Basics of Provable Security (II) & Computational Intractability

601.642/442: Modern Cryptography

Fall 2020

OTP is One-time Uniform Ciphertext Secure

Lemma

One-Time Pad encryption scheme satisfies uniform ciphertext security.

OTP is One-time Uniform Ciphertext Secure

Lemma

One-Time Pad encryption scheme satisfies uniform ciphertext security.

Proof. Given a fixed plaintext m and a fixed ciphertext c, we calculate the probability that c is the encryption of m.

$$\Pr[c = \mathsf{Enc}(k, m)] = \Pr[c = m \oplus k] = \Pr[k = m \oplus c]$$

- Note that the probability here is over the random choice of $k \in \{0,1\}^n$.

OTP is One-time Uniform Ciphertext Secure

Lemma

One-Time Pad encryption scheme satisfies uniform ciphertext security.

Proof. Given a fixed plaintext m and a fixed ciphertext c, we calculate the probability that c is the encryption of m.

$$\Pr[c = \mathsf{Enc}(k, m)] = \Pr[c = m \oplus k] = \Pr[k = m \oplus c]$$

- Note that the probability here is over the random choice of $k \in \{0,1\}^n$.
- Since k is chosen uniformly at random, the probability of choosing a $k = m \oplus c$ is $1/2^n$.

OTP is One-time Uniform Ciphertext Secure

Lemma

One-Time Pad encryption scheme satisfies uniform ciphertext security.

Proof. Given a fixed plaintext m and a fixed ciphertext c, we calculate the probability that c is the encryption of m.

$$\Pr[c = \mathsf{Enc}(k, m)] = \Pr[c = m \oplus k] = \Pr[k = m \oplus c]$$

- Note that the probability here is over the random choice of $k \in \{0,1\}^n$.
- Since k is chosen uniformly at random, the probability of choosing a $k = m \oplus c$ is $1/2^n$.
- In other words, for a given m, $\Pr[c = \mathsf{Enc}(k, m)] = 1/2^n$.

OTP is One-time Uniform Ciphertext Secure

Lemma

One-Time Pad encryption scheme satisfies uniform ciphertext security.

Proof. Given a fixed plaintext m and a fixed ciphertext c, we calculate the probability that c is the encryption of m.

$$\Pr[c = \mathsf{Enc}(k, m)] = \Pr[c = m \oplus k] = \Pr[k = m \oplus c]$$

- Note that the probability here is over the random choice of $k \in \{0,1\}^n$.
- Since k is chosen uniformly at random, the probability of choosing a $k = m \oplus c$ is $1/2^n$.
- In other words, for a given m, $\Pr[c = \mathsf{Enc}(k, m)] = 1/2^n$.
- Hence, the ciphertexts are uniformly distributed.

The Hybrid Technique

Example (Double OTP)

Prove uniform ciphertext security of the following scheme:

- $\mathsf{KeyGen}(1^n) : k_1 \xleftarrow{\$} \{0,1\}^n$, $k_2 \xleftarrow{\$} \{0,1\}^n$ and output (k_1, k_2)
- $\mathsf{Enc}((k_1, k_2), m) : c_1 = k_1 \oplus m$, $c_2 = k_2 \oplus c_1$ and output c_2.
- $\mathsf{Dec}((k_1, k_2), c) : c_1 = k_2 \oplus c$, $m = k_1 \oplus c_1$ and output m.

The Hybrid Technique

Example (Double OTP)

Prove uniform ciphertext security of the following scheme:

- $\mathsf{KeyGen}(1^n) : k_1 \xleftarrow{\$} \{0,1\}^n$, $k_2 \xleftarrow{\$} \{0,1\}^n$ and output (k_1, k_2)
- $\mathsf{Enc}((k_1, k_2), m) : c_1 = k_1 \oplus m$, $c_2 = k_2 \oplus c_1$ and output c_2.
- $\mathsf{Dec}((k_1, k_2), c) : c_1 = k_2 \oplus c$, $m = k_1 \oplus c_1$ and output m.

We need to show that for each m, the following distributions are identical:

1. $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$
2. $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

Proving Security using the Hybrid Technique

We consider the following set of distributions called **hybrids**.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

Proving Security using the Hybrid Technique

We consider the following set of distributions called **hybrids**.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

Our goal is to show that $\mathcal{H}_1$ and $\mathcal{H}_3$ are identical distributions. We will do this in two steps by using the "intermediate" hybrid $\mathcal{H}_2$.

