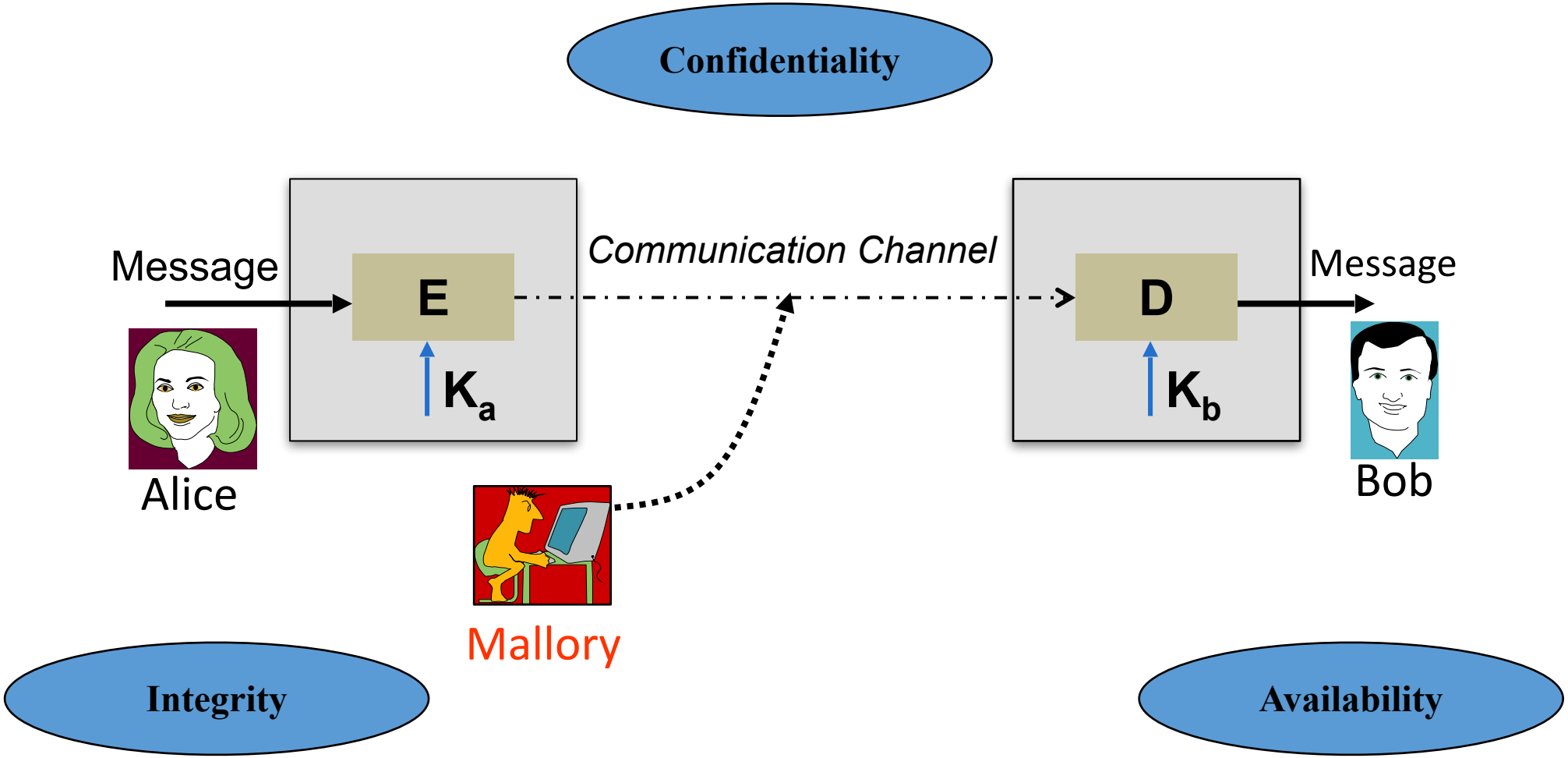
Implementation Security In Cryptography

Lecture 02: (Very informal) Intro to (Theoretical) Crypto

Why Theoretical Crypto?

- Just to create a bridge
- We shall only mention a few key results to set the stage..
- Ultimate goal is to reach something called block ciphers
  - The rest of the course will be on how to make and break block ciphers and similar primitives.

Cryptology Once Again...



Confidentiality

- The network is untrusted..
- The information should remain secret during transmission…
- Crypto-algorithms makes it gibberish looking

Integrity

- No change allowed by unauthorised person.

