What Makes Cryptography Secure?

Eleanor McMurtry
M.Sc. candidate (University of Melbourne)

eleanorm.info or Twitter @noneuclideangrl

Act I: a tale of two families

How to keep a secret

Two households, both alike in dignity,

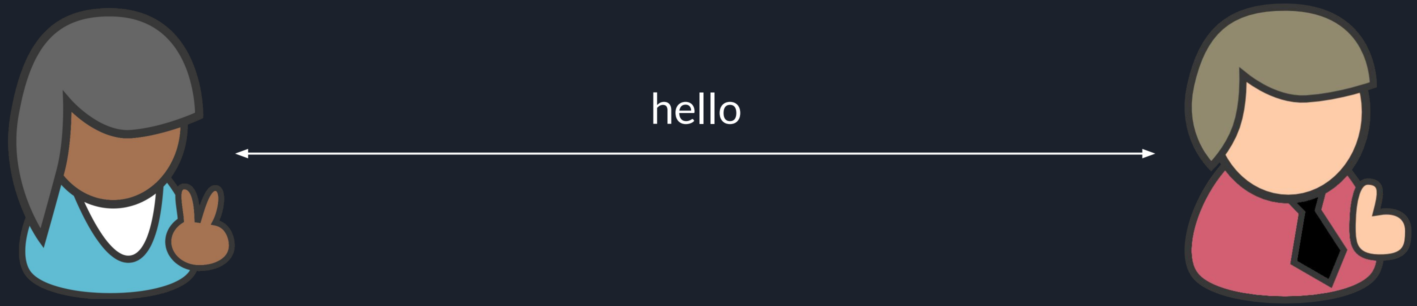
In fair Canberra, where we lay our scene...

How to keep a secret

Two households, both alike in dignity,

In fair Canberra, where we lay our scene...

Alice wants to talk to Bob, but their families hate each other, so it needs to be secret!

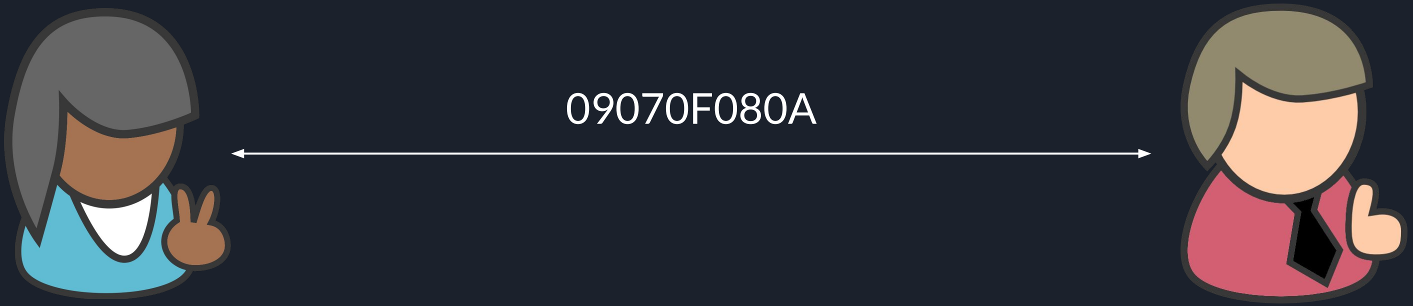


How to keep a secret

Two households, both alike in dignity,

In fair Canberra, where we lay our scene...

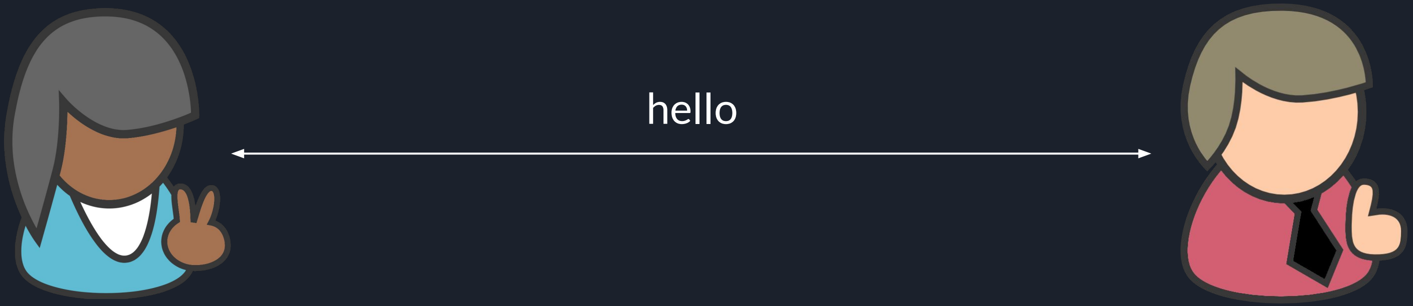
Alice wants to talk to Bob, but their families hate each other, so it needs to be secret!