Proving Security using the Hybrid Technique

We consider the following intermediate distributions called hybrids.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

$\mathcal{H}_1$ is identical to $\mathcal{H}_2$ because of the uniform ciphertext security of OTP.

Proving Security using the Hybrid Technique

We consider the following intermediate distributions called hybrids.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

$\mathcal{H}_2$ and $\mathcal{H}_3$ are trivially identical.

Proving Security using the Hybrid Technique

We consider the following intermediate distributions called hybrids.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \xleftarrow{\$} \{0,1\}^n\right\}$

$\mathcal{H}_2$ and $\mathcal{H}_3$ are trivially identical.

By transitivity, we have that $\mathcal{H}_1$ and $\mathcal{H}_3$ are also identical!

Proving Security using the Hybrid Technique

We consider the following intermediate distributions called hybrids.

$\mathcal{H}_1$: $\{c_2 = k_2 \oplus c_1; k_1 \leftarrow \mathsf{KeyGen}(1^n), k_2 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\}$

$\mathcal{H}_2$: $\left\{c_2 \stackrel{\$}{\leftarrow} \{0,1\}^n; k_1 \leftarrow \mathsf{KeyGen}(1^n), c_1 = k_1 \oplus m\right\}$

$\mathcal{H}_3$: $\left\{c_2 \stackrel{\$}{\leftarrow} \{0,1\}^n\right\}$

$\mathcal{H}_2$ and $\mathcal{H}_3$ are trivially identical.

By transitivity, we have that $\mathcal{H}_1$ and $\mathcal{H}_3$ are also identical!

The Hybrid Technique is very common in cryptographic proofs and we will see it again and again throughout the course.

Encryption: One-Time Perfect Security

Lets consider an alternate idea of security for encryption schemes.

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext.
- ~~The ciphertexts look like random values to Eve.~~
- Encryptions of m_0 look like encryptions of m_1 to Eve.

Encryption: One-Time Perfect Security

Lets consider an alternate idea of security for encryption schemes.

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext.
- ~~The ciphertexts look like random values to Eve.~~
- Encryptions of m_0 look like encryptions of m_1 to Eve.
- Eve is allowed to choose both m_0 and m_1.

Encryption: One-Time Perfect Security

Lets consider an alternate idea of security for encryption schemes.

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext.
- ~~The ciphertexts look like random values to Eve.~~
- Encryptions of m_0 look like encryptions of m_1 to Eve.
- Eve is allowed to choose both m_0 and m_1.

An encryption scheme is a good one if encryptions of m_0 look like encryptions of m_1 to Eve, when each key is secret and used to encrypt only one plaintext, even when Eve chooses both m_0 and m_1.

Encryption: One-Time Perfect Security

One-Time Perfect Security

We say that an encryption scheme is one-time perfectly secure if $\forall m_0, m_1 \in \mathcal{M}$ chosen by Eve, the following distributions are identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Encryption: One-Time Perfect Security

One-Time Perfect Security

We say that an encryption scheme is one-time perfectly secure if $\forall m_0, m_1 \in \mathcal{M}$ chosen by Eve, the following distributions are identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

As earlier, from adversary's viewpoint, the ciphertext carries no information about the plaintext.

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme is does not satisfy one-time perfect security, if $\exists m_0, m_1 \in \mathcal{M}$, such that the following distributions are not identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme is does not satisfy one-time perfect security, if $\exists m_0, m_1 \in \mathcal{M}$, such that the following distributions are not identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Example

Is the following encryption scheme secure?

- $\mathsf{KeyGen}(1^n) := k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Enc}(k, m) := c = k \wedge m$

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme is does not satisfy one-time perfect security, if $\exists m_0, m_1 \in \mathcal{M}$, such that the following distributions are not identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Example

Is the following encryption scheme secure?

- $\mathsf{KeyGen}(1^n) := k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Enc}(k, m) := c = k \wedge m$

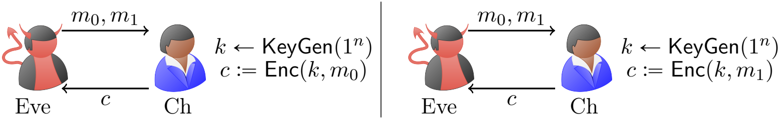
For $m_0 = 0^n$, $m_1 = 1^n$

$$\Pr[c = 0^n | \mathcal{D}_1] = 1$$
$$\Pr[c = 0^n | \mathcal{D}_2] = 1/2^n$$

Clearly the two distributions are not identical in this case.

