Cryptography - Khan Academy

Michel Liao*

April 2022

Contents

1 Ancient Cryptography **1**
1.1 General Introduction 1
1.2 The Enigma Encryption Machine 3

2 Ciphers **3**
2.1 Code vs. Cipher 3
2.2 Shift Ciphers 3
2.3 XOR Bitwise Operation 4

3 Modern Cryptography **4**
3.1 Public Key Cryptography 4
3.2 RSA Encryption 5
3.2.1 Example 6
3.3 Euler's Totient 6

4 Modular Arithmetic **6**

5 Primality Test **7**

6 Randomized Algorithms **7**

1 Ancient Cryptography

1.1 General Introduction

- Cryptography is the study of encoding and decoding messages.
- The **Caesar cipher** shifts each letter by a certain number.
  - E.g. A shifts to C, B shifts to D, etc.

*Thanks to Khan Academy for providing their cryptography course. All of the notes come from their videos and articles.

  - Every letter has a certain frequency, so we examine the frequency of letters to determine the shift (**frequency analysis**).

- The **polyalphabetic cipher** shifts each letter by a given word.
  - Assume the cipher word is 'SNAKE.' In numerical terms, where A is 1, B is 2, etc., SNAKE $\equiv$ 19 13 1 11 5.
  - If we want to send the message, "Meet me at elephant lake," then we shift M by 19, the first e by 13, the second e by 1, etc., where we repeat the pattern given by SNAKE.
  - We can analyze the message in intervals of 5 to decrypt it.[1]

- The **one-time pad** uses probability to generate a uniform distribution of letters used in the message.
  - Generate a sequence with as many integers as words. Shift each letter by its corresponding number in the sequence.

- The **frequency stability** property says that a truly random sequence will be equally likely to contain every sequence of any length.

- **Perfect secrecy** is where, regardless of computational power, a machine only serves as best as a guess.
  - On 9/12/1945, Claude Shannon proved, for the first time, that the one-time pad is perfectly secret.
  - The size of the message space is the same as the size of the key space and ciphertext space.

- We can generate a truly random sequence by sampling random fluctuations (**noise**) in nature. We visualize it by creating a **random walk**, or a path that changes according to each number.
  - John von Neumann generated a truly random sequence by the **middle-squares method**:
    1. Select a truly random number, or the **seed.**
       * E.g. measuring noise or the current time in milliseconds.
    2. Square the seed and output the middle of the resulting number.
    3. Use that number as the next seed and repeat the process.
  - The pseudorandom sequence repeats over time, unlike a random sequence. The **period** is the length before a pseudorandom sequence repeats.
    * The period is limited by the length of the seed[2].
  - We shrink the key space by using pseudorandom sequences.
  - As computers get faster, the practical safeness of the keys must increase.

[1] They're functionally a bunch of Caesar ciphers now.
[2] A n-digit seed can't have a greater period than 10^n.

1.2 The Enigma Encryption Machine

- A **rotor encryption machine** sums up three[3] numbers that are each a part of their own random sequence of integers 1 through 26. The sum serves as the shift.
  - The initial state of each device is called the **key setting.** All possible key settings are the **key space.**[4]
  - The daily key setting was distributed to each operator beforehand.
  - Operators were told to set their machine to a random position for each rotor before communication started. We, as humans, tend to not create random initial positions, however. It allowed the Allies to reverse engineer the rotor wirings.
  - The Enigma was also designed so that an input letter would never encrypt to itself, but that led to the Bombe.
    - It had multiple Enigma rotors chained together and took advantage of common words (**cribs**).
    - The cribs allowed the Allies to read the messages within hours of the Germans issuing them.

2 Ciphers

2.1 Code vs. Cipher

- A **code** is a mapping from a word/phrase into something else like symbols.
  - E.g. accountant could be equivalent to "Come at once. Do not delay."
  - A **codebook** lists codes.
- Ciphers do not involve meaning. They are algorithms performed on letters.
  - E.g. the Caesar Cipher is a **shift cipher** because each letter shifts to another: $A \to D, B \to E, C \to F$.

2.2 Shift Ciphers

- We can encrypt messages by doing the following to every letter in the message M:
  1. Convert the letter into a number where $A = 0, B = 1, C = 2$, etc.
  2. Calculate $Y \equiv X + K \pmod{26}$, where Y is the encrypted letter and K is the key.

[3]Not necessarily three–it's just used to illustrate the process.
[4]Think of column space in linear algebra.

3. Convert Y into its corresponding letter.

- We can decrypt messages by doing the following to every letter in the ciper text C:
  1. Convert the letter into a number Y.
  2. Calculate $X \equiv Y - K \pmod{26}$.
  3. Convert X into its corresponding letter.
- Shift ciphers are insecure via a **brute force attack**, or trying all 26 keys.

2.3 XOR Bitwise Operation

- **Bitwise** means we're dealing with individual bits, or **binary numbers.**
- The XOR bitwise operation outputs a 1 whenever the inputs do not match. It's functionally the same as addition modulo 2.
  - E.g. $100111001011010100111010 \, \textbf{XOR} \, 010110100001101111011000 = 110001101010111011100010$.
- XOR has a 50% chance of producing a 0 or a 1 while the AND and OR commands don't.

3 Modern Cryptography

3.1 Public Key Cryptography

- **The Fundamental Theorem of Arithmetic** says that every integer has a unique prime factorization.
- Internet grew, requiring cryptography.
- A **one-way function** is easy in one direction but hard in the other.
- Agree on a publicly on a starting color, then mix a private color into the starting color. Exchange the resulting mixtures. Mix your private color into the exchange mixture, resulting in the same mixture.
- The **Discrete Logarithm Problem**
  1. Find a prime modulus n, e.g. 17.
  2. Find a primitive root modulo n (aka the generator).
     - A number g is a primitive root modulo n if every number a coprime to n is equivalent to a pwoer of $g \pmod{n}$. I.e. $\exists k \in \mathbb{Z} \mid g^k \equiv a \pmod{n}$.
     - When raised to different exponents, g^k distributes uniformly to n's modulo residues.

