Applied Cryptography
Week 1: The basics

Bernardo Portela

M:ERSI, M:SI, M:CC - 25

Context

People usually think of *encryption* when they hear *cryptography*

... which is not unreasonable. But it is only one of many cryptographic techniques

Context

People usually think of *encryption* when they hear *cryptography*

... which is not unreasonable. But it is only one of many cryptographic techniques

Encryption guarantees *confidentiality*, but real-world applications often require other guarantees to be considered secure systems

- Authenticity, non-repudiation, unpredictability, anonymity, ...

Context

People usually think of *encryption* when they hear *cryptography*

... which is not unreasonable. But it is only one of many cryptographic techniques

Encryption guarantees *confidentiality*, but real-world applications often require other guarantees to be considered secure systems

- Authenticity, non-repudiation, unpredictability, anonymity, ...

Also, there are many kinds of encryption

- Symmetric, asymmetric, authenticated, homomorphic, ...

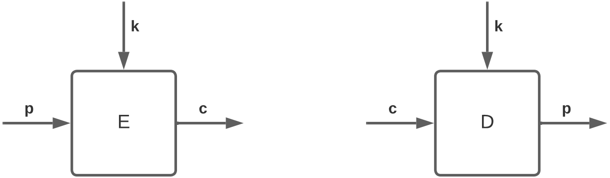
What is encryption?

Q1: What do you think encryption means?

What is encryption?

Q1: What do you think encryption means?

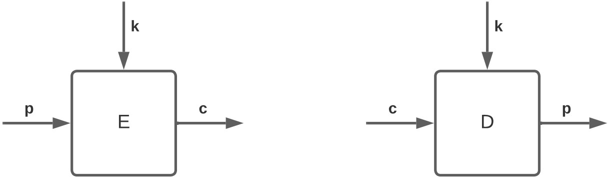
Encryption transforms *plaintexts* into *ciphertexts* using a *key*



What is encryption?

Q1: What do you think encryption means?

Encryption transforms *plaintexts* into *ciphertexts* using a *key*



We will use the following notation to talk about algorithms

- $c \leftarrow_{\$} \mathrm{E}(k, p)$ - Encryption is (usually) randomized
    - **Q2: Why?**
- $p \leftarrow \mathrm{D}(k, c)$ - Decryption is deterministic

We begin with **symmetric** encryption: same key on both ends

Classical Ciphers

- Historically, cryptography was concerned mainly with *confidentiality*
  - Informally, it means to protect the privacy of information
  - We will later define this in more concrete terms

Classical Ciphers

- Historically, cryptography was concerned mainly with *confidentiality*
  - Informally, it means to protect the privacy of information
  - We will later define this in more concrete terms
- We will now look at a few examples
  - Starting with a well-known one
- Key questions that we will ask:
  - What is the size of the key space?
  - How many times is the key used?
  - How are plaintext symbols in different positions transformed?
  - How can a modern computer break them?

Caesar Cipher

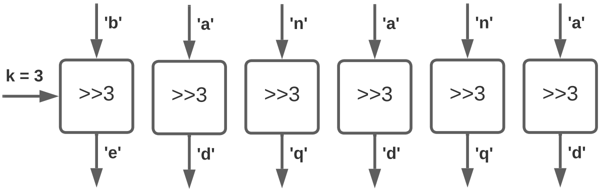
- One of the earliest known ciphers, used in Roman times

Caesar Cipher

- One of the earliest known ciphers, used in Roman times

Algorithm

- Take the plaintext, e.g., "banana"
- Shift the plaintext 3 letters upward
- Key is fixed, but we can choose other shift sizes as keys

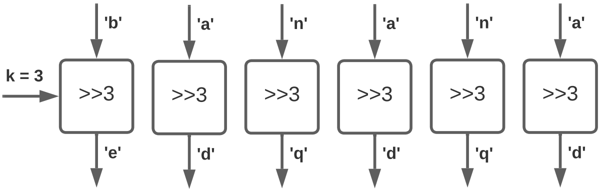


Caesar Cipher

- One of the earliest known ciphers, used in Roman times

Algorithm

- Take the plaintext, e.g., "banana"
- Shift the plaintext 3 letters upward
- Key is fixed, but we can choose other shift sizes as keys