Encryption: One-Time Perfect Security

Consider the following two interactions between Eve and a challenger.



- Interaction with a challenger helps us model what Eve can see during encryption, and what remains hidden.
- We say that an encryption scheme is secure if for any (m_0, m_1) chosen by Eve, the above two scenarios seem identical to Eve.

Comparing Both Security Notions

Theorem

If an encryption scheme achieves one-time uniform ciphertext security, then it also achieves one-time perfect security.

Comparing Both Security Notions

Theorem

If an encryption scheme achieves one-time uniform ciphertext security, then it also achieves one-time perfect security.

We are given that for each $m \in \mathcal{M}$ (where $\mathcal{M}$ is the message space), the following distributions are identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \left\{c \xleftarrow{\$} \mathcal{C}\right\}$

Comparing Both Security Notions

Theorem

If an encryption scheme achieves one-time uniform ciphertext security, then it also achieves one-time perfect security.

We are given that for each $m \in \mathcal{M}$ (where $\mathcal{M}$ is the message space), the following distributions are identical:

1. $\mathcal{D}_1 := \{c := \mathsf{Enc}(k, m); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}_2 := \left\{c \xleftarrow{\$} \mathcal{C}\right\}$

We want to show that for each $m_0, m_1 \in \mathcal{M}$, the following distributions are also identical:

1. $\mathcal{D}'_1 := \{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$
2. $\mathcal{D}'_2 := \{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Comparing Both Security Notions

Proof (Using hybrid technique): Consider the following distributions:

$\mathcal{H}_1$: $\{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$

$\mathcal{H}_2$: $\left\{c \xleftarrow{\$} \mathcal{C}\right\}$

$\mathcal{H}_3$: $\{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

Comparing Both Security Notions

Proof (Using hybrid technique): Consider the following distributions:

$\mathcal{H}_1$: $\{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$

$\mathcal{H}_2$: $\left\{c \xleftarrow{\$} \mathcal{C}\right\}$

$\mathcal{H}_3$: $\{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

- $\mathcal{H}_1 \equiv \mathcal{H}_2$: because of one-time uniform ciphertext security.

Comparing Both Security Notions

Proof (Using hybrid technique): Consider the following distributions:

$\mathcal{H}_1$: $\{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$

$\mathcal{H}_2$: $\left\{c \stackrel{\$}{\leftarrow} \mathcal{C}\right\}$

$\mathcal{H}_3$: $\{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

- $\mathcal{H}_1 \equiv \mathcal{H}_2$: because of one-time uniform ciphertext security.
- $\mathcal{H}_2 \equiv \mathcal{H}_3$: because of one-time uniform ciphertext security.

Comparing Both Security Notions

Proof (Using hybrid technique): Consider the following distributions:

$\mathcal{H}_1$: $\{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$

$\mathcal{H}_2$: $\left\{c \xleftarrow{\$} \mathcal{C}\right\}$

$\mathcal{H}_3$: $\{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

- $\mathcal{H}_1 \equiv \mathcal{H}_2$: because of one-time uniform ciphertext security.
- $\mathcal{H}_2 \equiv \mathcal{H}_3$: because of one-time uniform ciphertext security.
- Hence $\mathcal{H}_1 \equiv \mathcal{H}_3$

Comparing Both Security Notions

Proof (Using hybrid technique): Consider the following distributions:

$\mathcal{H}_1$: $\{c := \mathsf{Enc}(k, m_0); k \leftarrow \mathsf{KeyGen}(1^n)\}$

$\mathcal{H}_2$: $\left\{c \stackrel{\$}{\leftarrow} \mathcal{C}\right\}$

$\mathcal{H}_3$: $\{c := \mathsf{Enc}(k, m_1); k \leftarrow \mathsf{KeyGen}(1^n)\}$

- $\mathcal{H}_1 \equiv \mathcal{H}_2$: because of one-time uniform ciphertext security.
- $\mathcal{H}_2 \equiv \mathcal{H}_3$: because of one-time uniform ciphertext security.
- Hence $\mathcal{H}_1 \equiv \mathcal{H}_3$

Corollary

One-time pad satisfies one-time perfect security.

