Midterm Review

- **Midterm Review**
  - Mathematics
    - Great Common Divisor
    - Multiplicative Inverse
    - Invertible Matrix
    - Fermat's Little Theorem
    - Chinese Remainder Theorem
    - Group
    - Abelian Group
    - Ring
    - Commutative Ring
    - Field
    - Finite Field
    - Group order
    - Element order
    - Generator
    - Integer Group
    - Euler's Totient Function
    - Subgroup
    - Square Root Lemma
    - Square Root to Factors
    - Square Root modulo p
    - Legendre Symbol
    - Jacobi Symbol
    - Square and Multiply for Modular Exponentiation
    - Solovay-Strassen Primality Test
    - Miller-Rabin Test
    - Pollard's Rho Factorization
    - Pollard's Rho Discrete Logarithm Problem
    - Pollard's Hellman Algorithm
      - Groups of Prime Power Order
      - General Groups
  - Cryptography
    - Types of Attack
    - Kerckhoff's Principle
    - Caesar Cipher
    - Substitution Ciphers
    - One Time Pad
    - Block Ciphers
    - Hill Cipher
    - Symmetric Keys
    - Public Key Cryptography
    - Security Level
    - Zero Knowledge Proof
    - Block Ciphers
    - Randomness
    - BBS Generator
    - Feistel Network
    - Advanced Encryption Standard (AES)
      - Sub-bytes
      - Shift-rows
      - Mix-columns
      - Add Round Keys
      - Structures
    - RSA Cryptosystem
      - setup
      - encryption
      - decryption
    - Diffie-Hellman Key Exchange
    - Elgamal Cryptosystem
      - setup
      - encryption
      - decryption
    - Hash Function Identities
    - DLP Hash Function
    - Birthday Attack
    - Compression Function
    - Merkle-Damgard Construction
    - SHA-1
      - padding
      - compression
      - algorithm
  - Problem
    - Quadratic Residuosity Problem (QR)
    - RSA Problem
    - Discrete Logarithm Problem (DLP)
    - Computation Diffie-Hellman Problem (CDH)
    - Decisional Diffie-Hellman Problem (DDH)

Mathematics

Great Common Divisor

$d = \gcd(a, b)$ where a or b is non-zero integer

there exists integers s, t where $as + bt = d$

```
function ext_gcd(a, b):
    r0 = b
    r1 = a
    s0 = 0
    s1 = 1
    t0 = 1
    t1 = 0
    while r0 != 0:
        q = r1 / r0
        r1, r0 = r0, r1 - q * r0
        s1, s0 = s0, s1 - q * s0
        t1, t0 = t0, t1 - q * t0
    return r1, s1, t1
```

Multiplicative Inverse

a^{-1} is called the multiplicative inverse of a modulo p if $a^{-1}a \equiv 1 \bmod p$

a^{-1} doesn't exist if a is not invertible modulo p, indicating $\gcd(a, p) > 1$

Invertible Matrix

$A^{*}A = |A|E$

matrix A is invertible modulo p if and only if $|A|$ is invertible modulo p

the inverse of a matrix A modulo p is $A^{-1} \equiv tA^{-1} \bmod p$ where $t|A| \equiv 1 \bmod p$

Fermat's Little Theorem

Let $p \in \mathbb{N}$ and $a \in \mathbb{Z}$. If p is a prime and $p \nmid a$, then $a^{p-1} \equiv 1 \bmod p$

Generally $a^{p} \equiv a \bmod p$ for $p \in \mathbb{N}$ and $a \in \mathbb{Z}$

Chinese Remainder Theorem

If we have a modular equations for x modulo $m = \prod_{i} [m_i]$

$\{x \equiv a_i \bmod m_i\}$ where $m_i = p_i^{e_i}$ and p_i is a prime

calculate $M_i = \dfrac{m}{m_i}$ and $d_i \equiv M_i^{-1} \bmod m_i$

the result is given by $x \equiv \sum_{i} \left[a_i M_i d_i\right]$

Group

A group is defined as $(G, \circ)$ consisting of a set G and a group operation $\circ : G \times G \mapsto G$ satisfying

