CSC110 Lecture 20: Cryptography Wrap-Up

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!)
  - (**new**) Review the posted cover page
- PythonTA survey 1 (**due Sunday October 30!**)

Today you'll learn to...

1. Implement the RSA cryptosystem in Python.
2. Define the TLS (Transport Layer Security) protocol and explain its connection to secure online communications.
3. Explain how symmetric-key and public-key cryptosystems are used as part of TLS.

Implementing RSA

Quick review of RSA

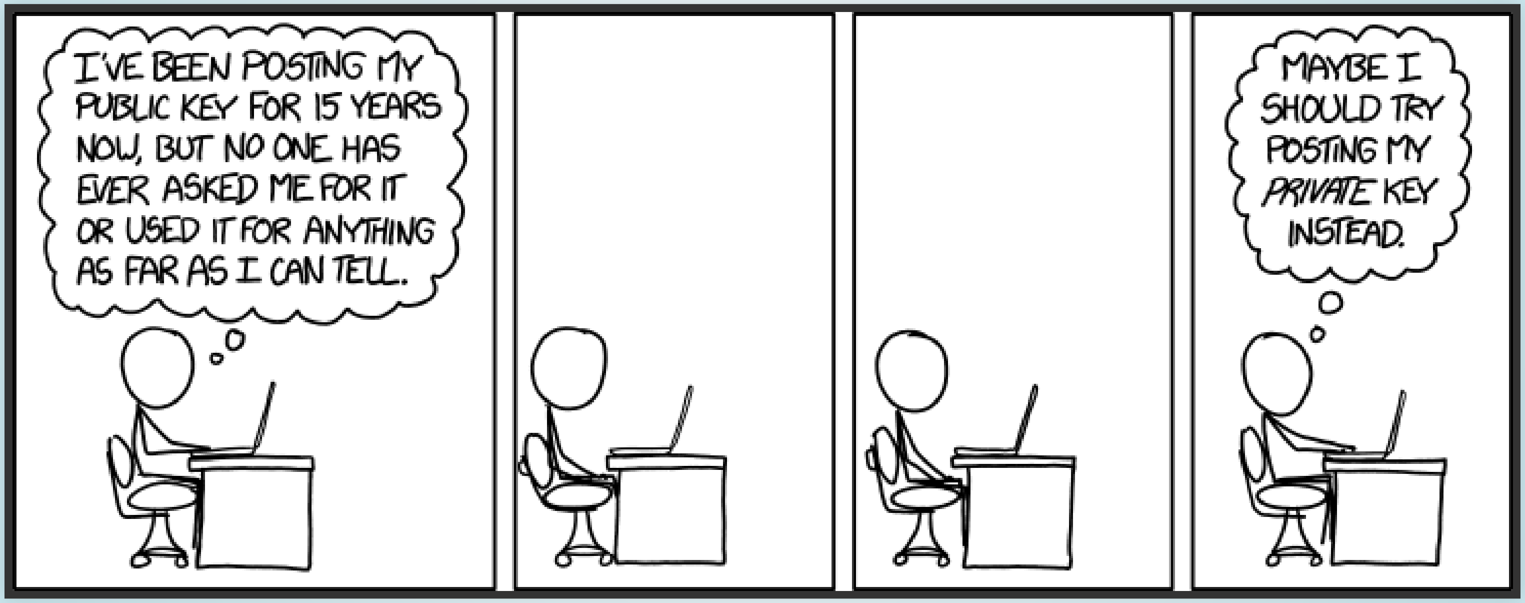
1. Pick two distinct primes p and q
2. Compute $n = pq$
3. Choose integer $e \in \{2, 3, \ldots, \varphi(n)\}$ such that $\gcd(e, \varphi(n)) = 1$
4. Compute $d \in \{2, 3, \ldots, \varphi(n)\}$ such that $ed \equiv 1 \pmod{\varphi(n)}$

Public key: (n, e). Private key: (p, q, d).

Encryption: $c = m^e \,\%\, n$

Decryption: $m' = c^d \,\%\, n$

Exercise 1: Implementing the RSA cryptosystem


I'VE BEEN POSTING MY PUBLIC KEY FOR 15 YEARS NOW, BUT NO ONE HAS EVER ASKED ME FOR IT OR USED IT FOR ANYTHING AS FAR AS I CAN TELL.
MAYBE I SHOULD TRY POSTING MY *PRIVATE* KEY INSTEAD.

