Part I

Introduction to Public-Key Cryptography

1. Introduction to Public-Key Cryptography

Symmetric and Asymmetric Cryptography

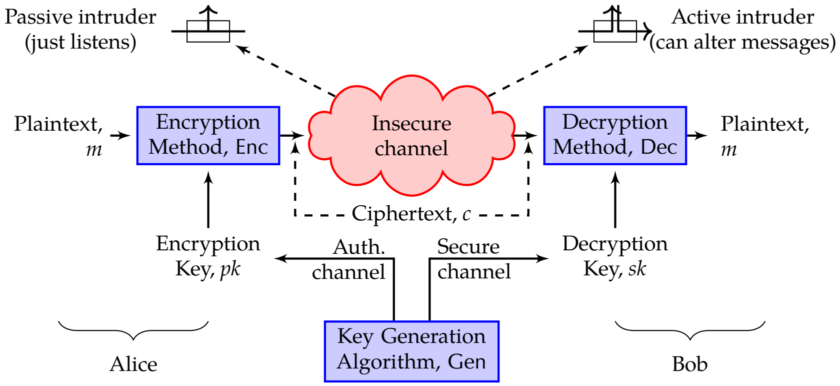
In symmetric (or private key) cryptography, security relies on shared secrets.

- Before any secure communication, the parties must agree on the secret.
- It is reasonable when the parties are few and the communication needs are known in advance (e.g. cellular communications, domestic WiFi, etc.)

In asymmetric (or public key) cryptography the parties do not need to share a secret.

- The parties need not know in advance that they will communicate.
- It is the natural choice when the communication needs are not known in advance (e.g. e-commerce over the internet)

Channel Model



Key Setup

The receiver generates a pair of keys (pk, sk):

pk is the public key

sk is the private key

A give keypair can be used several times or it can be generated just before communications begin (session key).
The senders use pk to cipher their messages.
The recevier uses sk to decipher the messages.

Advantages and Disadvantages

Advantages

- Simplified key management, because a single key is used to communicate with multiple parties.
- Simplified key distribution, because the public key does not need a confidential channel.

Disadvantages

Algorithms are slower.

Distribution of the Public Key

In case of passive attackers, distribution can be over an insecure channel.
In case of active attackers, distribution requires an authenticated channel.
One way to have an authenticated channel is having an trusted third party authenticate the keys. It is assumed that all the parties share some secure information with the trusted third party. Such an architecture is called a Public-Key Infrastructure (PKI).

Definition of Public-Key Encryption Scheme

A Public Key Encryption (PKE) Scheme is a tuple of efficient algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

- $(pk, sk) = \mathsf{Gen}(n)$ The key generation algorithm takes as input a security parameter n and outputs a pair of keys (pk, sk)
- $c = \mathsf{Enc}(pk, m)$ The encryption algorithms takes as input a message from the plaintext space $\mathcal{M}$, which may depend on pk, and outputs a ciphertext
- $m = \mathsf{Dec}(sk, c)$ The decription algorithm takes as input the private key and a message and outputs a message (or fails).

The scheme is correct if

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$$

for any keypair (pk, sk) generated by Gen

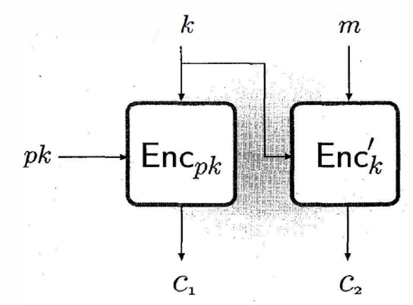
1. Introduction to Public-Key Cryptography

Security Considerations

- Perfect secrecy is impossible (an adversary with infinite computational power can find sk given pk)
- The Attacker can encrypt any message, so there is no one-time-key model, only many-time-key.
- Deterministic algorithms cannot be semantically secure.

Hybrid Encryption

PKE is slow and yields a ciphertext much larger than the plaintext. Therefore it is better to use PKE in tandem with symmetric cryptography.



The sender chooses a random session key k and encrypts k using the public key of the receiver. Then it uses k to encrypt the message with a symmetric cipher.

1. Introduction to Public-Key Cryptography

1. Introduction to Public-Key Cryptography

Textbook RSA

We now discuss a scheme based on the RSA Assumption. Note that *textbook* RSA is insecure and must not be used. To make RSA suitable for practical usage it is necessary (at least) to:

- use large enough and safe prime numbers,
- implement randomization and padding such as in the RSA-OAEP construction,
- be careful in the implementation to avoid side-channel attacks.