How to keep a secret

A cryptosystem is a set of three algorithms:

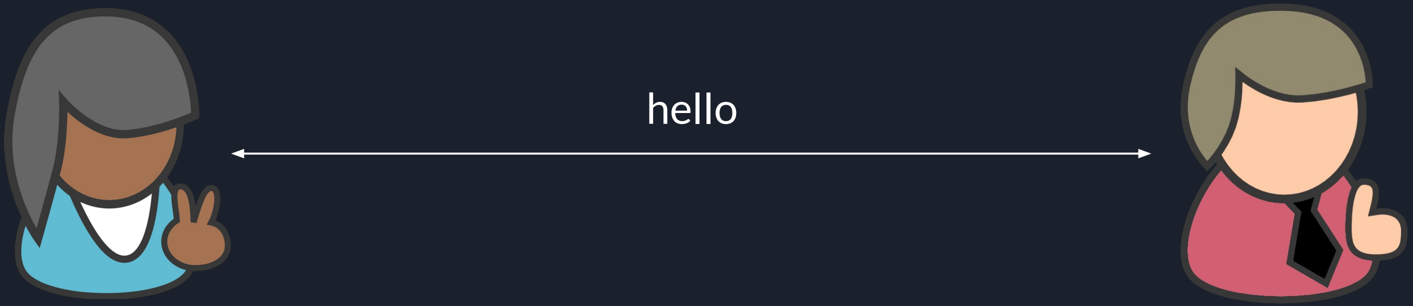
- `Gen(`λ`)` : generates a public key pk for encryption and a secret key sk for decryption (or just a secret key for symmetric systems)



How to keep a secret

A cryptosystem is a set of three algorithms:

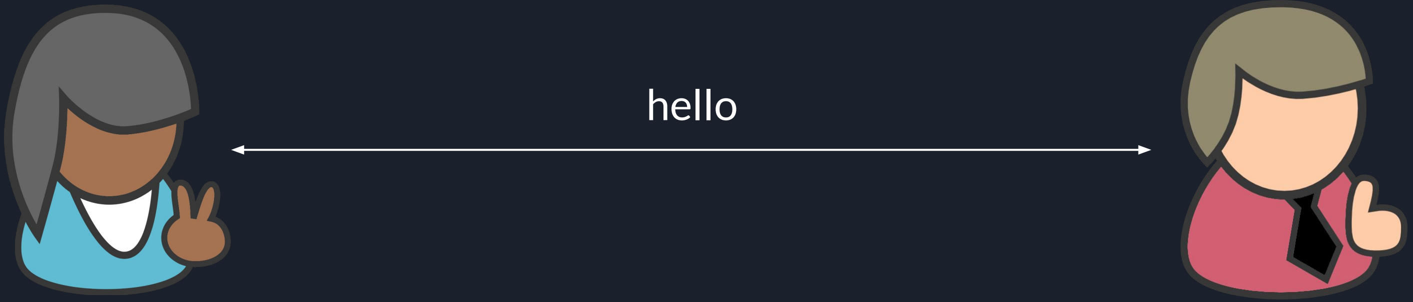
- `Gen`(λ) : generates a public key pk for encryption and a secret key sk for decryption (or just a secret key for symmetric systems)
- `Enc`(m, pk) : encrypts a message `m` with `pk` to produce a ciphertext $\{m\}$



How to keep a secret

A cryptosystem is a set of three algorithms:

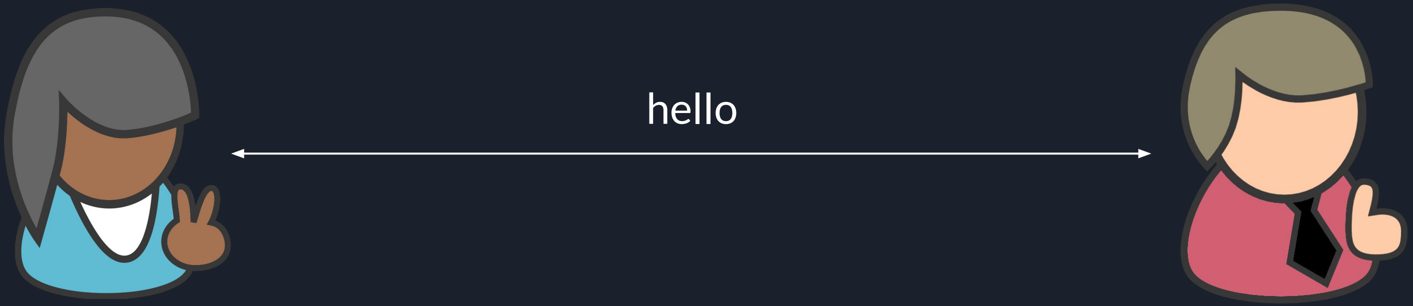
- $\texttt{Gen}(\lambda)$: generates a public key pk for encryption and a secret key sk for decryption (or just a secret key for symmetric systems)
- $\texttt{Enc}(m,\ pk)$: encrypts a message $\texttt{m}$ with pk to produce a ciphertext $\{m\}$
- $\texttt{Dec}(\{m\},\ sk)$: decrypts a ciphertext $\{m\}$ with sk to recover the message m



What is security?

What we would *like:*

- A cryptosystem is information-theoretic secure if **no** adversary can recover the message without the secret key.

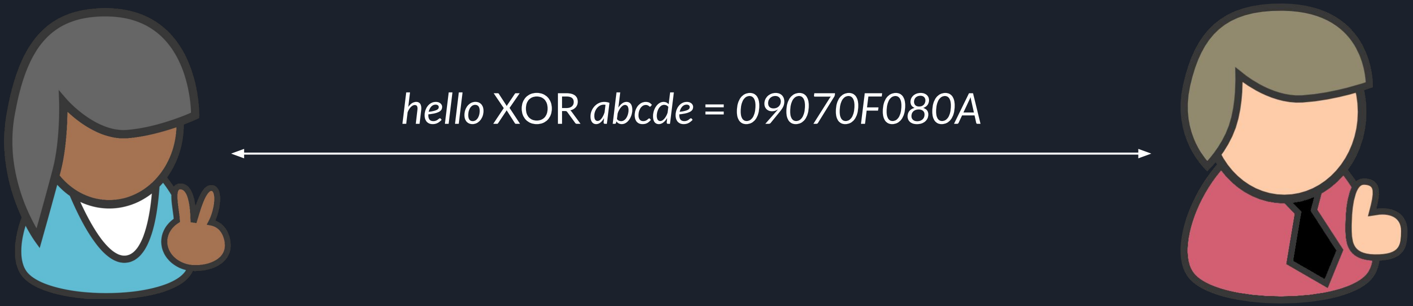


What is security?

What we would *like:*

- A cryptosystem is information-theoretic secure if **no** adversary can recover the message without the secret key.

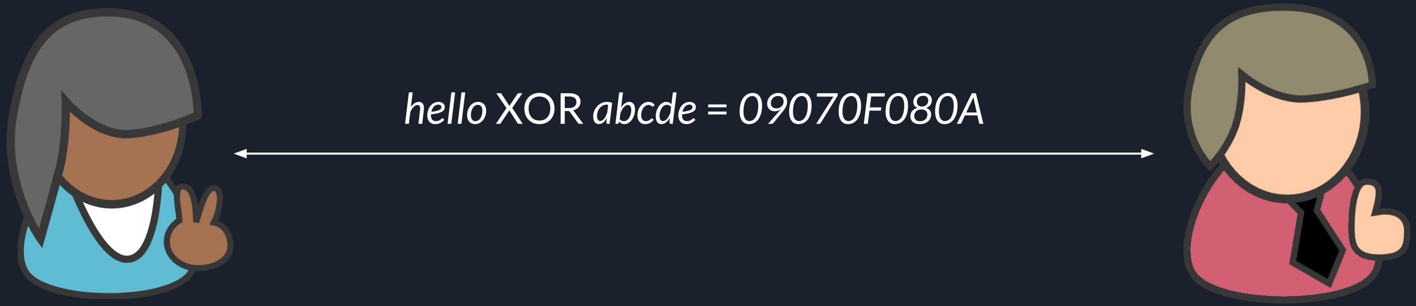
Example: with the secret key `abcde`, we can use XOR.