Comparing Both Security Notions

Theorem

One-time perfect security does not necessarily imply one-time uniform ciphertext security.

Comparing Both Security Notions

Theorem

One-time perfect security does not necessarily imply one-time uniform ciphertext security.

In other words, there exists an encryption scheme that satisfies one-time perfect security but not one-time uniform ciphertext security.

Comparing Both Security Notions

Theorem

One-time perfect security does not necessarily imply one-time uniform ciphertext security.

In other words, there exists an encryption scheme that satisfies one-time perfect security but not one-time uniform ciphertext security.

Proof. Consider the following encryption scheme.

- $\mathsf{KeyGen}(1^n) : k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Compute $c' = k \oplus m$ and output $c = c'||00$
- $\mathsf{Dec}(k,c)$: Compute $c' = c[0:n]$ and output $m = k \oplus c'$.

Comparing Both Security Notions

Theorem

One-time perfect security does not necessarily imply one-time uniform ciphertext security.

In other words, there exists an encryption scheme that satisfies one-time perfect security but not one-time uniform ciphertext security.

Proof. Consider the following encryption scheme.

- $\mathsf{KeyGen}(1^n) : k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Enc}(k,m)$: Compute $c' = k \oplus m$ and output $c = c'||00$
- $\mathsf{Dec}(k,c)$: Compute $c' = c[0:n]$ and output $m = k \oplus c'$.

Does this scheme satisfy one-time perfect security? Why?

Comparing Both Security Notions

Theorem

One-time perfect security does not necessarily imply one-time uniform ciphertext security.

In other words, there exists an encryption scheme that satisfies one-time perfect security but not one-time uniform ciphertext security.

Proof. Consider the following encryption scheme.

- $\mathsf{KeyGen}(1^n) : k \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{Enc}(k, m)$: Compute $c' = k \oplus m$ and output $c = c' || 00$
- $\mathsf{Dec}(k, c)$: Compute $c' = c[0 : n]$ and output $m = k \oplus c'$.

Does this scheme satisfy one-time perfect security? Why?

Does it also satisfy one-time uniform ciphertext security? Why not?

Observations

- Uniform ciphertext security might be too strong.

Observations

- Uniform ciphertext security might be too strong.
- Limitations of one-time pad

Observations

- Uniform ciphertext security might be too strong.
- Limitations of one-time pad
  - Key must be as long as plaintext (necessary for perfect security). Think Why?

Observations

- Uniform ciphertext security might be too strong.
- Limitations of one-time pad
  - Key must be as long as plaintext (necessary for perfect security). Think Why?
  - A key cannot be used to encrypt more than one plaintext (see HW1).

Computational Intractability

“Real” Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.

"Real" Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).

“Real” Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).
- How much computation is needed for a brute force attack on n-bit keys? 2^n

"Real" Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).
- How much computation is needed for a brute force attack on n-bit keys? 2^n
- In principle, the "real" security of OTP depends on how costly 2^n is.

"Real" Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).
- How much computation is needed for a brute force attack on n-bit keys? 2^n
- In principle, the "real" security of OTP depends on how costly 2^n is.
- Recall: n- length of the key is called the **security parameter**.

“Real” Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).
- How much computation is needed for a brute force attack on n-bit keys? 2^n
- In principle, the “real” security of OTP depends on how costly 2^n is.
- Recall: n- length of the key is called the **security parameter**.

"Real" Security of One-time pad

- One-time pad remains secure only until the key is kept hidden.
- It can be broken by Eve (or any **adversary**) if it performs a **brute force attack**, i.e., tries all possible keys (assume that the adversary has some a priori information on the message so that he knows when he is successful).
- How much computation is needed for a brute force attack on n-bit keys? 2^n
- In principle, the "real" security of OTP depends on how costly 2^n is.
- Recall: n- length of the key is called the **security parameter**.

Why?

It is like a knob that allows the user to tune the security to any desired level. Increasing n makes the difficulty of a brute force attack grow exponentially fast.

Cost of Computation

- How should we *measure the cost* of 2^n computations?

Cost of Computation

- How should we *measure the cost* of 2^n computations?
- It can be helpful to think of the cost of a computation in terms of monetary value, and a convenient way to assign such monetary costs is to use the pricing model of a cloud computing provider (e.g. Amazon EC2).

