One-Time Pad & Basics of Provable Security (I)

601.642/442: Modern Cryptography

Fall 2020

Today's Agenda

- Private Communication via One-Time Pads
- Basics of Provable Security Approach

Before we begin...

A few remarks

Before we begin...

A few remarks

- Ask questions!

Before we begin...

A few remarks

- Ask questions!
    - As many as you want

Before we begin...

A few remarks

- Ask questions!
  - As many as you want
  - Whenever you want

Before we begin...

A few remarks

- Ask questions!
  - As many as you want
  - Whenever you want
- Do not be fazed by seeming complexity (or seeming simplicity) of a cryptographic scheme

Before we begin...

A few remarks

- Ask questions!
    - As many as you want
    - Whenever you want
- Do not be fazed by seeming complexity (or seeming simplicity) of a cryptographic scheme
    - Complex-looking things may not be that complex. Simple-looking things may not be that simple.

Before we begin...

A few remarks

- Ask questions!
  - As many as you want
  - Whenever you want
- Do not be fazed by seeming complexity (or seeming simplicity) of a cryptographic scheme
  - Complex-looking things may not be that complex. Simple-looking things may not be that simple.
  - What matters is the underlying idea. Always ask "why?"

Before we begin...

A few remarks

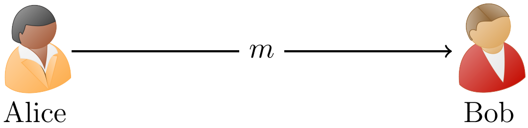
- Ask questions!
  - As many as you want
  - Whenever you want
- Do not be fazed by seeming complexity (or seeming simplicity) of a cryptographic scheme
  - Complex-looking things may not be that complex. Simple-looking things may not be that simple.
  - What matters is the underlying idea. Always ask "why?"
- View definitions as constructs. They are meant to capture the "rules" of the "game". Think about alternatives.

Before we begin...

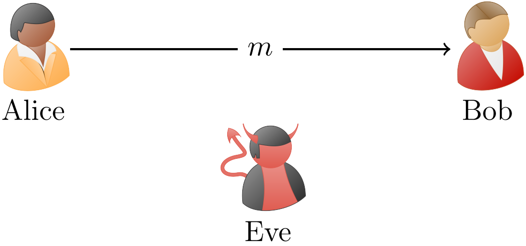
A few remarks

- Ask questions!
  - As many as you want
  - Whenever you want
- Do not be fazed by seeming complexity (or seeming simplicity) of a cryptographic scheme
  - Complex-looking things may not be that complex. Simple-looking things may not be that simple.
  - What matters is the underlying idea. Always ask "why?"
- View definitions as constructs. They are meant to capture the "rules" of the "game". Think about alternatives.
- Sometimes, the intuition may seem to not align with the proof. But eventually it will, once we make the intuition *robust*.

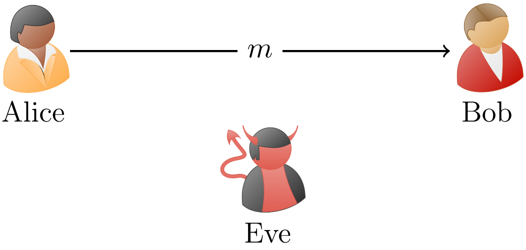
Private Communication



Private Communication

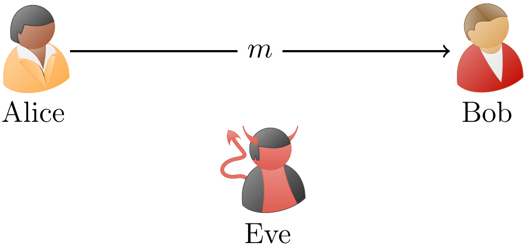


Private Communication



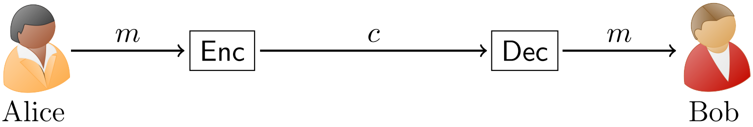
- Alice wants to send a message m to Bob.

Private Communication



- Alice wants to send a message m to Bob.
- How can Alice convey this message to Bob, while keeping it hidden from an eavesdropper Eve?

