A Mathematical Introduction to RSA Encryption

CS SAIL 2018

Patrick Feltes & Ben Pankow

Agenda

- Brief introduction
- What is encryption, and how does it work?
- Proving RSA encryption
- Implementing RSA

Introduction

Who are these people and why should I trust them?

- Patrick Feltes
  - pfeltes2@illinois.edu
- Ben Pankow
  - bpankow2@illinois.edu

- Course Information
  - https://github.com/sail-rsa/sail-rsa

Feel free to ask us questions during the presentation, talk to us afterwards, or email us with any follow-up questions

Encryption

What is encryption?

- Used to secure information
  - 'Scrambles' data so that only intended recipients can read it
- Encryption for security
  - Secret messaging (Wickr, Telegram, iMessage)
  - Secure data transfer (HTTPS, TLS/SSL)
  - Keeping files safe (Encrypt data on phone/computer/server)
- Encryption for digital signatures
  - Validating message sender
  - Ensuring accountability (contracts)
  - Blockchain (cryptocurrencies)

Encryption

How does it work?

- Algorithm is secret
  - Restricted ciphers ("security through obscurity")
- Algorithm is public w/ secret component
  - Key-based encryption

- Symmetric encryption
  - AES, DES
- Public-key encryption
  - Diffie-Hellman exchange, RSA

RSA Encryption

A brief background

- Public-key (asymmetric) cryptosystem
  - Each user has a "public key" and a "private key"
- Developed by Ron Rivest, Adi Shamir, Leonard Adleman in 1978
- Security based on difficulty of factoring large numbers
- Patent held by MIT until 2000 - now public domain
- Widely used (PGP, some TLS protocols)

Modular Arithmetic

Definition

$x \pmod{m} = a$, where a is the remainder when dividing x by m

Examples

$16 \pmod{7} = 2$

$15 \pmod{7} = 1$

$14 \pmod{7} = 0$

$13 \pmod{7} = 6$

Notice that the remainder is always < 7

Encrypting and Decrypting with RSA

Definition

$E(x) = x^e \pmod{n}$

$D(y) = y^d \pmod{n}$

In theory, $D\left(E(x)\right) = x$

x is cleartext (message), y is ciphertext (encrypted message)

p and q are large primes we pick and later throw out

$n = pq$

We choose e, d so that $ed \pmod{(p-1)(q-1)} = 1$

(e, n) is the public (encryption) key

(d, n) is the private (decryption) key

Proving RSA

Theorem

$D(E(x)) = (x^e)^d \pmod{n} = x^{ed} \pmod{n} = x$ *for any* $x < n$

To Prove Later

$a^{N(p-1)+1} \pmod{p} = a \pmod{p}$ for any integers a, N and prime p

Proof

$ed \pmod{(p-1)(q-1)} = 1$, so

$ed = L(p-1)(q-1) + 1$ for some integer L

$x^{ed} = x^{L(p-1)(q-1)+1}$

Pick $N = L(q-1)$

$x^{ed} = x^{N(p-1)+1} \pmod{p} = x \pmod{p}$ $\square$

Proving RSA (cotd.)

Theorem

$D(E(x)) = (x^e)^d \pmod{n} = x^{ed} \pmod{n} = x$ *for any* $x < n$

Proof (cotd.)

$x^{ed} = x^{L(p-1)(q-1)+1}$

Now pick $N = L(p-1)$

$x^{ed} = x^{N(q-1)+1} \pmod{q} = x \pmod{q}$

$x^{ed} - x$ is a multiple of p and q

$x^{ed} - x$ is then a multiple of $n = pq$

$x^{ed} \pmod{n} = x \pmod{n}$

$x < n$, so

$x^{ed} \pmod{n} = x$ $\square$

Fermat's Little Theorem

Theorem

$a^p \pmod{p} = a \pmod{p}$ *for any integer a and prime p*

Proof

Proof omitted, see additional material if interested $\square$

Example

$5^3 = 125$, $125 \pmod{3} = 2 = 5 \pmod{3}$

Extending Fermat's Little Theorem

Theorem

$a^{N(p-1)+1} \pmod p = a \pmod p$ *for any integers a, N and prime p*

Proof

$a^p \pmod p = a \pmod p$

$a^{p-1} \cdot a^p \pmod p$
$= a^{p-1} \cdot a \pmod p$
$= a^p \pmod p = a \pmod p$

$a^{k(p-1)} \cdot a^p \pmod p = a \pmod p$

$\square$

Extending Fermat's Little Theorem (cotd.)

Theorem

$a^{N(p-1)+1} \pmod p = a \pmod p$ *for any integers a, N and prime p*

Proof(cotd.)

$a^{k(p-1)} \cdot a^p \pmod p = a \pmod p$

$$
\begin{aligned}
&a^{k(p-1)} \cdot a^p \pmod p \\
&= a^{k(p-1)} \cdot a^{p-1} \cdot a \pmod p \\
&= a^{k(p-1)+(p-1)+1} \pmod p \\
&= a^{(k+1)(p-1)+1} \pmod p
\end{aligned}
$$

Define $N = k + 1$

$a^{N(p-1)+1} \pmod p = a \pmod p$ $\square$

Drawbacks

- Message size (x) limited by size of primes (n)
- Slower with larger primes
  - Solution: To send large messages, use RSA to distribute symmetric key. Then use symmetric encryption (AES, DSA) to encrypt messages
- Deterministic results - same message = same output
  - Solution: Random padding

Components of RSA

- Key generation
  - Given primes p, q, produce private key (d, n) and public key (e, n)
- Encryption
  - Given a message x and a public key (e, n), produce the ciphertext y
- Decryption
  - Given ciphertext y and a private key (d, n), produce the original message x