- associativity — $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$
- unit element — $\exists e \in G, \forall a \in G : a \circ e = e \circ c = a$
- inverse element — $\forall a \in G, \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$

Abelian Group

An abelian group is defined as $(G, \circ)$ which is a group satisfying

- commutativity — $\forall a, b \in G : a \circ b = b \circ a$

Ring

A ring is defined as $(R, \cdot, +)$ consisting of a set R and two group operations $\cdot, + : R \times R \mapsto R$ satisfying

- $(R, +)$ is an abelian group
- multiplicative unit — $\exists 1 \in R, \forall a \in R : a \cdot 1 = 1 \cdot a = a$
- associativity — $\forall a, b, c \in R : a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- distributivity — $\forall a, b, c \in R : a \cdot (b + c) = a \cdot b + a \cdot c$

Commutative Ring

A commutative ring is defined as $(R, \cdot, +)$ which is a ring satisfying

- commutativity — $\forall a, b \in R : a \cdot b = b \cdot a$

Field

A field is defined as $(F, \cdot, +)$ which is a commutative ring satisfying

- unit of addition $0 \neq$ unit of multiplication 1
- $\forall a \in F \backslash \{0\}, \exists a^{-1} : a \cdot a^{-1} = 1$

Finite Field

Prime Fields $\mathbb{F}_p = \{0, 1, \cdots, p-1\}$ where p is a prime

- addition — $a + b \bmod p$
- multiplication — $a \times b \bmod p$

Galois Field $\mathbb{F}_{\mathrm{p}^{\mathrm{n}}} = GF(p^n)$ can be constructed as $\mathbb{F}_{\mathrm{p}^{\mathrm{n}}} = \{a_0 x^0 + a_1 x^1 + \cdots a_{n-1} x^{n-1} \mid a_i \in \mathbb{F}_{\mathrm{p}}\}$

- irreducible polynomials $P(x)$ with degree n
- coefficients follows the computation in prime fields $\mathbb{F}_p$
- addition — $a + b \bmod P(x)$
- multiplication — $a \times b \bmod P(x)$

Group order

the order k of a group G is its cardinality, the number of its elements

Element order

the order of an element A in a group G is the smallest integer k such that $a^k = 1$

Generator

A generator (primitive element) is the element having the same order with its group

Integer Group

$\mathbb{Z}/n\mathbb{Z}$ contains all the integers modulo n

- its order is n

$U(\mathbb{Z}/n\mathbb{Z}) = \mathbb{Z}_n^* = \mathbb{Z}_n^\times$ contains all the invertible elements in $\mathbb{Z}/n\mathbb{Z}$

- its order is $\varphi(n)$
- if n is an odd prime, $\alpha \in U(\mathbb{Z}/n\mathbb{Z})$ is a generator if and only if $\alpha^{\frac{p-1}{q}} \not\equiv 1 \bmod p$ for all $q \mid p-1$

Euler's Totient Function

For a group $\mathbb{Z}/n\mathbb{Z}$, $\varphi(n)$ counts the number of invertible elements.

Given $n = \prod_i \left[p_i^{e_i}\right]$, we can calculate $\varphi(n) = n \prod_i \left[\frac{p_i - 1}{p_i}\right]$

For all elements $a \in U(\mathbb{Z}/n\mathbb{Z})$, $a^{\varphi(n)} = 1$

Subgroup

The order of a subgroup divides the order of its original group

Any element in $U(\mathbb{Z}/n\mathbb{Z})$ can generate a subgroup

Square Root Lemma

For $p \equiv 3 \bmod 4$, there doesn't exist x satisfying $x^2 \equiv 1 \bmod p$

Square Root to Factors

Let $n = pq$ where p and q are primes and $p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$

Let $x \equiv \pm a, \pm b \bmod n$ be the four solutions of $x^2 \equiv y \bmod n$