From numbers to text

```python
def rsa_encrypt_text(public_key: tuple[int, int],
                     plaintext: str) -> str:
    """Encrypt the given plaintext using the recipient's public k
    """
```

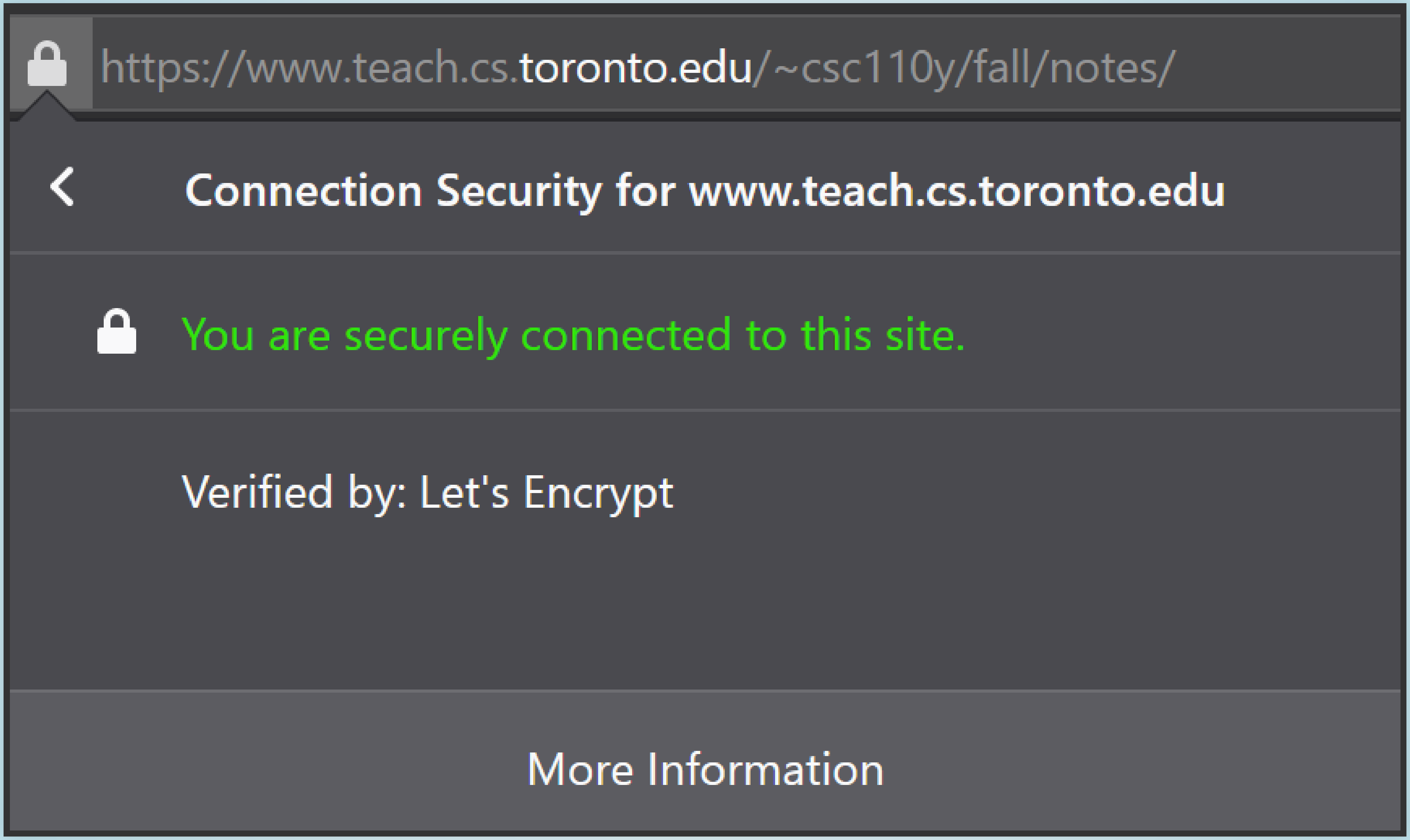
Idea: encrypt each letter separately (like Caesar cipher, one-time page).

RSA in practice

Problem: same as Caesar cipher! `'OLaTO+T^+NZZW'`

In practice, plaintext messages are divided into blocks, and each block is encrypted separately.

Securing online communications



https://www.teach.cs.toronto.edu/~csc110y/fall/notes/

Connection Security for www.teach.cs.toronto.edu

You are securely connected to this site.

