ITCS 531: Number Theory 4 - RSA encryption

Rob Egrot

Why encryption?

➤ Sometimes you want to send a message that you only want the intended receiver to be able to read.

Since we cannot usually make sure that nobody intercepts the message, we must use a code.

The goal is that the coded message will be easy to understand for the intended recipient (and us), and very hard to understand for everyone else.

Encryption functions

▶ We assume our 'messages' are numbers - it's easy to code English sentences as big numbers.

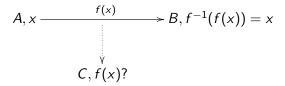
► An **encryption function** is a bijection between two subsets of N.

We use the encryption function to change the numbers that code our messages.

Only people who know the encryption function should easily be able to work out the original message.

Sending messages

A sends B a message x encrypted using function f. B can use f^{-1} to recover x from f(x). C intercepts f(x) but cannot recover x.



Problem

► This system (**private key encryption**) can work well when there are only a small number of people.

But the encryption function must be agreed in advance and kept secret.

Not dynamic - someone who knows function can decrypt all messages using it.

► The more people who know the function the more likely that it will become known by others - harder to keep the secret.

Asymmetrical information

Private key encryption is symmetrical - all parties must know encryption function and its inverse.

▶ Is there a way to exchange messages where each party only knows how to decrypt the messages intended for them?

Yes - Public key encryption solves this problem.

Public key encryption

- A wants to send B a message x.
- B tells A to encrypt using f.
- ▶ A sends f(x), B decrypts using g to get g(f(x)) = x.
- C also wants to send B a message y.
- ▶ *B* tells *C* to encrypt using *f*.
- ightharpoonup C send f(y) to B B decrypts using g.
- ▶ But, A and C don't know g, so they can't read each other's messages if they intercept them.

$$A \xrightarrow{f(x)} B \xleftarrow{f(y)} C$$

RSA encryption

▶ B can broadcast f so anyone can know it - only g is a secret, and only B has to know it.

► This is fine in theory, but do functions that are easy to calculate but hard to invert exist?

Yes - we can find example with basic number theory.

This gives us RSA encryption.

Choosing the public key

▶ B needs to choose a **public key** that he can broadcast to anyone who wants to send him a message - this public key defines B's encryption function.

ightharpoonup B chooses two large primes p and q, and defines N = pq.

▶ B chooses a number e < (p-1)(q-1) that is coprime with (p-1)(q-1) - an easy way to do this is to make e prime.

 \triangleright (N, e) is B's public key.

Encrypting with the public key

▶ A wants to send B the message x - assume x < N (B chose large primes p, q).

ightharpoonup A calculates $x^e \mod N$.

ightharpoonup She sends $x^e \mod N$ to B.

$$A \xrightarrow{x^e \mod N} B$$

Decrypting

- \triangleright B receives $x^e \mod N$ how can he recover x?
- ▶ There is a number d such that $(x^e)^d \equiv_N x$.
- ▶ *d* is *B*'s **private key**. He must keep this secret from everyone.
- ► What is *d*?
- ▶ d is the inverse of $e \mod (p-1)(q-1)$.
 - This exists because e and (p-1)(q-1) are coprime, and B can easily calculate it using the extended Euclidean algorithm (as in the proof of Bézout's identity).

Why does *d* work?

▶ Why is it true that $(x^e)^d \equiv_N x$?

▶ We will need to use some number theory to prove this.

▶ To prove this we will need a lemma.

A lemma

Lemma 1

Let p be prime, and let $a, m \in \mathbb{N}$. Then $a \equiv_{p-1} 1 \implies m^a \equiv_p m$.

Proof.

- ▶ Obviously true if p, m are not coprime as then $m \equiv_p 0$.
- ▶ If m, p are coprime then $m^{p-1} \equiv_p 1$ by Fermat's little theorem.
- ▶ If $a \equiv_{p-1} 1$ then a-1=(p-1)k for some k.
- $m^a m = m(m^{a-1} 1) = m(m^{(p-1)k} 1) \equiv_p m(1^k 1) = 0.$
- ▶ I.e. $m^a \equiv_p m$.

The main result

Lemma 2

If d is the inverse of e modulo (p-1)(q-1) then $x^{ed} \equiv_N x$ for all $x \in \{0,1,\ldots,N-1\}$.

Proof.

- ▶ As $ed \equiv_{(p-1)(q-1)} 1$ we have ed 1 = k(p-1)(q-1).
- ▶ So $ed \equiv_{p-1} 1$ and $ed \equiv_{q-1} 1$.
- ▶ By lemma 1 we get $x^{ed} \equiv_p x$ and $x^{ed} \equiv_q x$.
- ▶ I.e. $p \mid (x^{ed} x)$ and $q \mid (x^{ed} x)$.
- ▶ So $N | (x^{ed} x)$.
- ▶ I.e. $x^{ed} \equiv_N x$.

Cracking the code?

- ▶ Suppose C intercepts $x^e \mod N$. How can C recover x?
- ► C can calculate y^e for all y < N to find y such that $y^e \equiv_N x^e$.
 - In the worst case this involves checking $2^{L(N)}$ values, where L(N) the length of N when written in binary.
 - ▶ In computing terms this is a very slow process.
- ightharpoonup C can factor N into p and q, then work out d just like B did.
 - No efficient algorithm for finding prime factorizations exists. Or if it does it's a secret!
 - If you could find such an algorithm you would become famous
 or possibly the CIA would assassinate you first.

In practice

- Usually RSA is used to transmit private keys.
- ▶ What I have described here is *textbook RSA*.
- It has some vulnerabilities.
- ▶ E.g. if $x < N^{\frac{1}{e}}$ then x can be recovered from x^e just by finding $(x^e)^{\frac{1}{e}}$ in ordinary arithmetic.
- ▶ If the same message x is sent to several people using the same value e then the Chinese Remainder Theorem can be used to recover x.
- ➤ To avoid these and other attacks, messages are usually padded with additional random elements to distort the exploitable rigid structure of textbook RSA.