{ CSCI 6331 · 4331 | Lecture 2 }

Cryptography

Hoeteck Wee · hoeteck@gwu.edu

http://tinyurl.com/cryptogw/

Announcements

- Sign up for Piazza (participation affects final grade)
- Homework 1 is up (updated today, added new question)

Vigenère Cipher

Vignère Cipher.

- Key space = strings of letters, e.g. `SECRET`
- Generate a pad by repeating the key, use as one-time pad (`A=0`,...):

```
THIS IS THE PLAINTEXT TO BE ENCRYPTED
SECR ET SEC RETSECRET SE CR ETSECRETS
LLKJ ML LLG GPTARVVBM LS DV IGUVAGXXV
```

Cipertext-Only Attack. (assuming English language)

- Guess the key length, e.g. 6. What happens every 6'th letter? (shift cipher)

Chosen-Plaintext Attack. (adversary obtains encryption of plaintexts of its choice)

- Which plaintext?

Perfectly Secure Encryption

Goal. Protect secrecy of communication

Trust Model. Three entities, sender, receiver. Public channel, no tampering.

Who is the adversary? Eavesdroper.

Powers? Cipertext-only attacks. (adversary observes a single ciphertext)

What constitutes a break? Learns more about plaintext after seeing ciphertext.

- example: substitution cipher. Ciphertext `A A` reveals both letters are the same.

Perfectly Secure Encryption

Criterion. a perfect cipher / perfect secrecy

- for all $\mathrm{m}_1, \mathrm{m}_2 \in \mathcal{M}$, for all ciphertexts c,

$$\Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_1) = \mathrm{c}] = \Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_2) = \mathrm{c}]$$

- Idea. Regardless of plaintext, adversary sees same distribution of ciphertexts

One-Time Pad. example of perfect cipher

- $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^{\ell}$; Gen outputs a random ℓ-bit string k
- $\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}) = \mathrm{k} \oplus \mathrm{m}$ (bit-wise XOR)

Q1. What is $\Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(0^{\ell}) = 0^{\ell}]$? A. equals $\Pr_{\mathrm{k}}[\mathrm{k} = 0^{\ell}] = 1/2^{\ell}$

Q2. take any $\mathrm{m}, \mathrm{c} \in \{0,1\}^{\ell}$. What is $\Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}) = \mathrm{c}]$?

A. equals $\Pr_{\mathrm{k}}[\mathrm{k} = \mathrm{m} \oplus \mathrm{c}] = 1/2^{\ell}$

Perfectly Secure Encryption

Criterion. a perfect cipher / perfect secrecy

- for all $\mathrm{m}_1, \mathrm{m}_2 \in \mathcal{M}$, for all ciphertexts c,

$$\Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_1) = \mathrm{c}] = \Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_2) = \mathrm{c}]$$

- Idea. Regardless of plaintext, adversary sees same distribution of ciphertexts

One-Time Pad. example of perfect cipher

- $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^{\ell}$; Gen outputs a random ℓ-bit string k
- $\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}) = \mathrm{k} \oplus \mathrm{m}$ (bit-wise XOR)

Theorem. For a perfect cipher, the number of keys is at least the size of message space

- Corollary. Key length of one-time pad is optimal.
- In practice, we want to encrypt gigabytes of data.

  $\Longrightarrow$ need gigabyte-long key for perfect secrecy! (hard to store)

Computational Security

Computational Security. "practically unbreakable" (vs *impossible* to break)

— e.g. takes fastest available supercomputer > 200 years to break

- security against all *efficient* adversaries running in feasible amount of time
- adversary's success probability is tiny, e.g. $< 2^{-80}$
- Note. not enough to defeat a set of statistical tests, e.g. frequency analysis

More formally.. a scheme is (t, ϵ)-secure if every adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ

- practice: $t = 2^{80}$ CPU cycles, $\epsilon = 2^{-80}$ (2^{60} cycles ~ 35 years on 1 GHz PC)

  Q. What happens in 20 years with 200 GHz computers?
- theory: $t =$ polynomial-time algorithms,

  ϵ = negligible, i. e. smaller than any inverse polynomial

Computational Secrecy

Perfect Secrecy.

