Modern Cryptography: An overview

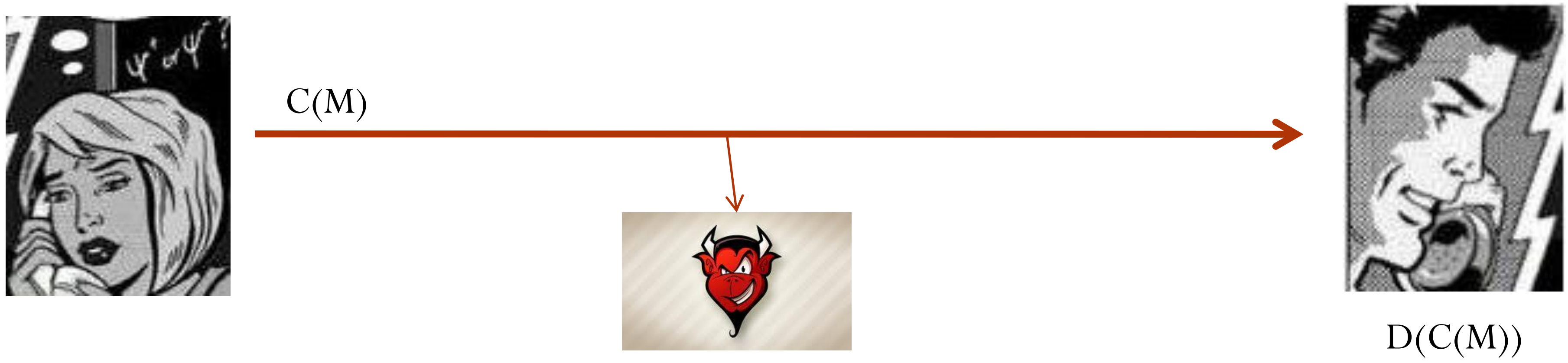
Ragesh Jaiswal

CSE, IIT Delhi

Classical Cryptography

- *Cryptography*: The art of writing or solving codes.

- Classical cryptography: The art of secret writing.



- The communication is secure as long as the encoding algorithm is a *secret*.
  - Disadvantages: Reverse engineering, coding algorithm leaks.

Modern Cryptography

- *Cryptography*: The scientific study of techniques for securing digital information, transaction, and distributed computations.

Modern Cryptography

- *Cryptography*: The scientific study of techniques for securing digital information, transaction, and distributed computations.
  - Classical cryptography was restricted to military. Modern cryptography is influences almost everyone.

Modern Cryptography

- *Cryptography*: The scientific study of techniques for securing digital information, transaction, and distributed computations.
  - Classical cryptography was restricted to military. Modern cryptography is influences almost everyone.
  - Classical cryptography was mostly about secret communication. With modern cryptography the scope has expanded. It now deals with digital signatures, digital cash, secure voting…

Modern Cryptography

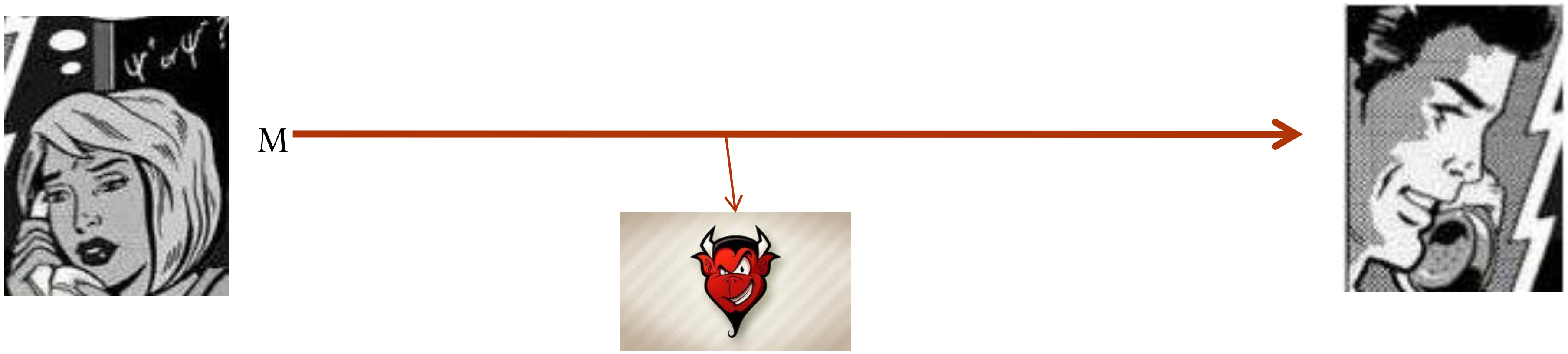
- *Cryptography*: The scientific study of techniques for securing digital information, transaction, and distributed computations.
  - Classical cryptography was restricted to military. Modern cryptography is influences almost everyone.
  - Classical cryptography was mostly about secret communication. With modern cryptography the scope has expanded. It now deals with digital signatures, digital cash, secure voting…
  - Modern cryptography breaks out of the "design-break-design" cycle model of classical cryptography.
    - The security is not based on the secrecy of the protocol details but based on sound mathematical and computational principles.
    - *Provable security*: It is now possible to formally argue about the security of protocols.

