CSC110 Lecture 19: Public-Key Cryptography and the RSA Cryptosystem

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
- **No tutorial this Friday!**

Today you'll learn to...

1. Define the components and requirements of a secure public-key cryptosystem.
2. Explain how the RSA cryptosystem works.
3. Prove the correctness of the RSA cryptosystem.
4. Explain why the RSA cryptosystem considered computationally secure by referring to the integer factorization problem.

Public-key cryptography

Story so far

Last class, we studied **symmetric-key cryptosystems**, which required that two communicating parties establish a shared secret key before communicating.

We also learned about the **Diffie-Hellman key exchange algorithm** for establishing shared secret keys with only public communication.

Is there a way to send encrypted messages to someone without sharing a secret key with them first?

Public-key cryptosystem, informal

David has two keys: a public key and a private key.

Everyone knows his public key, and can encrypt plaintext messages for David using the public key.

Only David knows his private key, and can use the private key to decrypt the encrypted ciphertexts.

Public-key cryptosystem, formal

A public-key cryptosystem consists of:

- $\mathcal{P}$ (set of plaintext messages), $\mathcal{C}$ (set of ciphertext messages)
- Sets $\mathcal{K}_1$ (of public keys) and $\mathcal{K}_2$ (of private keys)
- A subset $\mathcal{K} \subseteq \mathcal{K}_1 \times \mathcal{K}_2$ (of valid public-private key pairs)
- $Encrypt : \mathcal{K}_1 \times \mathcal{P} \to \mathcal{C}$
- $Decrypt : \mathcal{K}_2 \times \mathcal{C} \to \mathcal{P}$

Two properties

Correctness

For all $(k_1, k_2) \in \mathcal{K}$ and $m \in \mathcal{P}$, $Decrypt(k_2, Encrypt(k_1, m)) = m$

Security

For all $(k_1, k_2) \in \mathcal{K}$ and $m \in \mathcal{P}$, if an eavesdropper only knows the values of the public key k_1 and the ciphertext $c = Encrypt(k_1, m)$ but does not know k_2, it is computationally infeasible to find the plaintext message m.

Review: modular exponentiation

Cycles

2^a	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8
$2^a \% 7$	1	2	4	1	2	4	1	2	4

The powers of 2 modulo 7 enter a cycle of length 3: 1, 2, 4, 1, 2, 4, ...

(**Fermat's Little Theorem**) Let $p, a \in \mathbb{Z}$ and assume p is prime and that $p \nmid a$. Then $a^{p-1} \equiv 1 \pmod{p}$.

Examples:

- $2^6 \equiv 1 \pmod{7}$
- $5^{16} \equiv 1 \pmod{17}$

How can we extend this to non-prime numbers?

The **Euler totient function** (or **Euler phi function**) is defined as:

$\varphi : \mathbb{Z}^+ \to \mathbb{N}$

$\varphi(n) = \text{len}\Big(\big\{a \mid a \in \{1, \ldots, n-1\} \text{ and } \gcd(a, n) = 1\big\}\Big)$

Examples:

- $\varphi(5) = 4 \quad (\{1, 2, 3, 4\})$
- $\varphi(17) = 16 \quad (\{1, 2, \ldots, 16\})$
- For all primes p, $\varphi(p) = p - 1$

- $\varphi(6) = 2 \quad (\{1, 5\})$
- $\varphi(15) = 8 \quad (\{1, 2, 4, 7, 8, 11, 13, 14\})$
- For all distinct primes p and q, $\varphi(pq) = (p - 1)(q - 1)$.

Euler's Theorem

For all $a \in \mathbb{Z}$ and $n \in \mathbb{Z}^+$, if $\gcd(a, n) = 1$ then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

(special case) For all $a \in \mathbb{Z}$ and distinct primes $p, q \in \mathbb{Z}$, if $\gcd(a, pq) = 1$ then $a^{(p-1)(q-1)} \equiv 1 \pmod{n}$.

Exercise 1: Reviewing modular exponentiation

The Rivest-Shamir-Adleman (RSA) cryptosystem

Overview

The Rivest-Shamir-Adleman (RSA) cryptosystem is a public-key cryptosystem that relies on modular arithmetic.

In the (basic) RSA cryptosystem, plaintexts and ciphertexts are positive integers.

Demo

We'll illustrate RSA using an interactive demo between David and you (yes, you!).