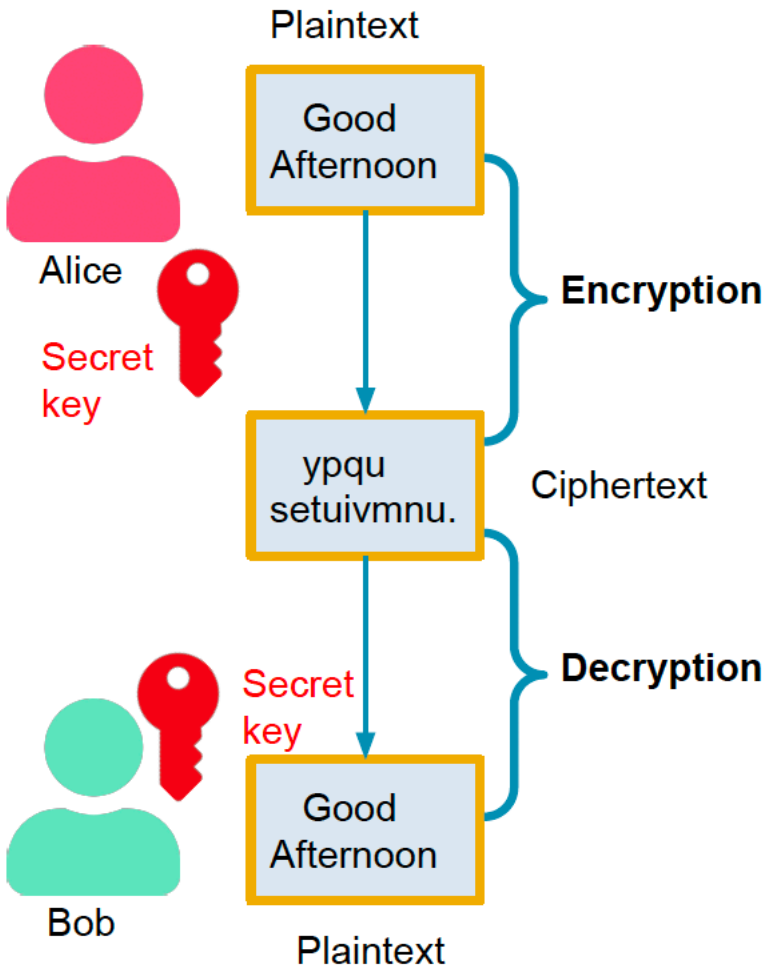
- But the change should be made by authorized users
  - modification: change made by unauthorized users.

- Need techniques to ensure the integrity of data:
  - detect any modification made

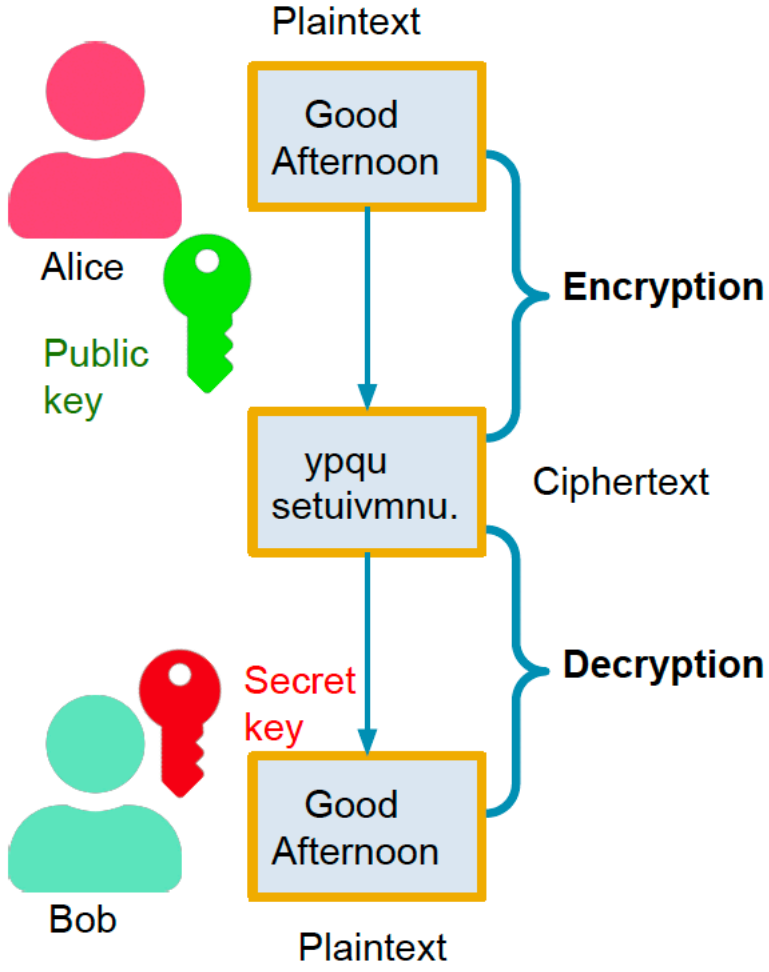
Availability

- Things should not become impractical.
  - Computational overheads of cryptography should be limited.