What is security?

This scheme is information-theoretic secure:

09070F080A XOR *6F75606F79* = *frogs*

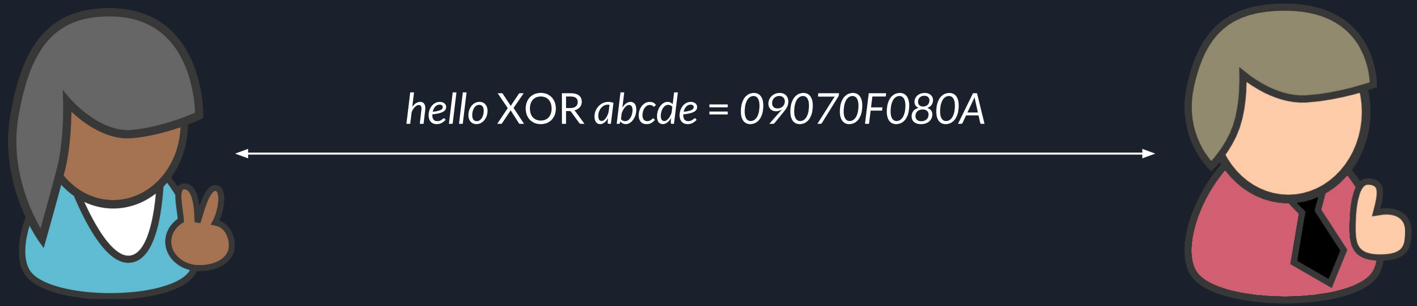


What is security?

This scheme is information-theoretic secure:

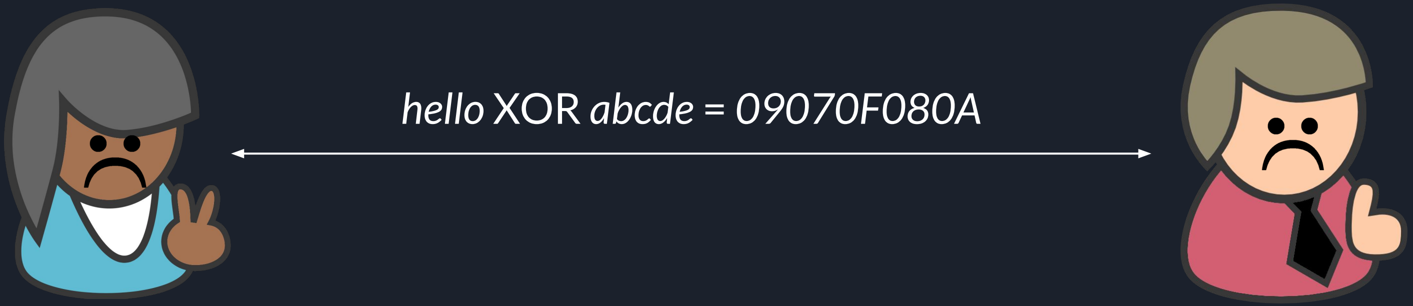
09070F080A XOR *6F75606F79* = *frogs*

Without the secret key, you have no way to guess which message was used.



What is security?

