RSA Encryption

Alex Shaffer

RSA encryption

- RSA encryption is very different from anything we have seen so far.
- Involves a public key and a private key.
- You make you public key public so someone can encrypt a message with it that may only be decrypted with the private key.

Key Generation

- Choose two very (very) large prime numbers p and q.
  - These numbers make up your private key.
- Define $n = pq$
  - this is part of the public key.
- Compute $\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p\ - 1)(q\ - 1)$
  - Keep this private
- Choose a number e where $1 < e < \varphi(n)$ and $gdc\big(e, \varphi(n)\big) = 1$
  - This is part of the public key
- Determine $d = e^{-1}\big(mod\ \varphi(n)\big)$, that is d is the modular multiplicative inverse of e
  - This is part of the private key

Key continued

- The private key is made up of $p, q, \varphi(n)$, and d.
- The public key is made up of n, and e.

RSA encryption

- Consider a plaintext message as a number m:
  - The ciphertext is determined by $c \equiv m^e \ (mod\ n)$
  - Notice that the only thing involved in this calculation is the public key.
- To get the decrypted message back compute it as follows:
  - $m \equiv c^d \ (mod\ n)$

Justification

- Why does the decryption work?
- $c \equiv m^e \ (mod\ n) \to c^d \equiv (m^e)^d \ (mod\ n)$
- We want to show that $m \equiv m^{ed}(mod\ n)$
- Recall how we chose e and $d.$
- $ed \equiv 1(mod\ \varphi(n))$
- For some natural number $h, ed = 1 + h\ \varphi(n)$
- $m^{ed} = m^{1+h\ \varphi(n)} = m\left(m^{\varphi(n)}\right)^h \equiv m(1)^h \equiv m\ (mod\ n)$
  - This is just Euler's theorem in disguise

Why is it Secure

- The private key is entirely determined by the prime numbers p and q.
- n is determined by p and q, but n is made public.
- If you can factor n, then you can easily determine all of the private key.
- However, when p and q are chosen well, then n is hard to factor.

Intractable Problems

- Prime factorization of very large numbers is an example of an intractable problem.
- It is technically possible to solve, but with computational capabilities it can be hard in any useful time.

End of Main Notes

- After this there is talk about computation, this is extra material, but you don’t necessarily need to worry about it.

Classical Computation

- A classical computer can in some sense be reduced down to something that keeps track of numbers and does a certain number of operations per unit of time.
- If an algorithm with an input of size n is said to run in $O(n)$ then it takes n time some constant number of operations to run.
- Some problems, however, take a greater number of operations to run.
  - For example, $O(n^2), or\ O(b^n)$
- In general, it is great to find an algorithm that can solve a problem in polynomial runtin - $O(n^m)$ - where m is some constant power. However, sometimes exponential solutions - $O(b^n)$ - are the best we can do.

Prime Factorization

- Prime factorization is an example of a problem where different algorithms may be used to solve it.
  - The size of the problem n, is the number that is being factored.
- With a classical computer, there is no known way to factor a number in a polynomial number of operations.
  - You can do it with a quantum computer