Then $\gcd(a - b, n)$ is a non-trivial factor of n

Square Root modulo p

For an odd prime p and $a \not\equiv 0 \bmod p$, we can deduce $a^{\frac{p-1}{2}} \equiv \pm 1 \bmod p$

a is a square modulo p if and only if $a^{\frac{p-1}{2}} \equiv 1 \bmod p$

Legendre Symbol

Legendre Symbol is defined for an odd prime p and $a \not\equiv 0 \bmod p$

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \exists x : x^2 \equiv a \bmod p & a \text{ is a square modulo } p \\ -1 & \forall x : x^2 \not\equiv a \bmod p & a \text{ is a not square modulo } p \end{cases}$$

- $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$ if $a \equiv b \bmod p$
- $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p$
- $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$
- $\left(\frac{-1}{p}\right) = 1$ if $p \equiv 1 \bmod 4$

Jacobi Symbol

Jacobi Symbol is defined for an odd integer p and $a \not\equiv 0 \bmod n$

$$\left(\frac{a}{p}\right) = \prod_i \left(\frac{a}{p_i}\right)^{e_i}$$

Jacobi Symbol only works with $\left(\frac{a}{p}\right) = -1$ to show that a is not a square modulo p

- $\left(\frac{a}{p}\right) = \left(\frac{b}{n}\right)$ if $\gcd(a, p) = 1$
- $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$ if $\gcd(ab, p) = 1$
- $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}$
- $\left(\frac{2}{p}\right) = \begin{cases} +1 & p \equiv 1, 7 \bmod 8 \\ -1 & p \equiv 3, 5 \bmod 8 \end{cases}$
- $\left(\frac{a}{p}\right) = \begin{cases} -\left(\frac{p}{a}\right) & m \equiv n \equiv 3 \bmod 4 \\ +\left(\frac{p}{a}\right) & \text{otherwise} \end{cases}$ if a is odd and $\gcd(a, p) = 1$

Square and Multiply for Modular Exponentiation

An algorithm calculate $c = m^d \bmod n$

Complexity — $O(\log(n)^2 \log d)$ bit operations

```
function mod_exp(m, d):
    res = 1
    d = binary(d)
    k = len(d)
    for i in range(k - 1, -1, -1):
        res = (res * res) % n
        if d[k] == 1:
            res = (m * res) % n
    return res
```

Solovay-Strassen Primality Test

Complexity — $O(k(\log n)^3)$ operations

```
function solovay_strassen_test(n):
    for _ in range(k):
        a = random.randint(2, n - 2)
        if gcd(a, n) != 1:
            return False
        x = legendre(a, n)
        y = a ** ((n-1) / 2) % n
        if x % n != y:
            return False
    // It shows that n has a high probability to be a prime if k large enough
    return True
```

Miller-Rabin Test

```
function miller_rabin_test(n):
    m = (n - 1) / 2
    s = 1
    while m % 2 == 0:
        m = m / 2
        n = n + 1
    for _ in range(k):
        a = random.randint(2, n - 2)
        if gcd(a, n) != 1:
            return False
        a = a ** m % n
        if a == 1 or a == n - 1:
            continue
        for j in range(1, s):
            a = a ** 2
            if a % n == 1:
                return False
            if a % n == -1:
                break
        else:
            return False
    // It shows that n has a high probability to be a prime if k large enough
    return True
```

Pollard's Rho Factorization

Complexity — $O(\sqrt[4]{n})$ operations

```
function pollards_rho_factorization(n, func=lambda x: x ** 2 + 1):
    a = b = 2
    d = None
    while d != 1:
        a = func(a)
        b = func(func(b))
        d = gcd(a - b, n)
    return False, None if d == n else True, d
```

Pollard's Rho Discrete Logarithm Problem

To calculate x satisfying $\alpha^x = \beta$ in G with prime order p

Partition the original group into three small groups S_1, S_2, S_3

For example $S_1 = \{x \in G, x \equiv 1 \bmod 3\}$, $S_2 = \{x \in G, x \equiv 0 \bmod 3\}$, $S_3 = \{x \in G, x \equiv 2 \bmod 3\}$

