Agenda

- Encryption schemes
- Computational security
- One-way functions
- Public-key encryption schemes
Cryptography

- **Cryptography** is the study of secure communication
  - Cryptography is *much* older than computer science
  - Traditionally, cryptography referred to the development of various ad-hoc encryption schemes
  - These schemes were usually broken sooner or later

- **Modern cryptography** was born in the 1970s, when computational complexity theory was applied to cryptography
  - Modern cryptography aims to develop *provably unbreakable* encryption schemes
  - Unbreakability is conditioned on complexity assumptions
Encryption Schemes

- Two parties, *Alice* and *Bob*, wish to communicate in the presence of a malevolent eavesdropper *Eve*
Encryption Schemes

- **Encryption scheme** consists of two algorithms:
  - Encryption algorithm $E$
  - Decryption algorithm $D$

- Both algorithms are parameterised by a randomly selected *secret key* $k \in \{0, 1\}^n$, known to Alice and Bob
  - For all $k \in \{0, 1\}^n$ and $x \in \{0, 1\}^m$, we have $D_k(E_k(x)) = x$
Encryption Schemes

Key: $k \in_R \{0, 1\}^n$

Ciphertext: $y = E_k(x)$

Message (plaintext):
$x \in \{0, 1\}^m$

Transmission of a secret message $x$:
- Alice computes ciphertext $y = E_k(x)$
- Alice sends $y$ to Bob and Bob computes $x = D_k(y)$

Requirements for encryption scheme $(E, D)$:
- $E$ and $D$ are polynomial-time
- Eve cannot obtain any information about $x$ from $y$, even if Eve knows $E$ and $D$
Perfect Secrecy

- What does “Eve cannot obtain any information about $x$ from $y$” mean?

Definition (Perfect secrecy)

Let $(E, D)$ be an encryption scheme for messages of length $m$ with key size $n$. We say that $(E, D)$ is perfectly secret if for any pair of messages $x, x' \in \{0, 1\}^m$, the distributions $E_{U_n}(x)$ and $E_{U_n}(x')$ are the same, where $U_n$ denotes the uniform distribution over $\{0, 1\}^n$.

- If the key is picked at random, Eve will see the same distribution of ciphertexts regardless of the actual message.
One-time Pad

A simple solution: one-time pad

- For message $x \in \{0, 1\}^n$, select key $k \in \{0, 1\}^n$ uniformly at random
- Let $E_k(x) = x \oplus k$ and $D_k(x) = x \oplus k$, where $\oplus$ is the bit-wise XOR
- Now $D_k(E_k(x)) = (x \oplus k) \oplus k = x$

Key: $k \in_R \{0, 1\}^n$
Ciphertext: $y = E_k(x)$

Alice

Eve

Bob

Message (plaintext):
$x \in \{0, 1\}^n$

$x = D_k(y)$
One-time Pad

- **One-time pad satisfies perfect secrecy:**
  - If $k$ is uniformly distributed over $\{0, 1\}^n$, then so is $E_k(x)$

- **One-time pads are one-time:**
  - If same key $k$ is used twice, then $E_k(x) \oplus E_k(x') = x \oplus x'$, which yields nontrivial information about the messages
Computational Security

- **Perfect security** means we can *fool* any adversary
  - Perfect security requires that key length is at least message length
  - However, it is reasonable to assume that adversary has limited computational power

- **Computational security**: aim is to fool any (randomised) polynomial-time adversary
  - One can define various forms of computational security
  - Strength of the required security can depend on application
  - Real definitions are somewhat complicated and subtle
A simple version of computational security:

- **Intuition:** adversary cannot guess any bit of the plaintext with probability significantly larger than $1/2$
- **Formally:** we say that a scheme $(E, D)$ for $m$-bit messages with $n$-bit keys is *computationally secure* if for any probabilistic polynomial-time algorithm $A$,

$$\Pr_{k \in U_n, x \in U_m} [A(E_k(x)) = (i, b) \text{ such that } x_i = b] \leq 1/2 + \varepsilon(n),$$

where $\varepsilon(n) = n^{-\omega(1)}$, that is, $\varepsilon(n) < n^{-c}$ for any $c$ and for sufficiently large $n$. 
One-way Functions

- One can show that if \( P = NP \), then computationally secure encryption schemes do not exist
  - Thus, modern cryptography requires \( P \neq NP \)
  - Assuming \( P \neq NP \) is not quite enough, as far as we know

- Standard assumption: *one-way functions* exist
One-way Functions

Definition
A polynomial-time computable function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is a one-way function if for every probabilistic polynomial-time algorithm $A$,

$$\Pr_{x \in U_n, y = f(x)} [A(y) = x' \text{ such that } f(x') = y] < \varepsilon(n),$$

where $\varepsilon(n) = n^{-\omega(1)}$.