Foundations of Modern Cryptography

Provable security

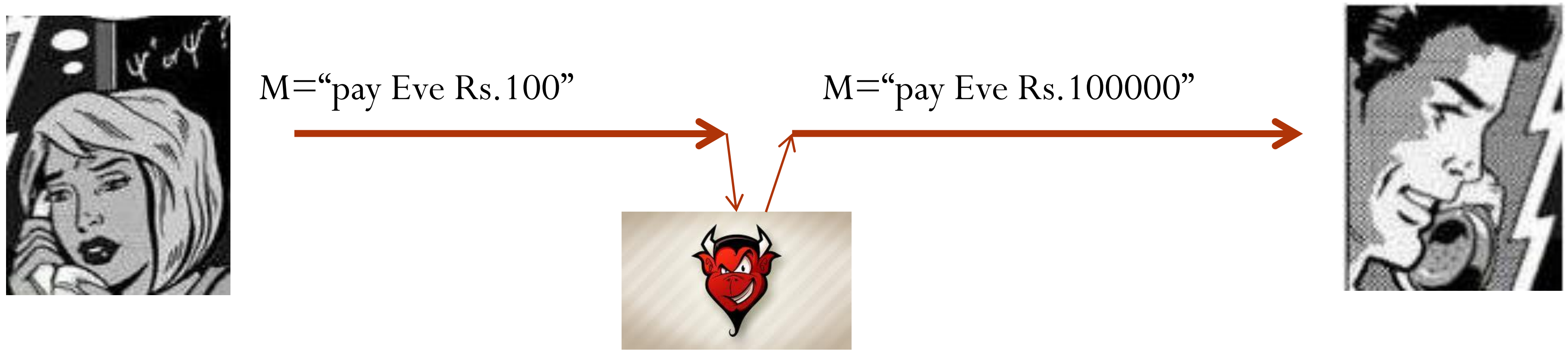
Privacy

- Alice wants to send a message to Bob without an adversary Eve figuring out the message.