Crypto: Symmetric Key Vs. Public Key

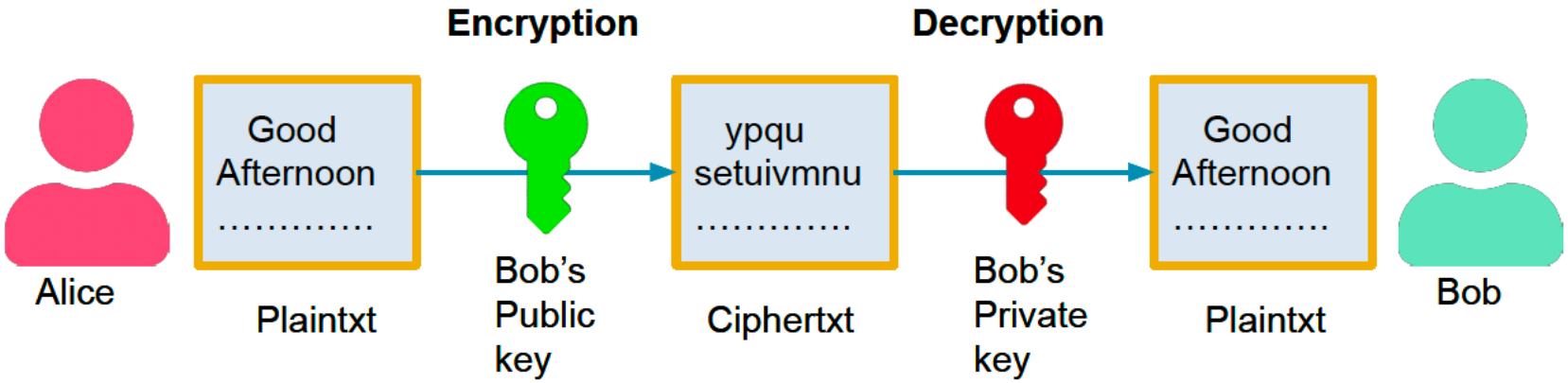


Symmetric-key scheme



Public-key scheme

Crypto: Symmetric Key Vs. Public Key



Public-key cryptography

used in

SSL/ TLS

Secure key exchange

Crypto-currencies

Digital signature

GPay

Digital payment

Crypto: Formal Definitions

- An SKE is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$
  - KGen: $k \xleftarrow{\$} \mathscr{K}$
  - Enc: for $m \in \mathscr{M}, C \leftarrow \mathsf{Enc}(m, k)$, Enc is a probabilistic algorithm.
  - Dec: for $c \in \mathscr{C}, m = Dec(c, k)$, Dec is a deterministic algorithm.
  - for all $k \in \mathscr{K}, m \in \mathscr{M}$, and any ciphertext c output by Enc(m, k), it holds that Dec(c, k) = m.

- A PKE is a tuple of algorithms $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$
  - KGen: $(pk, sk) \xleftarrow{\$} \mathscr{K}$
  - Enc: for $m \in \mathscr{M}, C \leftarrow \mathsf{Enc}(m, pk)$, Enc can be a probabilistic algorithm.
  - Dec: for $c \in \mathscr{C}, m = Dec(c, sk)$, Dec is a deterministic algorithm.
  - for all $k \in \mathscr{K}, m \in \mathscr{M}$, and any ciphertext c output by Enc(m, pk), it holds that Dec(c, sk) = m.

What do we mean by “Secure”?

- Well, it is a philosophical question.
- In cryptography, it has concrete definitions
  - Consider that I want my message being sent to remain confidential.
  - So, nobody except the intended recipient should understand anything about it
  - And this has to be ensured against someone who is “bad” — “the adversary”
  - If I can ensure that, (using say some algorithm with the aforementioned syntax)
    - Then I am secure…

Who Is “Adversary”?