Conjecture
There exists a one-way function.
One-way Functions

- Existence of one-way functions implies $P \neq NP$
  - No one-way functions are known
- Some candidates:
  - *Integer multiplication*: the function $(p, q) \mapsto pq$, where $p$ and $q$ are prime numbers (inverse: factoring)
  - *RSA function*: $f_{RSA}(x, e, p, q) = (x^e \mod pq, pq, e)$, where $p$ and $q$ are prime numbers, $e$ is a relative prime to $\phi(pq) = (p - 1)(q - 1)$ and $x < pq$ is an integer
One-way Functions and Security

- One-way functions are used as building block for encryption schemes

Theorem

Assume one-way functions exist. Then for every $c \in \mathbb{N}$, there exists a computationally secure encryption scheme $(E, D)$ for $n^c$-bit messages with $n$-bit keys.
Another application of one-way functions: pseudorandom generators

- Basic idea: turn a small number of random bits into a larger number of “random-looking” bits
- Specifically, we require that the pseudorandom bits cannot be distinguished from real random bits by polynomial-time algorithms
- Lots of practical applications
Pseudorandomness

Definition
Let $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a polynomial-time computable function, and let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial-time computable function with $\ell(n) > n$. We say that $G$ is a secure pseudorandom generator of stretch $\ell(n)$ if $|G(x)| = \ell(|x|)$ for every $x \in \{0, 1\}^*$ and for every probabilistic polynomial-time algorithm $A$, we have that

$$\left| \Pr_{x \in U_n} [A(G(x)) = 1] - \Pr_{z \in U_{\ell(n)}} [A(z) = 1] \right| < \varepsilon(n),$$

where $\varepsilon(n) = n^{-\omega(1)}$.

Definition
If one-way functions exist, then there is a secure pseudorandom generator with stretch $n^c$ for every $c \in \mathbb{N}$. 
The notion of encryption schemes we have been discussing so far is *private-key encryption*

- Alice and Bob need to share a secret key $k$
- Impractical: this key needs to be shared somehow!

Modern cryptography is based on *public-key encryption*

- Both the algorithms $(E, D)$ and the encryption key are known publicly
Public-key Encryption

In public-key encryption, Bob generates two keys:
- A private key $k_1$ and a public key $k_2$
- Bob sends the public key to Alice unencrypted

Alice can now send an encrypted message to Bob
- Alice encrypts the message as $y = E_{k_2}(x)$
- Bob decrypts the message as $x = D_{k_1}(y)$
Public-key Encryption

- The security requirement is now:
  - Public key should not reveal information about private key
  - Knowing public key should not reveal information about the message

- This can be achieved using one-way functions
Recall the definition of RSA function:

1. $f_{RSA}(x, e, p, q) = (x^e \mod pq, pq, e)$, where $p$ and $q$ are prime numbers, $e$ is a relative prime to $\phi(pq) = (p - 1)(q - 1)$ and $x < pq$ is an integer
2. $e$ can be selected to be a fixed prime number
The keys for the RSA system are now as follows:

- Bob generates two large primes $p$ and $q$
- Bob’s private key is $k_1 = (p, q, d)$, where $d = e^{-1} \mod \phi(pq)$, that is, $ed = 1 + k\phi(pq)$ for some $k$ (given $p, q$ and $e$, one can compute $d$ by extended Euclid’s algorithm)
- Bob’s public key is $k_2 = (pq, e)$
RSA Encryption

Alice encrypts a message $x$ as

$$y = x^e \mod pq$$

Bob decrypts a message $y$ by

$$y^d = x^{ed} = x^{1+k\phi(pq)} = x(x^{\phi(pq)})^k = x \mod pq,$$

where $x^{\phi(pq)} = 1 \mod pq$ by an extension of Fermat’s theorem.
Lecture 16: Summary

- Encryption schemes
- Basic idea of computational security
- One-way functions
- Pseudorandom generators
- Public-key encryption