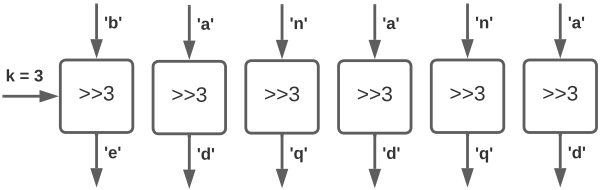
- **Q1: How can we decrypt?**

Caesar Cipher

- One of the earliest known ciphers, used in Roman times

Algorithm

- Take the plaintext, e.g., "banana"
- Shift the plaintext 3 letters upward
- Key is fixed, but we can choose other shift sizes as keys



- **Q1: How can we decrypt?**
- **Q2: Why is this cipher insecure?**

Substitution Ciphers

- We can choose different shifts for different letters
  - E.g. $'a' \rightarrow' f'$; $'b' \rightarrow' a'$; $'c' \rightarrow' z'$; ...
- Shift is a particular class of permutations over the alphabet
  - **Q: How many permutations are there over the alphabet?**
  - **A.k.a. how large is the key space?**

Substitution Ciphers

- We can choose different shifts for different letters
  - E.g. $'a' \rightarrow' f'$; $'b' \rightarrow' a'$; $'c' \rightarrow' z'$; ...
- Shift is a particular class of permutations over the alphabet
  - **Q: How many permutations are there over the alphabet?**
  - **A.k.a. how large is the key space?**

- $26! \approx 2^{88}$: It's a pretty big number
- Not possible to brute force without massive investment
- Surely it will be safe... **Q: Right?**

Frequency letter attacks

Q1: Which of these is most common in Portuguese?

1. 'l'
2. 'a'
3. 's'
4. 'z'

Frequency letter attacks

Q1: Which of these is most common in Portuguese?

1. 'l' - 2.78%
2. 'a' - 14.63%
3. 's' - 6.81%
4. 'z' - 0.47%

Frequency letter attacks

Q1: Which of these is most common in Portuguese?

1. 'l' - 2.78%
2. 'a' - 14.63%
3. 's' - 6.81%
4. 'z' - 0.47%

Q2: How can we use this to attack this encryption scheme?

Frequency letter attacks

Q1: Which of these is most common in Portuguese?

1. 'l' - 2.78%
2. 'a' - 14.63%
3. 's' - 6.81%
4. 'z' - 0.47%

Q2: How can we use this to attack this encryption scheme?

- Gather many ciphertexts and count the frequency of letters
- Match that frequency with the frequency of plaintext alphabet
    - With good odds, the most common letter in the ciphertexts will match the most common letter in the plaintext alphabet
- Can be done using a statistical hypothesis (χ^2) test

Vigenère Cipher

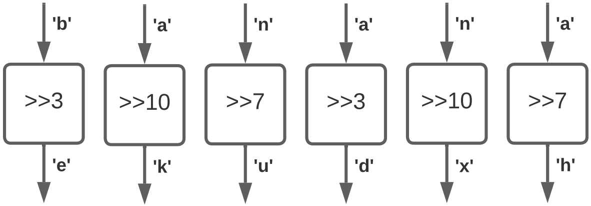
- Thought to be secure in the 16th-19th centuries
- Key spans over multiple letters, defining different shifts

Vigenère Cipher

- Thought to be secure in the 16th-19th centuries
- Key spans over multiple letters, defining different shifts

Algorithm

- Choose key in $\mathbb{Z}_{26}^{l}$, where l is the size of the key
- For our example, $l = 3$ and $K = (3, 10, 7)$
- To encrypt, shift the i-th plaintext letter by $K_{i \bmod l}$

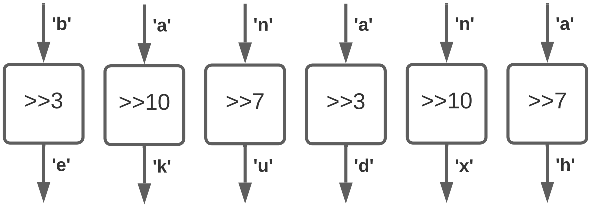


Vigenère Cipher

- Thought to be secure in the 16th-19th centuries
- Key spans over multiple letters, defining different shifts

Algorithm

- Choose key in $\mathbb{Z}_{26}^{l}$, where l is the size of the key
- For our example, $l = 3$ and $K = (3, 10, 7)$
- To encrypt, shift the i-th plaintext letter by $K_{i \bmod l}$



Q: How can we break it?