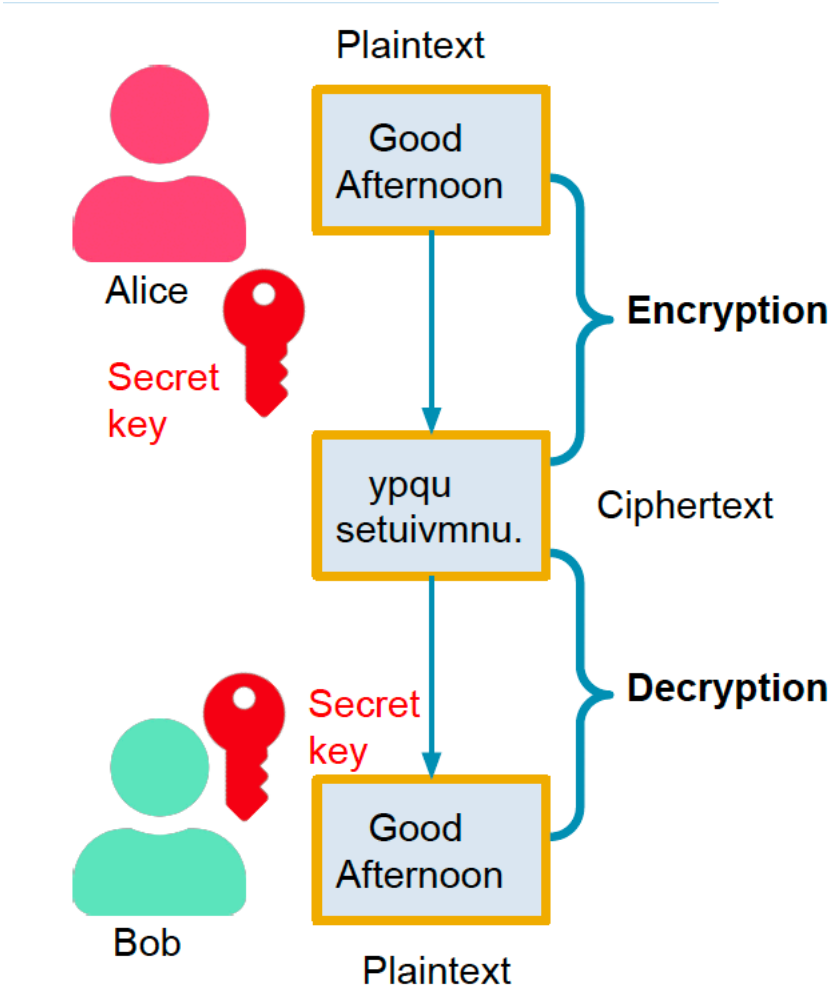
Formally, adversary is an algorithm (PPT) interacting with the cryptosystem

We define certain “minimal” capabilities for the adversary. And then define “security” with respect to that… Finally we prove that a given crypto algorithm adheres to this definition of security and adversary…

The Adversarys' Power: Threat Model

- **Ciphertext-only attack**: The adversary only knows ciphertext(s) and want to extract information about plaintext(s).
- **Known-plaintext attack**: The adversary knows some plaintext-ciphertext pairs (for some key), and want to deduce information about the plaintext of some *other* ciphertext.
- **Chosen-plaintext attack**: Adversary has obtained some plaintext-ciphertext pairs as per its choice. The goal is same as the previous attack.
- **Chosen-ciphertext attack**: Adversary can additionally obtain decryptions for some ciphertexts of its choice, but not the one it really wants to "expose".

Let us begin from Symmetric Key…



Symmetric-key scheme

What do we mean by “Secure”?

- A ciphertext *should leak no additional information* about the underlying plaintext.
- Let’s consider the “ciphertext only” adversary…
- Refine the statement:
  - Suppose the adversary has collected a ciphertext C. It wants to know something about the plaintext that it does not know already.
- *Now the question is, what information does the adversary “already has”??*
  - *Generally some generic information about the plaintext space*
    - *Such as if it is an English sentence*
    - *Some other specifics — such as the message space is “ATTACK” or “NO ATTACK”*
    - What is the encryption algorithm, except for the key — Kerckhoff's Principle

Let's See a Simple Cipher: Shift Cipher

- Let $\mathscr{K} = \{0,1,2,\cdots,25\}$
- Let $\mathscr{M} = \{0,1,2,\cdots,25\}$ — this can be english alphabets.
- KGen: choose a key k.
- Enc: for $m = m_1||m_2||\cdots||m_l$, outputs $c = c_1||c_2||\cdots||c_l$, such that $c_i = [m_i + k] \ mod \ 26$
- Dec: $m_i = [c_i - k] \ mod \ 26$