3. The question is: how do you do the reverse? How do you find the exponent?

- **Solution:**

  1.

Diffie-Hellman Key Exchange needs review and notes

3.2 RSA Encryption

- RSA is the most widely used public key algorithm in the world.
- 1970 **James Ellis** worked on an idea for non-secret encryption.
- You can split the 'key' idea into an encryption key and decryption key.
- **Clifford Cocks** wanted to create a **trapdoor one-way function** (easy in one way, difficult in the reverse unless you have special information, i.e. the trapdoor).
  - A message is converted into a number m. Then, raise m to e, where e is a public number, and evaluate it modulo N to give c, where N is a random number. Output c.
  - Forward: $m^e \equiv ? \pmod{N}$, which is easy.
  - Backward: $?^e \equiv c \pmod{N}$, which is hard and requires trial-and-error.
- If you want to reverse the encryption, you need to find d (the decryption), where $c^d \equiv m \pmod{N}$. Equivalently, $m^{ed} \equiv m \pmod{N}$.
- We need to find ed such that it's hard to find out what d is, so we turn to another one-way function.
- Prime factorization is a fundamentally hard problem because of the requirement of trial-and-error.
- Generate a 150-digit long prime number, p_1. Generate another prime number roughly the same size, p_2. Find N, where $N = p_1 \cdot p_2$. You can give N to anyone.
- Euler's Theorem gives us an easy way to calculate d: $e \cdot d = k \cdot \phi(n) + 1$. d should be the private key.

3.2.1 Example

- Bob has a message he converted into a number m.
- Alice generates p_1 and p_2, prime numbers of similar size, and multiplies them to create $n = p_1 p_2$.
- Alice calculates $\phi(n) = (p_1 - 1)(p_2 - 1)$ and picks some small public exponent e such that e is odd and relatively prime to $\phi(n)$.
- Alice finds the value of her private exponent d and hides everything except for n and e.
- Bob gets n and e and locks his message by finding $m^e \equiv c \pmod{n}$. He sends c back to Alice.
- Alice finds $c^d \equiv m \pmod{n}$.
- Note that the public can only find d if they know $\phi(n)$, but that's really hard if n is large.

3.3 Euler's Totient

- Measures the breakability of a number
- Outputs the number of integers that are coprime with the input.
- $\phi(p) = p - 1$, where p is a prime number.
- The function is multiplicative: $\phi(A \cdot B) = \phi(A) \cdot \phi(B)$.
- **Euler's Theorem**: $m^{\phi(n)} \equiv 1 \pmod{n}$.
  - Corollary: $m^{k \cdot \phi(n)+1} \equiv m \pmod{n}$.

4 Modular Arithmetic

- Fast modular exponentiation for $a^b \pmod{c}$. We'll do $5^{117} \pmod{19}$.
  1. Convert b into binary.
     $$117 = 1110101_2 \implies 5^{117} \equiv 5^1 \cdot 5^4 \cdot 5^{16} \cdot 5^{32} \cdot 5^{64} \pmod{19}.$$
  2. Calculate the expansion, exploiting powers of two.
     $$5^2 \equiv 5 \cdot 5 \equiv 6 \pmod{19},\ 5^4 \equiv 6 \cdot 6 \equiv 17 \pmod{19} \ldots.$$
  3. Evaluate the multiplication.
     $$5^{117} \equiv 5 \cdot 17 \cdot 16 \cdot 9 \cdot 5 \equiv 1 \pmod{19}.$$
- **Euclidean Algorithm**:
  $$\gcd(a, b) = \gcd(r, b),$$
  where $a > b$ and $r = a - qb$ for some integer q.

5 Primality Test

- We don't have to check if a number is prime by checking each integer between 1 and $n-1$, inclusive. We can the interval $[1, \sqrt{n}]$.
- Bits (0 or 1) are now stored in tiny magnetic cells.
- The **Sieve of Eratosthenes** allows us to generate a list of primes up to an integer n.
  - Start at 2 and mark it if it's prime. Then, eliminate all multiples of 2. Then, keep increasing. If the next integer is not marked, it's prime and you can mark all multiples of it.
- $\frac{1}{\ln x}$ approaches the prime density up to some number x as $x \to \infty$.
- The **Prime Number Theorem** says that the number of primes less than or equal to x is $\frac{x}{\ln x}$.
- **Time-space tradeoff** refers to trading time for an algorithm to finish for the amount of space that algorithm takes.

6 Randomized Algorithms

- We don't need to prove that a given number is prime. We just need a to show it is very unlikely to be composite.
- **Fermat's Little Theorem**: $a^p \equiv a \pmod{p}$.
  - Corollary: $a^{p-1} \equiv 1 \pmod{p}$.
- **Fermat Primality Test**:
  1. Given an integer p, generate a random integer a less than p.
  2. Check if $\gcd(p, a) = 1$. If no, then p is composite.
  3. If yes, then check if $a^{p-1} \equiv 1 \pmod{p}$. If no, then it's composite.
  4. If yes, then p is most likely prime. Note that p could be composite and Fermat's Little Theorem could hold, so it's not guaranteed. Call such p's "fools."
  5. The number of fools must divide the total size of the group we select from. Thus, the probability of getting a fool is less than or equal to $\frac{1}{2}$. So, we can just choose a lot of a's.