Encryption



- **Encryption:** Alice transforms (encrypts) the message m – also called the "plaintext" – into a "ciphertext" c.
- **Communication:** Alice sends c to Bob.
- **Decryption:** Bob recovers (decrypts) the original m from c.

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

No!

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

No! Why not?

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

No! Why not?

- If Eve eventually learns the details of Enc and Dec, we will have to *invent new algorithms*.

Kerckhoffs' Principle

- Goal: Bob should be able to decrypt c, but not Eve.
- To enable that, something must be kept secret from Eve.
- Should we keep the details of the Enc and Dec algorithms secret from Eve?

No! Why not?

- If Eve eventually learns the details of Enc and Dec, we will have to *invent new algorithms*.
- Inventing good encryption algorithms is not an easy task.

Kerckhoffs' Principle

Kerckhoffs' Principle:

Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

Kerckhoffs' Principle

Kerckhoffs' Principle:

Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

- If the algorithms themselves are not secret, then there must be some other secret information in the system.

Kerckhoffs' Principle

Kerckhoffs' Principle:

Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

- If the algorithms themselves are not secret, then there must be some other secret information in the system.
- That information is called the **secret key**.

Kerckhoffs' Principle

Kerckhoffs' Principle:

Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

- If the algorithms themselves are not secret, then there must be some other secret information in the system.
- That information is called the **secret key**.
- If a secret key gets compromised, we only need to choose a new one, not reinvent a new encryption algorithm.

Kerckhoffs' Principle

Kerckhoffs' Principle:

Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

- If the algorithms themselves are not secret, then there must be some other secret information in the system.
- That information is called the **secret key**.
- If a secret key gets compromised, we only need to choose a new one, not reinvent a new encryption algorithm.
- Put differently, the security of the system is concentrated in the secrecy of the key, not the algorithm.

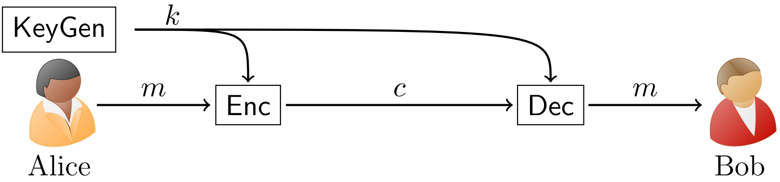
Kerckhoffs' Principle

Kerckhoffs' Principle:

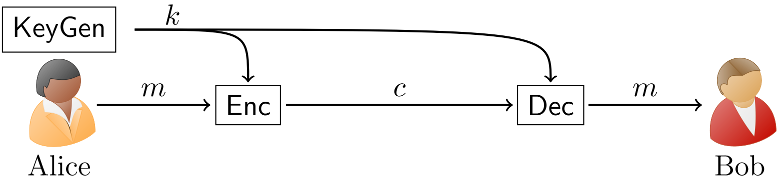
Design your system to be secure even if the attacker has complete knowledge of all its algorithms.

- If the algorithms themselves are not secret, then there must be some other secret information in the system.
- That information is called the **secret key**.
- If a secret key gets compromised, we only need to choose a new one, not reinvent a new encryption algorithm.
- Put differently, the security of the system is concentrated in the secrecy of the key, not the algorithm.
- The *length* of the secret key is called the **security parameter**.

Encryption: Syntax

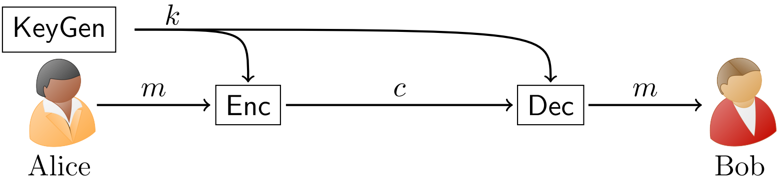


Encryption: Syntax



- **Key Generation:** KeyGen is the *randomized* process of choosing a secret key k, which is used as input to *both* Enc and Dec algorithms.

Encryption: Syntax



- **Key Generation:** KeyGen is the *randomized* process of choosing a secret key k, which is used as input to *both* Enc and Dec algorithms.

