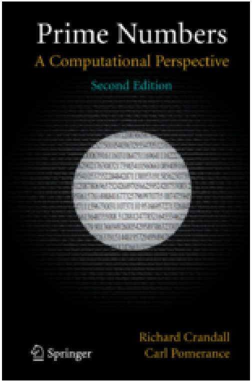
- Carmichael numbers are composite numbers n which fail the Fermat Test for every a not dividing n.
- There are **infinitely many** Carmichael Numbers
  - the first three are 561, 1105, 1729
- **Exercise:** Carmichael Numbers N have the following properties
  1. always odd
  2. are square free
  3. if p divides N then $p-1$ divides $N-1$.
  4. have at least three prime factors
- Need for other tests

References



A Computational Introduction to Number Theory and Algebra
Victor Shoup

References



Prime Numbers: A Computational Perspective
Crandall, Richard, Pomerance, Carl B.

Security of RSA

- Security of RSA relies on difficulty of finding d given N and e.
- If we can **factor** N then we can find p and q
  - Hence we can calculate d.
- i.e. **If factoring is easy we can break RSA.**
  - Currently 768 bit numbers are the largest that have been (2010) factored
  - Hence best to choose (at least) **2048 bit** numbers
- Is RSA as strong as factorization? Will next show that knowing d we can factor N.
  - Still does not rule out possibility that breaking RSA is easier than factoring

Integer Factoring

- Exponential methods:
  - **trial division**
  - **Pollard's $p-1$ method**
  - **Pollard's ρ method**

- Three most effective algorithms are:
  - **quadratic sieve**
  - **elliptic curve factoring algorithm** (ECM)
  - **number field sieve** (NFS)

- One idea many factoring algorithms use:
  - Suppose one fine $x^2 \equiv y^2 \mod N$ s.t. $x \neq \pm y \mod N$.
  - Then $N \mid (x-y)(x+y)$.
  - Neither $(x-y)$ nor $(x+y)$ is divisible by N; thus,

$$\boxed{\gcd(x-y, N) \textit{ has a non-trivial factor of } N.}$$

Integer Factoring

- Exponential methods:
  - **trial division**
  - **Pollard's $p-1$ method**
  - **Pollard's ρ method**

- Three most effective algorithms are:
  - **quadratic sieve**
  - **elliptic curve factoring algorithm** (ECM)
  - **number field sieve** (NFS)

- One idea many factoring algorithms use:
  - Suppose one fine $x^2 \equiv y^2 \mod N$ s.t. $x \neq \pm y \mod N$.
  - Then $N \mid (x-y)(x+y)$.
  - Neither $(x-y)$ nor $(x+y)$ is divisible by N; thus,

$\mathrm{gcd}(x-y, N)$ *has a non-trivial factor of N.*

Integer Factoring

- Exponential methods:
  - **trial division**
  - **Pollard's $p-1$ method**
  - **Pollard's ρ method**

- Three most effective algorithms are:
  - **quadratic sieve**
  - **elliptic curve factoring algorithm** (ECM)
  - **number field sieve** (NFS)

- One idea many factoring algorithms use:
  - Suppose one fine $x^2 \equiv y^2 \mod N$ s.t. $x \neq \pm y \mod N$.
  - Then $N \mid (x-y)(x+y)$.
  - Neither $(x-y)$ nor $(x+y)$ is divisible by N; thus,

$\gcd(x-y, N)$ *has a non-trivial factor of N.*

Time complexity of Integer Factoring

- **quadratic sieve:**

$$O(\exp((1+o(1))\sqrt{\ln N \ln\ln N}))$$

  [For $N \simeq 2^{1024}$, "$O(e^{68})$"]

- **elliptic curve factoring algorithm:**

$$O(\exp((1+o(1))\sqrt{2\ln p \ln\ln p})),$$

  where p is N's smallest prime factor
  [For $N = pq$ and $p, q \simeq 2^{512}$, "$O(e^{65})$"]

- **number field sieve:**

$$O(\exp((1.92+o(1))(\ln N)^{1/3}(\ln\ln N)^{2/3}))$$

  [For $N \simeq 2^{1024}$, "$O(e^{60})$"]

- Multiple 512-bit moduli have been factored
- Extrapolating trends of factoring suggests that:
  - 1024-bit moduli will be factored by **2018** ...

Knowledge of $\varphi(N)$

- We will show knowledge of $\varphi(N)$ allows us to factor N as well.
- We have

$$\varphi(N) = (p-1)(q-1) = N - (p+q) + 1.$$

- Hence

$$\begin{aligned} S &= p+q = N+1-\varphi(N) \\ P &= pq = N \end{aligned}$$

- p and q are the **roots** of $X^2 - SX + P = 0$.

Security of RSA

- Suppose you can find d for a given N and e.
- Then for some integer s
$$ed - 1 = s(p-1)(q-1).$$
- Hence for any $x \neq 0$
$$x^{ed-1} = 1 \mod N.$$
- We want to put
$$y_1 = \sqrt{x^{ed-1}} = x^{(ed-1)/2}$$
and then use
$$y_1^2 - 1 \equiv 0 \mod N$$
to recover a factor of N from $\gcd(y_1 - 1, N)$.
- This will only work when $y_1 \neq \pm 1 \mod N$.

