CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

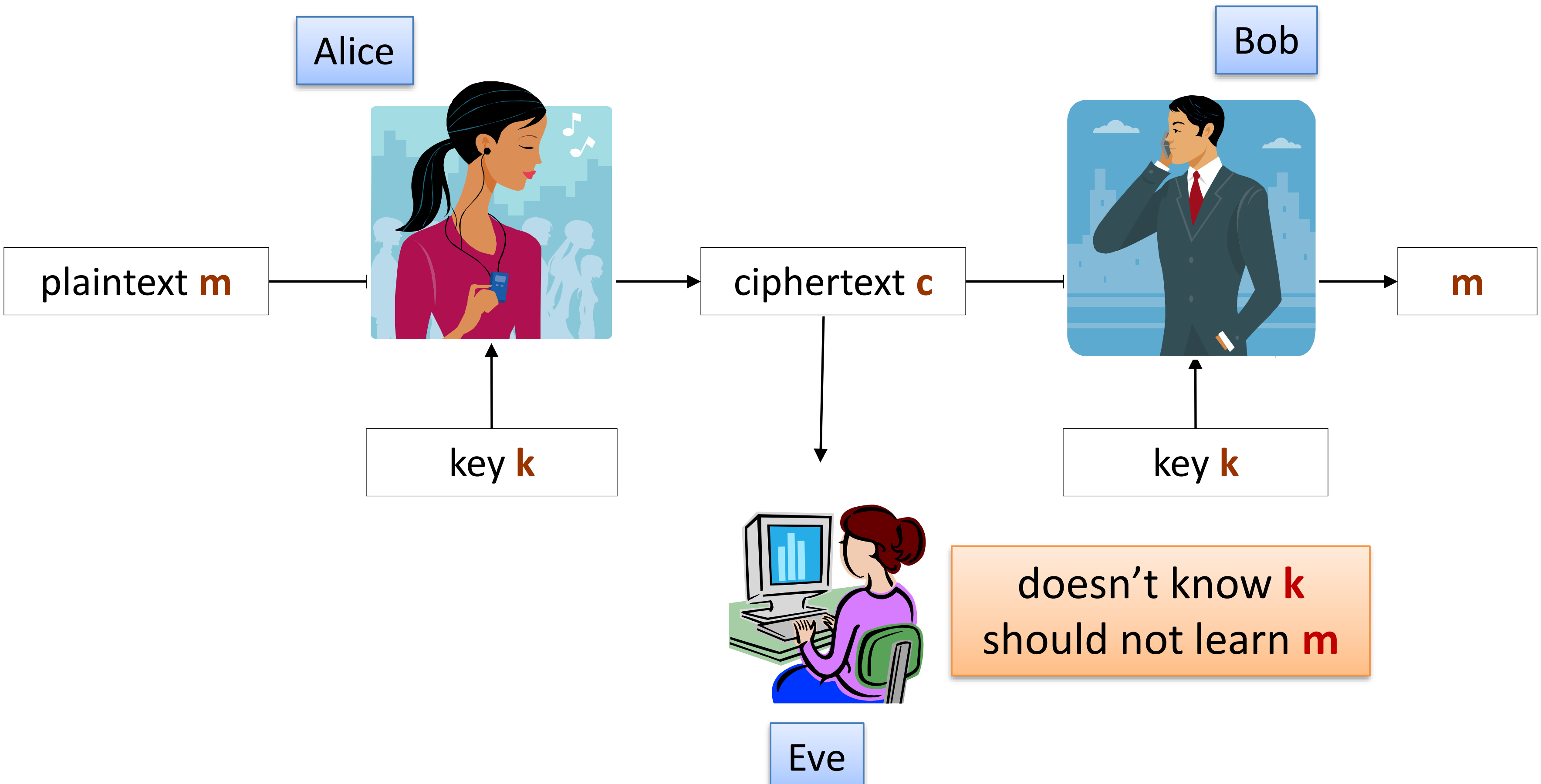
Alina Oprea
Associate Professor, CCIS
Northeastern University

January 18 2018

Outline

- Perfect security
  - Review
  - Optimality of one-time pad
- Computational security
  - Probabilistic polynomial-time attackers
  - Negligible probability of success
- Definition of security for encryption schemes
  - Security games
  - Computational indistinguishability
- Pseudorandom generators (PRG)
  - Definition
  - Constructing computational secure encryption schemes from PRG

Encryption setting



"The adversary should not learn any information about **m**."