- for all $\mathrm{m}_1, \mathrm{m}_2 \in \mathcal{M}$, for all ciphertexts c,

$$\Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_1) = \mathrm{c}] = \Pr_{\mathrm{k}}[\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_2) = \mathrm{c}]$$

- Idea. Regardless of plaintext, adversary sees same distribution of ciphertexts

Computational Secrecy. (t, ϵ)-hard to distinguish encryptions of m_1 vs m_2.

- for all $\mathrm{m}_1, \mathrm{m}_2 \in \mathcal{M}$, for all adversaries A running in time t:

$$|\Pr_{\mathrm{k}}[\mathrm{A}(\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_1)) = 1] - \Pr_{\mathrm{k}}[\mathrm{A}(\mathrm{Enc}_{\mathrm{k}}(\mathrm{m}_2)) = 1]| < \epsilon$$

- Note. implied by perfect secrecy

Q. How to construct ciphers with short keys achieving computational secrecy?

— new tool: pseudorandom generator

Computational Secrecy: Pseudorandomness

Pseudorandom Generator (PRG). generates strings that "look random"

- $\mathrm{G} : \{0,1\}^{\mathrm{k}} \to \{0,1\}^{\mathrm{m}}$ where $\mathrm{k} < \mathrm{m}$ and G efficiently computable
- (t, ϵ)-hard to distinguish output of G from truly random string
- for all adversaries A running in time t:

$$\left| \Pr_{\mathrm{s} \leftarrow \{0,1\}^{\mathrm{k}}} [\mathrm{A}(\mathrm{G}(\mathrm{s})) = 1] - \Pr_{\mathrm{u} \leftarrow \{0,1\}^{\mathrm{m}}} [\mathrm{A}(\mathrm{u}) = 1] \right| < \epsilon$$

- example: $\mathrm{k} = 80, \mathrm{m} = 128, \epsilon = 0.1$; A outputs the majority of bits.

  Q. What is $\Pr_{\mathrm{s} \leftarrow \{0,1\}^{80}} [\mathrm{A}(\mathrm{G}(\mathrm{s})) = 1]$?

Computational Secrecy: Encryption with PRG

Pseudorandom Generator (PRG). generates strings that "look random"

- $\mathrm{G} : \{0,1\}^{\mathrm{k}} \to \{0,1\}^{\mathrm{m}}$ where $\mathrm{k} < \mathrm{m}$ and G efficiently computable
- (t, ϵ)-hard to distinguish output of G from truly random string

Using PRG for encryption. use output of PRG as a one-time pad

- $\mathcal{M} = \{0,1\}^{\mathrm{m}}, \mathcal{K} = \{0,1\}^{\mathrm{k}}, \mathcal{C} \subset \{0,1\}^{\mathrm{m}}$
- Gen outputs a random k-bit string s
- $\mathrm{Enc}_{\mathrm{s}}(\mathrm{m}) = \mathrm{G}(\mathrm{s}) \oplus \mathrm{m}$ (What is $\mathrm{Dec}_{\mathrm{s}}(\mathrm{c})$?)

Claim. If G is (t, ϵ)-secure, then the cipher is (t, ϵ)-secure too.

