Perfect Secrecy: $\Pr[M = m \mid C = c] = \Pr[M = m]$, seeing ciphertext doesn't change prior of Adv

Formal: $k \xleftarrow{\$} K, \forall m_0, m_1 \in M$ where $|m_0| = |m_1|, \forall c \in C, \Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$

Shannon's theorem: perfectly secure requires $|K| \geq |M|$, OTP is perfectly secure

Security, but not **integrity** (change contents of msg w/o detection) or **authenticity** (forge msg)

TLS satisfies all 3 subgoals: (1) all algorithms are public (Kerckhoff's principle), (2) security only determined by key size, (3) if you roll your own, it'll be insecure/it's a public effort

/dev/random: can find seed from observing sequence of outputs → predictable

Also avoid **/dev/urandom**; use **getrandom syscall**

Pseudorandom Generator $G: \{0,1\}^s \to \{0,1\}^n$, seed size $s \ll n$, then $E'(k, m) = G(k) \oplus m$

G is **predictable** if $\exists i \in [1, n)$ s.t. $\left|\Pr[\text{Adv}(G_k|_{1\cdots i}) = G_k|_{i+1}] - \frac{1}{2}\right| > \varepsilon$

Cipher is **functionally correct** if $\forall k \in K, m \in M,\ D\big(k, E(k, m)\big) = m$

For $k \xleftarrow{\$} K$, **secure PRF** $F: K \times X \to Y$ induces random-looking $f(x) = F(k, x) \in \text{Func}[X, Y]$

PRP $E: K \times X \to X$ requires E to be bijective, $D = E^{-1}$ efficient, #permutations = $|X|! \approx \left(\frac{|X|}{e}\right)^{|X|}$

CHALLENGER	ADVERSARY	
$b \xleftarrow{\$} \{0,1\}$ If $b = 0, k \xleftarrow{\$} K, y \leftarrow E\|_k, y(x) = E(k, x)$ If $b = 1, y \xleftarrow{\$} \text{Perm}[X]$ or $\text{Func}[X, Y]$	For i in $\{0, \dots, N\}$: Send $x_i \in X$ Receive y_i Output $b' \in \{0,1\}$	$\text{Adv}^{\text{BG}} = \left\|\Pr[b = b'] - \frac{1}{2}\right\|$ Adv^{2W} $= \|\Pr[b' = 1 \mid b = 0]$ $- \Pr[b' = 1 \mid b = 1]$

Birthday attack: adversary can distinguish between a random function $X \to X$ from a PRP on X after Q queries with probability $\leq Q^2/2|X| \implies$ PRP E is secure if $|X| \geq Q^2/2$

Block ciphers: PRF $E: \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}, M \in (\{0,1\}^{128})^{\leq L} = (m[0], \dots, m[L-1])$

Electronic Code Book: $c \in C = (\{0,1\}^{128})^L, k \xleftarrow{\$} \{0,1\}^{128}, c[i] \leftarrow E(k, m[i]), i \in [0, L)$

World 0 CHALLENGER	World 1 CHALLENGER	ADVERSARY
$k \xleftarrow{\$} \{0,1\}^{128}$ Receive (m_0, m_1), send $E(k, m_0)$	$k \xleftarrow{\$} \{0,1\}^{128}$ Receive (m_0, m_1), send $E(k, m_1)$	Send (m_0, m_1) Receive c, output b'

Theorem: ECB is broken – say $L = 2$, then $m_0 = (x, x)$ and $m_1 = (x, y)$. $c = (c[0], c[1])$.
In world 0, $c = (E(k, x), E(k, x))$, and in world 1, $c = (E(k, x), E(k, y))$, Adv = 1.

IND-CPA (chosen plaintext): any **deterministic and stateless block cipher is insecure**

Semantic security (computational secrecy): negligible info. about m can be extracted from c

CHALLENGER	ADVERSARY
$b \xleftarrow{\$} \{0,1\}, k \xleftarrow{\$} \{0,1\}^{128}$ While receive (m_0, m_1): // L blocks long If $b = 0$, send $E(k, m_0)$, else, send $E(k, m_1)$	For i in $\{0, \dots, N\}$: Send (m_0, m_1), receive c_i Output $b' \in \{0,1\}$

Attack: select m_0, m_1, m_2 distinct, send (m_0, m_1), (m_0, m_2), if $c_0 = c_1$, guess $b' = 0$, Adv = 1

Vulnerability: if compression before encryption, choose messages with different length of repeats

Nonce: number used once, public; only (nonce, key) pair needs to be unique → fix k, vary nonce

- Stateful: beware of concurrency, 2 queries must have difference nonce, use atomic counter
- Random: beware of birthday paradox → randomness failures; reusing nonce = reusing key