Cost of Computation

- How should we *measure the cost* of 2^n computations?
- It can be helpful to think of the cost of a computation in terms of monetary value, and a convenient way to assign such monetary costs is to use the pricing model of a cloud computing provider (e.g. Amazon EC2).

Cost of Computation

- How should we *measure the cost* of 2^n computations?
- It can be helpful to think of the cost of a computation in terms of monetary value, and a convenient way to assign such monetary costs is to use the pricing model of a cloud computing provider (e.g. Amazon EC2).

CPU Cycles	Approx Cost	Reference
2^{50}	\$3.50	cup of coffee
2^{55}	\$100	tickets to a Portland Trailblazers game
2^{65}	$\$130,000$	median home price in Oshkosh, WI
2^{75}	\$130 million	budget of one of the Harry Potter movies
2^{92}	\$20 trillion	GDP of the United States
2^{99}	\$2 quadrillion	All human economic activity since 300,000 BC
2^{128}	A lot!!	a billion human civilizations' worth of effort

Cost of Computation

- How should we *measure the cost* of 2^n computations?
- It can be helpful to think of the cost of a computation in terms of monetary value, and a convenient way to assign such monetary costs is to use the pricing model of a cloud computing provider (e.g. Amazon EC2).

CPU Cycles	Approx Cost	Reference
2^{50}	\$3.50	cup of coffee
2^{55}	\$100	tickets to a Portland Trailblazers game
2^{65}	\$130,000	median home price in Oshkosh, WI
2^{75}	\$130 million	budget of one of the Harry Potter movies
2^{92}	\$20 trillion	GDP of the United States
2^{99}	\$2 quadrillion	All human economic activity since 300,000 BC
2^{128}	A lot!!	a billion human civilizations' worth of effort

2^{128} already seems like a lot.

Must we use (say) a 500-bit key to encrypt 500-bit messages, as in one-time pad? Or can we somehow use a smaller (say 128-bit) key to encrypt long messages and still get meaningful security?

Computational Infeasibiity

In 1955, Nobel laureate John Nash during his correspondence with the NSA said something to the effect of the following:

*"It doesn't really matter whether attacks are **impossible**, only whether attacks are **computationally infeasible**."*

Computational Infeasibiity

In 1955, Nobel laureate John Nash during his correspondence with the NSA said something to the effect of the following:

*"It doesn't really matter whether attacks are **impossible**, only whether attacks are **computationally infeasible**."*

- "Modern" cryptography is based on this principle, where security is based on intractable computations.

Computational Infeasibiity

In 1955, Nobel laureate John Nash during his correspondence with the NSA said something to the effect of the following:

*"It doesn't really matter whether attacks are **impossible**, only whether attacks are **computationally infeasible**."*

- "Modern" cryptography is based on this principle, where security is based on intractable computations.
- If his letters hadn't been kept classified until 2012, they might have accelerated the development of modern cryptography.

Asymptotic Cost of an Attack

- Thinking about the monetary cost of an enormous computation gives us an intuitive idea. Is that sufficient?

Asymptotic Cost of an Attack

- Thinking about the monetary cost of an enormous computation gives us an intuitive idea. Is that sufficient?

Asymptotic Cost of an Attack

- Thinking about the monetary cost of an enormous computation gives us an intuitive idea. Is that sufficient?No!

Asymptotic Cost of an Attack

- Thinking about the monetary cost of an enormous computation gives us an intuitive idea. Is that sufficient?No!

  We only looked at the retail cost of performing computation. A large organization (say a government) could be capable of manufacturing special-purpose hardware that could significantly reduce the computation's cost

Asymptotic Cost of an Attack

- Thinking about the monetary cost of an enormous computation gives us an intuitive idea. Is that sufficient?No!

  We only looked at the retail cost of performing computation. A large organization (say a government) could be capable of manufacturing special-purpose hardware that could significantly reduce the computation's cost

- In order to make security definitions that say *only feasible attacks are ruled out*, we need a concrete way to draw the line between **feasible attacks** (which we want to protect against) and **infeasible attacks** (which we agreed we don't need to care about).

Asymptotic Cost of an Attack

- **A Good measure**: *How does the running time of computation scale as the input length goes to infinity?*

Asymptotic Cost of an Attack

- **A Good measure**: *How does the running time of computation scale as the input length goes to infinity?*
- As we saw, exponential-time (e.g., $\Theta(2^n)$) brute-force attacks do not scale well.