- Proof by reduction: if we can break security of encryption (distinguish encryptions of m_1 and m_2), then we can break security of PRG (distinguish from random)

Computational Secrecy: Encryption with PRG

Pseudorandom Generator (PRG). generates strings that "look random"

- $\mathrm{G} : \{0,1\}^{\mathrm{k}} \to \{0,1\}^{\mathrm{m}}$ where $\mathrm{k} < \mathrm{m}$ and G efficiently computable
- (t, ϵ)-hard to distinguish output of G from truly random string

Using PRG for encryption. use output of PRG as a one-time pad

- $\mathcal{M} = \{0,1\}^{\mathrm{m}}, \mathcal{K} = \{0,1\}^{\mathrm{k}}, \mathcal{C} \subset \{0,1\}^{\mathrm{m}}$
- Gen outputs a random k-bit string s
- $\mathrm{Enc}_{\mathrm{s}}(\mathrm{m}) = \mathrm{G}(\mathrm{s}) \oplus \mathrm{m}$ (What is $\mathrm{Dec}_{\mathrm{s}}(\mathrm{c})$?)

Claim. If G is (t, ϵ)-secure, then the cipher is (t, ϵ)-secure too.

- Proof by reduction: if we can break security of encryption (distinguish encryptions of m_1 and m_2), then we can break security of PRG (distinguish from random)

Computational Secrecy: Encryption with PRG

Pseudorandom Generator (PRG). generates strings that "look random"

- $\mathrm{G} : \{0,1\}^{\mathrm{k}} \to \{0,1\}^{\mathrm{m}}$ where $\mathrm{k} < \mathrm{m}$ and G efficiently computable
- (t, ϵ)-hard to distinguish output of G from truly random string

Using PRG for encryption. use output of PRG as a one-time pad

- $\mathcal{M} = \{0,1\}^{\mathrm{m}}, \mathcal{K} = \{0,1\}^{\mathrm{k}}, \mathcal{C} \subset \{0,1\}^{\mathrm{m}}$
- Gen outputs a random k-bit string s
- $\mathrm{Enc}_{\mathrm{s}}(\mathrm{m}) = \mathrm{G}(\mathrm{s}) \oplus \mathrm{m}$ (What is $\mathrm{Dec}_{\mathrm{s}}(\mathrm{c})$?)

Claim. If G is (t, ϵ)-secure, then the cipher is (t, ϵ)-secure too.

- Proof by reduction: if we can break security of encryption (distinguish encryptions of m_1 and m_2), then we can break security of PRG (distinguish from random)

Computational Secrecy: Encryption with PRG

Using PRG for encryption. use output of PRG as a one-time pad

- $\mathcal{M} = \{0,1\}^m, \mathcal{K} = \{0,1\}^k, \mathcal{C} \subset \{0,1\}^m$
- Gen outputs a random k-bit string s
- $\mathrm{Enc}_s(m) = G(s) \oplus m$; $\mathrm{Dec}_s(c) = c \oplus G(s)$

Claim. If G is (t, ϵ)-secure, then the cipher is (t, ϵ)-secure too.

- if $G(s)$ "looks like" random, then $\mathrm{Enc}_k(m_1)$ "looks like" $\mathrm{Enc}_k(m_2)$
- Consider the four distributions:
  1. encrypt m_1 with pseudorandom one-time pad $G(s)$
  2. encrypt m_1 with truly random one-time pad
  3. encrypt m_2 with truly random one-time pad
  4. encrypt m_2 with pseudorandom one-time pad $G(s)$

PRG in Practice

Pseudorandom Generators in Practice

- heuristic design
- very efficient
- specific input/key lengths and output lengths

  Q. can we extend the output length?

Stream ciphers meant to implement a PRG

- examples: A5, RC4, SEAL
- intended for encryption in the same way as OTP
- single functions, require synchronization

Block Ciphers

Block Ciphers. family of functions $\mathcal{K} \times \mathrm{X} \to \mathrm{X}$

- AES: $\mathcal{K} = \mathrm{X} = \{0,1\}^{128}$
- DES: $\mathrm{X} = \{0,1\}^{64}, \mathcal{K} = \{0,1\}^{56}$
- 3DES: $\mathrm{X} = \{0,1\}^{64}, \mathcal{K} = \{0,1\}^{168}$