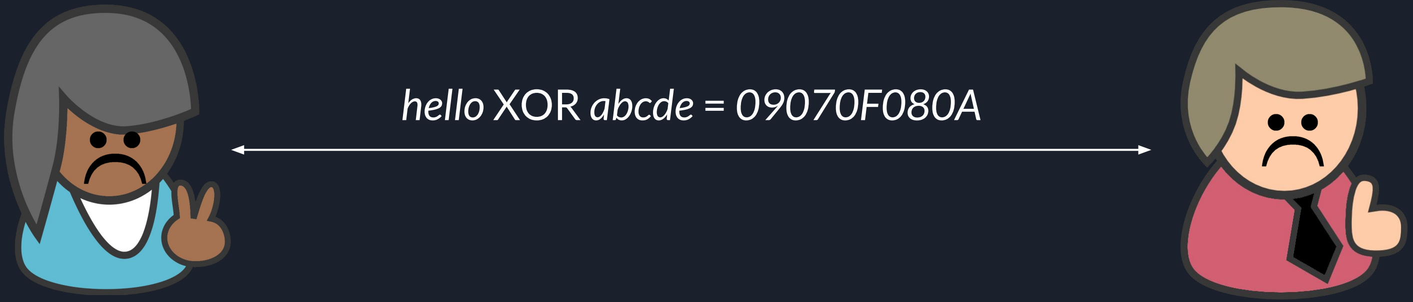
Theorem: No asymmetric cryptosystem can be information-theoretic secure.



What is security?

Theorem: No asymmetric cryptosystem can be information-theoretic secure.

Alice and Bob need to have already agreed on a secret key. It gets worse:

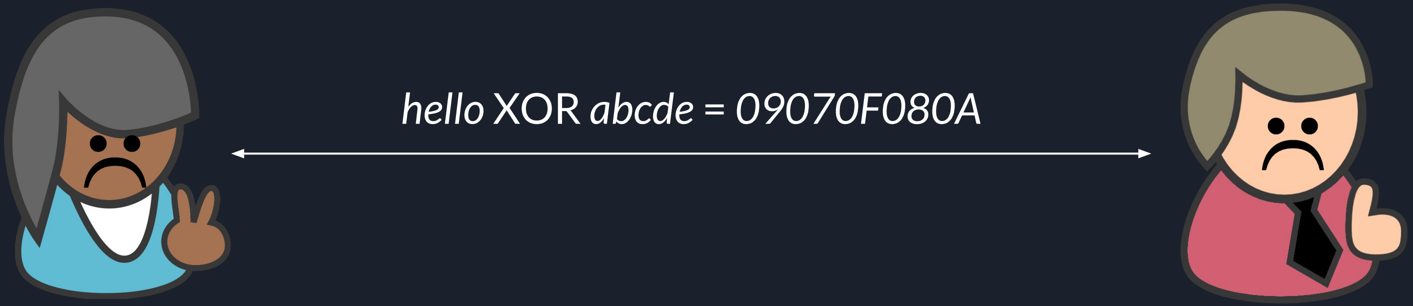


What is security?

Theorem: No asymmetric cryptosystem can be information-theoretic secure.

Alice and Bob need to have already agreed on a secret key. It gets worse:

Theorem: For a cryptosystem to be information-theoretic secure, its secret key must be at least as long as the message.



Act II: are you feeling lucky?

What we will settle for:

- A cryptosystem is computationally secure if no polynomial-time adversary can recover the message without the secret key, *unless they are extremely lucky.*

What we will settle for:

- A cryptosystem is computationally secure if no polynomial-time adversary can recover the message without the secret key, *unless they are extremely lucky.*
- Probability is negligible for a given key length: smaller than the reciprocal of any polynomial.

$$\texttt{negl}(\lambda)$$

Want to play a game?

We define **games** between a challenger C and an adversary A where the adversary wins if they can break the cryptosystem.

Basic security: if Alice sends either 0 or 1 to Bob, someone watching shouldn't be able to tell which one she sent.

The indistinguishability game

1. C runs $\mathtt{Gen}(\lambda)$ to get a secret key.
2. A sends two messages m_0 and m_1 to C.
3. C flips a coin. If heads, C sends $\{m_0\}$ to A; otherwise C sends $\{m_1\}$ to A.
4. A outputs either 0 or 1.

A wins if it outputs the number corresponding to the message.

The indistinguishability game

1. C runs $\texttt{Gen}(\lambda)$ to get a secret key.
2. A sends two messages m_0 and m_1 to C.
3. C flips a coin. If heads, C sends $\{m_0\}$ to A; otherwise C sends $\{m_1\}$ to A.
4. A outputs either 0 or 1.

