RSA

Recall from **Cryptography Basics (https://canvas.duke.edu/courses/24695/pages/cryptography-basics)** that asymmetric crypto systems need a reasonably hard problem at the core of preventing easy computation of the private key f^{-1} given the public key f. In RSA algorithm, the core problem is just *prime factorization*.

In a nutshell, the RSA algorithm operates as follows:

- Find two (large) **primes** p and q. Compute their **product** $n = pq$.
- Find a positive number r that is coprime with $(p-1)(q-1)$. In other words, $\mathrm{GCD}\big(r, (p-1)(q-1)\big) = 1$.
- Find the **multiplicative inverse** of $r \bmod (p-1)(q-1)$. Let's call this number s, so that $rs \equiv 1 \pmod{(p-1)(q-1)}$.
  - We should use the Euclidean Algorithm to find s (because $(p-1)(q-1)$ is very much not a prime).

- The **public key** can be represented by the pair (r, n). More specifically, assume our plaintext is encoded as a number $x < n$. (Think of sending small chunks of information at a time.) Then the encryption function is $y = f(x) = x^r \bmod n$.
- The **private key** can be represented by the number s. Given any ciphertext y, we can decipher it back to the plaintext $f^{-1}(y) = y^s \bmod n$.

That's it. Now let's verify that f and f^{-1} are actually inverses of each other. In other words, we verify $f^{-1}\big(f(x)\big) = x$ for all possible plaintext f.

Proof. Since $rs \equiv 1 \pmod{(p-1)(q-1)}$, we can write it as $rs = k(p-1)(q-1) + 1$ for some integer k. Then we have

$$\begin{aligned}
x^{rs} &= x^{k(p-1)(q-1)+1} && \texttt{//} \\
&= x \times x^{(p-1)k(q-1)} && \texttt{// factoring out one } x \texttt{ and rearr} \\
&= x \times \left(x^{(p-1)}\right)^{k(q-1)} && \texttt{// treating } x^{(p-1)} \\
&= x \times \left(x^{(p-1)}\right) \times \left(x^{(p-1)}\right) \times \cdots \times \left(x^{(p-1)}\right)
\end{aligned}$$

Now, recall Fermat's Little Theorem tells us $x \times x^{(p-1)} = x^p \equiv x \pmod{p}$. Leveraging this fact, we have:

$$
\begin{aligned}
x^{rs} &= x \times \left(x^{(p-1)}\right) \times \left(x^{(p-1)}\right) \times \cdots \times \left(x^{(p-1)}\right) \\
&\equiv x \times \left(x^{(p-1)}\right) \times \cdots \times \left(x^{(p-1)}\right) \pmod{p} \qquad \texttt{// the first two terms} \\
&\equiv x \times \cdots \times \left(x^{(p-1)}\right) \pmod{p} \qquad \texttt{/} \\
&\vdots \\
&\equiv x \pmod{p}
\end{aligned}
$$

The process above tells us $x^{rs} \equiv x \pmod{p}$, which implies $p \mid (x^{rs} - x)$.

Now, let us realize that the entire process can be applied on q instead of p. Try replacing all p in the above by q and all q by p. This gives us $q \mid (x^{rs} - x)$.

But p and q are both primes. If $p \mid (x^{rs} - x)$ and $q \mid (x^{rs} - x)$, and both p and q are primes, we can conclude $pq \mid (x^{rs} - x)$.

Finally, notice $n = pq$. So $pq \mid (x^{rs} - x)$ actually means $x^{rs} \equiv x \pmod{n}$. In other words, we have

$$
\begin{aligned}
f^{-1}\left(f(x)\right) &= f^{-1}\left(x^r \bmod n\right) && \texttt{// encryption} \\
&= (x^r \bmod n)^s \bmod n && \texttt{// decryption} \\
&= x^{rs} \bmod n && \texttt{// mod rules} \\
&= x && \texttt{// because } x^{rs} \equiv x \pmod{n}
\end{aligned}
$$

Why is RSA (conceptually) secure? Note that only (r, n) is public information. To decrypt, one needs to know s, the multiplicative inverse of $r \bmod (p-1)(q-1)$. Given r and $(p-1)(q-1)$ this is easily computable by Euclidean Algorithm... except malicious parties do not know $(p-1)(q-1)$. We have only announced $n = pq$. **Assuming prime factorization is reasonably hard**, no one can figure out what p and q are in reasonable time, so no one knows what $(p-1)(q-1)$ is.

Remark.

- What is described here is the original concept of RSA. It is not the actual RSA used in real life now.
- We haven't addressed many issues:
    - How large should p and q be? (Nowadays, they should be about the size of 2048 binary digits.)
    - What is a good r? (Choosing a too small r will make the whole scheme prone to some deliberate attacks.)

- Is the encryption/decryption actually fast enough? (Not really. In practice, asymmetric systems are used to communicate the keys of symmetric systems.)
- Since the public key is, well, public, anyone can send the receiver some encrypted information. How do we prevent fake/malicious information being sent? (Read about digital