An encryption scheme is **perfectly secret** if

for every distribution of **M**

and every $\mathbf{m} \in \mathcal{M}$ and $\mathbf{c} \in \mathcal{C}$

$$\Pr[\, M = m \,] = \Pr[\, M = m \mid C = c \,]$$

Ciphertext-only attack

Equivalently:

For all **m**, **c**: **Pr[M = m] = Pr[M = m | C = c]**

$\Updownarrow$

M and **C=Enc(K,M)** are independent

$\Updownarrow$

For every **m** , **m'** , **c** we have:

Pr[Enc(K, m) = c] = Pr[Enc(K, m') = c]

A perfectly secret scheme: one-time pad

ℓ – a parameter

$\mathcal{K} = \mathcal{M} = \{0,1\}^\ell$

component-wise **xor**

Vernam's cipher:

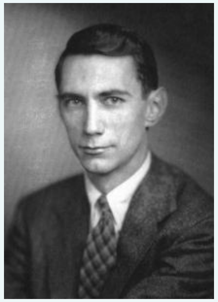
$$\mathbf{Enc}_k(m) = k \oplus m$$
$$\mathbf{Dec}_k(c) = k \oplus c$$

Gilbert Vernam

(1890 –1960)

Correctness:

$$\mathbf{Dec}_k(\mathbf{Enc}_k(m)) = k \oplus (k \oplus m)$$
$$m$$



Theorem (Shannon 1949)

"One time-pad is optimal"

In every perfectly secret encryption scheme

$$\textbf{Enc} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}\,,\ \textbf{Dec} : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$$

we have $|\mathcal{K}| \geq |\mathcal{M}|$.

Intuitive Proof:

Otherwise can do "exhaustive search". Given ciphertext c, try decrypting with every key k. Will rule-out at least 1 message and learn some information about m.

Proof:

Let M be the uniform distribution over $\mathcal{M}$ and c be some ciphertext such that $\Pr[C = c] > 0$.

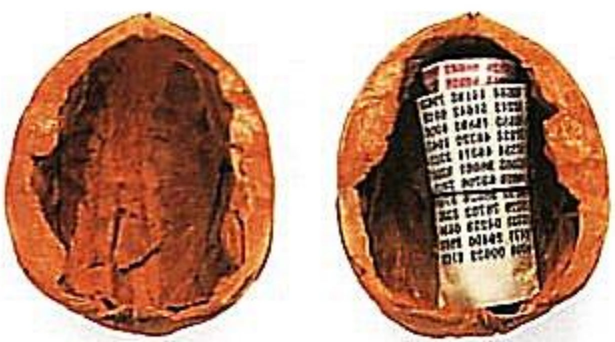
Consider the set $\mathcal{M}' = \{ \textbf{Dec}(k, c) : k \in \mathcal{K} \}$.

If $|\mathcal{K}| < |\mathcal{M}|$ then exists $m \in \mathcal{M} / \mathcal{M}'$. We have:

$$\Pr[M = m \mid C = c] = 0\,, \Pr[M = m] = 1/|\mathcal{M}|.$$

Practicality?

Generally, the **one-time pad** is **not very practical**, since the key has to be as long as the **total** length of the encrypted messages.



a **KGB** one-time pad hidden in a walnut shell

However, it is sometimes used because of the following advantages:

- **perfect secrecy**,
- short messages can be encrypted using **pencil and paper** .