Encryption: Syntax

An **encryption scheme** consists of the following three algorithms:

- $\mathsf{KeyGen}(1^n) \to k$
- $\mathsf{Enc}(k, m) \to c$
- $\mathsf{Dec}(k, c) \to m$

What are we (*not*) trying to do?

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)
- We are not guaranteeing that Bob will necessarily receive the ciphertext.

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)
- We are not guaranteeing that Bob will necessarily receive the ciphertext.
- We are assuming that Eve is *passive* and cannot tamper with communication. Later in the course, we will consider *active* Eve.

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)
- We are not guaranteeing that Bob will necessarily receive the ciphertext.
- We are assuming that Eve is *passive* and cannot tamper with communication. Later in the course, we will consider *active* Eve.
- We are, for now, ignoring the issue of how Alice and Bob get to share a secret key in the first place. Later in the course, we will look at *key exchange*.

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)
- We are not guaranteeing that Bob will necessarily receive the ciphertext.
- We are assuming that Eve is *passive* and cannot tamper with communication. Later in the course, we will consider *active* Eve.
- We are, for now, ignoring the issue of how Alice and Bob get to share a secret key in the first place. Later in the course, we will look at *key exchange*.
- We are assuming that keys can be kept *private* in a reliable manner. We are not discussing how to do key management.

What are we (*not*) trying to do?

- We are not trying to hide the *existence* of private communication (i.e, Steganography)
- We are not guaranteeing that Bob will necessarily receive the ciphertext.
- We are assuming that Eve is *passive* and cannot tamper with communication. Later in the course, we will consider *active* Eve.
- We are, for now, ignoring the issue of how Alice and Bob get to share a secret key in the first place. Later in the course, we will look at *key exchange*.
- We are assuming that keys can be kept *private* in a reliable manner. We are not discussing how to do key management.
- We are assuming that all users have the ability to generate random bits. Will discuss randomness generation and usage in more detail later in the course.

One-Time Pad

- One-time pad (OTP) also called *Vernam's cipher* is a simple encryption scheme where secret keys, plaintexts and ciphertexts are all strings of same length.

One-Time Pad

- One-time pad (OTP) also called *Vernam's cipher* is a simple encryption scheme where secret keys, plaintexts and ciphertexts are all strings of same length.

One-Time Pad: Construction

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

- Recall: $k \xleftarrow{\$} \{0,1\}^n$ refers to sampling k uniformly at random from the set of all n-bit strings.

One-Time Pad: Correctness

- **Correctness (Intuitive):** Does the receiver (Bob) recover the intended plaintext when decrypting the ciphertext?

One-Time Pad: Correctness

- **Correctness (Intuitive):** Does the receiver (Bob) recover the intended plaintext when decrypting the ciphertext?

Claim

For all $k, m \in \{0,1\}^n$, it holds that $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$.

One-Time Pad: Correctness

- **Correctness (Intuitive):** Does the receiver (Bob) recover the intended plaintext when decrypting the ciphertext?

Claim

For all $k, m \in \{0,1\}^n$, it holds that $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m$.

Proof. For all $k, m \in \{0,1\}^n$, we have:

$$\begin{aligned} \mathsf{Dec}(k, \mathsf{Enc}(k, m)) &= \mathsf{Dec}(k, (k \oplus m)) \\ &= k \oplus (k \oplus m) \\ &= 0^n \oplus m \\ &= m \end{aligned}$$

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.
- **Security (Intuitive):** The ciphertext doesn't reveal any information about the plaintext to Eve, no matter what Eve does with the ciphertext.

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.
- **Security (Intuitive):** The ciphertext doesn't reveal any information about the plaintext to Eve, no matter what Eve does with the ciphertext.
- Let us (informally) argue security by analyzing Eve's *view*.

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.
- **Security (Intuitive):** The ciphertext doesn't reveal any information about the plaintext to Eve, no matter what Eve does with the ciphertext.
- Let us (informally) argue security by analyzing Eve's *view*.
- From the viewpoint of Eve, the following happens: Alice has an n-bit message message m. She samples some key k *uniformly at random* from the space of n-bit strings and then outputs $c = k \oplus m$. Let's analyze this *distribution*.

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.
- **Security (Intuitive):** The ciphertext doesn't reveal any information about the plaintext to Eve, no matter what Eve does with the ciphertext.
- Let us (informally) argue security by analyzing Eve's *view*.
- From the viewpoint of Eve, the following happens: Alice has an n-bit message message m. She samples some key k *uniformly at random* from the space of n-bit strings and then outputs $c = k \oplus m$. Let's analyze this *distribution*.