Textbook RSA

Key Generation

Textbook RSA Scheme

Gen Takes as input the security parameter n (known as the *key size*).

1. Generate a pair or different random primes p and q of similar size such that $N = pq$ has n bits.
2. Compute $\varphi(N) = (p-1)(q-1)$
3. Choose e such that $\gcd(e, \varphi(N)) = 1$
4. Compute $d = e^{-1} \bmod \varphi(N)$
5. $pk := (N, e)$ and $sk := (N, d)$.

Textbook RSA

Encryption and Decription

Textbook RSA Scheme

Enc Takes as input *pk* and a message $m \in \mathbb{Z}_N^*$. Outputs

$$c := m^e \bmod N$$

Dec Takes as input *sk* and a ciphertext c. Outputs

$$m := c^d \bmod N$$

Necessary proofs

We must prove that

1. $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$
2. $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are polynomial-time algorithms
3. recovering sk from pk is hard
4. obtaining m from c is hard without knowing sk

Proof that it works

Since $ed \bmod \varphi = 1$, then there exists k such that $ed = 1 + k\varphi$. By Fermat's theorem, $m^{p-1} \bmod p = 1$, therefore:

$$c^d \equiv m^{ed} \equiv m^{1+k(p-1)(q-1)} \equiv m \pmod{p}$$

Similarly, for q. Since p and q are distinct primes:

$$c^d \equiv m^{ed} \equiv m \pmod{N}$$

Computational Aspects

Enc and Dec are modular exponentiations that can be performed in polynomial time with the `square-and-multiply` algorithm.
We will see that random prime numbers can be generated with a probabilistic polynomial-time algorithm GenPrime.

Gen general structure

1. Generate random primes p and q. Can be performed in polynomial time by GenPrime
2. Compute $\varphi(N) = (p-1)(q-1)$ Integer multiplication
3. Choose e Generally it is a fixed constant. Otherwise GenPrime can be used.
4. Compute $d = e^{-1} \bmod \varphi(N)$ Extended euclidean algorithm

GenPrime

Finding a random prime

The general idea is to generate an odd random number of the desired size and then test for primality. The algorithm is repeated until a prime number is found.

Density of primes

According to the *prime number theorem* we can say that the probability that a random n-bit integer is prime is larger than 1/n. Therefore, if we try n^2 numbers, the probability of failure is at most:

$$\left(1 - \frac{1}{n}\right)^{n^2} \leq e^{-n}$$

which is negligible.

GenPrime

Testing for primality

Testing for primality can be done deterministically in polynomial time. However, there are faster probabilistic tests whose error rate can be made negligible by repeating the test multiple times. The most common one is the Miller-Rabin test.

Output of the Miller-Rabin test

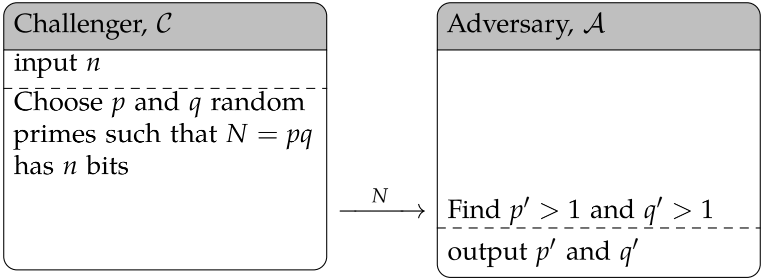
Suppose the Miller Rabin test is repeated t times for a given candidate N.

- If N is prime, then the test always outputs “prime”.
- If N is composite, then the test outputs “prime” with probability at most 2^{-t}.

By choosing t large enough, the probability of erroneously choosing a false prime can be made negligible.

1. Introduction to Public-Key Cryptography

The Factoring Experiment


Challenger, $\mathcal{C}$
input n
Choose p and q random primes such that $N = pq$ has n bits
N
Adversary, $\mathcal{A}$
Find $p' > 1$ and $q' > 1$
output p' and q'

The attacker wins if $p'q' = N$.

The Factoring Assumption

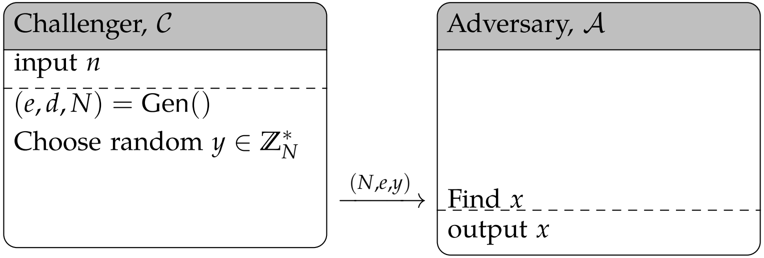
It is assumed that the factoring experiment succeeds with negligible probability.

