CSC110 Lecture 18: Introduction to Cryptography

David Liu, Department of Computer Science

Navigation tip for web slides: press ***?*** *to see keyboard navigation controls.*

Announcements and Today’s plan

Announcements

- Assignment 3 has been posted
  - Check out the A3 FAQ (+ corrections)
  - Additional TA office hours
  - Review advice on academic integrity
- Term Test 2 is next Monday!
  - Check out the Term Test 2 Info Page
    - Test time and location (not MY 150!)
    - Test coverage
    - Advice for preparing for the test
  - Review the posted reference sheet (this will be provided to you at the test!)
- PythonTA survey 1

Announcements

No tutorial this Friday! (To give you more time for Assignment 3/Term Test 2.)

We will post last year's tutorial for additional practice with this week's material.

Today you'll learn to...

1. Define the components and requirements of a secure symmetric-key cryptosystem.
2. Define and implement the one-time pad symmetric-key cryptosystem.
3. Define and trace the Diffie-Hellman key exchange algorithm.
4. Define the terms perfect secrecy and discrete logarithm problem, and explain how they are related to the algorithms we study today.

Reviewing symmetric-key cryptosystems

Encryption and decryption

Two people, Alice and Bob, want to communicate with each other.

Alice and Bob share a **secret key** $k \in \mathcal{K}$.

Alice **encrypts** a plaintext message $m \in \mathcal{P}$ using k to obtain a ciphertext $c \in \mathcal{C}$, and sends c to Bob.

Bob **decrypts** the ciphertext c using k to obtain the original plaintext message m.

Two properties for a symmetric-key cryptosystem

Correctness

For all $k \in \mathcal{K}$ and $m \in \mathcal{P}$, $Decrypt(k, Encrypt(k, m)) = m$.

Security

For all $k \in \mathcal{K}$ and $m \in \mathcal{P}$, if an eavesdropper only knows the value of $c = Encrypt(k, m)$ but does not know k, it is computationally infeasible to find m.

Example: Caesar cipher

Plaintext and ciphertext messages are strings of ASCII characters.

Secret key k is a numeric shift of each letter:

$$c[i] = (m[i] + k) \ \% \ 128$$

The One-Time Pad Cryptosystem

Problem with the Caesar cipher

Ciphertext: `'OLaTO+T^+NZZW'`

0	1	2	3	4	5	6	7	8	9	10	11	12
O	L	a	T	O	+	T	^	+	N	Z	Z	W

Any cryptosystem based on character substitution reveals information about the structure of the original message.

The one-time pad cryptosystem

The secret key k is now a string. We can encrypt a message m up to the same length as k:

$$c[i] = (m[i] + k[i]) \ \% \ 128$$

We call the secret key a “one-time pad”.

Example

Encrypt message `'HELLO'` with secret key `'david'`.

Plaintext		Secret key		Ciphertext	
H	72	d	100	$(72 + 100) \% 128 = 44$	,
E	69	a	97	$(69 + 97) \% 128 = 38$	&
L	76	v	118	$(76 + 118) \% 128 = 66$	B
L	76	i	105	$(76 + 105) \% 128 = 53$	5
O	79	d	100	$(79 + 100) \% 128 = 51$	3

Exercise 1: The One-Time Pad Cryptosystem

Perfect secrecy

Given the ciphertext `'AAAAA'` from the one-time pad encryption, what plaintext message could we have started with?

For every string of length 5, there exists a secret key that yields the ciphertext `'AAAAA'`.

The one-time pad has **perfect secrecy**: the ciphertext yields no information about the plaintext. An eavesdropper seeing the ciphertext can't determine anything about the plaintext!

Limitations of the one-time pad cryptosystem

1. The length of the secret key must be $\geq$ the length of the plaintext message.
2. If a secret key is reused, we no longer have perfect secrecy.

Stream ciphers

Stream ciphers are based on the one-time pad, but use a small shared secret key as a starting point to generate new “random” numbers.