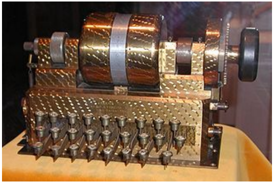
Rotor machines

Hebern machine (left)

- Key is the disk, encoding a substitution table
- On key press, the output is encrypted and the disk rotates

The Enigma (right)

- Key is the initial setting of rotors by multiple rotors (3-5)
- Rotors rotate with different frequencies




The one-time pad - Part 1

- Patent issued in 1917 by Gilbert Vernam

Algorithm

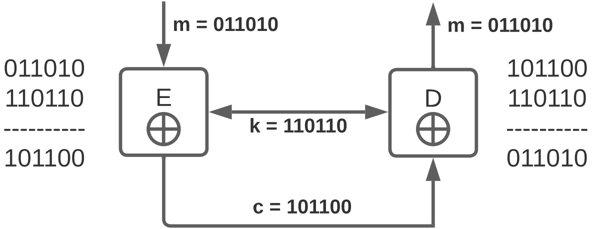
- Lets work with 0s and 1s
- Choose a random bit string $k \leftarrow \{0,1\}^m$
- To encrypt, compute the bit-wise XOR of m and k: $m \oplus k$
- To decrypt, compute the bit-wise XOR of c and k: $c \oplus k$

The one-time pad - Part 1

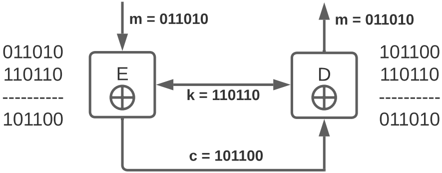
- Patent issued in 1917 by Gilbert Vernam

Algorithm

- Lets work with 0s and 1s
- Choose a random bit string $k \leftarrow \{0,1\}^m$
- To encrypt, compute the bit-wise XOR of m and k: $m \oplus k$
- To decrypt, compute the bit-wise XOR of c and k: $c \oplus k$

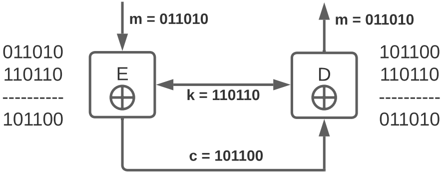


The one-time pad - Part 2



Q1: Is this secure?

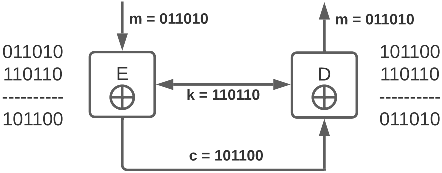
The one-time pad - Part 2



Q1: Is this secure?

- It is *perfectly secure* (as long as keys are used only once)

The one-time pad - Part 2

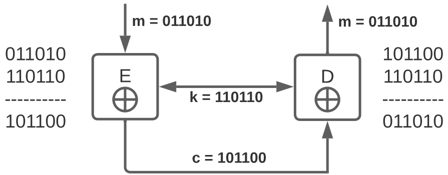


Q1: Is this secure?

- It is *perfectly secure* (as long as keys are used only once)

Q2: Why is this not used to encrypt everything?

The one-time pad - Part 2



Q1: Is this secure?

- It is *perfectly secure* (as long as keys are used only once)

Q2: Why is this not used to encrypt everything?

- Keys must have *the same size* as the messages
  - To send a 2 Gb file, I must use a 2 Gb key!
- How can we pre-share and store such huge keys?
- But it is used everywhere in cryptography as a building block

Making OTP Practical

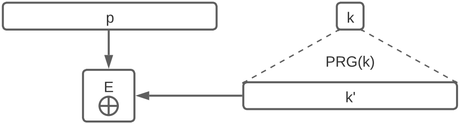
A big problem of OTP is that we need that $|k| = |p|$

Making OTP Practical

A big problem of OTP is that we need that $|k| = |p|$

Pseudo Random Generators

- Something we will look more in-depth next class
- Take randomness k and expand to k' such that $|k'| >> |k|$

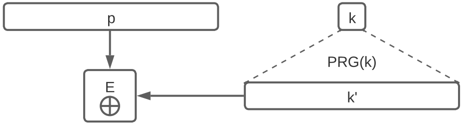


Making OTP Practical

A big problem of OTP is that we need that $|k| = |p|$

Pseudo Random Generators

- Something we will look more in-depth next class
- Take randomness k and expand to k' such that $|k'| >> |k|$



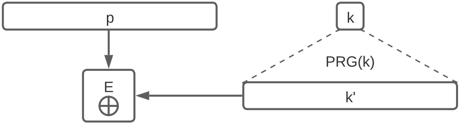
Q: Given this PRG, is the OTP still secure?