Theorem

Recovering d from (e, N) is computationally equivalent to factoring N.

Therefore recovering *sk* from *pk* is as hard as factoring N. Note that the above does not imply semantic security. There might exist a way to decrypt some messages without knowing the key.

The RSA Experiment


Challenger, $\mathcal{C}$
input n
$(e, d, N) = \text{Gen}()$
Choose random $y \in \mathbb{Z}_N^*$
(N,e,y)
Adversary, $\mathcal{A}$
Find x
output x

The attacker wins if $x^e \bmod N = y$.

The RSA Assumption

It is assumed that the RSA experiment succeeds with negligible probability.

Note that RSA Assumption implies Factoring Assumption, but there is no proof of the inverse. There might exists a way to solve the RSA experiment without factoring N (or solving an equivalent problem such as finding φ or d).

RSA might be insecure even if the Factoring Assumption is ture.

1. Introduction to Public-Key Cryptography

Encoding of messages as elements of $\mathbb{Z}_N^*$.

Any message in $\{0,1\}^{n-1}$ can be represented as an element of $\mathbb{Z}_N$; shorter messages can be padded.
An issue is that $\mathcal{M} = \mathbb{Z}_N^*$ does not include the elements of $\mathbb{Z}_N$ that have $\gcd(m, N) \neq 1$.
However finding one is unlikely and, worse, would reveal a factor of N.

Choice of Encryption and Decryption Exponents

Choosing a small exponent results in a lower computational complexity for encryption.
Exponentiation complexity is $O(n^3)$ with `Square-and-Multiply` but becomes $O(n^2)$ if the exponent is small.
Choosing $e = 3$ makes textbook RSA vulnerable to attacks.

Attack to RSA with low exponent

Suppose e and m small. If $m^e < N$, then m can be calculated as $m = \sqrt[e]{N}$

For non-textbook RSA there are fewer problems, but too small exponents are avoided.
A popular choice is the prime $e = 2^{16} + 1 = 65547$.
Choosing a small d is appealing, but there is an attack exploiting continuous fractions, which is effective if $d < \sqrt[4]{N}$.

Speeding calculations using the Chinese Reminder Theorem

Once d has been calculated, p and q are no longer necessary. The receiver can keep them to speed up calculations.
Instead of calculating $m = c^d \bmod N$ one can calculate

$$m \bmod p = c^{d \bmod p-1} \bmod p$$
$$m \bmod q = c^{d \bmod q-1} \bmod q$$

and recombine them using the Chinese Reminder Theorem. This saves about $1/4$ of the calculations and can be parallelized.

Using the *Universal Exponent*

It is common practice to use the number

$$\lambda(N) = \operatorname{lcm}(p-1, q-1)$$

called *universal exponent* of N in place of $\varphi(N)$ in the key generation algorithm. Since $\lambda|\varphi$, then $\lambda < \varphi$ and the resulting d is smaller. However, since $\gcd(p-1, q-1)$ is likely to be small, then λ and φ have similar size.

Fermat Factorization

If $p - q$ is small, the Fermat factorization algorithm can find p and q efficiently.

Fermat factorization

If $N = pq$, then

$$N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2 = t^2 - s^2$$

If $p \simeq q$, the second term,s^2, is small.
Starting from $t = \sqrt{N}$, increase t until you find t and s. Solve a linear system to find p and q.

Since algorithm complexity is roughly $O(s)$, in the general case (p and q chosen randomly), complexity is $O(e^n)$.

Pollard's $p - 1$ Factorization Method

If $p - 1$ has only *small* factors, then Pollard's method can find p efficiently.

```
function POLLARD($N$)
    Choose $B$                              ▷ $B$ is the upper bound
    Choose $a \in \mathbb{Z}_N^*$                 ▷ For example $a = 2$
    Calculate $y = a^{B!} \bmod N$
    Calculate $p = \gcd(y - 1, N)$
    if $1 < p < N$ then
        return $p$
    else
        fail
    end if
end function
```

Pollard's $p-1$ Method

Why it works

If $(p-1)$ has only small factors, then all the factors of $p-1$ are factors of $B!$. Consequently

$$a^{B!} \bmod p = a^{B! \bmod (p-1)} = 1$$

If $q-1$ does not have only small factors, in general

$$a^{B!} \bmod q \neq 1$$

Therefore p is a factor of y and q is not, thus $\gcd(y-1, N) = p$.