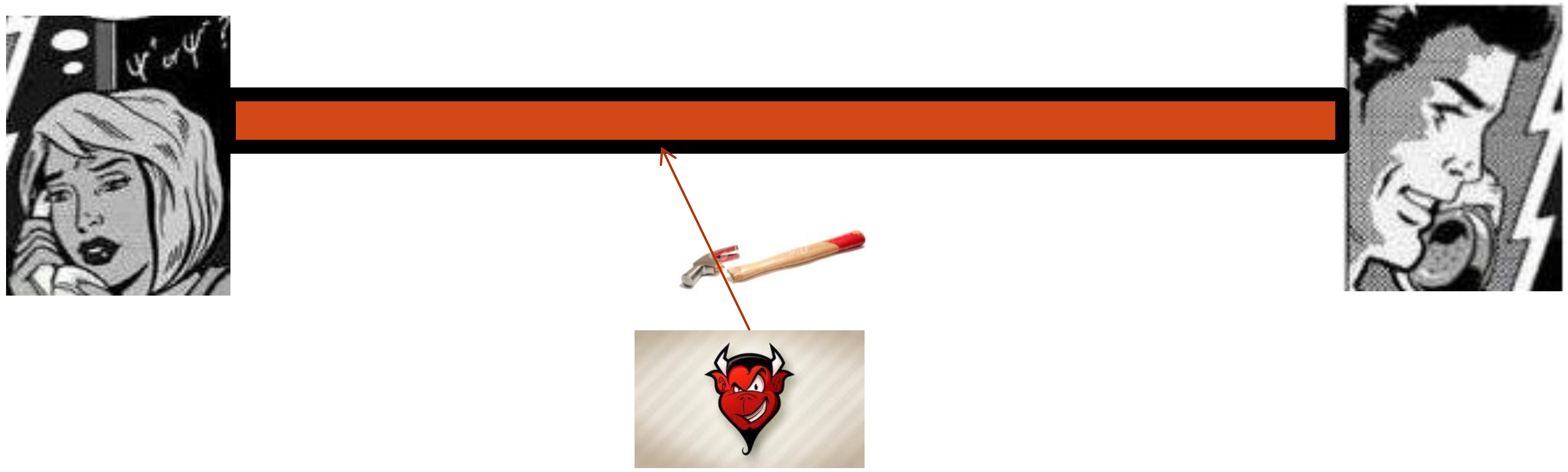
Integrity and Authenticity

- Bob wants to make sure that the message that he received from Alice is indeed sent by her and not modified during transit.



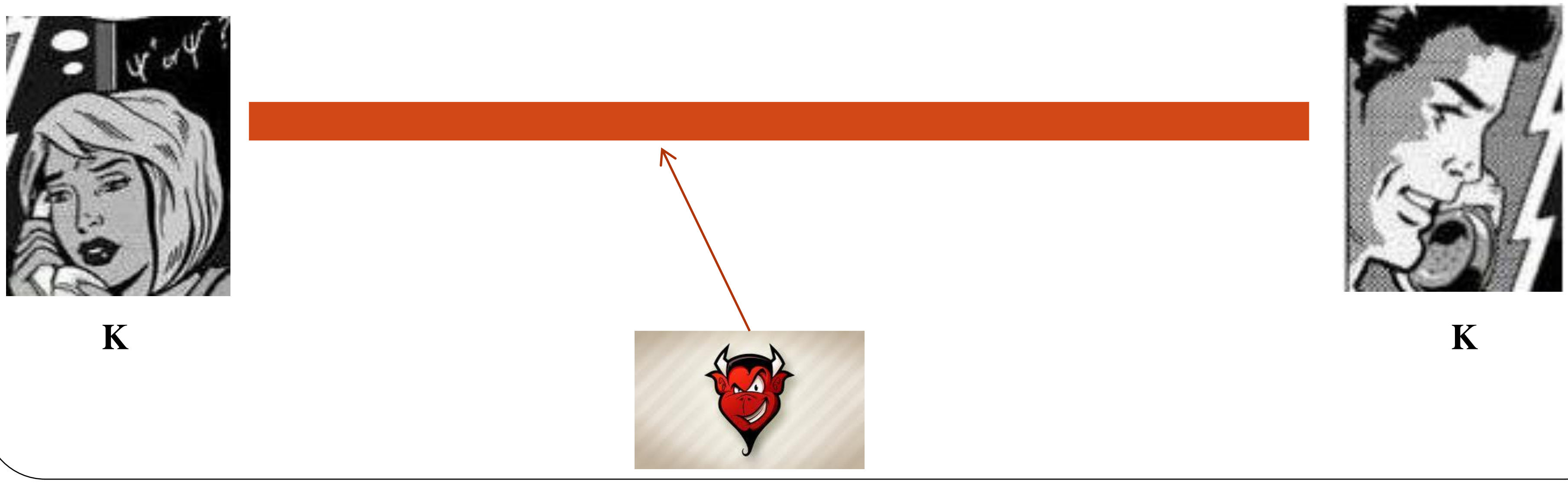
Perfect world

- There is a super-strong pipe between Alice and Bob.
  - Both privacy and authenticity goals are met.



Real world

- The channel between Alice and Bob is public.
  - Assume that Alice and Bob share some secret **K**.
  - Alice encodes her message **M** using a public encryption algorithm **E** and **K**. We write $\mathbf{C} = \mathbf{E}_{\mathbf{K}}(\mathbf{M})$.
  - Bob decrypts Alice's message using a public decryption algorithm **D** and **K**. We write $\mathbf{M} = \mathbf{D}_{\mathbf{K}}(\mathbf{C})$.



