Encryption, ethics, etc.!

Robert Y. Lewis

CS 0220 2023

March 15, 2023

Overview

One-time pads

- Alice and Bob agree on a secret in advance.
- Alice can send one private message to Bob, or Bob to Alice, over public channels.
- Another message to send? Need another secret!

- Upsides: very strong encryption, very fast to encrypt/decrypt, messages go both directions.
- Downsides: can’t reuse secrets, need a private channel to arrange secret in the first place.
- Trusting in: the secret is actually secret.

RSA

- Bob generates a *public* key and a *private* key.
- Alice uses the public key to send a message to Bob over public channels.
- Bob decrypts the message with his private key.

- Upsides: Alice can send many messages, no need for a private channel.
- Downsides: could be broken (?), slower than a one-time pad, one directional.
- Trusting in: the difficulty of factoring huge numbers, ownership of the private key.

RSA summary

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).

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.)

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

As presented: if Bob is the one generating keys, RSA allows Alice to securely message Bob. What else can we do with this encryption scheme?

- 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."
- Encrypt a one-time pad that can now be sent over public channels.

Breaking RSA

The security of this encryption scheme depends on a “hard problem."

Wiki: In 2019, ... factored a 240-digit number (RSA-240) utilizing approximately 900 core-years of computing power. The researchers estimated that a 1024-bit RSA modulus would take about 500 times as long. (With classical computing techniques.)

Quantum computers can change how this scales. Known ways (Shor’s algorithm) to make this efficient in principle.

So, who can build a real quantum computer?

Musings about algorithms

It's easy to see algorithms (especially mathematical ones) as being neutral, equalizing.

RSA comes with a mathematical guarantee, so it must be safe, right? Not really.

Important to think of these things—bias, safety, correctness—in context. Is "trust" justified? Who is the adversary?

Closed or open?

COMPAS: a “tool” for predicting likelihood of recidivism for people convicted of crimes.

Notoriously closed-source and unexplained.

Claim: when an algorithm structures lived experience like this, developers have a responsibility for their work.

- Particularly if the algorithm is designed to let others avoid responsibility?

Can we analyze RSA (or encryption more generally) under similar terms? Does the “openness” (and/or simplicity) of RSA help or hinder it?