- Let's first try some example: The message space is English alphabets…

Let's See a Simple Cipher: Shift Cipher

- Let $\mathscr{K} = \{0,1,2,\cdots,25\}$
- Let $\mathscr{M} = \{0,1,2,\cdots,25\}$ — this can be english alphabets.
- KGen: choose a key k.
- Enc: for $m = m_1||m_2||\cdots||m_l$, outputs $c = c_1||c_2||\cdots||c_l$, such that $c_i = [m_i + k]\ mod\ 26$
- Dec: $m_i = [c_i - k]\ mod\ 26$

- Can you break this cipher? — say I have told you that it only encrypts English, and it is a ciphertext only attack??

Let’s Try…

- First approach — find the key by *brute force:* possible, there are only 26 keys
  - *Key Lesson:* Have a large key space

Let's Try...

- First approach — find the key by *brute force:* possible, there are only 26 keys
  - *Key Lesson:* Have a large key space
- Second approach: Suppose, I am sending only three possible messages with the following probabilities:
  - Pr[M = kim] = 0.5
  - Pr[M = ann] = 0.2
  - Pr[M = boo] = 0.3
- Let's say the observed ciphertext c = dqq

Let's Try...

- First approach — find the key by *brute force:* possible, there are only 26 keys
  - *Key Lesson:* Have a large key space
- Second approach: Suppose, I am sending only three possible messages with the following probabilities:
  - Pr[M = kim] = 0.5
  - Pr[M = ann] = 0.2
  - Pr[M = boo] = 0.3
- Let's say the observed ciphertext c = dqq
- Can we get to know something more about the plaintext than we already know?
  - Remember: The adversary does not know the key… every key is equiprobable..
    - Pr[K = k] = 1/26 for any k.

Let's Try...

- First approach — find the key by *brute force:* possible, there are only 26 keys
  - *Key Lesson:* Have a large key space
- Second approach: Pr[M = kim] = 0.5, Pr[M = ann] = 0.2, Pr[M = boo] = 0.3
- c = dqq
- The only way this ciphertext can occur if M = ann and K = 3, or M = boo and K = 2.
- No possibility for "kim".
- Now, Pr[K = 3] = Pr [K = 1] = 1/26 — you do not know the key!!
- So, Pr[c = dqq] = Pr[M = ann]* Pr[K = 3] + Pr[M = boo]* Pr [K = 2] = 0.5*1/26 = 1/52
- Now, Pr[M = kim| c = dqq] = 0, Pr[M = ann| c = dqq] = 0.4, Pr[M = boo| c = dqq] = 0.6.

Let’s Try…

- What is the takeaway here?
  - Just observing the ciphertext, and knowing some trivial information on the plaintext distribution you can learn a lot!! — **ciphertext only attack**
  - Subtle point: the attack become a bit easy as two ciphertext characters were repeated — it exposed some pattern.
  - **Ideally, the adversary should not learn anything about the plaintext space seeing the ciphertext.**
  - Even easier attack: Let say you know some message m and corresponding ciphertext c. You can recover the key K!! — **known/chosen-plaintext attack**

The Notion of Perfect Secrecy

DEFINITION 2.3 *An encryption scheme* (Gen, Enc, Dec) *with message space* $\mathcal{M}$ *is* perfectly secret *if for every probability distribution over* $\mathcal{M}$*, every message* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$ *for which* $\Pr[C = c] > 0$*:*

$$\Pr[M = m \mid C = c] = \Pr[M = m].$$

- But what does it mean??
- No information about the message space “leaks” from the ciphertext.

The Alternative Notion of Perfect Secrecy

$$\Pr[\mathsf{Enc}_K(m) = c] = \Pr[\mathsf{Enc}_K(m') = c]$$

- Must hold true for $\forall m, m' \in \mathscr{M}, \forall c \in \mathscr{C}$, and for any key k
- Probability distribution of the plaintext does not depend on the ciphertext.
  - The distributions of the ciphertexts for m and m' are identical.
  - The ciphertext distributions are *indistinguishable*…
    - That means that the adversary learns nothing seeing the ciphertext distributions.

Is There Anything Perfect? One-time Pad

CONSTRUCTION 2.8

Fix an integer $\ell > 0$. The message space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ are all equal to $\{0,1\}^\ell$ (the set of all binary strings of length ℓ).