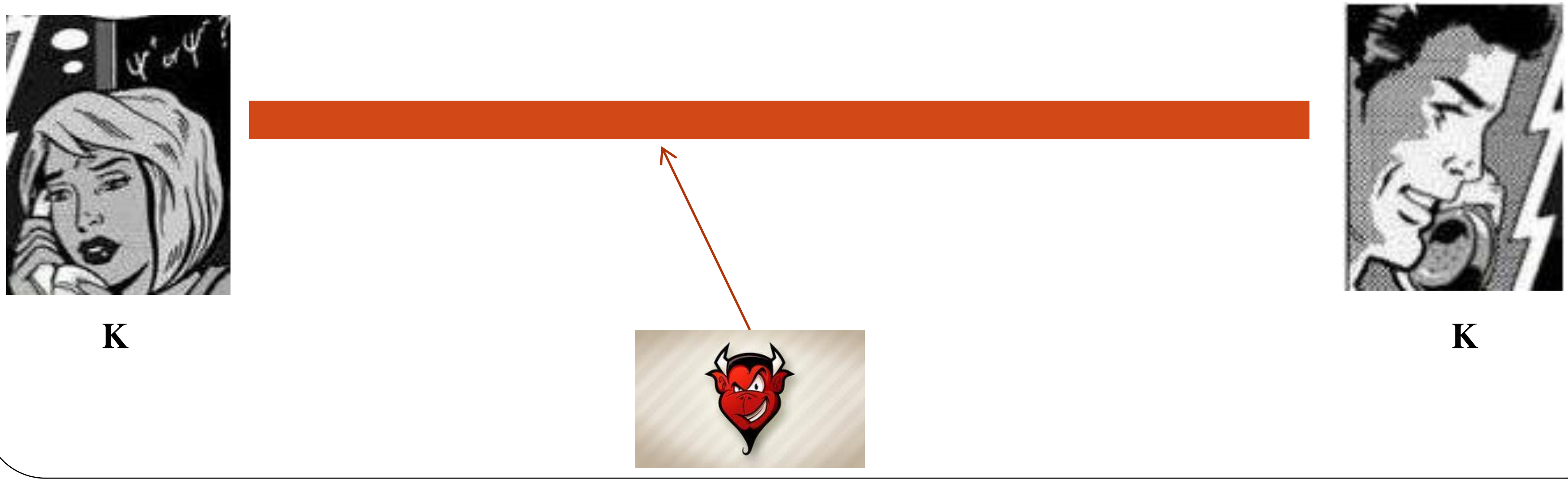
Shannon's one time pad

- $E_K(M)$ = **K** (XOR) **M** and $D_K(C)$ = **K** (XOR) **C**
  - Example:
    - **101** (XOR) **111** = **010**
    - **101** (XOR) **010** = **111**
- Is this protocol secure?

K **K**

Shannon's one time pad

- $\mathbf{E_K(M) = K}$ (XOR) $\mathbf{M}$ and $\mathbf{D_K(C) = K}$ (XOR) $\mathbf{C}$
  - Example:
    - **101** (XOR) **111** = **010**
    - **101** (XOR) **010** = **111**
- Is this protocol secure?
  - Yes. The adversary can only guess each bit with probability ½.
  - Problem: The key is as long as the message.



Pseudorandomness

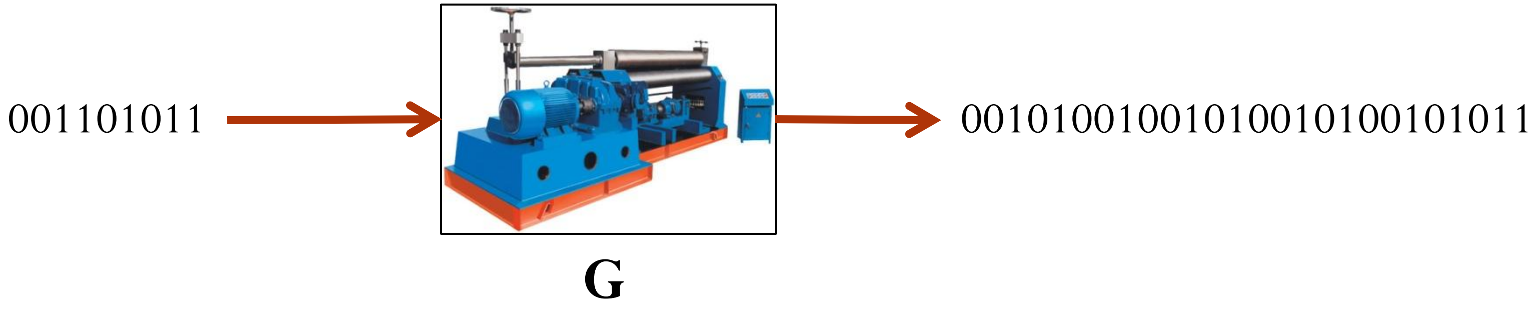
- Suppose there was a *generator* that *stretches* random bits.