Three predefined functions

$$f(x) = \begin{cases} \beta x & x \in S_1 \\ x^2 & x \in S_2 \\ \alpha x & x \in S_3 \end{cases} \qquad g(a, x) = \begin{cases} a \bmod p & x \in S_1 \\ 2a \bmod p & x \in S_2 \\ a + 1 \bmod p & x \in S_3 \end{cases} \qquad h(b, x) = \begin{cases} b + 1 \bmod p & x \in S_1 \\ 2b \bmod p & x \in S_2 \\ b \bmod p & x \in S_3 \end{cases}$$

```
function pollards_rho_dlp(alpha, beta, p):
    a1 = b1 = a2 = b2 = 0
    x = y = 1
    start = True
    while (x - y) % p != 0 or start:
        start = False
        a1 = g(a1, x)
        b1 = h(b1, x)
        x = f(x)
        a2 = g(a2, y)
        b2 = h(b2, y)
        y = f(y)
        a2 = g(a2, y)
        b2 = h(b2, y)
        y = f(y)
    r = (b1 - b2) % p
    if r != 0:
        return True, inverse(r, p) * (a1 - a2) % p
    return False, None
```

Pollard's Hellman Algorithm

Assume the problem $h \equiv g^x \bmod a$ is given and we want to calculate $x = \log_g h$

Groups of Prime Power Order

If order $n = p^e$ and p is a prime, the solution x can be written in the form of

$$x = \sum_{k=0}^{e-1} d_k p^k \quad d_k \in \mathbb{F}_p$$

We can compute d_k and x in the following iterated algorithm

1. Determine all parameters a, n, p, e, g, h
2. Compute $\gamma \equiv g^{p^{e-1}} \bmod a$
3. Set $x_0 = 0$
4. For $k = 0 \to e - 1$
    1. Calculate inverse $m_k \equiv (g^{x_k})^{-1} \bmod a$
    2. Calculate $h_k \equiv (m_k \cdot h)^{p^{e-1-k}} \bmod a$
    3. Choose $d_k \in \mathbb{F}_p$ so that $\gamma^{d_k} \equiv h_k \bmod a$
    4. Calculate $x_{k+1} = x_k + d_k p^k$
5. We get all d_k and the solution $x = x_e$

General Groups

If order $n = \prod_{i=1}^{r} p_i^{e_i}$

We can decompose $n - 1$ to several prime power groups problems using the following algorithm

For $i = 1 \to r$

1. Calculate $u_i = \dfrac{n}{p_i^{e_i}}$
2. Calculate $g_i \equiv g^{u_i} \bmod a$ and $\mathrm{ord}(g_i) = p_i^{e_i}$
3. Compute $h_i \equiv h^{u_i} \bmod a$
4. Determine x_i by prime power group algorithm where $g_i^{x_i} = h_i$

We may find that for all i

$(g_i)^x = h_i \qquad (g_i)^{x_i} = h_i \qquad (g_i)^{p_i^{e_i}} \equiv (g_i)^{\mathrm{ord}(g_i)} \equiv 1 \bmod a$

Then we can conclude $x \equiv x_i \bmod p_i^{e_i}$

We can give the final answer x by solving

$x \equiv x_i \bmod p_i^{e_i} \quad i = 1, 2, \cdots, r$

with Chinese Remainder Theorem

Cryptography

Types of Attack

- blind attack — only have the cipher-text
- Known Plain-text Attack (KPA) — have plain-text and corresponding cipher-text
- Chosen Plain-text Attack (CPA) — can choose plain-text to be encrypted
- Chosen Cipher-text Attack (CCA) — can choose cipher-text to be decrypted
- Chosen Plain-text and Cipher-text Attack (CPCA) — can choose plain-text to be encrypted and cipher-text to be decrypted

Kerckhoff's Principle

A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

Caesar Cipher

corresponds letters and integers (a-z) ~ (0-25)

a private key $0 \le \kappa \le 25$

encryption $c \leftarrow m + \kappa$

decryption $m \leftarrow c - \kappa$

Substitution Ciphers

replace each letter with another symbol

frequency attack

monogram — e, t, a, o, i, n, s, r, h, ...

digram — th, ...