- Gen: the key-generation algorithm chooses a key from $\mathcal{K} = \{0,1\}^\ell$ according to the uniform distribution (i.e., each of the 2^ℓ strings in the space is chosen as the key with probability exactly $2^{-\ell}$).
- Enc: given a key $k \in \{0,1\}^\ell$ and a message $m \in \{0,1\}^\ell$, the encryption algorithm outputs the ciphertext $c := k \oplus m$.
- Dec: given a key $k \in \{0,1\}^\ell$ and a ciphertext $c \in \{0,1\}^\ell$, the decryption algorithm outputs the message $m := k \oplus c$.

Is There Anything Perfect? One-time Pad

- In OTP, for each message m', I have a key k
- For arbitrary ciphertext $c \in \mathscr{C}$ and message $m' \in \mathscr{M}$:

  Pr[C = c | M = m'] = Pr[Enc(m', K) = c] = $Pr[m' \oplus K = c] = Pr[K = c \oplus m'] = 2^{-l}$

Now $$Pr[C = c] = \sum_{m' \in \mathscr{M}} Pr[C = c \,|\, M = m'] Pr[M = m'] = 2^{-l} \sum_{m' \in \mathscr{M}} Pr[M = m'] = 2^{-l}$$

Finally, $$Pr[M = m' \,|\, C = c] = \frac{Pr[C = c \,|\, M = m'] Pr[M = m']}{Pr[C = c]} = Pr[M = m']$$

But There are Caveats

- What if I use the **same key** to encrypt m0 and m1
  - $m_0 \oplus k = c_0, m_1 \oplus k = c_1$
  - Adversary learns $m_0 \oplus m_1$
  - Suppose $m_0 = 0011, m_1 = 1100$
  - Then you basically get $m0||m1$
- You need the key as long as the message!!
  - That means if you want to communicate a movie to your friend, you first establish a key of 8 Gb with your friend!!!
  - Basically, you cannot repeat and need a key size same as message size — **inefficient!!**

But There are Caveats

THEOREM 2.10 *If* (Gen, Enc, Dec) *is a perfectly secret encryption scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$*, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

- Is the other direction of the theorem true?

Notion of Computational Security

- We need encryption algorithms which can use one key of fixed length to encrypt several plaintext
- Perfect secrecy cannot give that….So, what to do? Compromise
- Let’s say we have a magic algorithm:
  - Given a (fixed) key, it can generate uniformly random strings of arbitrary length.
  - This solves all problem!! Two parties share this fixed key, and generate the random strings as they want!!!
  - But this cannot happen in real world.

Notion of Computational Security

- We need encryption algorithms which can use one key of fixed length to encrypt several plaintext
- Perfect secrecy cannot give that….So, what to do? Compromise
- Let’s say we have a magic algorithm:
  - Given a (fixed) key, it can generate uniformly random strings of arbitrary length.
  - This solves all problem!! Two parties share this fixed key, and generate the random strings as they want!!!
  - But this cannot happen in real world. Can you see why??

Notion of Computational Security

- We need encryption algorithms which can use one key of fixed length to encrypt several plaintext
- Perfect secrecy cannot give that….So, what to do? Compromise
- Let’s say we have a magic algorithm:
  - Given a (fixed) key, it can generate random strings of arbitrary length.
  - This solves all problem!! Two parties share this fixed key, and generate the random strings as they want!!!
  - But this cannot happen in real world. — randomness cannot be controlled!!!

Notion of Computational Security

- Let's say we have a magic algorithm:
  - Given a (fixed) key, it can generate pseudorandom strings of arbitrary length.
    - $G : \{0,1\}^n \rightarrow \{0,1\}^m$ with $m > n$.
  - The strings generated are not random, but random looking
    - So, G is basically a deterministic function…
  - Also for the fixed key, both parties generate the same string
  - This really solves the problem. — But then what is this "pseudo randomness"?
    - Something indistinguishable from random!!
    - But can that really exist??
  - First, let's assume that the adversary has unlimited computational power…

Notion of Computational Security

- Let's say we have a magic algorithm:
  - Given a (fixed) key, it can generate pseudorandom strings of arbitrary length.
    - $G : \{0,1\}^n \rightarrow \{0,1\}^m$ with $m > n$.
  - The strings generated are not random, but random looking
    - So, G is basically a deterministic function…
  - Also for the fixed key, both parties generate the same string
  - This really solves the problem. — But then what is this "pseudo randomness"?
    - Something indistinguishable from random!!
    - But can that really exist??
  - First, let's assume that the adversary has unlimited computational power…
    - What happens once you generate more than 2^n keys??