Making OTP Practical

A big problem of OTP is that we need that $|k| = |p|$

Pseudo Random Generators

- Something we will look more in-depth next class
- Take randomness k and expand to k' such that $|k'| >> |k|$



Q: Given this PRG, is the OTP still secure?

Trick question, sorry! Depends on the power of the adversary

- If the power of the adversary is unlimited, then no
- Otherwise, yes, if the PRG is secure – next class

Will be used to construct stream ciphers (in a couple of weeks)

Key Takeaways

- Encryption is a combination of two main algorithms:
  - Encryption takes plaintexts and produces ciphertexts
  - Decryption takes ciphertexts and produces plaintexts

Key Takeaways

- Encryption is a combination of two main algorithms:
    - Encryption takes plaintexts and produces ciphertexts
    - Decryption takes ciphertexts and produces plaintexts
- Classical ciphers fail because of multiple reasons:
    - Small key space - key is easy to guess
    - Ciphertexts reveal patterns in the original message

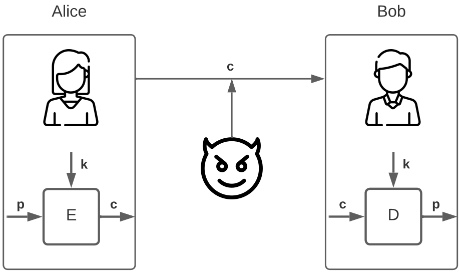
Key Takeaways

- Encryption is a combination of two main algorithms:
  - Encryption takes plaintexts and produces ciphertexts
  - Decryption takes ciphertexts and produces plaintexts
- Classical ciphers fail because of multiple reasons:
  - Small key space - key is easy to guess
  - Ciphertexts reveal patterns in the original message
- One-time pad, on the other hand, is *perfectly secure*
  - Might as well guess the original message
  - Keys **can only be used once**
  - Keys **must be the same size** as the message

What we talk about when we talk about Security - Part 1

Meet Alice and Bob

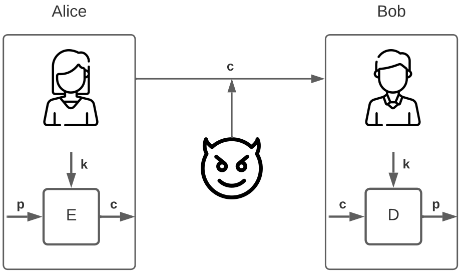
- Alice wants to send a message to Bob
- The message must be secure against an attacker (the devil)



What we talk about when we talk about Security - Part 1

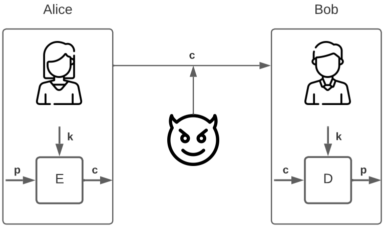
Meet Alice and Bob

- Alice wants to send a message to Bob
- The message must be secure against an attacker (the devil)



Q: What do we mean for the encryption to be “secure”?

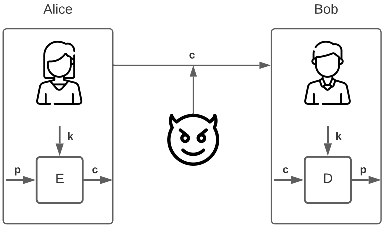
What we talk about when we talk about Security - Part 2



Suppose my message is "banana"

- Attempt #1: I don't want "*banana*" to be revealed

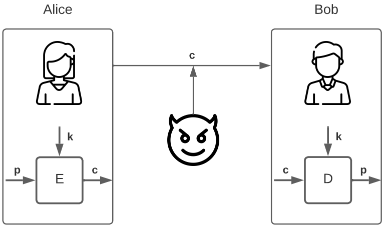
What we talk about when we talk about Security - Part 2



Suppose my message is "banana"

