Goals in Crypto

1. Confidentiality: Keeping secret data secret
2. Integrity: Preventing modification
3. Authentication: Preventing frauds
4. Non-repudiation: Preventing denials of messages sent

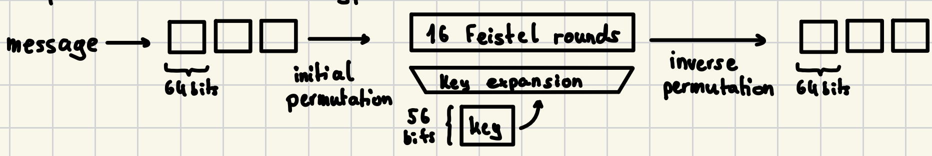
Symmetric crypto

$$
\begin{tikzcd}
\text{Plain text} \arrow[r] & \text{Encryption} \arrow[d] & \text{Key} \arrow[l] \arrow[r] & \text{Decryption} \arrow[r] & \text{Plain text} \\
 & \text{Ciphertext} \arrow[rr] & & \text{Ciphertext} \arrow[u] &
\end{tikzcd}
$$

Examples:

- **Substitution cipher:** Substitute each letter with the corresponding letter according to SK, a bijective 1-to-1 function
- **OTP:** $K = M = C = \{0,1\}^n$, $E(k,m) = k \overset{\text{XOR}}{\oplus} m$, $D(k,c) = k \oplus c$
  → Has perfect secrecy: The ciphertext reveals no info about the plaintext
  ↳ No ciphertext only attacks
  ↳ $|K| \geq |M|$
  → Never use same K twice because $c_1 \oplus c_2 = m_1 \oplus m_2$
  → K has to be long for perfect secrecy
- **Stream cipher:** Same as OTP, but K is generated based on seed s
  → Uses PRGs
- **Block cipher:** Encrypt data in fixed-size chunks
  → Different mode of operation:
  - Electronic codebook (ECB): Each block is encrypted independently
    → vulnerable to pattern attacks
  - Cipher block chaining (CBC): Each block is XORed with previous block before encryption
    → requires init vector (IV)
  - Counter mode (CTR): Counter value (counter + nonce) is encrypted and XORed with plaintext

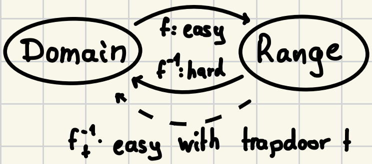
  → Implementation: Data encryption standard (DES)



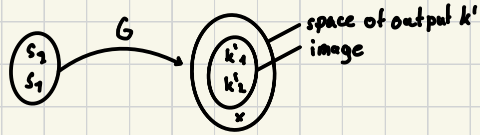
Unsafe → Triple DES (3DES):
- $3E((k_1,k_2,k_3), \text{block}) = E(k_1, D(k_2, E(k_3, \text{block})))$
- backwards compatability: 3DES = DES if $k_1 = k_2 = k_3$
- attack requires 2^{118} time, so it's safe ($> 2^{90}$) but not efficient

- 2DES would be enough, 2^{112}, but vulnerable to Meet-in-the-Middle attacks:
  - $E(k_1, E(k_2, m)) = c \iff E(k_2, m) = D(k_1, c)$
  - Build lookup table for all k_1, k_2 and find match
  - Can be done in 2^{63}

Trapdoor Functions



Pseudorandom Generators (PRG)



- $G: \{0,1\}^l \rightarrow \{0,1\}^n$ $n >> l$
- called secure if for any efficient statistical test D (Distinguisher) it holds that $|P[D(G(s))=1] - P[D(r)=1]|$ is negligible.
  $s = \{0,1\}^l$ $r = \{0,1\}^n$
- an unpredictable PRG means that a part of k' gives no info about the rest

Semantic Security

An advisory cannot derive meaningful information from ciphertexts.
Test: Distinguish between encryptions of two chosen plaintexts.

If the advisory can deduce sensitive information or even the PK, then this is called a chosen plaintext attack (CPA)

Public Key Crypto

$$\overset{\text{Alice}}{\text{Plaintext}} \longrightarrow \overset{\text{Bob PK}}{\text{Encryption}} \longrightarrow \overset{\text{Bob SK}}{\text{Decryption}} \longrightarrow \text{Plaintext}$$

$KeyGen(\lambda) = (pk, sk)$
$Enc(pk, m) = c$
$Dec(sk, m)$

Public Key Infrastructure

- Certificate authorities (CA) issues certificates that Alice's PK is valid. This creates a single source of truth and single point of failure.
- Distributed CAs can all issue, but may or may not trust each other
- Web of trust means everyone can issue certs and there is a chain of trust