Notion of Computational Security

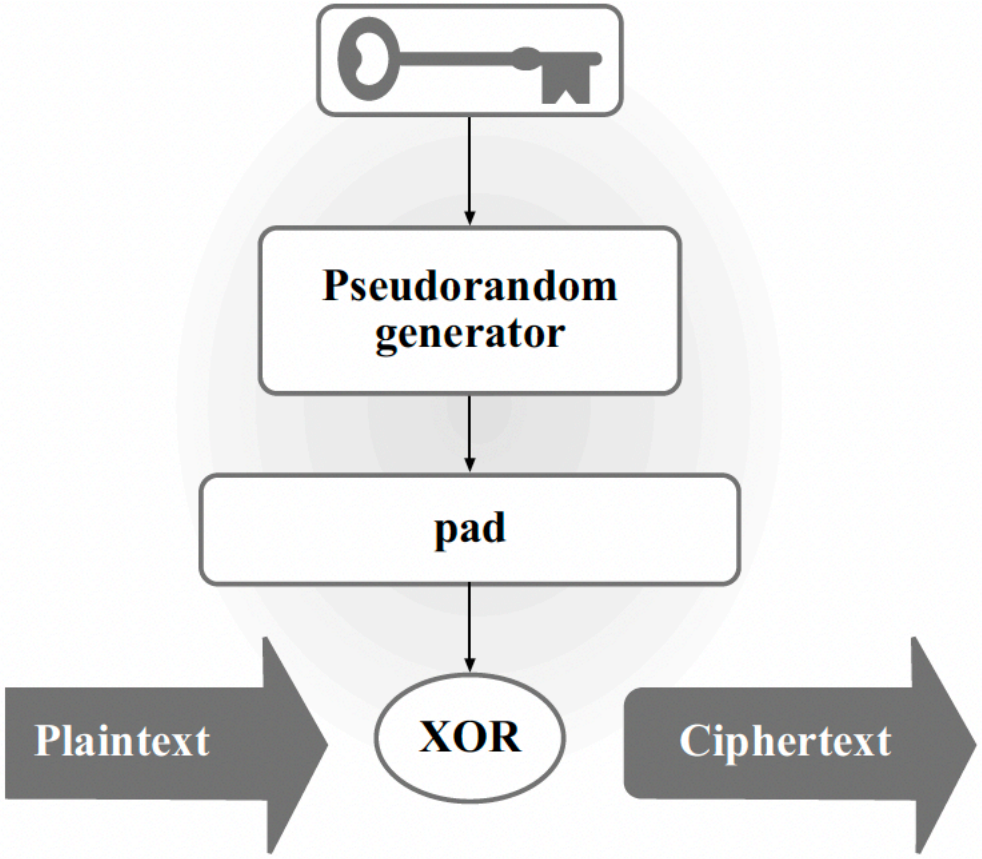
- Let's first compromise a bit with the security:
  - The adversary a polynomial time algorithm — so, it cannot generate more than 2^n strings if *n* is large.
  - *n is called the security parameter.*
  - Negligible function (*negl(n)*): For large values of n, the function is smaller than all 1/p(n), where p(n) is any polynomial function in *n*.
  - We say that the probability of distinguishing an output of G from a truly random string is negligibly small.

Notion of Pseudorandom Generator

- Now we are in the position of defining a pseudorandom generator (PRG)
  - It takes a bit string s of length n.
  - It outputs a bit string of length l(n) > n
  - What the adversary *D* does?
    - It tries to distinguish this string from a random string of the same length.
      - Given a string — either a string output of G, or a purely random string
      - Outputs 1, when it can correctly identify if it G's output or a random string r.
    - *But for PRG :*

$$\left| \Pr[D(G(s)) = 1] - \Pr[D(r) = 1] \right| \leq \mathsf{negl}(n),$$

Pseudorandom Generator to Secure Encryption



Notion of Pseudorandom Function

- PRGs can give you the indistinguishability…But then…
  - The state of the PRGs important to maintain at both side of communication. — PRGs are stateful.
    - Practical PRGs — **stream ciphers**
  - That means perfect synchronisation…
    - At both side you should be generating the same pseudorandom string at the same time.
    - *Can we get rid of this issue?*
    - *Yes,* but we need another theoretical object — *Pseudorandom Function (PRF)*

