

CSC110 Lecture 20:

Cryptography Wrap-Up

David Liu and Tom Fairgrieve, 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 this coming Monday (Halloween)! 🎃 👻
 - 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

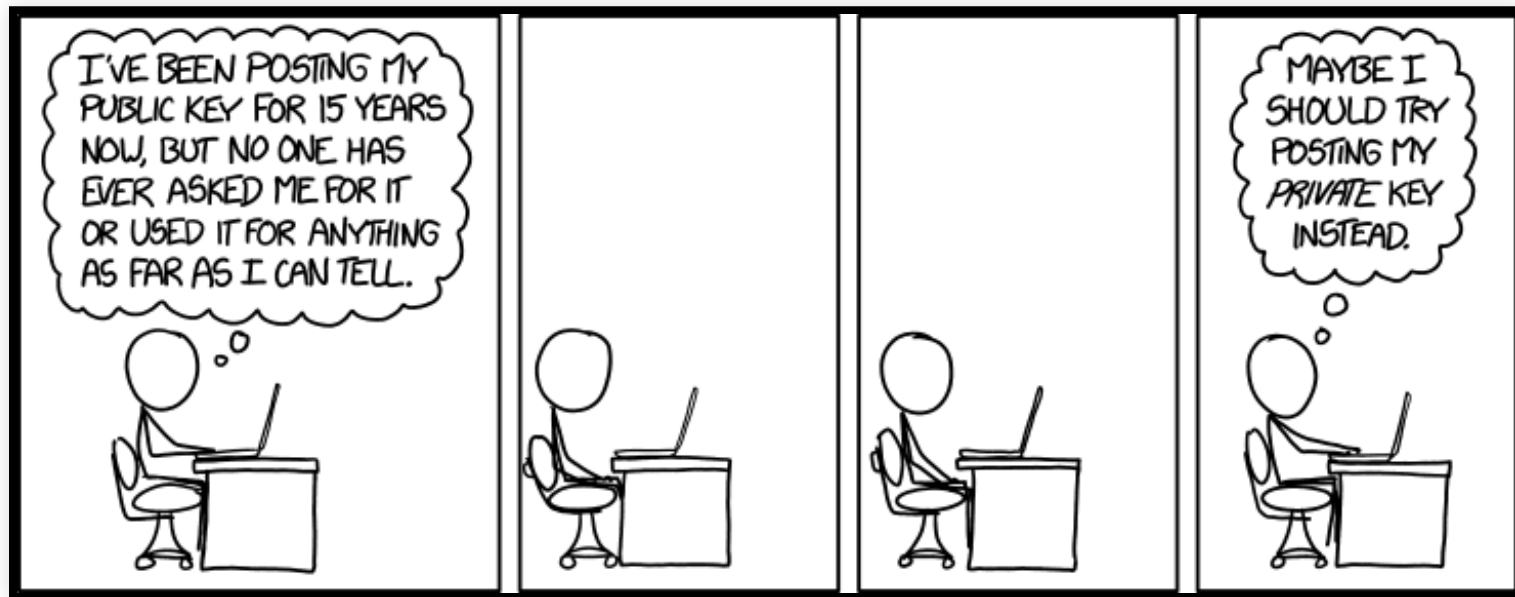
1. Pick two distinct primes p and q
2. Compute $n = pq$
3. Choose integer $e \in \{2, 3, \dots, \varphi(n)\}$ such that $\gcd(e, \varphi(n)) = 1$
4. Compute $d \in \{2, 3, \dots, \varphi(n)\}$ such that $ed \equiv 1 \pmod{\varphi(n)}$

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



From numbers to text

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

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

RSA in practice

Problem: same issue as with Caesar cipher!

Recall: 'DAVID IS COOL'

Became: 'OLaTO+T^+NZZW'

We could use letter frequency expectations to extract some information.

In practice, plaintext messages are divided into **blocks** of characters, 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