A wins if it outputs the number corresponding to the message.

If any polynomial-time adversary has a probability less than $½ + \texttt{negl}(\lambda)$ of winning, the cryptosystem is indistinguishable-secure.

A simple cryptosystem

Choose a number between 0 and 25; this is your secret key. To encrypt a message, rotate the letters rightwards by that number:

A B C D E F G H I J K L M N O

sk = 2

A B C D E F G H I J K L M N O

hello → jgnnq

Playing the indistinguishability game

1. C generates the secret key 2.

Playing the indistinguishability game

1. C generates the secret key 2.
2. A sends the messages ab and ac to C.

Playing the indistinguishability game

1. C generates the secret key 2.
2. A sends the messages ab and ac to C.
3. C flips a coin — heads. C sends $\{ab\} = cd$ to A.

Playing the indistinguishability game

1. C generates the secret key 2.
2. A sends the messages `ab` and `ac` to C.
3. C flips a coin — heads. C sends `{ab}` = `cd` to A.
4. The distance between the encrypted letters is 1, so A knows C chose the first message. A outputs `0` and wins.

Playing the indistinguishability game

1. C generates the secret key 2.
2. A sends the messages ab and ac to C.
3. C flips a coin — heads. C sends $\{ab\}$ = cd to A.
4. The distance between the encrypted letters is 1, so A knows C chose the first message. A outputs `0` and wins.

This cryptosystem — the Caesar cipher — is **not** indistinguishable-secure.

Chosen plaintext attacks

- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.

Chosen plaintext attacks

- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.
- The adversary knew the plaintext and used that to break the system, running a chosen plaintext attack.

Chosen plaintext attacks

- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.
- The adversary knew the plaintext and used that to break the system, running a chosen plaintext attack.
- Real-world systems (e.g. AES-128-GCM) **are not** vulnerable to these attacks.

Chosen plaintext attacks

- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.
- The adversary knew the plaintext and used that to break the system, running a chosen plaintext attack.
- Real-world systems (e.g. AES-128-GCM) **are not** vulnerable to these attacks.
  - **Example:** the first message sent over HTTPS almost certainly starts with `GET / HTTP/1.1`.

Chosen plaintext attacks

- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.
- The adversary knew the plaintext and used that to break the system, running a chosen plaintext attack.
- Real-world systems (e.g. AES-128-GCM) **must not be** vulnerable to these attacks.
    - **Example:** the first message sent over HTTPS almost certainly starts with `GET / HTTP/1.1`.

Chosen plaintext attacks

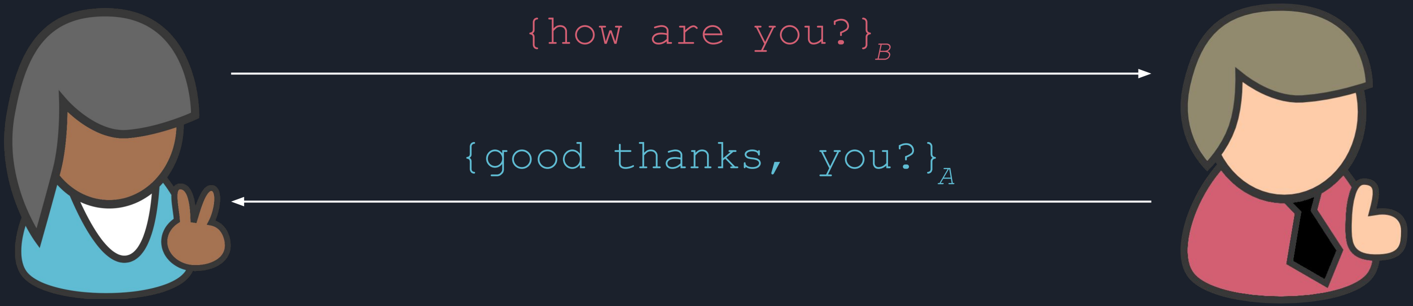
- The Caesar cipher is broken because the relationship between `ab` and `ac` was preserved by encryption.
- The adversary knew the plaintext and used that to break the system, running a chosen plaintext attack.
- Real-world systems (e.g. AES-128-GCM) **are not** vulnerable to these attacks*.
    - **Example:** the first message sent over HTTPS almost certainly starts with `GET / HTTP/1.1`.