$$\begin{tikzcd} 001101011 \arrow[r] & \mathbf{G} \arrow[r] & 00101001001010010100101011 \end{tikzcd}$$

- Idea:
  - Choose a short key **K** randomly.
  - Obtain **K’=G(K)**.
  - Use **K’** as key for the one time pad.
- Issue: ?

Pseudorandomness

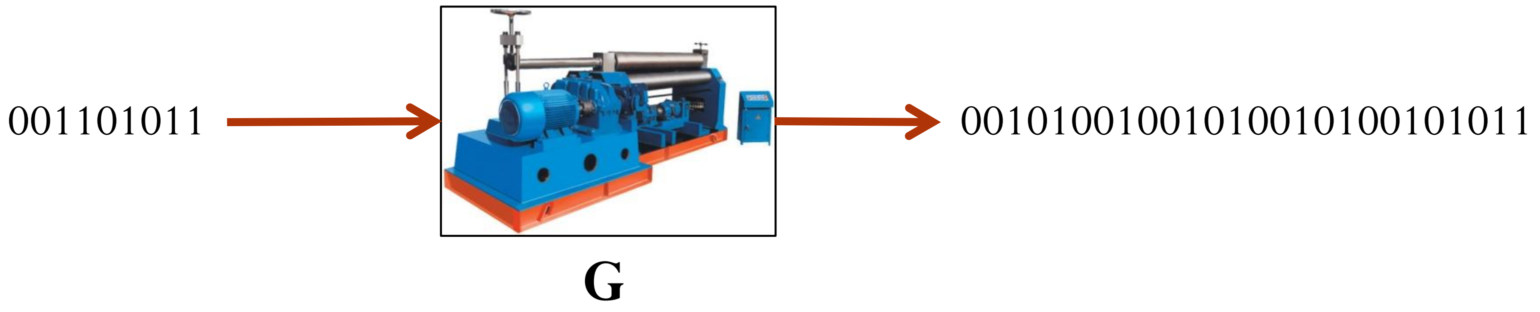
- Suppose there was a *generator* that *stretches* random bits.



- Idea:
  - Choose a short key **K** randomly.
  - Obtain **K'=G(K)**.
  - Use **K'** as key for the one time pad.
- Issue:
  - Such a generator is not possible!
  - Any such generator produces a longer string but the string is not *random*.

Pseudorandomness

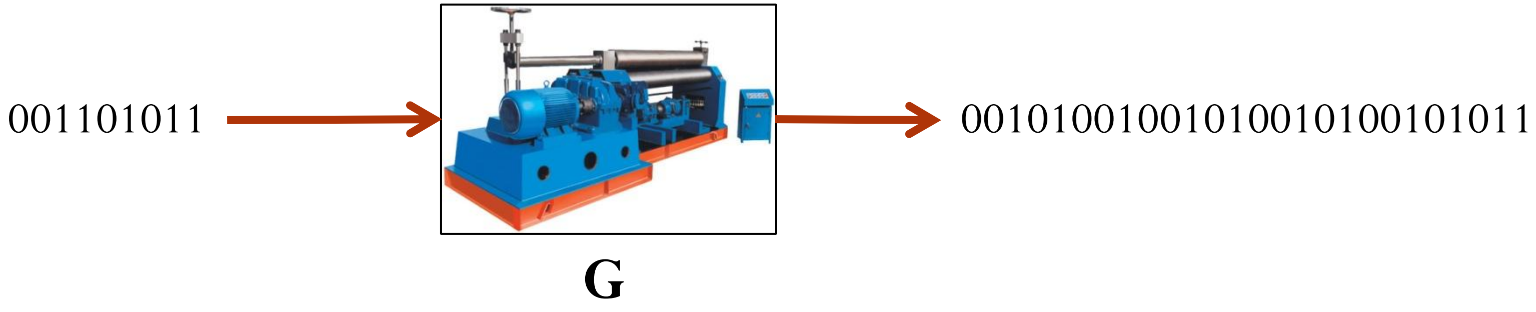
- Suppose there was a *generator* that *stretches* random bits.