One-Time Pad: Security

- We cannot presume what Eve might do after seeing the ciphertext.
- **Security (Intuitive):** The ciphertext doesn't reveal any information about the plaintext to Eve, no matter what Eve does with the ciphertext.
- Let us (informally) argue security by analyzing Eve's *view*.
- From the viewpoint of Eve, the following happens: Alice has an n-bit message message m. She samples some key k *uniformly at random* from the space of n-bit strings and then outputs $c = k \oplus m$. Let's analyze this *distribution*.

Example

Let $n = 3$, $m = 010$. What is the value of c for every possible value of k?

One-Time Pad: Security

Pr	k	$c = k \oplus 010$
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

One-Time Pad: Security

Pr	k	$c = k \oplus 010$
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

- Each possible value of k is sampled by the key generation algorithm with probability 1/8.

One-Time Pad: Security

Pr	k	$c = k \oplus 010$
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

- Each possible value of k is sampled by the key generation algorithm with probability 1/8.
- Every string in $\{0,1\}^3$ appears *once* in the last column.

One-Time Pad: Security

$\Pr$	k	$c = k \oplus 010$
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

- Each possible value of k is sampled by the key generation algorithm with probability 1/8.
- Every string in $\{0,1\}^3$ appears *once* in the last column.
- In other words, for any string in $\{0,1\}^3$, the probability that the ciphertext is equal to that string is 1/8, i.e., the ciphertext is **uniformly distributed** over $\{0,1\}^3$.

One-Time Pad: Security

Pr	k	$c = k \oplus 010$
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

- Each possible value of k is sampled by the key generation algorithm with probability 1/8.
- Every string in $\{0,1\}^3$ appears *once* in the last column.
- In other words, for any string in $\{0,1\}^3$, the probability that the ciphertext is equal to that string is 1/8, i.e., the ciphertext is **uniformly distributed** over $\{0,1\}^3$.
- This holds for each $m \in \{0,1\}^3$, and not just $m = 010$.

One-Time Pad: Security (contd.)

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)
- But Eve can herself sample from this distribution (without knowing the message)!

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)
- But Eve can herself sample from this distribution (without knowing the message)!
- Thus, the ciphertext that Eve sees *does not carry any information about the message*!

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)
- But Eve can herself sample from this distribution (without knowing the message)!
- Thus, the ciphertext that Eve sees *does not carry any information about the message*!
- Paradox?

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)
- But Eve can herself sample from this distribution (without knowing the message)!
- Thus, the ciphertext that Eve sees *does not carry any information about the message*!
- Paradox?
  1. Doesn't the above claim conflict with the claim on correctness?

One-Time Pad: Security (contd.)

- Regardless of what message m Alice starts with, the the resulting ciphertext is uniformly distributed. (Will argue this more formally for general n later.)
- But Eve can herself sample from this distribution (without knowing the message)!
- Thus, the ciphertext that Eve sees *does not carry any information about the message*!
- Paradox?
  1. Doesn't the above claim conflict with the claim on correctness?
  2. No, because Eve's view does include the key.

Basics of Provable Security

- How to generalize and formalize the above intuitive properties, so that they hold for any encryption scheme.

Basics of Provable Security

- How to generalize and formalize the above intuitive properties, so that they hold for any encryption scheme.
- There are two types of properties:

Basics of Provable Security

- How to generalize and formalize the above intuitive properties, so that they hold for any encryption scheme.
- There are two types of properties:
  1. Ones that should hold in the absence of an attacker. E.g., correctness.

Basics of Provable Security

- How to generalize and formalize the above intuitive properties, so that they hold for any encryption scheme.
- There are two types of properties:
  1. Ones that should hold in the absence of an attacker. E.g., correctness.
  2. Ones that specify what can happen to a system in the presence of an attacker. E.g., security.

Basics of Provable Security

- How to generalize and formalize the above intuitive properties, so that they hold for any encryption scheme.
- There are two types of properties:
  1. Ones that should hold in the absence of an attacker. E.g., correctness.
  2. Ones that specify what can happen to a system in the presence of an attacker. E.g., security.
- **Eventual Goal:** How to formally define any *secure* system, in a way that captures all the properties that we need from that system.