Three phases:

1. Key generation: David will create a public-private key pair
2. Encryption: you will send David an encrypted message using his public key
3. Decryption: David will decrypt your message using his private key

RSA Phase 1: Key generation

1. Pick two distinct primes p and q.
   - e.g., $p = 5$, $q = 13$
2. Compute $n = pq$.
   - e.g., $n = 65$
3. Choose integer $e \in \{2, 3, \ldots, \varphi(n)\}$ such that $\gcd(e, \varphi(n)) = 1$. (Note: $\varphi(n) = (p-1)(q-1)$.)
   - e.g., $\varphi(n) = 4 \cdot 12 = 48$; pick $e = 5$
4. Compute $d \in \{2, 3, \ldots, \varphi(n)\}$ such that $ed \equiv 1 \pmod{\varphi(n)}$ (i.e., d is an inverse of e modulo $\varphi(n)$).
   - e.g., $d = 29$ (since $ed = 145 \equiv 1 \pmod{48}$)

RSA Phase 1: Key generation

- **Public key**: (n, e)
  - e.g., $(65, 5)$
- **Private key**: (p, q, d)
  - e.g., $(5, 13, 29)$

Everyone will know the public key. Only the person who generates the key pair knows the private key.

RSA Phase 1: Key generation

New example: David's public key is $(50381, 11)$.

So $n = 50381$ and $e = 11$.

RSA Phase 2: Encryption

Plaintext: $m \in \{1, 2, \ldots, n-1\}$

You take your plaintext m and compute $c = m^e \,\%\, n$, and send c to David.

Pick a message m, encrypt it using $(n, e) = (50381, 11)$, and type in your c (but not m!) into Campuswire.

RSA Phase 3: Decryption

David takes the ciphertext c, and computes $m' = c^d \% n$.

Then $m' = m$, the original plaintext message.

Moment of truth!

RSA Phase Summary

Key generation

- Choose primes p, q. Let $n = pq$.
- Choose numbers e, d such that $ed \equiv 1 \pmod{\varphi(n)}$.
- Public key: (n, e). Private key: (p, q, d).

Encryption

- $c = m^e \% n$

Decryption

- $m' = c^d \% n$

RSA is asymmetric

If David wants to send you an encrypted message, he can't use his public-private key pair.

He needs to use your public key!

Exercise 2: Reviewing the RSA Cryptosystem

RSA: Correctness and Security

RSA Correctness theorem

Let $(p, q, d) \in \mathbb{Z}^+ \times \mathbb{Z}^+ \times \mathbb{Z}^+$ be a private key and $(n, e) \in \mathbb{Z}^+ \times \mathbb{Z}^+$ its corresponding public key. Let $m, c, m' \in \{1, \ldots, n-1\}$ be the original plaintext message, ciphertext, and decrypted message, respectively.

Then $m' = m$ (i.e., the decrypted message is the same as the original message).

We'll prove this theorem in the **special case** when $\gcd(m, n) = 1$.

RSA Correctness theorem (proof)

Proof assuming $\gcd(m, n) = 1$.

From the encryption step, we know

$$c \equiv m^e \pmod{n}$$

From the decryption step, we know

$$m' \equiv c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}$$

We will show that

$$m^{ed} \equiv m \pmod{n}$$

RSA Correctness theorem (proof)

From the key generation step, we know $ed \equiv 1 \pmod{\varphi(n)}$.

So (by the definition of modular equivalence and divisibility), there exists $k \in \mathbb{Z}$ such that $ed - 1 = k \times \varphi(n)$.

Substituting this into the power m^{ed}, we have:

$$\begin{aligned} m^{ed} &= m^{k \times \varphi(n) + 1} \\ &= (m^{\varphi(n)})^k \times m \end{aligned}$$

Key idea: by Euler's theorem, we know that $m^{\varphi(n)} \equiv 1 \pmod{n}$!!!

RSA Correctness theorem (proof)

Putting this all together, we have:

$$\begin{aligned} m' &\equiv m^{ed} && \pmod{n} \\ &\equiv (m^{\varphi(n)})^k \times m && \pmod{n} \\ &\equiv 1^k \times m && \pmod{n} \\ &\equiv m && \pmod{n} \end{aligned}$$

So $m' \equiv m \pmod{n}$.

Then since $1 \leq m, m' \leq n-1$, we can conclude $m' = m$.