Block counter mode with nonce with secure PRP is IND-CPA secure, e.g. counter (CTR) mode

$\text{Enc}(k \in \{0,1\}^{128}, m \in (\{0,1\}^{128})^L)$: $n \xleftarrow{\$} \{0,1\}^{96}$ For $m[i]$ in m: $nc \leftarrow n\|\|i$, i is a 32-bit integer $c[i] = m[i] \oplus E(k, nc)$ // or PRF Return (n, c)	$\text{Dec}\big(k, (n \in \{0,1\}^{96}, c \in (\{0,1\}^{128})^L)\big)$: For $c[i]$ in c: $nc \leftarrow n\|\|i$ $m[i] \leftarrow c[i] \oplus E(k, nc)$ // inv not used Return m **not IND-CCA secure*

	Symmetric (share secret k)	Asymmetric
Privacy	Stream/Block cipher	Public key encryption
Integrity, Authenticity	Message Authentication Code (MAC)	Digital signature scheme

Cipher Block Chaining: encryption is slow due to dependencies/forced to be sequential

```
for b in range(numblocks - 1, 0, -1):
  for i in range(15, -1, -1):
    corr_pad = 16 - i
    for pad in range(1, 256):
      newprev[i] = pad
      for j in range(i+1, 16):
        newprev[j] = found[j] ^ corr_pad
      send: blocks[:b-2] + newprev + currblock
      if (accepted without pad_error):
        dec[i] = pad ^ corr_pad
        message[16*(b-1)+i] = dec[i] ^ prevblock[i] // break
```

$M[b] = C[b-1] \oplus \text{Dec}(C[i])$
$C'[b-1] \oplus \text{Dec}(C[i]) = \text{corr\_pad}$

IND-CCA (ciphertext chaining): allowed to query ciphertext whose message was unseen

CHALLENGER	ADVERSARY
$b \xleftarrow{\$} \{0,1\}, k \xleftarrow{\$} K$ Loop: Receive (m_0, m_1), send $E(k, m_b)$ Receive c^*, send $D(k, c^*)$	For i in $\{0, \dots, N\}$: // poly- log K Send (m_0, m_1), receive c_i Send c_i^*, receive m_i^* Output $b' \in \{0,1\}$

$$\text{Adv}^{\text{BG}}_{\text{CCA}} = \left|\Pr[\text{win}] - \frac{1}{2}\right|, \text{win} \Leftrightarrow (b = b') \wedge (\forall i, c_i^* \notin \{c_i\})$$

MAC for EUF (existential unforgeability): require key to create/verify tag, integrity over message

CHALLENGER	ADVERSARY
$k \leftarrow$ keygen() While receive m_i: send createTag(k, m_i) Receive (m^*, t^*) Output VerifyTag$(k, m^*, t^*) \wedge m^* \notin \{m_i\}$	For i in $\{0, \dots, N\}$: send m_i, receive t_i Send (m^*, t^*) $\text{Adv}_{\text{EUF}} = \|\text{Challenger outputs 1}\|$

PRF-MAC: Given $H: \{0,1\}^* \to X$ and $F': K \times X \to Y$, we can construct an EUF-secure MAC with createTag $F: K \times X^{\leq L} \to Y$ via $F(k, m) = F'(k, H(m))$, F is a secure "variable-length" PRF

- Adversary either (1) finds collision in H, or (2) guesses output of F'
- (1) Collision resistance: $\forall A \in \text{PPT}, \Pr[A \text{ outputs } (m_0, m_1) \text{ s.t. } H(m_0) = H(m_1)] < \varepsilon$
  - By birthday, probability of collision after $\sqrt{|X|}$ queries is approx. 1/2
- (2) (1 ⇒ 2) 2nd pre-image resistance: given x, hard to find y such that $H(x) = H(y)$
- (3) Pre-image resistance: given hash output h, hard to find x such that $H(x) = h$
- (4) Random oracle: public, indistinguishable from random function, $H: \{0,1\}^* \to X$

CRHF-MAC (collision-resistant hash function): createTag$(k, m) = H(k||m)$ with compression function $H: \{0,1\}^b \times \{0,1\}^n \to \{0,1\}^n$ (e.g. Merkle–Damgård) is vulnerable to length extension

- Can construct $H(m||m'')$ from $H(m)$ without knowing $m \to \left(k||m||m', H\left(k||m||m'\right)\right)$
- Solution: throw away bits (z = SHA-512, output $z[1 \cdots 384]$), use SHA-3/Blake (build to be resistant to length-extension/sponges), always use HMAC$(k \in \{0,1\}^{256}, m \in \{0,1\}^*)$
  - tmp $\leftarrow H(k \oplus \text{IPAD}||m)$, returns $H(k \oplus \text{OPAD}||\text{tmp})$, where I/OPAD are public

