Message Passing, RSA Encryption

Robert Y. Lewis

CS 0220 2023

March 13, 2023

Overview

What have we done so far?

We started off talking about propositional (boolean) logic.

- Connected to circuits, foundations of computing
- Conditional statements: *everywhere* in CS
- Boolean satisfiability: a “perfect” generic problem

Then we talked about sets, relations, and proof techniques.

- A language to describe groups of objects and ways to manipulate them
- A sandbox for writing rigorous logical arguments
- Functions and relations, related terminology: essential for reasoning about program design

Why number theory?

Why number theory?

Very relevant for computer science!

- A more interesting sandbox to talk about proofs, without losing ourselves in details
- On one level of abstraction, computers are number-crunching machines (functions)
- Computers like finite things; mathematicians like infinite things; integers are somewhere in between

Modular arithmetic, computing inverses in particular: canonical “hard and important” computations.

Building up to encryption algorithms!

Sending messages

Motivations:

- Why do we send each other messages? Communication is a pretty human activity. Coordination is a practical application.
- Why would we want them to be secret? Competition. Gossiping. Surprise. Private information.
- Why might the message need to be encrypted? Message can be intercepted, stolen/broadcast by a third party, accidentally revealed like by being left on screen connected to projector. Communication channel is open.

Encoding messages

We'll assume all messages are fixed-length bit strings.

Is that sufficiently general? Can we encode arbitrary messages this way?

The Alphabet

00000	space	01000	H	10000	P	11000	X
00001	A	01001	I	10001	Q	11001	Y
00010	B	01010	J	10010	R	11010	Z
00011	C	01011	K	10011	S	11011	.
00100	D	01100	L	10100	T	11100	!
00101	E	01101	M	10101	U	11101	?
00110	F	01110	N	10110	V	11110	,
00111	G	01111	O	10111	W	11111	@

One-time pad: Encryption

Alice and Bob share 60 random bits (the “one-time pad”) in advance:

10110 00011 00000 00110 10010 00010
00000 11000 00110 01001 11111 11110

Alice: Wants to send a private message to Bob. She turns it into a sequence of 60 bits. She then computes the bitwise “xor” of her message and the one-time pad and transmits it:

10101 01100 00110 00000 10111 00111
00000 11000 00110 01001 11111 11110

One-time pad: Decryption

Bob: Wants to read Alice's message.

10101 01100 00110 00000 10111 00111
00000 11000 00110 01001 11111 11110

How can he recover it? Bitwise "xor" with the one-time pad will undo the encryption operation.

encrypted line 1:	10101 01100 00110 00000 10111 00111
pad line 1:	10110 00011 00000 00110 10010 00010
xor line 1:	00011 01111 00110 00110 00101 00101
text line 1:	C O F F E E
encrypted line 2:	00000 11000 00110 01001 11111 11110
pad line 2:	00000 11000 00110 01001 11111 11110
xor line 2:	00000 00000 00000 00000 00000 00000
text line 2:	_ _ _ _ _ _

One-time pad: Cracking

Eve: Sees the encrypted message and wants to understand it. She doesn't have the one-time pad.

The encrypted message gives *no information* about the unencrypted message. All possible messages are *equally likely*.

Although, if one-time pad is reused, information is leaked.

Public key cryptography

The one-time pad is essential to the one-time pad scheme. How can Alice and Bob agree on the one-time pad if Eve is listening? Could send it encrypted with a one-time pad, but they'd use up n bits of pad to transmit n bits of pad, so that doesn't help.

Can they put their key out in the open??

Bob wants to be able to receive secret messages. Bob creates a private key, which must remain secret. Bob also creates a public key, which is made public. Anyone, Alice say, who wants to send a secret message to Bob can encrypt it with Bob's public key. Only Bob can decrypt such messages (using his private key).

No other communication or agreements or secret emails are needed.

Beforehand

1. Bob generates two distinct (hundreds of digits long) primes, p and q and keeps them hidden.
2. Bob sets $n ::= pq$.
3. Bob selects an integer $e \in [1, n)$ such that $\gcd(e, (p-1)(q-1)) = 1$. The public key is the pair (e, n).
4. Compute $d \in [1, n)$ such that $de \equiv 1 \pmod{(p-1)(q-1)}$. The private key is the pair (d, n).

Time to send and receive

Encryption: Alice wants to send unencryted message m. She computes and then sends the encrypted message $m^* = \text{rem}(m^e, n)$. (Uses e and n, which are public.)

Decryption: Bob receives m^*. He decrypts by computing $m' = \text{rem}((m^*)^d, n)$. (Uses d and n, where d is private.)

Decrypting works

Claim: $m' = m$.
Partial Proof (for case when m is relatively prime to n):

$$
\begin{aligned}
m' &= \text{rem}((m^*)^d, n) && \text{defn } m' \\
&= \text{rem}((\text{rem}(m^e, n))^d, n) && \text{def of } m^* \\
&= \text{rem}((m^e)^d, n) && \text{“pre-mod” property} \\
&= \text{rem}(m^{ed}, n) && \text{exponentation} \\
&= \text{rem}(m^{1+k(p-1)(q-1)}, n) && \text{selection of } e \\
&= \text{rem}(m^{1+k\phi(n)}, n) && \text{property of } \phi(pq) \\
&= \text{rem}(m \times (m^{\phi(n)})^k, n) && \text{rearrange factors} \\
&= \text{rem}(m, n) && \text{Euler's thm: } m^{\phi(n)} \equiv 1 \pmod{n} \\
&= m && \text{assuming } m < n
\end{aligned}
$$

Can handle $\gcd(m, n) > 1$, too, but this argument doesn't show why.

How to compute?

1. Generate big primes p and q? Factoring is hard, but can do (randomized) primality test quickly—Miller-Rabin. For d-digit prime, need about d tries to find one.
2. Multiply big primes pq to get n? Standard multiplication algorithm scales well to big numbers.
3. Find number relatively prime to $(p-1)(q-1)$? Generate and test works again, but the chance of getting a hit at least as good and the test is much faster.
4. Compute $d = e^{-1} \bmod (p-1)(q-1)$? Pulvarize! (aka gcdcombo.)
5. Compute $m^* = m^e \bmod n$ or $m' = (m^*)^d \bmod n$? Exponentiation by repeated squaring.

All of the steps are fast enough.

Breaches

Private: p, q, d.

Public: n, e.

- If p revealed? Can get $q = n/p$. Can get d as $e^{-1} \bmod (p-1)(q-1)$.
- If q is revealed? Same as p.
- If d is revealed? Can get us p and q, though less clear how. But, can decrypt messages, so that's pretty bad.

We're pretty sure it's hard to get p and q from n. Factoring. We don't know if knowing e makes factoring easier, but no one has found a way to do it yet.

Other RSA uses

Encrypt with Bob's public key. Can only be decrypted with Bob's private key. Sending private messages to Bob.

Encrypt with Bob's private key. Can be decrypted with Bob's public key. Digital signature—only Bob can make such a message.

Encrypt with Bob's public key, then Alice's. Bob and Alice must work together to decrypt. Like an encrypted "and".