- Attempt #1: I don't want "*banana*" to be revealed
    - But if a scheme reveals "bananb" I am also not happy.
- Attempt #2: I don't want any characters to be retrieved

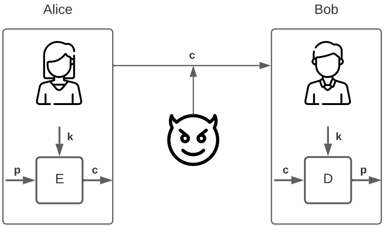
What we talk about when we talk about Security - Part 2



Suppose my message is "banana"

- Attempt #1: I don't want "*banana*" to be revealed
  - But if a scheme reveals "bananb" I am also not happy.
- Attempt #2: I don't want any characters to be retrieved
  - But if a scheme reveals "cbobob" I am also not happy.

What we talk about when we talk about Security - Part 2



Suppose my message is “banana”

- Attempt #1: I don’t want “*banana*” to be revealed
  - But if a scheme reveals “bananb” I am also not happy.
- Attempt #2: I don’t want any characters to be retrieved
  - But if a scheme reveals “cbobob” I am also not happy.
- A more rigorous approach to define security must be taken

Cryptographic Security

Security must be rigorously defined before it can be analyzed.

Cryptographic Security

Security must be rigorously defined before it can be analyzed.

It has two dimensions:

- The power of the attacker - the *attack model*
- What it should not be able to do - the *security goal*

Cryptographic Security

Security must be rigorously defined before it can be analyzed.

It has two dimensions:

- The power of the attacker - the *attack model*
- What it should not be able to do - the *security goal*

The attack model defines:

- Computational power, e.g. poly-time/quantum adversaries
- Ability to collect outputs, such as ciphertexts
- Ability to influence inputs, such as plaintexts

Cryptographic Security

Security must be rigorously defined before it can be analyzed.

It has two dimensions:

- The power of the attacker - the *attack model*
- What it should not be able to do - the *security goal*

The attack model defines:

- Computational power, e.g. poly-time/quantum adversaries
- Ability to collect outputs, such as ciphertexts
- Ability to influence inputs, such as plaintexts

Security analysis is made based on a *security model*

- Experiment runs an adversary according to its model
- It then checks the security goal

Security Experiments

The *ciphertext leaks nothing about the plaintext*

Security Experiments

The *ciphertext leaks nothing about the plaintext*

- This notion of security is formalized with an experiment
- A game played between a challenger and an adversary
- The goal of the adversary is to win the game

Security Experiments

The *ciphertext leaks nothing about the plaintext*

- This notion of security is formalized with an experiment
- A game played between a challenger and an adversary
- The goal of the adversary is to win the game
- Within the experiment, what the adversary can do reflects its assumed power – the *attack model*
- The security goal depends on how often the adversary wins the experiment. Winning the game does not imply that the encryption scheme is broken, as we will see

Semantic Security

Ciphertext indistinguishability, also known as Semantic Security

Experiment

- Challenger chooses a random key k
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow \mathrm{E}(k, m_b)$
- Attacker gets challenge c and outputs b'
- The attacker wins if $b = b'$

Semantic Security

Ciphertext indistinguishability, also known as Semantic Security

Experiment

- Challenger chooses a random key k
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow \mathrm{E}(k, m_b)$
- Attacker gets challenge c and outputs b'
- The attacker wins if $b = b'$

Q: Program an attacker that wins with probability $\frac{1}{2}$

Semantic Security

Ciphertext indistinguishability, also known as Semantic Security

Experiment

- Challenger chooses a random key k
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow \mathrm{E}(k, m_b)$
- Attacker gets challenge c and outputs b'
- The attacker wins if $b = b'$

Q: Program an attacker that wins with probability $\frac{1}{2}$

Encryption is broken if the attacker wins the experiment with visible bias from $\frac{1}{2}$

A Concrete Example - The One-time Pad - Part 1

Consider the previous experiment (succinct version for OTP)

- Challenger chooses a random key k and bit b
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow k \oplus m_b$

Q: Can we win this experiment with probability over $\frac{1}{2}$?

A Concrete Example - The One-time Pad - Part 1

Consider the previous experiment (succinct version for OTP)

- Challenger chooses a random key k and bit b
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow k \oplus m_b$

Q: Can we win this experiment with probability over $\frac{1}{2}$?