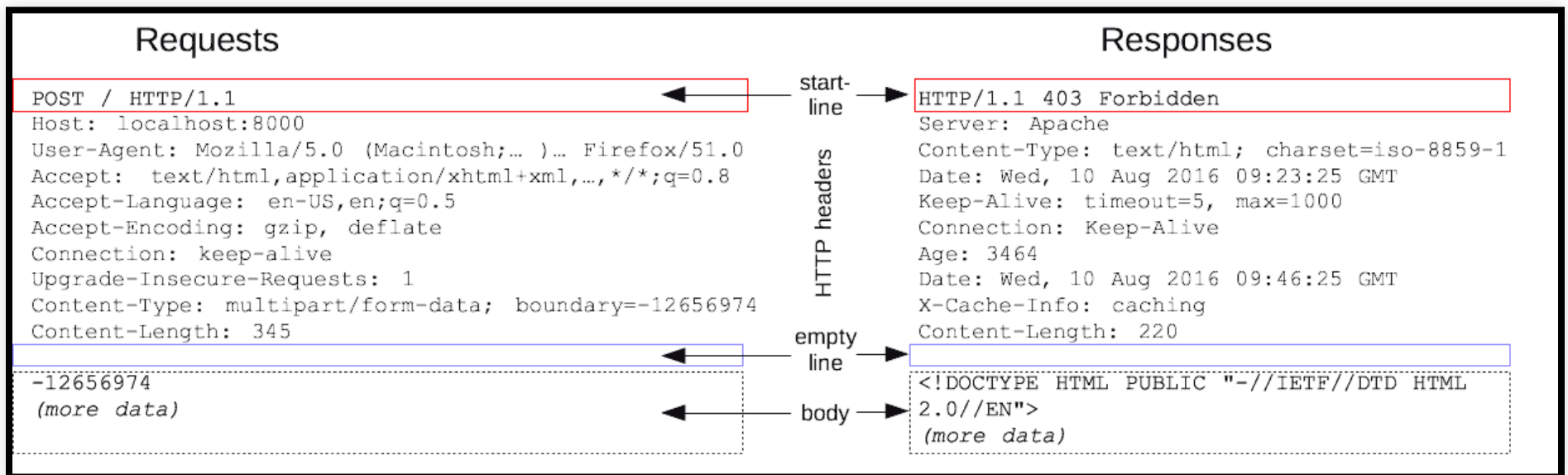
More Information

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 Transfer Protocol **Secure**”.



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** with Alice: you encrypt with Alice's public key, Alice decrypts with her private key.

For **signing/verification** of Alice: Alice signs with her private key, you verify with her 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

Alice publishes her public key $(n, e) = (50381, 11)$

Alice posts the unencrypted text:

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

Alice **also** posts a signature of the message using her private key:

```
Signature: '蹂셀橐樓ǔ壓♯7樓략ω藿壓粤꺇长壓셀깅뎡壓邱꺇꺇淬'
```

This signature was generated by encrypting the message with Alice's private key (that only Alice knows)! You can decrypt with Alice's public key and can compare the decrypted message with the first post.

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

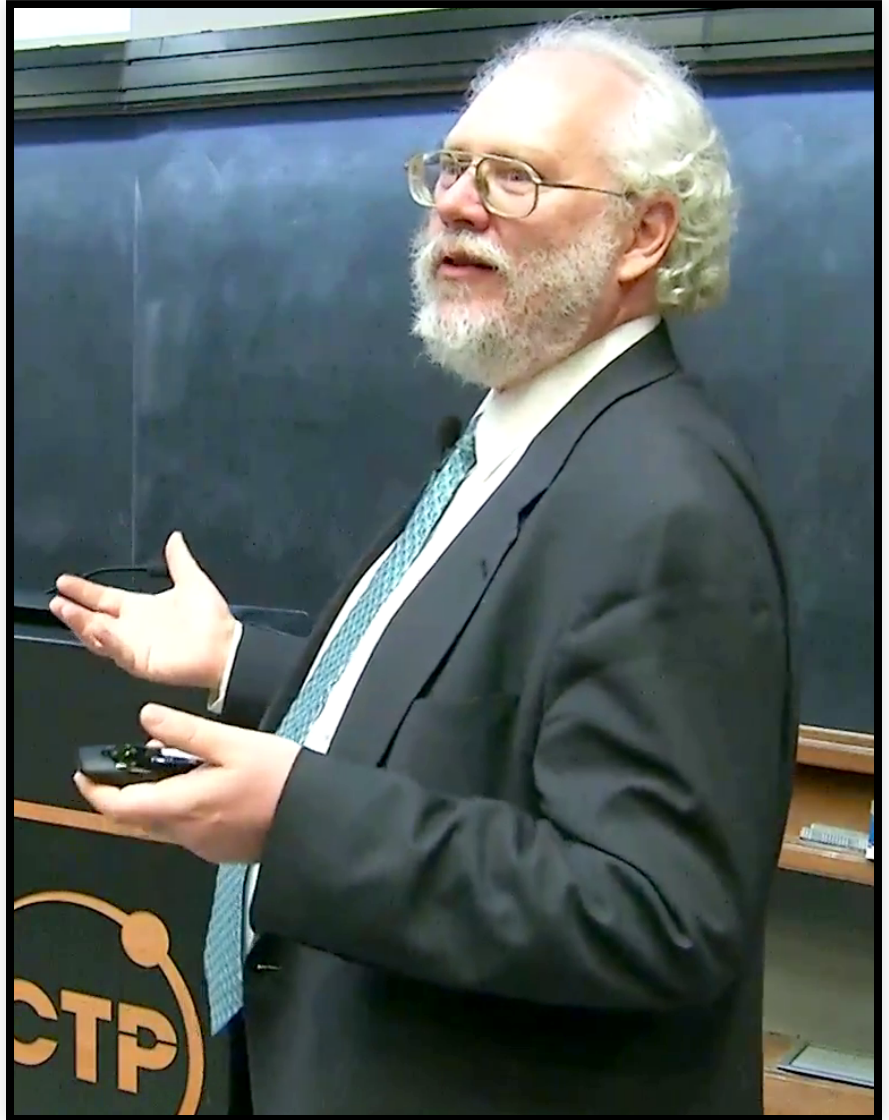
AES stands for **Advanced Encryption Standard**, and is a symmetric-key cryptosystem first published in 1998, and still widely used today.

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!