Textbook RSA

1) Distinct primes p, q: Compute $N = pq$, $\phi(N) = (p-1)(q-1)$ amount of numbers relatively prime
2) Choose $e \in \mathbb{Z}_{\phi(N)}$ such that $gcd(e, \phi(N)) = 1$: Compute $d = e^{-1} \bmod \phi(N)$
3) $pk = (N, e)$, $sk = (N, d)$

   $Enc(pk, m) = m^e \bmod N$ $\quad$ $Dec(sk, c) = c^d \bmod N$

Finding N is a NP problem called the factoring problem, but not semantically secure

Extended Euclidean

$ax + by = gcd(a, b)$
$r = a \bmod b$
$a \leftarrow b$
$b \leftarrow r$
stop if $r = 0$

Modular Inversion

$a x \equiv 1 \bmod m$
exists only if $gcd(a, m) = 1$
EE on $ax + my = 1$
Reduce $x \bmod m$

Chinese Remainder Theorem

$x \equiv a_1 \bmod n_1$
$x \equiv a_2 \bmod n_2$
where n are pairwise coprime ($gcd(n_i, n_j) = 1$)

$N = n_1 \cdot \ldots \cdot n_k$
$N_i = \frac{N}{n_i}$

Find $N_i \cdot M_i \equiv 1 \bmod n_i$
$x = \sum_{i=1}^{k} a_i \cdot N_i \cdot M_i \bmod N$

Security Concepts

IND-CPA: Indistinguishable chosen plaintext attack

→ Test: Can attacker distinguish the ciphertexts of 2 chosen plaintexts

IND-CCA: Indistinguishable chosen ciphertext attack

→ Test: Can attacker distinguish the ciphertexts of 2 chosen plaintexts with access to decryption algo (except for the 2 ciphertexts)

Passing the tests requires success prabability > 50%

RSA fails both. El Gamal is secure against IND-CPA.

Homomorphism

$f(a) + f(b) = f(a+b)$ $\quad f(a) \cdot f(b) = f(a \cdot b)$

RSA and El Gamal are homomorphic:

$E(pk, m_1) \cdot E(pk, m_2) = E(pk, m_1 \cdot m_2)$

El Gamal

1. Generate description of cyclic group $G = \langle g \rangle$ of order q
2. Choose $x \in \{1, \dots, q-1\}$ and compute $h = g^x$
3. $pk = (G, g, q, h)$ $\quad sk = (x)$

Enc (pk, m):

1. Pick random $r \in \{1, \dots, q-1\}$, compute $c_1 = g^r$
2. Compute $c_2 = m \cdot h^r$ $\rightarrow c = (c_1, c_2)$

Dec (sk, c):

1. $k = c_1^x$
2. $m = c_2 k^{-1} = c_2 \cdot c_1^{-x}$

Relies on the discrete log problem:

1. Given h and g, it is infeasable to compute x
2. The shared secret $k = c_1^x \bmod p$ remains secure

Data Integrity

Confidentiality (i.e. encryption) does not imply integrity.
The adversary doesn't have to break the cipher to modify the message.

Cryptographic Hash Functions

Maps arbitrary long inputs into fixed size bit strings. A small change in input should yield significantly different output.

These functions are one-way, meaning they are easy to calculate, but hard to reverse.

Problem: $H: M \to T$, but $\|M\| \gg \|T\|$, so collisions exist:
$H(m_0) = H(m_1)$ $m_0 \neq m_1$. If that's not true, then H is a strong hash.
H is collision resistant if there is no efficient algorithm to find collisions. If that's the case then h is a weak hash.
If we take $2^{n/2}$ inputs and compute $t_i = H(m_i) \in \{0,1\}^n$, we have a 50% chance of a collision (similar to birthday paradox).

Diffie - Hellman

Secure way to exchange cryptographic keys.
Public parameters:

$\left.\begin{array}{l} - \ p: \text{prime} \\ - \ g: \text{generator} \end{array}\right\}$ cyclic multiplicative group G

Private params:

- $a, b \in \{1, \dots, q-1\}$

Public keys:

- $A = g^a \bmod p$
- $B = g^b \bmod p$

Shared key:

- $K = B^a \bmod p \iff K = A^b \bmod p$

→ Again making use of Dlog problem ⇒ computationally infeasable
→ Also works for 3 users, but not more

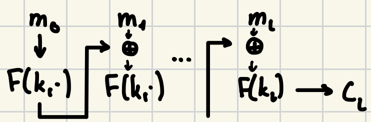
Message Authentication Code (MAC)

Create a tag to ensure message integrity and authentication.
A MAC is a triple of efficient algorithms: KeyGen, MAC, Verify:
- $KeyGen(\lambda) \to k$: λ = security parameter, k = secret key
- $MAC(k, m) \to t$: tag generation
- $Verify(k, m, t) \to \{0, 1\}$: deterministic

It has a correctness property if $Verify(k, m, MAC(k, m)) = 1$.
A MAC must not allow for an existential forgery, meaning an attacker can't produce a valid message-tag pair (m', t') for a new message m' without knowing k.