Notion of Pseudorandom Function

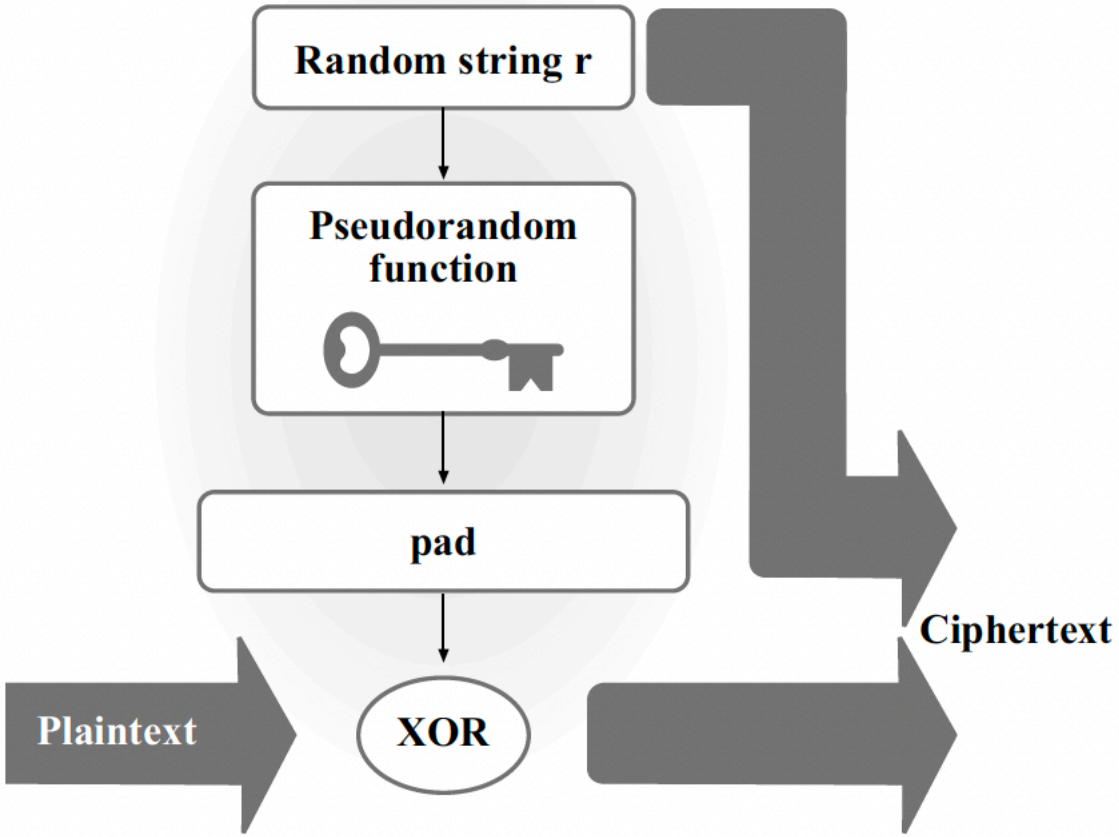
- PRF is a generalisation of PRGs. Now you select from the space of random-looking functions
  - Practical PRGs — **stream ciphers**
  - That means perfect synchronisation…
    - Can we get rid of this issue?
    - Yes, but we need another theoretical object — *Pseudorandom Function (PRF)*
    - *The idea is simple — it is a function that cannot be distinguished from a random function given its input output behaviour.*

DEFINITION 3.25 *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function.* F *is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers* D*, there is a negligible function* negl *such that:*

$$\left| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \right| \leq \mathsf{negl}(n),$$

where the first probability is taken over uniform choice of $k \in \{0,1\}^n$ *and the randomness of* D*, and the second probability is taken over uniform choice of* $f \in \mathsf{Func}_n$ *and the randomness of* D*.*

What Can be Achieved with Pseudorandom Functions



How to Practically Realise Pseudorandom Functions

- Many ways exist…But the practical way is to use a **block cipher**
  - Its basically a permutation — encrypt n bit plaintexts to n-bit ciphertexts.
  - The key is always the same.
  - **Block cipher is instrumental in developing other stronger security notions too —like chosen ciphertext security.**
  - **You can also realise Hash functions with block ciphers, to achieve integrity.**
  - **Even for public key cryptography, you need to have pseudorandom functions or hash functions — so these are one of the most fundamental building blocks of cryptography.**

Finally, …

- We shall make and break these block ciphers
  - If block cipher breaks —
    - You can violate the assumptions of pseudorandom functions
      - You break your theory crypto
- Next steps…
  - We shall see how to implement these block ciphers
  - How to break them with practical attacks

References

- *Introduction to Modern Cryptography*, Jonathan Katz and Yehuda Lindell
  - Chapter 2, 3 (up to 3.5)