One Time Pad

given a message m with bit length l

private key k with bit length l

encryption $c \leftarrow m \oplus k$

decryption $m \leftarrow c \oplus k$

Block Ciphers

can avoid frequency attack

one changed letter in plain-text may affect multiple changes in cipher-text

one erroneous letter in cipher-text may affect multiple errors in decrypted cipher-text

Hill Cipher

private key $n \times n$ matrix K with $\gcd(|K|, 26) = 1$

filling the messages to length l to satisfy $n \mid l$

splitting message m into several blocks m_i each with length n

encryption $c_i = m_i K$

decryption $m_i = c_i K^{-1}$

Breaking Hill Cipher by KPA

1. Guess value n by $n \mid l$
2. fetch n sequential blocks of messages to be a $n \times n$ matrix A until A is invertible modulo 26
3. get corresponding n sequential blocks of cipher-texts to be a $n \times n$ matrix C
4. calculate $K = A^{-1}C$ where $AK = C$

Symmetric Keys

The symmetric keys scheme is one of the following

- use the same key both in encryption and decryption
- the decryption key is easily derived from the encryption key

Drawbacks

- need to discuss the keys in advance

- n people needs $n(n-1)$ keys for absolute security

Public Key Cryptography

Also referred to asymmetric key cryptography
use public keys to encrypt the message and private keys to decrypt the message
hard to generate private keys from public keys

Security Level

secure — 2^{128}

very hard — 2^{80}

hard — 2^{64}

easy — 2^{56}

Zero Knowledge Proof

B can prove his identity to A
A cannot steal any information from B
Nobody can cheat A

Block Ciphers

Electronic Code Block (ECB)
Cipher Block Chaining (CBC)
- use an initialization vector IV
- iterative encryption but parallel decryption

Counter (CTR)

Randomness

random — We say that x is random if and only if it is not larger than any program that can produce it in any language

entropy — The entropy of x is the minimum number of bits necessary to describe x

BBS Generator

Two large primes p and q and set $n = pq$

Choose a random integer x coprime to n

$\begin{cases} x_0 \equiv x^2 \bmod n \\ x_{i+1} \equiv x_i^2 \bmod n \end{cases}$
At each iteration choose the least significant bit of x_i
BBS is secure based on QR problem

Feistel Network

a bijection over $2n$-bit message based on a one-way function $F : \{0,1\}^n \mapsto \{0,1\}^n$

partition the message into two parts L and R, each with n bits

one iteration $\Psi_F : \{0,1\}^{2n} \mapsto \{0,1\}^{2n}$ is defined as $[L, R] \leftarrow [R, L \oplus F(R, K)]$ and K is private keys

define $\sigma : \{0,1\}^{2n} \mapsto \{0,1\}^{2n}$ as $[L, R] \leftarrow [R, L]$, then $\Psi_F^{-1} = \sigma \circ \Psi_F \circ \sigma$

generally $\Psi^{-n} = \sigma \circ \Psi^n \circ \sigma$ for n-round Feistel Network

Rounds	KPA	CPA	CPCA
1	1	1	1
2	$O\left(\sqrt{2^n}\right)$	2	2
3	$O\left(\sqrt{2^n}\right)$	$O\left(\sqrt{2^n}\right)$	3
4	$O\left(2^n\right)$	$O\left(\sqrt{2^n}\right)$	$O\left(\sqrt{2^n}\right)$

Advanced Encryption Standard (AES)

Finite Fields $\mathbb{F}_{2^8}$ with $P(x) = x^8 + x^4 + x^3 + x + 1$ is used for calculation

For each layer, it will receive a 128-bit message divided into 4×4 blocks each with one byte

The 4×4 blocks is represented as $a_{i,j}$, and the output is represented as $c_{i,j}$

Sub-bytes

S-box calculation for each given byte $a_{i,j}$

- find $b = a_{i,j}^{-1} \bmod P(x)$ and transform it to be a column vector
- calculate $c = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} b + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$

We can directly get c by looking up S-box table S

$c = S[a] = S[\text{row}=\overline{a_7 a_6 a_5 a_4}, \text{column}=\overline{a_3 a_2 a_1 a_0}]$