- Observe that, without knowledge of the key, c follows a uniform distribution
  - Consider that the size of messages is 1 (one bit)
  - If the encrypted message is p, then c is either $p \oplus 0$ or $p \oplus 1$
  - The key is randomly sampled, so outcomes are equally likely
  - This is the case for any message size

A Concrete Example - The One-time Pad - Part 2

Consider the previous experiment (succinct version for OTP)

- Challenger chooses a random key k and bit b
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow k \oplus m_b$
- Additional encryptions of messages can be requested
  - But not of m_0 or m_1

Q: Can we win this experiment with probability over $\frac{1}{2}$?

A Concrete Example - The One-time Pad - Part 2

Consider the previous experiment (succinct version for OTP)

- Challenger chooses a random key k and bit b
- Attacker chooses two messages of equal length (m_0, m_1)
- Challenger chooses a random bit b and encrypts $c \leftarrow k \oplus m_b$
- Additional encryptions of messages can be requested
  - But not of m_0 or m_1

Q: Can we win this experiment with probability over $\frac{1}{2}$?

- That's why it's called the *ONE*-time pad...
  - Adversary got $c = m_b \oplus k$
  - Ask to encrypt $m_0 \oplus m_1$: $c' = m_0 \oplus m_1 \oplus k$
  - Now the adversary can do $c \oplus c' = m_b \oplus k \oplus m_0 \oplus m_1 \oplus k$
  - If $b = 0$, $m_1 \cancel{\oplus m_b \oplus m_0 \oplus k \oplus k}$
  - Check if $c \oplus c' = m_1$, return 0, and 1 otherwise
  - Victory with probability 1

Attack models for (symmetric) encryption

Attacks of increasing attacker power:

- Ciphertext-only attack: sees only challenge ciphertexts
- Known-message attack: + some plaintext/ciphertext pairs
- Chosen-plaintext attacks: + ciphertexts for chosen plaintexts
- Chosen-ciphertext attacks: + plaintext for chosen ciphertexts

Attack models for (symmetric) encryption

Attacks of increasing attacker power:

- Ciphertext-only attack: sees only challenge ciphertexts
- Known-message attack: + some plaintext/ciphertext pairs
- Chosen-plaintext attacks: + ciphertexts for chosen plaintexts
- Chosen-ciphertext attacks: + plaintext for chosen ciphertexts

Models are black-box

- Mathematical abstractions to facilitate rigorous reasoning
- Attacker only sees what we give it
- Ignores many (possibly nuanced) real-world concerns
  - Algorithm execution time
  - Fault-injection

Kerckhoffs's Principle

- Long ago, it was common for encryption systems to be secret
- The idea is: the less people know, the harder it is to attack
- Also known as *Security through obscurity*
- We now know that **this is a bad idea**

Kerckhoffs's Principle

- Long ago, it was common for encryption systems to be secret
- The idea is: the less people know, the harder it is to attack
- Also known as *Security through obscurity*
- We now know that **this is a bad idea**

Kerckhoffs's Principle

- All details of a cryptosystem's operation are public
- The only secret is the key
- Why? Public knowledge promotes scrutiny
  - Designs of systems we will study are all public knowledge
  - Cryptographic schemes can be analyzed by everyone
  - Real-world security built on top of open standards
  - Methodology that revolutionized the way we approach security

Never Use Your Own Crypto

A point we will hammer home.

- It is very easy to make mistakes
- It is very hard to find mistakes

Never Use Your Own Crypto

A point we will hammer home.

- It is very easy to make mistakes
- It is very hard to find mistakes

A plethora of bad apples

- Cryptography can be poorly designed
  - 2G mobile phone standard using A5/2 encryption
  - Broken after 10 years

Never Use Your Own Crypto

A point we will hammer home.

- It is very easy to make mistakes
- It is very hard to find mistakes

A plethora of bad apples

- Cryptography can be poorly designed
  - 2G mobile phone standard using A5/2 encryption
  - Broken after 10 years
- Cryptography may not match the application
  - TLS servers try to decrypt incoming traffic
  - What if errors are observed on the network?
  - Security experiment does not capture this
  - Padding-oracle attacks are an example of this mismatch

Never Use Your Own Crypto

A point we will hammer home.