Ciphertext Integrity (CI): adversary cannot produce a valid ciphertext

CHALLENGER	ADVERSARY
$k \leftarrow$ keygen() While receive m_i: send $E(k, m_i)$ Receive c^* Output $(c^* \notin \{c_i\}) \wedge D(k, c^*) \neq$ ERROR	For i in $\{0, \dots, N\}$: send m_i, receive c_i Send $c^* \notin \{c_i\}$ $\text{Adv}_{\text{CI}} = \lvert\text{Challenger outputs 1}\rvert$

Authenticated Encryption: IND-CPA security + CI, **AE ⟹ IND-CCA**

- Encrypt-then-MAC: $m \to c = E(k_E, m) \to c||S(k_I, c)$ is always correct
- MAC-then-Encrypt, $E\left(k_E, m||S(k_I, m)\right)$, Encrypt-and-MAC, $E(k_E, m)||S(k_I, m)$ may be unsafe

Key-Derivation Function: KDF : $\{0,1\}^* \to \{0,1\}^s$ such that $h_g\left(\text{KDF}(M)\right) \approx \min\left(s, h_g(M)\right)$

- $h_g(M) = -\log(\Pr[A \text{ guess } M])$: higher $h_g \implies$ harder to guess
- If $h_g(M) > s$, KDF(m) uniformly random
- Build from HMAC(k, M), either $k = 0^s$ (safe default), or public or secret $k \leftarrow \{0,1\}^s$

Public-Key Cryptography: EUF-CMA (chosen message attack) for signatures

CHALLENGER	ADVERSARY
$(pk, sk) \leftarrow$ KeyGen, send pk While receive m_i: send sign(sk, m_i) Receive (m^*, σ^*) Output $(m^* \notin \{m_i\}) \wedge$ verify(pk, m^*, σ^*)	Receive pk For i in $\{0, \dots, N\}$: Send m_i, receive σ_i Send $(m^* \notin \{m_i\}, \sigma^*)$

Malleability: A ***mauls*** σ into σ' if A takes valid (m, σ) and produces valid (m, σ'), $\sigma \neq \sigma'$

- A is EUF-secure, but breaks *strong*-EUF: $\left((m^*, \sigma^*) \notin \{m_i, \sigma_i\}\right) \wedge$ verify(pk, m^*, σ^*) to win

Trapdoor Permutation (TDP): $\forall(pk, sk) \in K, \forall x \in X, \text{invert}\left(sk, \text{permute}(pk, x)\right) = x$

$$\text{Adv}_{\text{TDF}} = \Pr\left[(pk, sk) \leftarrow \text{keyGen}, x \xleftarrow{\$} X, A\left(pk, \text{permute}(pk, x)\right) = x\right]$$

Given a secure TDF, to construct EUF: sign = TDF.invert$(sk, H(m))$, verify = permute$(pk, \sigma) = H(m)$

- Insecure without random oracle H: choose $\sigma \in X$, $m \leftarrow$ permute(pk, σ), send (m, σ)
- With RO, $x \leftarrow$ permute(pk, σ), still need to find m such that $x = H(m)$

RSA TDP: keygen choose $p, q \leftarrow \{\text{all } \ell - \text{bit primes}\}$, $N = pq$, $d \leftarrow e^{-1} \bmod \varphi(N)$, then $ed \equiv_{\varphi(N)} 1$ $\implies ed + k\varphi(N) = 1 \implies x^{ed} \equiv_N x$ since $x^{k\varphi(N)} \equiv_N 1$ by Euler's theorem for $x \perp N$ $\left(e \perp \varphi(N)\right)$

$$\text{Adv}_{\text{RSA}} = \Pr\left[((N, e), (N, d)) \leftarrow \text{keyGen}(\ell, e), x \xleftarrow{\$} \mathbb{Z}_N, y \leftarrow x^e \bmod N, A(N, e, y) = x\right]$$

- To avoid collisions + $k < 0$ when $e, d > 0$: require $\gcd(x, N) = 1 \iff x \in \mathbb{Z}_N^*$
- Assumption: hard to find $x \in \mathbb{Z}_N \backslash \mathbb{Z}_N^*$ ($|\mathbb{Z}_N| \approx |\mathbb{Z}_N^*|$), else N can be factored

Diffie-Hellman Key Exchange: $G = (g)$ of prime order q, $g^q = q^0$, $g^a = g^b \iff a \equiv_q b$

- KeyGen returns $sk \xleftarrow{\$} \mathbb{Z}_q$, $pk \leftarrow g^{sk}$, Exchange$(sk_A, pk_B) = pk_B^{sk_A} =$ Exchange(sk_B, pk_A)
- **Security**: Adversary knows pk^A, pk^B, cannot compute $X_{AB} = g^{sk_A \cdot sk_B}$, DLP ⇏ security
- A, B derive "traffic keys" for encryptions from $A \to B$, $B \to A$ + MAC via $k \leftarrow$ KDF(X_{AB})
  - **Unauthenticated DHKE is vulnerable to man-in-the-middle attacks**!