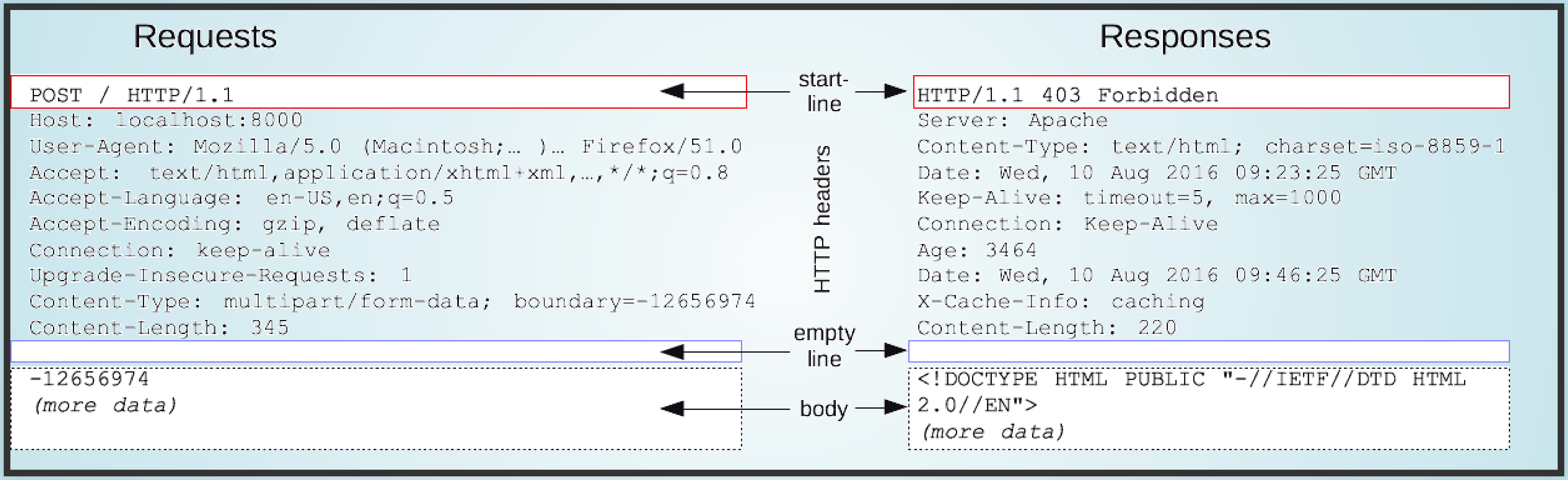
Verified by: Let's Encrypt

HTTPS

HTTPS is a communication protocol divided into two parts:

- **HTTP** (Hypertext Transfer Protocol): how data is formatted
- **TLS** (Transfer Layer Security): how formatted data is encrypted

Note: HTTPS stands for “Hypertext Fransfer Protocol Secure”.



```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

```
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
(more data)
```

HTTP diagram. Source: MDN Web Docs.

TLS Protocol

Context: a client (e.g., computer) makes a request to a server (e.g., Google).

1. When the client initiates the request, the server sends a “proof of identity” that the client has connected with the intended server, which the client verifies.

   This communication is not encrypted.

2. The client and server perform the Diffie-Hellman key exchange algorithm to establish a shared secret key.

   This communication is not encrypted either.

3. All remaining communication is encrypted using an agreed-upon symmetric-key cryptosystem, like a stream cipher.

Verifying identities

First two steps of TLS are unencrypted:

1. When the client initiates the request, the server sends a “proof of identity” that the client has connected with the intended server, which the client verifies.
2. The client and server perform the Diffie-Hellman key exchange algorithm to establish a shared secret key.

How does the client know they’re communicating with the correct server?!

Public-key cryptography: digital signatures

For encryption/decryption: you encrypt with David’s public key, David decrypts with his private key.

For signing/verification: David signs with his private key, you verify with his public key.

In general, an entity can claim the authenticity of a digital message by attaching a digital signature to the message using its private key.

Digital signature example

Example

David publishes his public key$(n, e) = (50381, 11)$

David posts:

```
Message: 'David thinks you are cool'
```

David *also* posts a signature of the message:

```
Signature: '搡쉥櫜褸ű靨ʃƬ褸롹□藿靨粤ɀ长靨쉥轚댆靨邱ɀɀ淬'
```

This signature was generated by encrypting the message with David’s private key (which only David knows)!

TLS Phase 1: “proof of identity”

1. When the client initiates the request, the server sends a “proof of identity” that the client has connected with the intended server, which the client verifies.

What actually happens:

- The server sends a **digital certificate** containing identifying information and its public key.
- The server’s digital certificate is signed by a trusted certificate authority.

Certificate authorities and web browsers

A **certificate authority** is an entity that creates digital certificates.

But anyone can sign a piece of data using their private key. How do we know who to trust?

Web browsers and other networking software come with a set of **pre-installed trusted certificates**.

TLS Phase 2: Diffie-Hellman, with signatures

2. The client and server perform the Diffie-Hellman key exchange algorithm to establish a shared secret key.

What actually happens:

- The server signs each message it sends using its private key
- The client verifies each message using the server’s public key (from the digital certificate)

TLS Phase 3: Why symmetric-key encryption?

(Why not use RSA, for example?)

Efficiency.

The public-key cryptosystems used in practice take longer to encrypt/decrypt data than the common symmetric-key cryptosystems.

TLS Phase 3: Demo

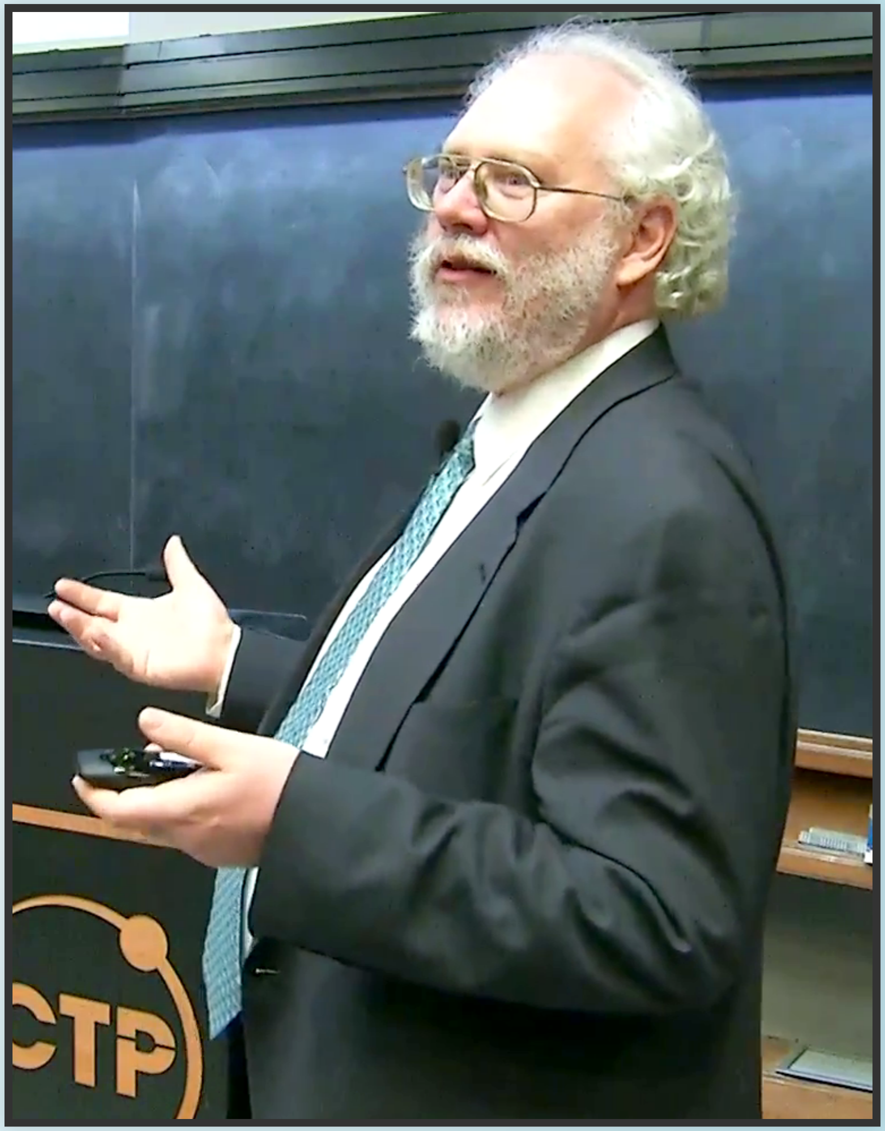
AES stands for **Advanced Encryption Standard**, and is a symmetric-key cryptosystem first published in 1998, and still widely used today.

Post-Quantum Cryptography

Diffie-Hellman’s security relies on the (assumed) hardness of the **Discrete Logarithm Problem**: given $g^a \equiv A \pmod{n}$, find a.

RSA’s security relies on the (assumed) hardness of the **Integer Factorization Problem**: given $n = pq$, find p and q.

In 1994, computer scientist Peter Shor developed a quantum computing algorithm for efficiently factoring large integers and solving the Discrete Logarithm Problem.



Post-Quantum Cryptography is an active area of research into cryptographic techniques and algorithms that remain secure even when allowing for quantum computing algorithms!

Summary

Today you learned to…

1. Implement the RSA cryptosystem.
2. Define the TLS (Transport Layer Security) protocol and explain its connection to secure online communications.
3. Explain how symmetric-key and public-key cryptography are used as part of TLS.

Homework

- Readings:
  - Today: 8.5, 8.6
  - Prep: 9.1, 9.2
  - Next week: 9.1, 9.2, 9.3, 9.5
- Work on Assignment 3
- Study for Term Test 2
- Complete PythonTA Survey 1 (due **Sunday**)
- Complete Prep 8 (due **Tuesday 9am**)

Good luck on Monday!