Implementation of MAC

- Raw CBC-MAC: $F_{CBC}\ K \times M' \to M$, where $M' = \{0,1\}^{l n}$. Message m is split into l blocks of length n.
  $C_0 = 0^n \to C_i = F_k(m_i \oplus C_{i-1})$ $i = 1, \dots, l$ $\to t = C_l$

  However, this is insecure. Given a valid (m, t) pair, the attacker can choose $m' = m \,\|\, (t \oplus m)$:
  $$F_{CBC}(k, m') = F_{CBC}(k, F_{CBC}(k, m) \oplus (t \oplus m)) = F_{CBC}(k, t \oplus (t \oplus m))$$
  $$= F_{CBC}(k, m) = t$$
  Thus, (m', t) is a valid pair.

  This can be fixed by encrypting T with another secret key: $F(k_2, CBC\text{-}MAC_{k_1}(M))$. This is called ECBC-MAC.

- HMAC: $S(k, m) = H(k \oplus opad \,\|\, H(k \oplus ipad) \,\|\, m)$, where opad and ipad are fixed constants used for padding.
  It is proven to be secure

Digital Signatures

$$\begin{array}{ccccc} & m & & \text{Alice PK} & \\ & \downarrow & & \downarrow & \\ \text{Alice SK} \longrightarrow & \text{Sign} & \longrightarrow & \text{Verify} & \rightarrow T/F \end{array}$$

A public-key signature scheme is an efficient algo triplet:

- $\text{KeyGen}(\lambda) \to (pk, sk)$
- $\text{Sign}(sk, m) \to \sigma$
- $\text{Verify}(pk, m, \sigma) \to \{0, 1\}$

Textbook RSA Signature:

1. Generate two λ-bit primes p and q, Compute $N = pq$ and $\Phi(N)$.
2. Choose an integer $e \xleftarrow{R} \mathbb{Z}_{\Phi(N)}$ such that $\gcd(e, \Phi(N)) = 1$ and compute $d = e^{-1} \bmod \Phi(N)$
3. $pk = (N, e)$ and $sk = (N, d)$

   $\text{Sign}(sk, m) = m^d \bmod N = \sigma$

   $\text{Verify}(pk, m, \sigma) = \sigma^e \bmod N \stackrel{?}{=} m$

Correctness: $\text{Verify}(pk, m, \text{Sign}(sk, m)) = 1$

$$\sigma^e \equiv m^{d \cdot e} \bmod N$$

This is again homomorphic: $\text{Sign}(sk, m_1) \cdot \text{Sign}(sk, m_2) = m_1^d \cdot m_2^d \bmod N$
$= (m_1 \cdot m_2)^d \bmod N = \text{Sign}(m_1, m_2)$

This is not secure against existential forgery. We can get the signature of their product without a key. To solve this, we can hash m first:

$$\text{Sign}(sk, H(m_1)) \cdot \text{Sign}(sk, H(m_2)) = H(m_1)^d \cdot H(m_2)^d$$
$$= (H(m_1) \cdot H(m_2))^d \neq (H(m_1 \; m_2))^d$$

This is called hash-and-sign and is secure.

Digital Signature Algorithm (DSA)

$KeyGen(\lambda) \to (pk, sk)$

1. Choose a group G of order q with generator g and a random $x \in \{1, \dots q-1\}$ and compute $X = g^x$.
2. Specify a hash function $H: \{0,1\}^* \to \mathbb{Z}_q$
3. Set $pk = (G, q, g, X)$ and $sk = x$

$Sign(sk, m) \to (R, s)$

Compute $R = (g^r \bmod p) \bmod q$ and $s = \frac{(H(m) + x \cdot R)}{r} \bmod q$

$Verify(pk, m, \sigma)$

$R \stackrel{?}{=} (g^{H(m) \cdot s^{-1}} \cdot X^{R \cdot s^{-1}} \bmod p) \bmod q$

Digital Signatures vs. MACs

Digital Signatures:

- simpler distribution
- only sign once
- publicly verifiable (also transferable!)
- Non-repudiation: Author cannot deny signing it

MACs:

- One key for each recipient
- New MAC for each recipient
- Only receiver can verify
- Author can deny having created a MAC for a message