- It is very easy to make mistakes
- It is very hard to find mistakes

A plethora of bad apples

- Cryptography can be poorly designed
  - 2G mobile phone standard using A5/2 encryption
  - Broken after 10 years
- Cryptography may not match the application
  - TLS servers try to decrypt incoming traffic
  - What if errors are observed on the network?
  - Security experiment does not capture this
  - Padding-oracle attacks are an example of this mismatch
- Cryptography can be poorly implemented
  - Timing attacks used to break theoretically secure crypto
  - Implementation errors can leak secret keys (e.g. heartbleed)

Key Takeaways

- Defining security is not trivial:
    - Modern crypto relies on two main concepts
        - *attack model:* what the attacker can do
        - *security goal:* the circumstances that constitute an attack
    - Security experiments - an algorithmic description of security model, which can then be used to prove security
    - Models are abstractions and disregard concerns such as time and fault injection

Key Takeaways

- Defining security is not trivial:
  - Modern crypto relies on two main concepts
    - *attack model:* what the attacker can do
    - *security goal:* the circumstances that constitute an attack
  - Security experiments - an algorithmic description of security model, which can then be used to prove security
  - Models are abstractions and disregard concerns such as time and fault injection
- Kerkhoff's Principle
  - Algorithms should always be open-source
  - Security comes from the strength of the key
  - Many systems (today) rely on closed-source crypto

Key Takeaways

- Defining security is not trivial:
  - Modern crypto relies on two main concepts
    - *attack model:* what the attacker can do
    - *security goal:* the circumstances that constitute an attack
  - Security experiments - an algorithmic description of security model, which can then be used to prove security
  - Models are abstractions and disregard concerns such as time and fault injection
- Kerkhoff's Principle
  - Algorithms should always be open-source
  - Security comes from the strength of the key
  - Many systems (today) rely on closed-source crypto
- Do not write your own crypto!
  - It's easy to f-up
  - Testing correctness and security is very nuanced

Syllabus overview

- Classical schemes and security definitions $\leftarrow$ **today!**
- Randomness and quantifying security
- Block ciphers
- Stream ciphers
- Hash functions and message authentication
- Authenticated encryption
- Computational hardness
- Public-key encryption
- Key agreement protocols
- Elliptic curve cryptography
- Public key infrastructures and TLS

Methodology

Theoretical classes - Friday, 14:30 $\approx$ 16:00

- Explore and discuss topics related to cryptography
- Some exercises, but mostly the ball is in the professors' court

Methodology

Theoretical classes - Friday, 14:30 $\approx$ 16:00

- Explore and discuss topics related to cryptography
- Some exercises, but mostly the ball is in the professors' court

Practical classes - Friday, 16:30 $\approx$ 18:00

- Work on the practical assignments established for each class
- Gain practical experience on the topics covered in T classes

Evaluation - Part 1

Exams

- Assess knowledge of topics discussed during classes
- Will cover both theoretical and practical questions
- Worth 15 points (75%) of the final grade
- Two exams:
  - Midterm exam – 7.5 points
  - Final exam – 7.5 points

Evaluation - Part 2

Practical exercises

- Practical assignments are divided in two parts
- To be done in groups of up to two students
- Mandatory
  - Students must submit at least 50% of all mandatory exercises (and be accepted by the lecturers) to be eligible for exams
  - Simple exercises to practice various cryptographic concepts
- Extra
  - Slightly more challenging exercises
  - Worth the remaining 5 points (25%) of the final grade

Class Page and Bibliography

All class material can be found in:

`https://www.dcc.fc.up.pt/~rvr/aulas/Cripto2526/`

This includes:

- Slides of all theoretical classes
  - This does not preclude the need to take notes!
- Work assignments for practical classes and deadlines
- Class notifications
- Other useful links

Bibliography

- Jean-Philippe Aumasson; Serious Cryptography: A Practical Introduction to Modern Encryption
- Jonathan Katz; Introduction to modern cryptography
- Oded Goldreich; Foundations of Cryptography - Volume 1

EU Chat Control

EU is currently discussing a legislation for Chat Control (link).
Summarily (and loosely), what this entails:

- Every message, photo and file scanned automatically
- Breaking end-to-end encryption (e.g. Watsapp)
- Information scanned by AI

As such, It is important to understand that:

- The technologies you study have profound societal impact
- This can be used for harm: criminal activity in the deep web
- **But!** This is a tool to protect citizens from abuses of power

Applied Cryptography
Week 1: The basics

Bernardo Portela

M:ERSI, M:SI, M:CC - 25