Discrete Log Problem: given $pk \in G$, hard to find sk such that $g^{sk} = pk$, DLP must be hard in G

Decisional DH game: Adv cannot distinguish X_{AB} from random (easier than DLP)

CHALLENGER	ADVERSARY
$b \xleftarrow{\$} \{0,1\}$, $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q$ If $b = 0$, $h \leftarrow g^{\alpha\beta} \in G$ If $b = 1$, $r \xleftarrow{\$} \mathbb{Z}_q$, $h \leftarrow g^r \in G$ Send $\left(g, g^\alpha, g^\beta, h\right)$	Receive (g, a, b, h) Output $b' \in \{0,1\}$ $\text{Adv}_{\text{DDH}}^{\text{BG}} = \left\lvert\Pr[b = b'] - \frac{1}{2}\right\rvert$

A, B have public identities id_A, id_B and private/stable identities $\text{sid}_A, \text{sid}_B$

- A: $(pk_A, sk_A) \leftarrow$ KeyGen(), $\sigma_A =$ sign(sid_A, pk_A), send (pk_A, σ_A),
- ABORT if verify$(\text{id}_B, pk_B, \sigma_B) \neq 1$, else, $X_{AB} \leftarrow$ Exchange(sk_A, pk_B)
- **Forward secrecy**: if A uses long-term pk, Attacker who steals sid can decrypt all previous convo.; long-term id: even if sid stolen, random sign used, can only impersonate, not encrypt
  - **Safety**: generate fresh DH keys for each session
- Learning id_A: trusted party (Certificate Authority/Trent) signs certificate $(A \text{ is } \text{id}_A)_{\text{id}_T}$
- What if A's key gets stolen? Thief has both secret and certificate: certs have expiry date

TLS Handshake: website operator gets certificate cert_v for address (validation by CA)

ALICE	VSRS.BIZ
$(pk_A, sk_A) \leftarrow$ KeyGen, send pk_A Receive $(pk_v, \sigma_v, \text{cert}_v)$, $\text{id}_v \leftarrow$ verifyCert$(\text{id}_{\text{CA}}, \text{cert}_v)$ Verify$(\text{id}_v, pk_v, \sigma_v)$ or ABORT $k_{\text{main}} \leftarrow$ KDF(Exch$(sk_A, pk_v)||\sigma_v)$	Receive pk_A, $(pk_v, sk_v) \leftarrow$ KeyGen $\sigma_v \leftarrow$ sign(sid_v, pk_v) Send $(pk_v, \sigma_v, \text{cert}_v)$ $k_{\text{main}} \leftarrow$ KDF(Exch$(sk_v, pk_A)||\sigma_v)$

Blockchain: use if you require auditable, tamper-proof database with multiple, untrusted writers and no central authority (or in-band transfer of currency)

- Wallets: represented by addresses (public key); goals: availability, security, convenience

Byzantine General's Problem: consensus not guaranteed if more than 1/3 generals are traitors

- Broadcasting not instantaneous, no global notion of time, some participants malicious
- BUT: our system is high latency and probabilistic (good!)

- New transactions are broadcast to all nodes
- Each node collects new transactions into a block
- In each round a random node gets to broadcast its block
- Other nodes accept block only if all transactions in it are valid (unspent, valid signatures)
- Nodes express acceptance of block by including a hash of it in the next block they create

Adversary def.: can't create/modify transactions from someone else, can't suppress transaction

Double-spending (forking attack): solution- honest nodes always extend longest valid branch

Verification + "proof of work" to (1) pick random node, (2) currency production is rate-limited

- Find x such that $H(x||\text{prev\_hash}||\text{block transactions}) < y$, where H is cryptographic hash
- No better algorithm than brute force (if RO: pick $z < y$, compute $H^{-1}(z)$)

Smart contracts: user-defined program running on top of a blockchain that acts as independent agent; "owner" of shared account. **Invoke** via a transaction (includes arguments (and money))
Requires: (1) availability (SC should have an address), (2) transparency (A, B should be able to see the code), (3) consistency (SC is not altered), (4) enforcement (code is correctly executed)

- Eth: miners execute SC- user defines gas price (Eth/gas), gas cost depends on CPU cycles
- Re-entrancy vulnerability: sends funds before updating balance