In the 1960s the Americans and the Soviets established a hotline that was encrypted using the one-time pad.

Venona project (1946 – 1980)

Ethel and Julius Rosenberg

American **National Security Agency** decrypted **Soviet** messages that were transmitted in the 1940s.

That was possible because the Soviets reused the keys in the one-time pad scheme.

Outlook

- Saw: limits of “perfect” or “statistical” security.

- Are there other meaningful security notions?

“Real” cryptography starts here!

Restriction: **Eve is computationally-bounded**

We will construct schemes that in **principle can be broken** if the adversary has a huge computing power or is extremely lucky.

- E.g., break the scheme by enumerating all possible secret keys. (“**brute force attack**”)
- E.g., break the scheme by guessing the secret key.

Goal: cannot be broken with reasonable computing power with reasonable probability.

Computationally-bounded adversary

Eve is computationally-bounded

But what does it mean?

Ideas:

- "She has can use at most 1000
  Intel Core 2 Extreme X6800 Dual Core Processors
  for at most **100** years..."

- "She can buy equipment worth **$10 million** and use it for **30** years..".

it's hard to reason
formally about it

First idea – concrete security

Adversary runs for limited amount of time t.

More generally, we could have definitions of a type:

"a system **<u>X is (t,ε)-secure</u>** if every **adversary**

that operates in time **t**

can break it with probability at most **ε**."

This would be mathematically precise, **but...**

- Exact run-time is not very robust
- Depends on low-level details of hardware and changes over time
- Does not consider parallelization or other computing paradigm shifts

*<u>Difficult to work with, **ugly formulas**...</u>*

What to do?

Idea:

t steps of an algorithm → "**efficient computation**"

ε → a value "**very close to zero**".

How to formalize it?

Use the asymptotics, as in complexity theory!

Efficiently computable?

“efficiently computable” = “polynomial-time algorithm”

Polynomial in what?

Security Parameter n

- A flexible parameter that dictates the security of the scheme.
- The scheme and the attacker get n (for example the key length)

Probabilistic algorithms

- Our cryptosystems rely on randomness
- The attacker should also get randomness

Probabilistic Polynomial Time (PPT) Algorithms

Very small probability?

"very small"

=

"negligible"

=

approaches 0 faster than the inverse of any polynomial

<u>Formally</u>

A function $\mu : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for every positive integer c there exists an integer n_0 s.t. for all integer $n > n_0$

$$\mu(n) < \frac{1}{n^c}$$

Negligible or not?

$$f(n) := \frac{1}{n^2}$$

$$f(n) := 2^{-n}$$

$$f(n) := 2^{-\sqrt{n}}$$

$$f(n) := n^{-\log n}$$

$$f(n) := \frac{1}{n^{1000}}$$

Nice properties of these notions

- A sum of two polynomials is a polynomial:
  poly + poly = poly
- A product of two polynomials is a polynomial:
  poly * poly = poly
- A sum of two negligible functions is a negligible function:
  negl + negl = negl

Moreover:

- A negligible function multiplied by a polynomial is negligible
  negl * poly = negl

Computational Security

Typically, we will say that a **scheme C is secure** if

$\forall$

Probabilistic polynomial-time algorithm A

Pr[A(n) "breaks the scheme" C(n)] is **negligible** in n.

- Scheme **C** and the **adversary A** take input **security parameter**.
- 2 relaxations of perfect security
  - PPT adversary
  - Adversary can succeed, but with very small probability (negligible)

Example

security parameter **n** = the length of the secret key **k**

in other words: **k** is always a random element of $\{0,1\}^n$

Adversary can always **guess** **k**.

- Running time is polynomial.
- Probability of success is 2^{-n} = **negligible**.

Adversary can **enumerate all possible keys** **k.**
(the "brute force" attack)

- Probability of success is **1**.
- Running time is 2^n (not polynomial).

Computational security is resilient against these

Is this the right approach?

