1 Cryptography

Alice (A) and Bob (B) want to send messages (m) privately over a wire, but Charlie (C) is a snooper who has access to the data going across that wire. Alice and Bob want to communicate in a way which doesn't allow Charlie to learn about the message.

1.1 One Time Pad

- Communication
  - Prior to communicating Alice and Bob come up with S.
  - Alice computes $m_1 \oplus S$.
  - Bob decodes the message by doing $m_1 \oplus S \oplus S$ which gives m_1.
- From this exchange, Charlie learns $m_1 \oplus S$ which provides little to no information (depending on how much entropy is in S).
- There are several problems with this.
  - The biggest problem is reuse. If Bob communicates m_2 back to Alice with the same S, Charlie will have $m_1 \oplus S$ and $m_2 \oplus S$. Charlie could compute $(m_1 \oplus S) \oplus (m_2 \oplus S)$, giving $m_1 \oplus m_2$ which is much more vulnerable to being deciphered.
  - The second problem is that S has to be as long as M. If it is not, then part of the message will not be encoded

1.2 Symmetric Key Encryption

- A symmetric encryption scheme is comprised of a generator, encryptor, and decrypter.
- $SE = (G, E, D)$ (G is the generator, E is the encryptor, and D is the decrypter)
  - $k \leftarrow G$ (the generator generates keys; k denotes a key)
  - $c \leftarrow E_k(m)$ (the encryptor with a given key takes a message and creates a ciphertext; c denotes the ciphertext)
  - $m \leftarrow D_k(c)$ (the decrypter with a given key takes a ciphertext and gives the message)
- Communication
  - Both Alice and Bob agree on a key, k, from G.
  - Alice receives c from $E_k(m)$ from the encryptor.
  - Alice transmits c ($E_k(m)$) over the wire.
  - Bob decodes the message with $m = D_k(c)$.
- In this scheme Charlie sees only the ciphertext, $E_k(m)$, which will have high entropy, so he should learn little.
- This scheme is used today with AES because it solves the spying.
- The problem is it doesn't solve pre-sharing where Alice and Bob need to agree on some key beforehand (although the key will likely be much smaller than the OTP).

1.3 Public Key Cryptography

- $PKC = (G, E, D)$
  - $(pk, sk) \leftarrow G$ (generates a public key pk and secret key sk)
  - $c \leftarrow E_{pk}(m)$
  - $m \leftarrow D_{sk}(c)$
- Communication
  - Bob generates a public key and a secret key.
  - He broadcasts the public key to Alice.
  - Alice can now communicate c created from Bob's public key which only B can decode.
- Charlie will see pk and $E_{pk}(m)$.
- Problems
  - An integrity issue can arise if Charlie is able to repeat a message of Alice and be assumed to be Alice.
  - Another issue could arise if there are a small number of options which are sent often. Say a vote is taking place with two options and one message is sent 60% of the time and another message is sent 40% of the time.

1.4 Security

- One Time Pad achieves perfect secrecy (or information theoretic secure) which says that there is no positive correlation between the amount of ciphertext communicated and the amount learned about the message.
- In reality though, we can assume that snoopers like Charlie are bounded by realistic computational limits like Probabilistic Polynomial Time (PPT).
- If a Cryptographic Scheme the keym k, is n-bits, the probability of simply guessing the key is $Pr[\text{guessing } k] = 2^{-n}$.
- We say a Cryptographic Scheme is secure if $\forall$ PPT adversaries $Pr[\text{learn anything}] < 2^{-n}$.
- This essentially says the system is secure if the best option for someone without the key to decode the ciphertext is to guess keys.
- Although many people believe this holds for cryptographic systems, it is hard to prove and doing so would cause many major breakthroughs in computing.
- Many cryptographic schemes are made using algorithms and ideas from number theory.