Asymptotic Cost of an Attack

- **A Good measure**: *How does the running time of computation scale as the input length goes to infinity?*
- As we saw, exponential-time (e.g., $\Theta(2^n)$) brute-force attacks do not scale well.
- But polynomial-time algorithms do (especially if exponent is small). Recall, they are also called efficient algorithms.

Asymptotic Cost of an Attack

- **A Good measure**: *How does the running time of computation scale as the input length goes to infinity?*
- As we saw, exponential-time (e.g., $\Theta(2^n)$) brute-force attacks do not scale well.
- But polynomial-time algorithms do (especially if exponent is small). Recall, they are also called efficient algorithms.
- Disclaimer: Using polynomial time as a synonym for efficient might be confusing since, e.g., $\Theta(n^{1000})$ is poly-time and $\Theta(n^{\log\log n})$ is not, yet the latter might be more "efficient" in practice.

Asymptotic Cost of an Attack

- **A Good measure**: *How does the running time of computation scale as the input length goes to infinity?*
- As we saw, exponential-time (e.g., $\Theta(2^n)$) brute-force attacks do not scale well.
- But polynomial-time algorithms do (especially if exponent is small). Recall, they are also called efficient algorithms.
- Disclaimer: Using polynomial time as a synonym for efficient might be confusing since, e.g., $\Theta(n^{1000})$ is poly-time and $\Theta(n^{\log\log n})$ is not, yet the latter might be more "efficient" in practice.
- Nevertheless, the reason why polynomial time is very useful is because of **closure property**: *repeating a poly-time algorithm polynomial times is still polynomial time!*

Some Examples

Efficient Algorithms known	**Efficient Algorithms not known**
Computing GCDs	Factoring Integers
Arithmetic mod N	Discrete Logarithm
Inverses mod N	Square roots mod composite N
Exponentiation mod N	Solving “noisy” linear equations

Some Examples

Efficient Algorithms known	**Efficient Algorithms not known**
Computing GCDs	Factoring Integers
Arithmetic mod N	Discrete Logarithm
Inverses mod N	Square roots mod composite N
Exponentiation mod N	Solving "noisy" linear equations

- Later in this course, we will use the fact that no efficient algorithm is known for some of the above examples, to construct cryptographic primitives.

Some Examples

Efficient Algorithms known	**Efficient Algorithms not known**
Computing GCDs	Factoring Integers
Arithmetic mod N	Discrete Logarithm
Inverses mod N	Square roots mod composite N
Exponentiation mod N	Solving "noisy" linear equations

- Later in this course, we will use the fact that no efficient algorithm is known for some of the above examples, to construct cryptographic primitives.
- In this class, we will mostly focus on algorithms on classical computers. Indeed, even in the second category, most problems, except last one are known to have efficient quantum algorithms.

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.
- For example, consider an attacker who just tries to guess a victim's secret key, making a single guess. This attack is extremely cheap, but it still has a nonzero chance (e.g., 2^{-128}) of breaking security.

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.
- For example, consider an attacker who just tries to guess a victim's secret key, making a single guess. This attack is extremely cheap, but it still has a nonzero chance (e.g., 2^{-128}) of breaking security.
- Essentially, for security, we don't need to worry about the following:

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.
- For example, consider an attacker who just tries to guess a victim's secret key, making a single guess. This attack is extremely cheap, but it still has a nonzero chance (e.g., 2^{-128}) of breaking security.
- Essentially, for security, we don't need to worry about the following:
  - Attacks that are as expensive as a brute-force attack.

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.
- For example, consider an attacker who just tries to guess a victim's secret key, making a single guess. This attack is extremely cheap, but it still has a nonzero chance (e.g., 2^{-128}) of breaking security.
- Essentially, for security, we don't need to worry about the following:
  - Attacks that are as expensive as a brute-force attack.
  - Attacks whose success probability is as low as a blind-guess attack.

Success Probability of an Attack

- It is not enough to consider only the running time of an attack.
- For example, consider an attacker who just tries to guess a victim's secret key, making a single guess. This attack is extremely cheap, but it still has a nonzero chance (e.g., 2^{-128}) of breaking security.
- Essentially, for security, we don't need to worry about the following:
  - Attacks that are as expensive as a brute-force attack.
  - Attacks whose success probability is as low as a blind-guess attack.
- While an attack with success probability 2^{-128} should not really count as an attack, one with success probability $1/2$ should. Where should we draw the line?