CS 198 Fall 2022 | Codebreaking | Ryan Cottone, Imran Khaliq-Baporia | Note 3

RSA

Heavy inspiration taken from the CS70 RSA note!

Now, we are equipped to define RSA. Unlike what we have been studying so far, RSA is not symmetric encryption – there is no single key to decrypt a ciphertext. Instead, there are public and private keys, where messages are tailored to specific keypairs.

Definition 1: Asymmetric Encryption

Asymmetric encryption does not rely on a shared secret, rather, parties publish a public key which others use to encrypt messages to them. Said party can then decrypt it with their private key, which is entirely separate and only known to themselves.

Let's take two people who wish to communicate in secret – Alice and Bob. Eve will try and eavesdrop on them.

First, Alice finds two prime numbers p, q which **must be kept secret**. Then, she finds $N = pq$, and chooses some exponent e such that e is relatively prime to $(p-1)(q-1)$. e is usually fixed as 65537.

She publishes both N and e. Her private key d is defined as $e^{-1} \pmod{(p-1)(q-1)}$. Notice that if Eve was able to find p and q, she could trivially find d the same way Alice derived it.

Bob this takes his message x and computes $x^e \pmod{N}$. This is the ciphertext sent to Eve. Since there exists no efficient method to finding x given this number, Eve cannot theoretically decrypt this. (More on this in the next note)

To decrypt this, Alice raises x^e to the power of d:

$$(x^e)^d \equiv x \pmod{N}$$

and recovers the original message. The proof of this can be found in the appendix – we highly recommend reading this if you are interested. However it is not crucial to understanding the content of this course.

Attacking RSA with CRT

Recall our RSA scheme in which Alice encrypts her message m to an end user by sending $m^e \pmod{N}$ for their N. Now, let's consider what happens when she encrypts the *same message* to multiple people using the same exponent $e = 3$:

$$C_1 = m^3 \pmod{n_1}$$
$$C_2 = m^3 \pmod{n_2}$$
$$C_3 = m^3 \pmod{n_3}$$

Notice that this looks similar to our CRT setup. We then set $x = m^e$ and re-arrange:

$$x = C_1 \pmod{n_1}$$
$$x = C_2 \pmod{n_2}$$
$$x = C_3 \pmod{n_3}$$

By the Chinese Remainder Theorem, we now know the value of $m^e \pmod{n_1 \cdot n_2 \cdot n_3}$. By itself, this doesn't seem like we've made much progress. However, there is a crucial difference between this and something like $x^e \pmod{n_1}$. The key is that x^3 *is less than* $n_1 \cdot n_2 \cdot n_3$.

Why does this matter? Well, there is no currently known algorithm for efficiently finding x given $x^e \pmod{n}$, mostly because x^e by itself could be greater than N, which means we cant just take the e-th root. In this example, however, it is **guaranteed** that N is greater than x^e.

This can be shown by expanding x^3 to $x \cdot x \cdot x$, and noting that $x < n_1$, $x < n_2$, and $x < n_3$ since x is reduced modulo each. If x were larger, it would be reduced modulo n and become strictly less than.

It follows that $x * x * x < n_1 \cdot n_2 \cdot n_3$. Knowing this, it is actually legal to take the cube root and recover x:

$$\sqrt[3]{x} \equiv x \pmod{n_1 \cdot n_2 \cdot n_3}$$

This x is our original message – we have successfully broken RSA in the case of a low exponent and a message repeatedly sent to different people. That might sound like a bit of a far-fetched case, but it is still a serious flaw that is often overlooked in older implementations. If (god forbid) you used RSA to encrypt a password to a secret group chat and sent it to your friends using a low exponent, it would be trivial for an eavesdropper to learn the key themselves.

This attack is most often mitigated by using $e = 65537$, which is often prohibitively high to perform this attack. Unless we had 65537 repeated messages, there would be no easy way to take the e-root of x and be sure it was correct.

Diffie-Hellman Key Exchange

Instead of using RSA to share a symmetric encryption key, there exists a much better scheme named the Diffie-Hellman Key Exchange (it even has key exchange in the name)!

This scheme is similar, but not quite equal to RSA. It uses the properties of modular arithmetic to share a value between two parties while keeping it safe from an eavesdropper.

Alice and Bob must first agree on three values: g and N, where g is the generator (base) and N is the modulus. Alice and Bob each choose a separate, private number a and b respectively. Alice computes $g^a \pmod{n}$ and sends it to Bob, while Bob computes $g^b \pmod{n}$ and sends it to Alice. Now they both know g^a and g^b modulo N.

The key to the system is that Alice and Bob then raise the other's result to the power of their own secret. For example, Alice now computs $(g^b)^a \pmod{n}$ and Bob $(g^a)^b \pmod{n}$. Both of these values are equivalent to $g^{ab} \pmod{n}$, which is their shared secret.

Eve knows both $g^a \pmod{n}$ and $g^b \pmod{n}$, but cannot easily find $g^{ab} \pmod{n}$. If she multiplied it, she would only find $g^{a+b} \pmod{n}$. The security of this system relies on the discrete logarithm problem, which we will cover next week.

Appendix: Proof of RSA

Recall that Bob has sent Alice $x^e \pmod{n}$ and Alice knows $d \equiv e^{-1} \pmod{(p-1)(n-1)}$.

$$\begin{aligned} & x^{ed} \\ & \quad = x^{(1+k(p-1)(q-1))} \\ & \quad = x(x^{k(p-1)(q-1)} - 1) \end{aligned}$$

We now prove that $x(x^{k(p-1)(q-1)} - 1)$ is divisible by p. This leaves two cases: x is divisible by p, and x is not divisible by p.

Case 1: x is divisble by p

$$\begin{aligned} x(x^{k(p-1)(q-1)} - 1) &\equiv 0 \pmod{N} \\ 0(x^{k(p-1)(q-1)} - 1) &\equiv 0 \pmod{N} \\ 0 &\equiv 0 \pmod{N} \end{aligned}$$

Case 2: x is not divisble by p

$$\begin{aligned} x(x^{k(p-1)(q-1)} - 1) &\equiv 0 \pmod{p} \\ x(1-1) &\equiv 0 \pmod{p} \\ 0 &\equiv 0 \pmod{p} \end{aligned}$$

We used FLT in the second step to reduce $x^{k(p-1)(q-1)} \pmod{N}$ to 1, since $x^{(p-1)} \equiv 1 \pmod{p}$. The same works for q, which means $x(x^{k(p-1)(q-1)} - 1) \equiv 0 \pmod{p}$.

Thus, we prove that $x^{ed} \equiv x \pmod{N}$, and that Alice can always decrypt a ciphertext that is encrypted with her public key.