Security of RSA (1)

Eavesdropper knows: public key (n, e), ciphertext $c = m^e \,\%\, n$

1. Given n, e, and c, can we efficiently compute m directly?

   No! We don't have an efficient way of computing "e-th roots" in modular arithmetic.

Security of RSA (2)

Eavesdropper knows: public key (n, e), ciphertext $c = m^e \,\%\, n$

2. Given n and e, can we efficiently compute the private number d? ($m' = c^d \,\%\, n$)

   d is the modular inverse of e modulo $\varphi(n)$, and we can calculate modular inverses...

   Not yet! We know n, not $\varphi(n)$.

Security of RSA (3)

Eavesdropper knows: public key (n, e), ciphertext $c = m^e \,\%\, n$

3. How can we calculate $\varphi(n)$ efficiently?

```
varphi_n = len({d for d in range(1, n) if math.gcd(d, n)
```

Not efficient (for large n)!

We could calculate $\varphi(n)$ efficiently by first finding p and q, since $\varphi(n) = (p-1)(q-1)$...

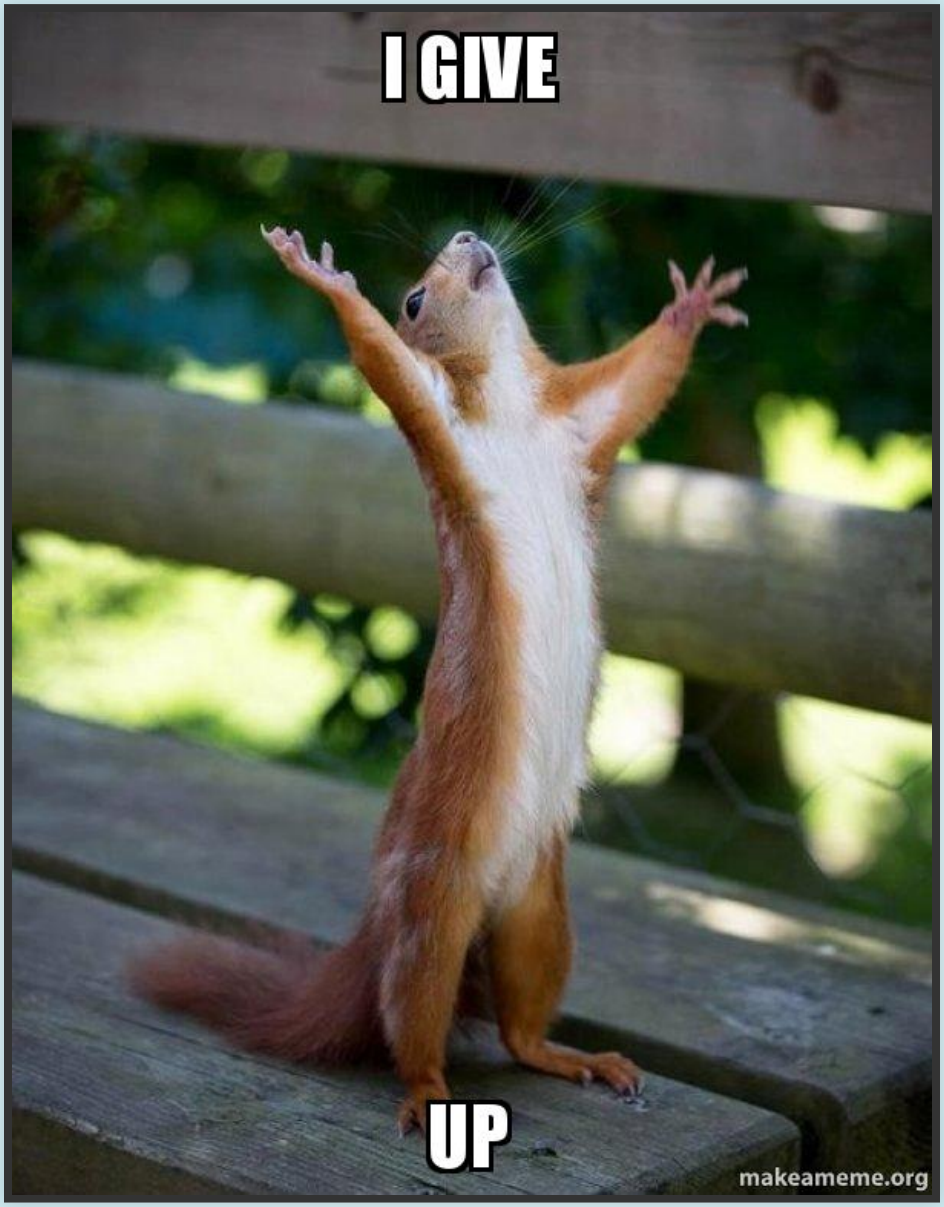
Security of RSA (4)

Eavesdropper knows: public key (n, e), ciphertext $c = m^e \,\%\, n$

4. Given n, can we find effciently p and q?

   No!