Shift-rows

For each row i, $c_{i,j} = a_{i,m}$ where $m = j + i \bmod 4$

Mix-columns

For each column $a_j = \begin{pmatrix} a_{1,j} & a_{2,j} & a_{3,j} & a_{4,j} \end{pmatrix}^T$

$$c_j = \begin{pmatrix} c_{1,j} & c_{2,j} & c_{3,j} & c_{4,j} \end{pmatrix}^T = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} a_j$$

Add Round Keys

The original 128-bit is organized to 4×4 blocks each with one byte

arranged by columns $K(0), K(1), K(2), K(3)$, we need to calculate $K(i)$ where $4 \le i \le 43$ for ten more rounds

For $i \not\equiv 0 \bmod 4$, $K(i) = K(i-1) \oplus K(i-4)$

For $i \equiv 0 \bmod 4$

- Compute $r(i) = 00000010^t$ where $t = \dfrac{i-4}{4}$
- left shift $K(i-1)$ by 1
- S-box transformation to $\begin{pmatrix} a & b & c & d \end{pmatrix}^T$
- $T(K(i-1)) = \begin{pmatrix} a \oplus r(i) & b & c & d \end{pmatrix}^T$
- $K(i) = K(i-4) \oplus T(K(i-1))$

In each round i, we get the keys $k = \begin{pmatrix} K(i) & K(i+1) & K(i+2) & K(i+3) \end{pmatrix}^T$ and calculate $c = a \oplus k$

Structures

```
function AES_encryption(m, k)
    AddRoundKey(m, round=0)
    for r in range(1, 10):
        Subbytes(m)
        ShiftRows(m)
        MixColumns(m)
        AddRoundKey(m, round=r)
    Subbytes(m)
    ShiftRows(m)
    AddRoundKey(m, round=10)
```

RSA Cryptosystem

setup

two large primes p, q and $n = pq, \varphi(n) = (p-1)(q-1)$

public key e

private key $d \equiv e^{-1} \bmod \varphi(n)$

encryption

given a message m, send $c \equiv m^e \bmod n$

decryption

receiving a cipher-text c, calculate $c^d \equiv m^{ed} \equiv m \bmod n$

Diffie-Hellman Key Exchange

A and B publicly agree on — a group G with prime order p and one generator α

A randomly choose $x \in G$ and B randomly choose $y \in G$

A sends α^x to B and B sends α^y to A

Both A and B get secret data α^{xy}

Elgamal Cryptosystem

Elgamal is based on the security of CDH

setup

group G with prime order p and its generator α

secret integer x and public key $\beta \equiv \alpha^x \bmod p$

encryption

select a random integer k

calculate $r \equiv \alpha^k \bmod p$

compute $t \equiv \beta^k m \bmod p$

set $c = (r, t)$

decryption

compute $tr^{-x} \equiv \alpha^{xk} m \alpha^{-xk} \equiv m \bmod p$

Hash Function Identities

For a cryptographically secure hash function h

- efficiently computed for any input
- pre-image resistant — given y, finding x such that $h(x) = y$ is computationally infeasible
- second pre-image resistant — given x, finding x' such that $h(x) = h(x')$ is computationally infeasible
- collision resistant — finding any pair x and x' such that $h(x) = h(x')$ is computationally infeasible

DLP Hash Function

Let p be a prime and $q = \dfrac{p-1}{2}$ is also a prime. We find two generators α and β for $U(\mathbb{Z}/p\mathbb{Z})$

For any input $x \in U(\mathbb{Z}/q^2\mathbb{Z})$, we represent it as $x = x_0 + x_1 q$

The DLP hash function $h : U(\mathbb{Z}/q^2\mathbb{Z}) \mapsto U(\mathbb{Z}/p\mathbb{Z})$ is defined as $h(x) = \alpha^{x_0}\beta^{x_1} \bmod p$

Birthday Attack

Can save storage by Pollard's Rho idea

Can find a collision in complexity $O(\sqrt{n})$

One can add insignificant noises over the texts to avoid birthday attack

Compression Function

A function $g : \{0,1\}^{m+t} \mapsto \{0,1\}^m$ with $t > 0$ is called a compression function

Merkle-Damgard Construction

Merkle-Damgard construction is based on a collision resistant compression function $g : \{0,1\}^{m+t} \mapsto \{0,1\}^m$ and $t \geq 2$