Pollard's $p - 1$ Method

Countermeasures

The attack is thwarted if $p - 1$ and $q - 1$ both have large factors. Ideally, one could choose p and q strong primes, i.e. $(p-1)/2$ and $(q-1)/2$ are also prime.

However, finding a strong prime is computationally much more expensive than finding a random prime. Further, a prime p randomly chosen is unlikely to have only small factors, therefore security risks of this attack are generally ignored.

1. Introduction to Public-Key Cryptography

Drawbacks of Textbook RSA

The main drawbacks of Textbook RSA

- it is not randomized, thus it is not semantically secure
- it is easy to forge a valid ciphertext (Chosen Ciphertext Attack, CCA)

Optimal Asymmetric Encryption Padding (OAEP)

Optimal Asymmetric Encryption Padding (OAEP) is reversible, randomized method for encoding a plaintext message m into a longer, randomized string $\hat{m}$.

$$\hat{m} := \mathsf{OAEP}(m, r)$$

$\hat{m}$ is the encoding of m with randomness r.
When using RSA-OAEP with public key (N, e) with $\|N\| \geq n$ the ciphertext becomes

$$c := \mathsf{OAEP}(m, r)^e \bmod N$$

The receiver computes $\hat{m} = c^d \bmod N$ and then tries to find m. If the inverse does not exist, the receiver knows that the ciphertext is invalid (probably a CCA). Otherwise, the unique inverse is m.

OAEP

Mask Generation Functions and CCA

RSA-OAEP uses two functions G and H, called Mask Generation Functions, which are assumed to behave as independent random oracles.

Given an n bit key, we will consider the (inefficient) special case of m having $n/4$ bits and r having $n/2$ bits. Then G and H are functions mapping n bits into n bits.

Since OAEP maps $3n/4$ bits into n bits, the probability that a random element in $\mathbb{Z}_N$ is in the image of OAEP is $2^{-n/4}$, which is negligible.

Theorem

If the RSA Assumption is true and H, G are random oracles, then RSA-OAEP is semantically secure and secure against a chosen ciphertext attack.

RSA-OAEP Encryption Scheme (1)

```
function RSA-OAEP ENCRYPTION($m$)    ▷ $m$ has $n/4$ bits
    repeat
        Choose random $r$ with $n/2$ bits
        $m' = m \| 0^{n/4}$
        $\hat{m}_1 = G(r) \oplus m'$
        $\hat{m} = \hat{m}_1 \| (r \oplus H(\hat{m}_1))$
    until $\hat{m} < N$
    return $\hat{m}^e \bmod N$
end function
```

RSA-OAEP Encryption Scheme (2)

```
function RSA-OAEP DECRYPTION($c$)
    $\hat{m} = c^d \bmod N$
    $\hat{m}_1$ is the first half of $\hat{m}$        $\triangleright$ $\hat{m}_1$ has $n/2$ bits
    $\hat{m}_2$ is the second half of $\hat{m}$       $\triangleright$ $\hat{m}_2$ has $n/2$ bits
    $r = H(\hat{m}_1) \oplus \hat{m}_2$
    $m' = \hat{m}_1 \oplus G(r)$
    if final $n/4$ bits of $m'$ are zeros then
        return the first $n/4$ messages of $m'$
    else
        reject message
    end if
end function
```

1. Introduction to Public-Key Cryptography

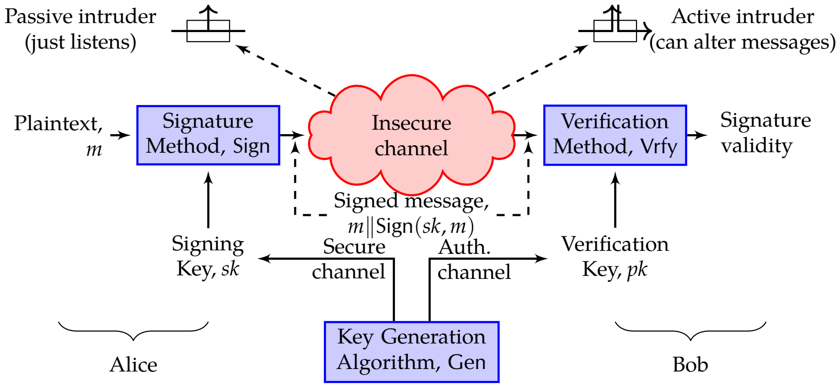
Introduction

Digital signatures allow a signer S who has published a public key pk to sign a message m so that any party knowing pk can verify that the message

- originated from S and
- has not been modified after the signature.