Example: starting with the integer key $k \in \{1, 2, \ldots, 127\}$, generate the sequence

$$k, (k^2 \% 128), (k^3 \% 128), (k^4 \% 128), \ldots$$

But modular exponentiation repeats—not a “random” sequence!

Establishing shared keys

Two people want to use a symmetric-key cryptosystem to communicate securely.

Problem: how do they establish a shared secret key?

The Diffie-Hellman key exchange algorithm

The **Diffie-Hellman key exchange algorithm** is an algorithm that allows two people to establish a shared secret key while only communicating publicly.

Diffie-Hellman (Step 1)

Context: David and you (yes, you!) want to establish a shared secret key, but can only communicate publicly.

1. David chooses p, a prime number greater than 2, and $g \in \{2, 3, \ldots, p-1\}$. David sends p and g to you.

$p = 6553$, and $g = 10$

Diffie-Hellman (Step 2)

2. David chooses a secret number a, and sends you $A = g^a \% p$.

$A = 6433$ (but I'm not sending a!)

Diffie-Hellman (Step 3)

3. You choose a secret number b, and send David $B = g^b \,\%\, p$.

Type your B (but not b) into the Campuswire chat!

(Remember, $p = 6553$, and $g = 10$.)

Diffie-Hellman (Step 4)

4. David calculates $k_A = B^a \,\%\, p$. You calculate $k_B = A^b \,\%\, p$.

   $k_A = k_B$, and this is our shared secret key!

(Remember, $p = 6553$, $g = 10$, and $A = 6433$.)
Moment of truth!

Why is Diffie-Hellman correct?

David has p, g.

David has a.

David has $B = g^b \% p$.

David has $k_A = B^a \% p$.

You have p, g.

You have $A = g^a \% p$.

You have b.

You have $k_B = A^b \% p$.

Theorem (Correctness of Diffie-Hellman key exchange).
For all $p, g, a, b \in \mathbb{Z}^+$, $(g^b \% p)^a \% p = (g^a \% p)^b \% p$.

Proof key idea (see Section 8.3 for full proof):

$$(g^a)^b \equiv g^{ab} \equiv (g^b)^a \pmod{p}$$

Why is Diffie-Hellman secure?

David has:	**You** have:	**Eavesdropper** has:
p, g	p, g	p, g
a	$A = g^a \,\%\, p$	$A = g^a \,\%\, p$
$B = g^b \,\%\, p$	b	$B = g^b \,\%\, p$
$k_A = B^a \,\%\, p$	$k_B = A^b \,\%\, p$	...?

E.g., eavesdropper has $p = 6553$, $g = 10$, $A = 6433$, and your B.

From p, g, A, and B, can the eavesdropper compute k_A/k_B?

Or, given $g^a \,\%\, p$ and $g^b \,\%\, p$, can the eavesdropper compute $g^{ab} \,\%\, p$?

Why is Diffie-Hellman secure?

Discrete logarithm problem: given $p, g, A \in \mathbb{Z}^+$, find $a \in \mathbb{Z}^+$ such that $g^a \equiv A \pmod{p}$, if such an a exists.

There is no known efficient algorithm for solving the discrete logarithm problem!

We say that Diffie-Hellman is **computationally secure**: for large enough primes (e.g., $p \approx 2^{2048}$), there is no computationally efficient way of determining the secret key from just the public communication.