- What if there is a generator that produces strings that "appear to be random". The bits are *pseudorandom*.
- General idea: The bits are not really random but they are as good as random so we'll just use them for our purpose.

Pseudorandomness

- Suppose there was a *generator* that *stretches* random bits.



- Approach for proving security:
  - Carefully define pseudorandomness (“appears to be random”).
  - Argue that if there is an adversary that *breaks* the protocol (our one time pad), then the bit string produced by **G** is not really pseudorandom.

Defining security

Privacy

Defining Security

- When do we say that an adversary has *broken* a secret communication protocol?

Defining Security

- When do we say that an adversary has *broken* a secret communication protocol?
    1. If this adversary is able to figure out the secret key.

Defining Security

- When do we say that an adversary has *broken* a secret communication protocol?
  1. If this adversary is able to figure out the secret key.
  2. If the adversary is able to figure out the secret message.

Defining Security

- When do we say that an adversary has *broken* a secret communication protocol?
  1. If this adversary is able to figure out the secret key.
  2. If the adversary is able to figure out the secret message.
  3. If the adversary is able to figure out the first bit of the message.

Defining Security

- When do we say that an adversary has *broken* a secret communication protocol?
  1. If this adversary is able to figure out the secret key.
  2. If the adversary is able to figure out the secret message.
  3. If the adversary is able to figure out the first bit of the message.
  4. If the adversary is able to figure out the xor of the first bit and 100th bit of a secret message.
  5. .
  6. .
  7. .

Defining Security

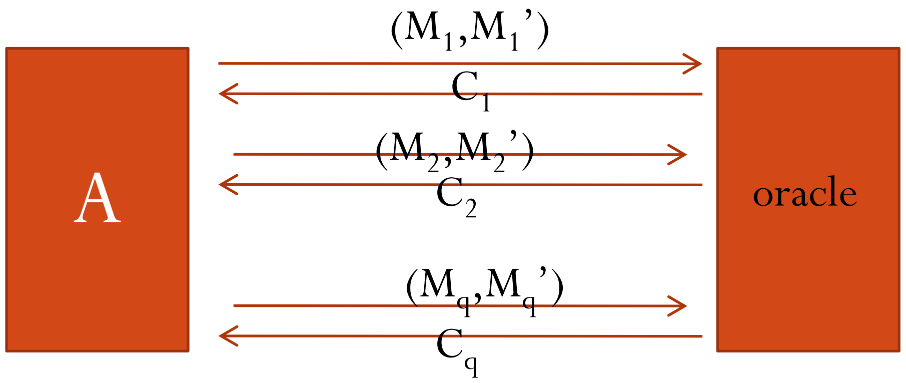
- When do we say that an adversary has *broken* a secret communication protocol?
  1. If this adversary is able to figure out the secret key.
  2. If the adversary is able to figure out the secret message.
  3. If the adversary is able to figure out the first bit of the message.
  4. If the adversary is able to figure out the xor of the first bit and 100th bit of a secret message.
  5. .
  6. .
  7. .
- We need to show that our protocol is secure against all such attacks!