1.5 Modular Exponentiation

```
def modexp(x, y, N):
  if y = 0, return 1
  z = modexp(x, y / 2, N)
  if y is even return z^2 (mod N)
  else         return z^2 * x (mod N)
```

- The first problem we'll discuss is $x^y \pmod N$
- Correctness

– Suppose y is even

$$z \equiv x^{y/2} (\text{mod } N)$$
$$z^2 \equiv x^{y/2} x^{y/2} (\text{mod } N)$$
$$z^2 \equiv x^2 (\text{mod } N)$$

– Suppose y is odd

$$z \equiv x^{(y-1)/2} (\text{mod } N)$$
$$z^2 \equiv x(x^{(y-1)/2})^2 \equiv x^{(y-1)/2+(y-1)/2+1}$$
$$(\text{mod } N) z^2 \equiv x^2 (\text{mod } N)$$

- Runtime: there are n recursive calls each doing multiplication of n bits, so the runtime is $O(n^3)$.

1.6 Primality

```
def isprime():
  choose a randomly
  return a^(n-1) mod n = 1
```

- Testing primality with certainty is difficult.
- Here is an algorithm which is fast (polynomial time by `modexp`) but it relies on random numbers. It also doesn't give certainty.
- There's a set of numbers (called Carmichael numbers) which are not prime but pass this test.

1.7 Euler Theorem (totient)

- $\varphi(n) = \#$ of numbers relatively prime to N which are less than N
- For a prime p, $\varphi(p) = p - 1$.
- For two primes p, q, $\varphi(pq) = (p-1)(q-1)$ (this does not work for p^2)
- $a^{\varphi(N)} \equiv 1 (\text{mod } N)$

1.8 Computation

- Showing that Public Key Cryptography is secure is a hard problem and would show that $P \neq NP$.
- Modern cryptographic algorithms rely on certain problems being hard.
  - Factoring: given $N = pq$, determine p, q
  - RSA: given $y, x^y (\text{mod } N), N$ determine x
  - Diffie-Hellman: given g^x, g^y, g, N determine $g^{xy} (\text{mod } N)$
- However, we're not sure if these are actually hard because we don't know if $P = NP$ or $P \neq NP$.
- Factoring is interesting because we can easily factor numbers with many small factors, so to make it hard, we often use two large factors.

1.9 RSA

- Communication
  - Bob generates large primes p, q
  - B then computes $N = pq$
  - B computes e a number relatively prime to $(p-1)(q-1)$
  - B computes $d \equiv e^{-1} (\text{mod } (p-1)(q-1))$
  - $pk = (N, e), sk = d$
  - B broadcasts pk
  - A computes and sends $m^e (\text{mod } N)$
  - B computes $m \equiv (m^e)^d (\text{mod } N)$
- Correctness

$$\begin{aligned} ed &\equiv 1 (\text{mod } (p-1)(q-1)) \\ ed &= k(p-1)(q-1) + 1 \\ (m^e)^d &\equiv m^{ed} (\text{mod } N) \\ (m^e)^d &\equiv m^{k(p-1)(q-1)+1} (\text{mod } N) \\ (m^e)^d &\equiv m^{k\varphi(N)+1} (\text{mod } N) \\ (m^e)^d &\equiv m(1)^k (\text{mod } N) \\ (m^e)^d &\equiv m (\text{mod } N) \end{aligned}$$

- Charlie will only see $N, e, m^e (\text{mod } N)$.
- Messages can be encoded in many ways (note that the cs 2050 message encoding technique is not used and somewhat problematic).
- We can't prove it's secure.

1.10 Diffie-Hellman

- A, B agree on p, g such that the two sets $(1, 2, ..., p-1)$ and $(g^1, g^2, ...g^{p-1}) (\text{mod } p)$ which are bijective.
- Communication
  - A computes x, g^x and communicates them to B.
  - B computes y, g^y and communicates them to A.
  - A computes $g^{xy} \equiv (g^y)^x (\text{mod } p)$
  - B computes $g^{xy} \equiv (g^x)^y (\text{mod } p)$
- Charlie sees g, p, g^x, g^y but can't compute $g^{xy} (\text{mod } p)$.
- This is commonly used to establish a symmetric encryption algorithm which is usually faster.