Exercise 2: The Diffie-Hellman key exchange algorithm

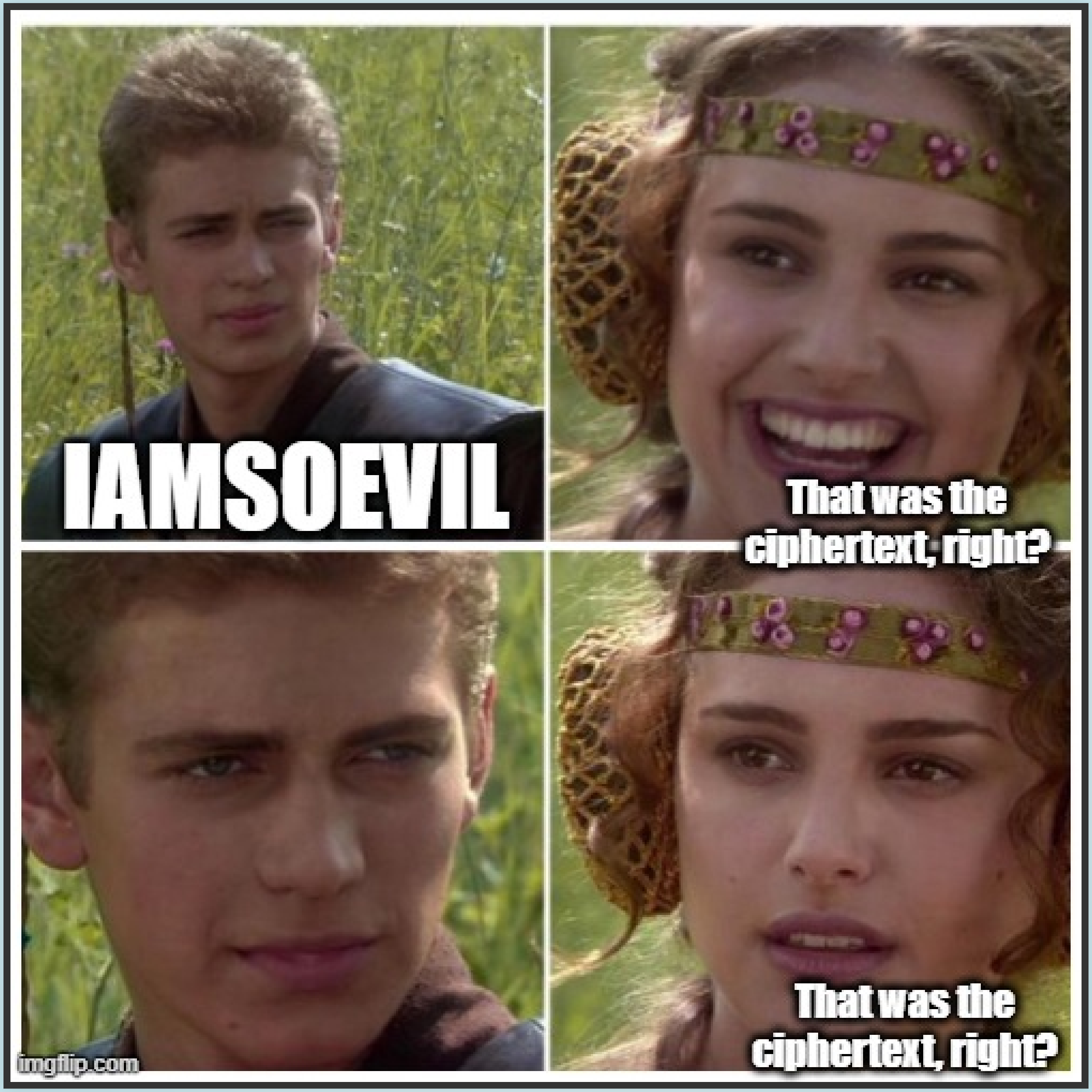
Summary

Today you learned to...

1. Define the components and requirements of a secure symmetric-key cryptosystem.
2. Define and implement the one-time pad symmetric-key cryptosystem.
3. Define and trace the Diffie-Hellman key exchange algorithm.
4. Define the terms perfect secrecy and discrete logarithm problem, and explain how they are related to the algorithms we study today.

Homework

- Readings from today: 8.1 (prep), 8.2, 8.3
- Readings for tomorrow: 7.5 (review), 8.4
- Work on Assignment 3
- Study for Term Test 2


IAMSOEVIL
That was the ciphertext, right?
imgflip.com
That was the ciphertext, right?