Defining Security

- General Idea: Try to come up with a master security property and then follow this line of argument.
  - If our protocol is secure with respect to the master property, then we can argue that our protocol is secure against any attack in the previous slide.
  - Conversely, if there is an adversary that breaks our protocol with respect to any properties given on the previous slide, then there is another adversary that breaks the protocol with respect to the master property.
- One such master security property for privacy task is called IND-CPA

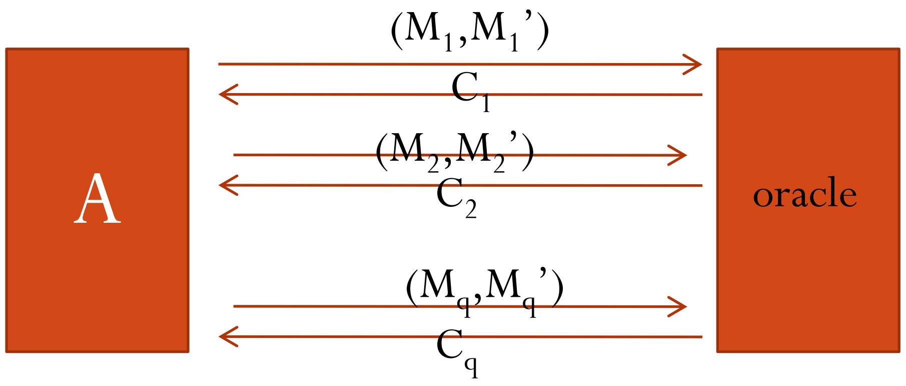
IND-CPA security

- We make the adversary **A** play a game and we say that our protocol is broken with respect to IND-CPA if **A** wins in this game.
  - We randomly choose a bit **b.**
  - We allow the adversary to send tuples of messages **(M, M')**.
  - For any tuple **(M,M')**, if **b=0** then we return the encryption of **M**, else we send the encryption of **M'**.
  - The adversary wins if it guesses the bit **b** correctly.



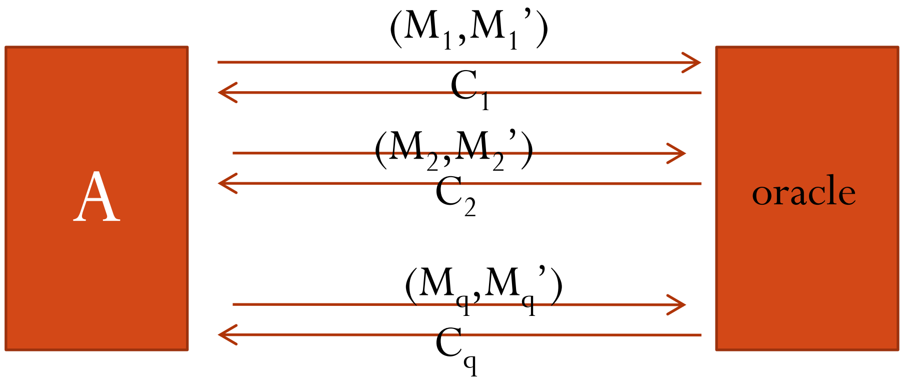
IND-CPA security

- Why is IND-CPA stronger notion of security compared to the “the secret key is not guessable” notion?



IND-CPA security

- Why is IND-CPA stronger notion of security compared to the "the secret key is not guessable" notion?
  - Suppose there is an adversary that is able to guess the secret key then there is a simple adversary that wins our game.
  - This means IND-CPA security ➔ "secret key not guessable" property.
  - We can argue similarly for all the simple security notions.



Pseudorandomness

Pseudorandomness

- How do we define "appears to be random"?

Pseudorandomness

- How do we define "appears to be random"?
  - Appears to whom?

Pseudorandomness

- How do we define "appears to be random"?
  - Appears to whom?
    - Time/space efficient algorithms.
  - Suppose there are two worlds.
    - Random world: In this world, whenever an adversary (algorithm) asks for new bits, random bits are returned.
    - Real world: In this world, whenever an adversary (algorithm) asks for new bits, small random seed **s** is chosen, and **G(s)** is returned.
  - An adversary wins if it can determine which world it is in after requesting bit strings a *few* times.