*if you use them properly

Act III: public keys

How to keep a secret

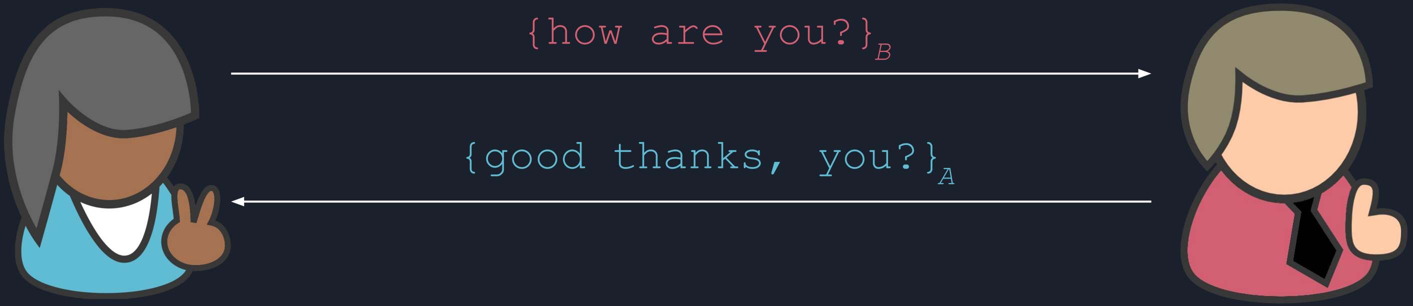
Public-key cryptography lets Alice and Bob talk to each other without agreeing on a secret key.



How to keep a secret

Public-key cryptography lets Alice and Bob talk to each other without agreeing on a secret key.

They have public keys A and B which everybody knows. Only Alice knows the secret key that can decrypt messages encrypted with A, and similarly for Bob.



A better game

1. C runs $\texttt{Gen}(\lambda)$ to get a public key and a secret key. C *gives the public key to A, as well as access to the encryption algorithm* $\mathit{Enc}(m,\ pk)$.
2. A sends two messages m_0 and m_1 to C.
3. C flips a coin. If heads, C sends $\{m_0\}$ to A; otherwise C sends $\{m_1\}$ to A.
4. A outputs either 0 or 1.

A wins if it outputs the number corresponding to the message. If any polynomial-time adversary has a probability less than $½ + \texttt{negl}(\lambda)$ of winning, the cryptosystem is IND-CPA secure.

A better game

1. C runs $\texttt{Gen}(\lambda)$ to get a public key and a secret key. C *gives the public key to A, as well as access to the encryption algorithm* $\texttt{Enc}(m,\ pk)$.
2. A sends two messages m_0 and m_1 to C.
3. C flips a coin. If heads, C sends $\{m_0\}$ to A; otherwise C sends $\{m_1\}$ to A.
4. A outputs either 0 or 1.

A wins if it outputs the number corresponding to the message. If any polynomial-time adversary has a probability less than $½ + \texttt{negl}(\lambda)$ of winning, the cryptosystem is IND-CPA secure.

The challenger lets the adversary encrypt whatever they want; this is a stronger guarantee.

Mathematical aside

- We'll see how we can prove that a cryptosystem is IND-CPA secure. But first, modular arithmetic!

Mathematical aside

- We’ll see how we can prove that a cryptosystem is IND-CPA secure. But first, modular arithmetic!
- **Like a clock:** 5 o’clock + 8 hours = 13 mod 12 = 1 o’clock
- 13 mod 12 says “divide 13 by 12, and take the remainder”. We call this addition modulo 12.

Mathematical aside

- We'll see how we can prove that a cryptosystem is IND-CPA secure. But first, modular arithmetic!
- **Like a clock:** 5 o'clock + 8 hours = 13 mod 12 = 1 o'clock
- 13 mod 12 says "divide 13 by 12, and take the remainder". We call this addition modulo 12.
- With the right numbers, **exponentiation is one-way:** if you know g, p, and $y = g^x \bmod p$, it's hard to guess x.