Given an input message x with n bits

- split x into $k = \left\lceil \frac{n}{t-1} \right\rceil$ blocks, each with $t-1$ bits
- calculate remaining bits needed to be filled $d = n - (t-1)k$
- set y from x — $\begin{cases} y_i = x_i & 1 \leq i \leq k-1 \\ y_k = x_k || 0^d \\ y_{k+1} = (d)_2 \end{cases}$
- compute $z_1 = g(0^{m+1} || y_1)$
- iteratively compute $z_{i+1} = g(z_i || 1 || y_i)$ for $1 \leq i \leq k$
- output $h(x) = z_{k+1}$

Merkle-Damgard construction is ensured to yield a collision resistant hash function

SHA-1

given an input message x

padding

We first perform padding operation to input message x

- record the original length d
- $x_1 = x || 1$
- $x_2 = x || 0^k$ where $k \equiv 447 - d \bmod 512$ satisfying the length d' of x_2 satisfying $d' - 64 \equiv 0 \bmod 512$
- $x_3 = x || (d)_2$ where $(d)_2$ takes up 64 bits
- output the message $y = x || 1 || 0^k || (d)_2$

The output message y can be divided into $k = \left\lfloor \frac{d}{512} \right\rfloor + 1$ each with 512 bits

compression

one predefined function

$$f_i(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \leq i \leq 19 \\ B \oplus C \oplus D & 20 \leq i \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leq i \leq 59 \\ B \oplus C \oplus D & 60 \leq i \leq 79 \end{cases}$$

one predefined constant

$K =$ `5A827999 6ED9EBA1 8F1BBCDC CA62C1D6`

```
// H is a 5-element list, each with 32-bit values
// y is a 512-bit input
function compress(H, y):
    // split y into 16 words W (1 word = 32 bit)
    W = split(y, size=16)
    // extension for W
    for i in range(16, 80):
        W[i] = ROTL(W[i-3] ^ W[i-8] ^ W[i-14] ^ W[i-16], n=1)
    (A, B, C, D, E) = H
    for i in range(80):
        T = ROTL(A, n=5) + f(i, B, C, D) + E + W[i] + K[i]
        E = D
        D = C
        C = ROTL(B, n=30)
        B = A
        A = T
    H[0] = H[0] + A
    H[1] = H[1] + B
```

```
    H[2] = H[2] + C
    H[3] = H[3] + D
    H[4] = H[4] + E
    return H
```

algorithm

```
function SHA_1(x):
    H = []
    H.append(0x67452301)
    H.append(0xEFCDAB89)
    H.append(0x98BADCFE)
    H.append(0x10325476)
    H.append(0xC3D2E1F0)
    // padding
    y = padding(x)
    k = int(len(y) / 512)
    // split y into k blocks, each with 512 bits
    y = split(y, size=k)
    for i in range(k):
        // compression
        H = compress(H, y[i])
    // concatination
    return concat(H)
```

Problem

Quadratic Residuosity Problem (QR)

Let $n = pq$ where p and q are primes. Let y be an integer $\left(\frac{y}{n}\right) = 1$, determining whether y is a square modulo n.

QR problem is hard, as hard as factorizing n

RSA Problem

Let n is a large integer and e is a positive integer coprime to $\varphi(n)$. Given $y \in U(\mathbb{Z}/n\mathbb{Z})$, determine x satisfying $x^e \equiv y \bmod n$

Discrete Logarithm Problem (DLP)

Let $\mathbb{F}_q$ be a finite field with $q = p^n$, and G is a subgroup of $\mathbb{F}_q^*$. Given a generator α of G and $\beta \in G$, determine x satisfying $\beta = \alpha^x$ in $\mathbb{F}_q$

Computation Diffie-Hellman Problem (CDH)

Let G be a group of prime order p and α be a generator of G, given α^x and α^y with unknown x, y, determine α^{xy}

Decisional Diffie-Hellman Problem (DDH)

Let G be a group of prime order p and α be a generator of G, given α^x and α^y with unknown x, y, determine whether $\alpha^c = \alpha^{xy}$ in G for any given $c \in G$