The signature is a string of bits that is generally appended to the message. In some less common schemes, the message can be recovered from the signature, therefore the signature replaces the message.

Channel Model



Properties

Digital signatures are similar to MACs because they provide data integrity and data origin authentication. In addition digital signatures are

- publicly verifiable: if a party verifies the signature as legitimate, then all the parties will do the same
- transferable: if a party verifies the signature as legitimate, then it can copy the signature to a third party who can verify the signature by herself

Therefore digital signatures have the property of non-repudiation. If S signs a message, at a later time he/she cannot deny having done so. This property cannot be achieved using symmetric encryption unless a TTP is involved in the signature process.

Definition

A signature scheme is a set of efficient algorithms:

- $(pk, sk) = \mathsf{Gen}(n)$ The key generation algorithm takes as input a security parameter n and outputs a pair of keys (pk, sk), called public and private key, respectively.
- $\sigma = \mathsf{Sign}(sk, m)$ The signing algorithm takes as input a private key sk and a message m. It outputs a signature σ.
- $\mathsf{Vrfy}(pk, m, \sigma)$ The verification algorithm Vrfy takes as input a public key pk, a message m, and a signature σ. It outputs true if the signature is valid meaning or false if the signature is not valid.

For every (pk, sk) generated by Gen, it holds that

$$\mathsf{Vrfy}(pk, m, \mathsf{Sign}(sk, m)) = \mathsf{true}$$

Security

Given pk, an adversary outputs an existential forgery if it outputs a message m along with a valid signature σ and m was not previously signed.
A signature scheme is existentially unforgeable under an adaptive chosen-message attack if, for any adversary, the probability of success of the following experiment is negligible.

The Existential Forgery Experiment

1. Run $\mathsf{Gen}(n)$ and obtain (pk, sk)
2. Adv is given pk and oracle access to $\mathsf{Sign}_{sk}(\cdot)$. Define $\mathcal{Q}$ the set of messages whose signature was asked to the oracle. Adv outputs (m, σ).
3. The experiment is successful if $\mathsf{Vrfy}_{pk}(m, \sigma) = 1$ and $m \notin \mathcal{Q}$.

Textbook RSA Signature

Similarly to textbook RSA encryption this scheme is not secure.

Textbook RSA signature

Gen takes as input a security parameter n. It is identical to the key generation algorithm in RSA encryption.

Sign takes as input sk and a message $m \in \mathbb{Z}_N^*$. Outputs the signature

$$\sigma := m^d \bmod N$$

Vrfy takes as input (pk, m, σ). It outputs 1 if and only if

$$m = \sigma^e \bmod N$$

It is easy to see that if σ is legitimate, then $\sigma^e = m^{de} = m$.

Attacks to textbook RSA signatures

Existential forgery

It is trivial to get an existential forgery, given *pk* alone. This attack does not even require oracle access to $\text{Sign}_{sk}(\cdot)$.

1. Choose random $\sigma \in \mathbb{Z}_N^*$
2. Compute $m := \sigma^e \bmod N$
3. Output (m, σ).

One may argue that the probability of getting a meaningful message is low, but what if the message is a key for a symmetric cipher?

Attacks to textbook RSA signatures

Selective forgery

Given oracle accesses to $\mathsf{Sign}_{sk}(\cdot)$ the attacker can forge a signature for any message m.

1. Choose random $m_1 \in \mathbb{Z}_N^*$
2. Compute $m_2 = m \cdot m_1^{-1} \bmod N$
3. Obtain σ_1 and σ_2
4. Output $\sigma = \sigma_1 \sigma_2 \bmod N$

This is because:

$$\sigma^e \equiv (m_1^d m_2^d)^e \equiv m_1 m_2 \equiv m \pmod N$$

RSA with Full Domain Hash (RSA-FDH)

RSA-FDH

Gen	as in textbook RSA
Sign	$\sigma := H(m)^d \bmod N$
Vrfy	output 1 if $\sigma^e = H(m) \bmod N$

If H is collision-resistant, then the previous attacks against textbook RSA are ineffective. However it has not been proved that collision-resistance of H is a sufficient condition for security.
RSA-FDH is secure under the assumption that $H\colon \{0,1\}^* \to \mathbb{Z}_N^*$ is a random oracle.
RSA-FDH also has the advantage of allowing the signature of arbitrarily-sized messages.

RSA signatures in practice

RSA-FDH is not used in practice, but variations are used:

- RSA with PKCS#1 v. 1.5 padding is very popular and widely scrutinized though no formal proof exists
- RSA-PSS (Probabilistic Signature Scheme), which is randomized and formally proved