Exponentiation is one-way

With the right numbers*, **exponentiation is one-way:** if you know g, p, and $y = g^x \bmod p$, it's hard to guess x.

*e.g. $p = 2q + 1$ is prime where q is also prime, and $g^q \bmod p = 1$

Exponentiation is one-way

With the right numbers*, **exponentiation is one-way:** if you know g, p, and $y = g^x \bmod p$, it's hard to guess x.

By "hard to guess", we mean the decisional Diffie-Hellman game is hard to win:

1. C chooses a, b, c and sends g^a, g^b to A.
2. C sends either g^{ab} or g^c to A.
3. A guesses which was sent.

*e.g. $p = 2q + 1$ is prime where q is also prime, and $g^q \bmod p = 1$

The ElGamal cryptosystem

We can make this a cryptosystem! Choose a random x to be your secret key, and $y = g^x$ is your public key.

The ElGamal cryptosystem

We can make this a cryptosystem! Choose a random x to be your secret key, and $y = g^x$ is your public key.

To encrypt m, choose a random r and output $(g^r, m \times y^r) \bmod p$.

The ElGamal cryptosystem

We can make this a cryptosystem! Choose a random x to be your secret key, and $y = g^x$ is your public key.

To encrypt m, choose a random r and output $(g^r, m \times y^r) \bmod p$.

To decrypt $(g^r, m \times y^r)$, calculate $(g^r)^x$; then $m \times y^r / (g^r)^x = m$.

Proving security!

The grand finale: we will show if you can break IND-CPA for ElGamal, then you can break DDH.

You have been given g^a, g^b, and $x = g^{ab}$ or g^c, and you need to work out what x is.

Proving security!

The grand finale: we will show if you can break IND-CPA for ElGamal, then you can break DDH.

You have been given g^a, g^b, and $x = g^{ab}$ or g^c, and you need to work out what x is.

You can play IND-CPA with Alice! Instead of encryption, give her access to $\mathtt{Enc'}(m) = (g^b, m \times x)$. If she wins, guess g^{ab}; otherwise, guess g^c.

Proving security!

Alice is trying to break $\texttt{Enc'}(m) = (g^b, m \times x)$. If she wins, we guess g^{ab}; otherwise, we guess g^c.

- If $x = g^c$, then Alice doesn't see a real encryption, so we just guess randomly.

Proving security!

Alice is trying to break $\texttt{Enc'}(m) = (g^b, m \times x)$. If she wins, we guess g^{ab}; otherwise, we guess g^c.

- If $x = g^c$, then Alice doesn't see a real encryption, so we just guess randomly.
- If $x = g^{ab}$, Alice does see a real encryption, and wins more often than not.

Proving security!

Alice is trying to break $\texttt{Enc}'(m) = (g^b, m \times x)$. If she wins, we guess g^{ab}; otherwise, we guess g^c.

- If $x = g^c$, then Alice doesn't see a real encryption, so we just guess randomly.
- If $x = g^{ab}$, Alice does see a real encryption, and wins more often than not.
- **But** decisional Diffie-Hellman is hard, so no matter how good she is, *Alice can't win more than a negligible amount of the time!*

Proving security!

Alice is trying to break $\mathtt{Enc'}(m) = (g^b, m \times x)$. If she wins, we guess g^{ab}; otherwise, we guess g^c.

- If $x = g^c$, then Alice doesn't see a real encryption, so we just guess randomly.
- If $x = g^{ab}$, Alice does see a real encryption, and wins more often than not.
- **But** decisional Diffie-Hellman is hard, so no matter how good she is, *Alice can't win more than a negligible amount of the time!*

Q.E.D.

Conclusions

- Cryptography is *probabilistic*: you can break it, but only if you're extremely lucky
- Cryptography is *sensitive to assumptions*: if decisional Diffie-Hellman is solvable, ElGamal evaporates
- Read cryptography specs carefully, *and don't try to implement it yourself if you can avoid it*
- Cryptography is really interesting!

For more, see Katz & Lindell's *Introduction to Modern Cryptography*. For a more comprehensive but difficult read, see Boneh & Shoup's *A Graduate Course in Applied Cryptography*.