From pseudorandomness to IND-CPA security

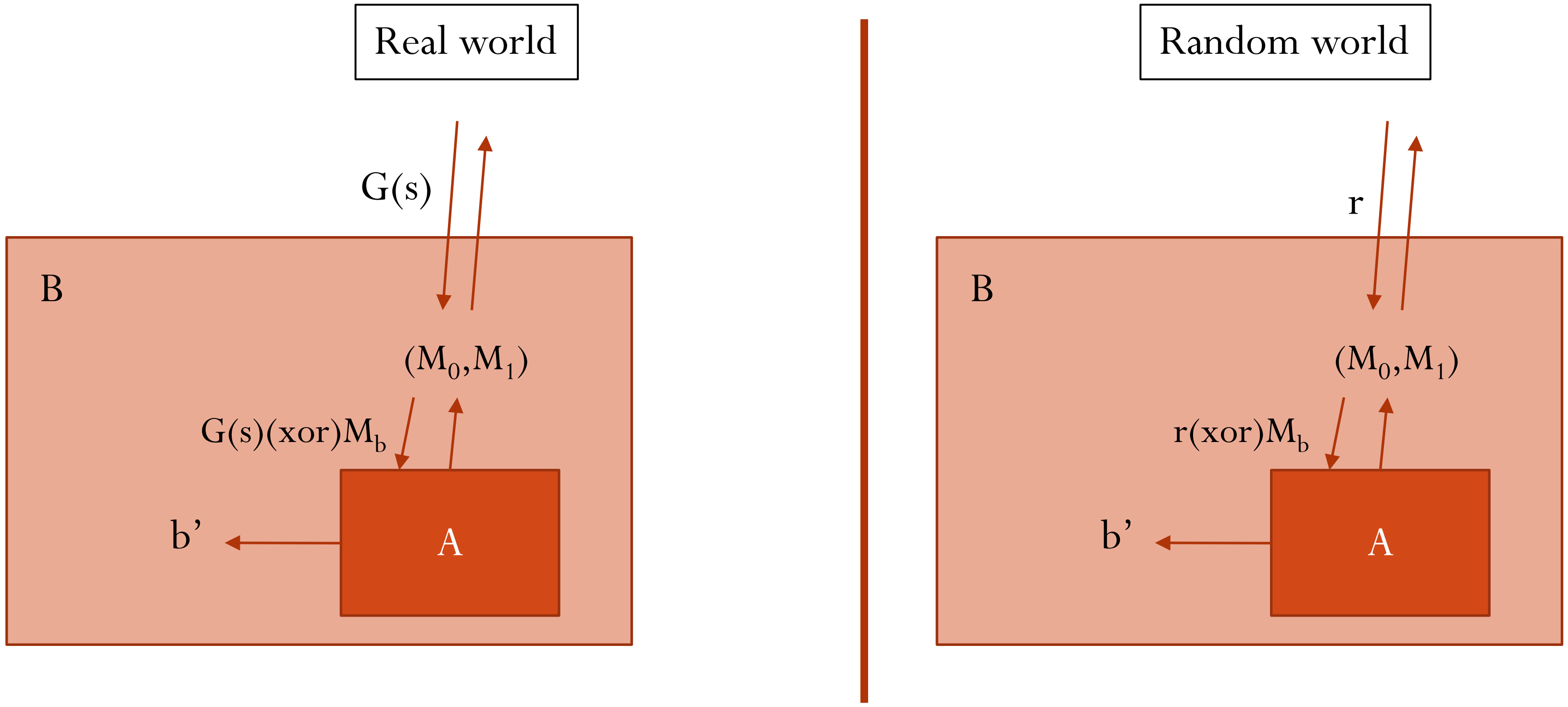
IND-CPA security from pseudorandom generator

- Suppose there *exists* a secure pseudorandom generator **G**.
  - For all efficient algorithms **A**, it cannot determine in which world it is interacting.
- We will show that the one time pad scheme with the generator **G** is secure in the IND-CPA sense.

- Main Idea: If our protocol is insecure in the IND-CPA sense, the G is not a pseudorandom generator. Alternatively, if there is an efficient algorithm **A** that wins in our IND-CPA game, then there is another efficient algorithm **B** that can determine the world it is interacting in.

IND-CPA security from pseudorandom generator

- Main Idea: Alternatively, if there is an efficient algorithm **A** that wins in our IND-CPA game, then there is another efficient algorithm **B** that can determine the world it is interacting in.



Summary

Summary: Provable security

- For any Cryptographic task, define security goals.
- Formulate a strong security property (master property).
- Argue that if certain standard cryptographic assumptions hold, then your protocol is secure with respect to the master property.
- In other words, if there is an attack on your protocol, then that means that the standard cryptographic assumption does not hold (which is unlikely).
- Some standard cryptographic assumptions:
  - AES is a pseudorandom permutation.
  - RSA is a one-way function.
  - Diffie-Hellman and discrete logarithm.

Thank You