Encryption: Correctness

Encryption: Correctness

An encryption scheme satisfies correctness if *for all possible* keys k, and *for all possible* messages m, the following holds:

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

Encryption: Correctness

Encryption: Correctness

An encryption scheme satisfies correctness if *for all possible* keys k, and *for all possible* messages m, the following holds:

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

- Decrypting a ciphertext, using the same key that was used for encryption, must always result in the original plaintext.

Encryption: Correctness

Encryption: Correctness

An encryption scheme satisfies correctness if *for all possible* keys k, and *for all possible* messages m, the following holds:

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

- Decrypting a ciphertext, using the same key that was used for encryption, must always result in the original plaintext.
- The definition is expressed in terms of a probability because Enc is allowed to be a *randomized* algorithm.

Encryption: Correctness

Encryption: Correctness

An encryption scheme satisfies correctness if *for all possible* keys k, and *for all possible* messages m, the following holds:

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

- Decrypting a ciphertext, using the same key that was used for encryption, must always result in the original plaintext.
- The definition is expressed in terms of a probability because Enc is allowed to be a *randomized* algorithm.

Does $\mathsf{Enc}(k, m) = 0^n$ satisfy correctness? Is it a useful encryption scheme?

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext. Why?

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext. Why?
- The ciphertexts look like random values to Eve.

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext. Why?
- The ciphertexts look like random values to Eve.
- A *pessimistic* choice: Let's allow Eve to choose the plaintexts. This gives a lot of power to Eve. However, if the encryption scheme is secure when Eve chooses the plaintexts, then it's also secure in more realistic scenarios where it only has some partial information about the plaintexts.

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext. Why?
- The ciphertexts look like random values to Eve.
- A *pessimistic* choice: Let's allow Eve to choose the plaintexts. This gives a lot of power to Eve. However, if the encryption scheme is secure when Eve chooses the plaintexts, then it's also secure in more realistic scenarios where it only has some partial information about the plaintexts.

Encryption: Security

Properties that we need from an encryption scheme such as OTP:

- The secret key should be kept hidden from Eve.
- The key is only used to encrypt one plaintext. Why?
- The ciphertexts look like random values to Eve.
- A *pessimistic* choice: Let's allow Eve to choose the plaintexts. This gives a lot of power to Eve. However, if the encryption scheme is secure when Eve chooses the plaintexts, then it's also secure in more realistic scenarios where it only has some partial information about the plaintexts.

An encryption scheme is a good one if its ciphertexts look like random values to Eve, when each key is secret and used to encrypt only one plaintext, even when Eve chooses the plaintexts.

Encryption: One-Time Uniform Ciphertext Security

One-Time Uniform Ciphertext Security

We say that an encryption scheme is one-time uniform ciphertext secure if $\forall m \in \mathcal{M}$ chosen by Eve, the ciphertext is uniformly distributed (over the ciphertext space $\mathcal{C}$), i.e., the following distributions are identical:

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

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme does not satisfy uniform ciphertext security, if $\exists m \in \mathcal{M}$, such that the following distributions are not identical:

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

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme does not satisfy uniform ciphertext security, if $\exists m \in \mathcal{M}$, such that the following distributions are not identical:

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

Example

Is the following encryption scheme secure?

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

Insecure Encryption

Insecure Encryption Scheme

An encryption scheme does not satisfy uniform ciphertext security, if $\exists m \in \mathcal{M}$, such that the following distributions are not identical:

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

Example

Is the following encryption scheme secure?

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

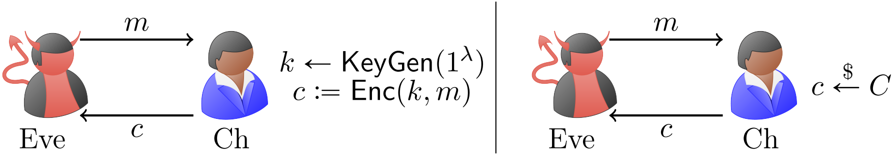
For $m = 0^n$,

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

Clearly the two distributions are not identical in this case.

Encryption: One-Time Uniform Ciphertext 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 chosen by Eve, the above two scenarios seem identical to Eve.