Advantages

1. Polynomial time is well-established notion in complexity theory and algorithm analysis.
2. The formulas get much simpler.

Disadvantage

Asymptotic results don't tell us anything about security of the **concrete systems**.

However

Usually one can prove **formally** an asymptotic result and then argue **informally** how to choose the "security parameter"

(and can be calculated based on best attacks).

Computationally Secure Encryption

$\mathcal{K}$ – **key** space, $\mathcal{M}$ – **plaintext** space, $\mathcal{C}$ - **ciphertext** space

All spaces can be parameterized by security parameter n.

Often consider $\mathcal{K} = \{0,1\}^n$, $\mathcal{M} = \mathcal{C} = \{0,1\}^*$

An **encryption scheme** is a tuple **(Gen, Enc,Dec)**, where

- **Gen:** $\mathcal{N} \rightarrow \mathcal{K}$, **Enc :** $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, **Dec :** $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$

Algorithms **Enc** and **Dec** can be randomized. Usually **Dec** is deterministic

Correctness

For every **k, m** we should have **Pr[** $Dec_k(Enc_k(m)) = m$ **] =1**.

Perfect vs. Computational Security

we will require that m_0,m_1 are chosen by a **poly-time adversary**

Recall: An encryption scheme is **perfectly secret** if for all m_0,m_1, c

$$\Pr[\mathrm{Enc}(K, m_0) = c] = \Pr[\mathrm{Enc}(K, m_1) = c]$$

Meaning: no attacker can distinguish $\mathrm{Enc}(K, m_0)$ from $\mathrm{Enc}(K, m_1)$

New: no PPT attacker can distinguish $\mathrm{Enc}(K, m_0)$ from $\mathrm{Enc}(K, m_1)$ with better then negligible probability.

Security Game

Π= (Gen,Enc,Dec): an encryption scheme

security parameter
n

PPT Adversary A

Alice
Challenger

chooses $\mathbf{m_0,m_1}$ such that $\mathbf{|m_0|=|m_1|}$

$\mathbf{m_0,m_1}$

1. Choose **k** ← **Gen(n)**
2. chooses random **b** ← **{0,1}**
3. calculate **c** ← $\mathbf{Enc(k,m_b)}$

c

Makes a guess **b’**

Security definition:
We say that **(Gen,Enc,Dec)** is **indistinguishable against eavesdropping (EAV-secure)** if for any **polynomial time** adversary, **| Pr[b’=b] - ½ |** is negligible in n.

The security definition

- Experiment $\mathrm{Exp}_{\Pi,A}^{\mathrm{EAV}}(n)$:
  1. Choose $k \leftarrow Gen(n)$
  2. $m_0, m_1 \leftarrow A_1\ (n)$
  3. $b \leftarrow^R \{0,1\}; c \leftarrow Enc_k(m_b)$
  4. $b' \leftarrow A_2\ (m_0, m_1, c)$
  5. Output 1 if $b = b'$ and 0 otherwise

We say that **(Gen, Enc,Dec)** is **EAV-secure** (secure against eavesdropping) if:

For every **PPT** adversary $A = (A_1, A_2)$:

|Pr[$\mathrm{Exp}_{\Pi,A}^{\mathrm{EAV}}(n)$ = 1]- ½ | negligible in n

Testing the definition

Suppose the adversary can compute k **from** Enc(k,m). Can he win the game?

YES!

Suppose the adversary can compute **some bit of** m from Enc(k,m). Can he win the game?

YES!

Is it possible to prove security?

Bad news:

<u>Theorem</u>

If EAV-secure encryption exists
with $|k| < |m|$

then

$P \neq NP$

Long-standing open problem

What can we prove?

We can't prove security of crypto schemes from scratch. But…

- Can prove security of a complicated primitive assuming security of a simpler one.

- Can prove security of a primitive assuming some basic algorithmic task is computationally hard.

This is what modern cryptography is all about

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/