   There is no known efficient algorithm for factoring large numbers! This is the **integer factorization problem**.


I GIVE
UP
makeameme.org

“Quantum computers break Internet security”

There exist efficient algorithms for integer factorization that can run on quantum computers!

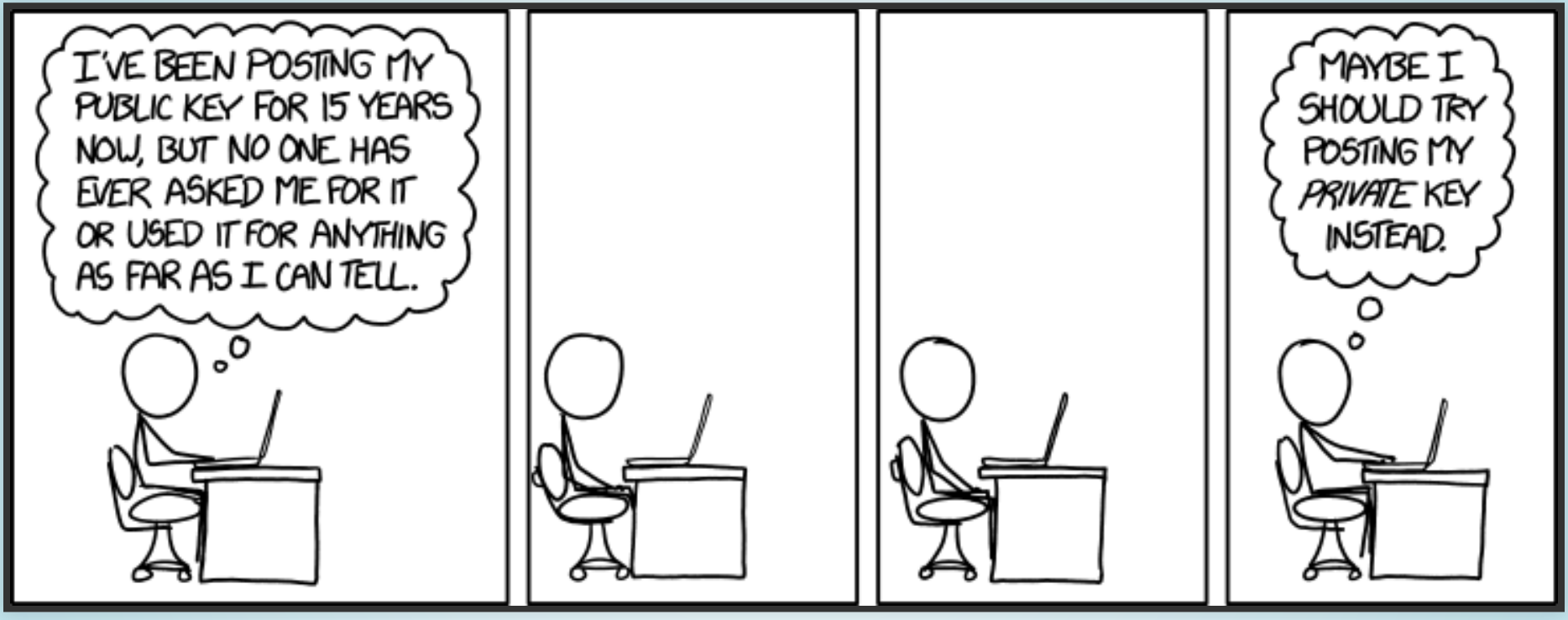
Summary

Today you learned to...

1. Define the components and requirements of a secure public-key cryptosystem.
2. Explain how the RSA cryptosystem works.
3. Prove the correctness of the RSA cryptosystem.
4. Explain why the RSA cryptosystem considered computationally secure by referring to the integer factorization problem.

Homework

- Readings from today: 8.4
- Readings for tomorrow: 8.5, 8.6
- Work on Assignment 3
- Study for Term Test 2


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.