Security of RSA

- Now suppose $y_1 = 1 \mod N$, then we take a square root of y_1

$$y_2 = \sqrt{y_1} = x^{(ed-1)/4}$$

- We know $y_2^2 = y_1 = 1 \mod N$. Hence we compute $\gcd(y_2 - 1, N)$ and see if this gives a factor of N.
- We repeat until
  - either we have factored N
  - or $(ed-1)/2^s$ is no longer divisible by 2.
- We will factor N with probability $1/2$.

Shared Modulus

- Assume for efficiency that each user has
  - The **same modulus** N
  - Different public/private exponents (e_i, d_i)

- Suppose I am user number one, and I want to find user number two's d_2.
  - User one computes p and q since they know d_1.
  - User one computes $\varphi(N) = (p-1)(q-1)$
  - User one computes $d_2 = e_2^{-1} \mod \varphi(N)$

- So each user can then find every other users key.

What about an eavesdropper ?

Shared Modulus

- **Now suppose the attacker is not one of the people who share a modulus**

- Suppose Alice sends the message m to two people with public keys
  - $(N, e_1), (N, e_2)$, i.e. $N_1 = N_2 = N$.

- Eve can see the messages c_1 and c_2 where
  - $c_1 = m^{e_1} \mod N$
  - $c_2 = m^{e_2} \mod N$

Shared Modulus

- Eve can now compute
  - $t_1 = e_1^{-1} \mod e_2$
  - $t_2 = (t_1 e_1 - 1)/e_2$

- Eve can then **retrieve the message** from

$$
\begin{aligned}
c_1^{t_1} c_2^{t_2} &\equiv m^{e_1 t_1} m^{-e_2 t_2} \mod N \\
&\equiv m^{1+e_2 t_2} m^{-e_2 t_2} \mod N \\
&\equiv m^{1+e_2 t_2 - e_2 t_2} \mod N \\
&\equiv m \mod N
\end{aligned}
$$

Small Public Exponent

Hastad (1988)
Solving Simultaneous Modular Equations of Low Degree.
SIAM J. Comput. 17(2): 336-341

- Suppose we have three users
  - With public moduli N_1, N_2 and N_3
  - All with public exponent $e = 3$
- Suppose Alice sends them the **same** message m
- Eve sees the messages
  - $c_1 = m^3 \mod N_1$
  - $c_2 = m^3 \mod N_2$
  - $c_3 = m^3 \mod N_3$
- Now Eve, using the **CRT**, computes the solution to

$$X = c_i \mod N_i$$

to obtain

$$X \mod N_1 N_2 N_3$$

Small Public Exponent

- So the attacker has

$$X \mod N_1 N_2 N_3.$$

- But since $m^3 < N_1 N_2 N_3$ we must have

$$X = m^3$$

over the integers. Hence

$$m = X^{1/3}.$$

- This attack is interesting since we find the message without factoring the modulus.

- This is evidence that breaking RSA is **easier than factoring** .

Small Private Exponent

Wiener (1990)
Cryptanalysis of short RSA secret exponents.
IEEE Transactions on Information Theory 36(3): 553-558

- To **reduce** the work load of the exponentiation, one may wish to use a small value of d rather than a random value
- Since modular exponentiation takes time linear in $\log(d)$, a small private key can improve performance
- If the card has limited computing power, a relatively small value of d would be handy.
- We present an attack, due to Wiener, that succeeds in computing the secret decryption exponent under certain conditions.

Small Private Exponent

- $N = pq$ with $q < p < 2q$, $d < (1/3) \cdot N^{0.25}$.
  $\rightsquigarrow$ Given (N, e) with $ed \equiv 1 \mod N$, attacker can efficiently recover d.

- The proof is using **continued fractions technique**

- There is an integer k such that $ed - k\varphi(N) = 1$.

- We have
$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)}.$$

- Since $N = pq > q^2$, we have $q < \sqrt{N}$ hence
  $N - \varphi(N) = p + q - 1 < 2q + q - 1 < 3\sqrt{N}$.

- Now we see that
$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed - kN}{dN}\right| = \left|\frac{1 + k(\varphi(N) - N)}{dN}\right| < \frac{3k\sqrt{N}}{dN} = \frac{3k}{\sqrt{N}}.$$

Small Private Exponent

- Since $k < d$, we have that $3k < 3d < N^{0.25}$, and hence

$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{1}{dN^{0.25}}.$$

- Finally, since $3d < N^{0.25}$, we have that

$$\left| \frac{e}{N} - \frac{k}{d} \right| < \frac{1}{3d^2}.$$

- **Legendre theorem on continued fractions:** there are at most $\log N$ fractions k/d with $d < N$ approximately e/N so tightly, and they can be obtained by computing the $\log N$ convergents of the continued fraction expansion of e/N (*i.e.* Euclidean algorithm).