CSCI 1515

$\{0,1\}^n$: set of n-bit binary strings, $\{0,1\}^*$ set of all (finite-length) binary strings. when x, y are strings of same length s.t $|x|=|y|$, $x \oplus y$ (bitwise exclusive-or), $x \oplus x = 000\ldots$; $x \oplus 000\ldots = x$; $x \oplus 111\ldots = \bar{x}$ (all bits flipped); $x \oplus y = y \oplus x$; $(x \oplus y) \oplus z = x \oplus (y \oplus z)$; $a = b \oplus c \iff b = a \oplus c \iff c = a \oplus b$

$\therefore$ Keys $\in \{0,1\}^\lambda$

. One-time-Pad (OTP): secret keys are strings of bytes (λ length)

$Enc_k(m_1) = m_1 \oplus k$, $\oplus = m_1 \oplus m_2$

key cannot be reused: $Enc_k(m_2) = m_2 \oplus k$

KeyGen: $k \leftarrow \{0,1\}^\lambda$, return k

$Enc(k, m \in \{0,1\}^*)$: return $k \oplus m$

$Dec(k, c \in \{0,1\}^*)$: return $k \oplus c$

($k \leftarrow \{0,1\}^\lambda$) sample k uniformly from the set of λ-bit strings

Correctness: $\forall k, m \in \{0,1\}^\lambda$, $Dec(k, Enc(k,m)) = m$: $Dec(k, enc(k,m)) = Dec(k, k \oplus m) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0^\lambda \oplus m = m$.

Eavesdrop($m \in \{0,1\}^*$): $k \leftarrow \{0,1\}^\lambda$, $c := k \oplus m$, return c. (Attacker receives output from this algorithm). Ex: $\lambda = 3$, $k \leftarrow \{0,1\}^3$

Eavesdrop(010):

Pr	k	c = k⊕010
1/8	000	010
1/8	001	011
1/8	010	000
1/8	011	001
1/8	100	110
1/8	101	111
1/8	110	100
1/8	111	101

Claim: $\forall m \in \{0,1\}^\lambda$ the distribution EAVESDROP(m) is the uniform on $\{0,1\}^\lambda$ $\therefore$ $\forall m, m' \in \{0,1\}^\lambda$, EAVESDROP(m) = EAVESDROP(m')

Proof: Arbitrarily fix $m, c \in \{0,1\}^\lambda$. Calculate pr. that E(m) produces c. That event happens only when $c = k \oplus m \iff k = m \oplus c$

$P[E(m)=c] = P[k = m \oplus c]$, where pr. is over uniform choice of $k \leftarrow \{0,1\}^\lambda$. $\exists!$ k s.t $k = m \oplus c$. Since k is chosen uniformly from $\{0,1\}^\lambda$, pr. choosing the particular value $k = m \oplus c$ is $1/2^\lambda$. $\therefore$ $\forall m, c$, the pr. that E(m) outputs c is $1/2^\lambda$. $\therefore$ Output of E(m), $\forall m$, follows the uni. distribution

Shannon Theorem: Under certain conditions keyGen must choose the key uniformly from the set of all possible keys; $\forall$ m, $\exists!$ key mapping $m \to c$

The theorem assumes $|M| = |K| = |C|$. Let (Gen, Enc, Dec) be an encryption scheme with message space M, $|M| = |K| = |C|$. The scheme is perfectly secret iff: (i) $\forall k \in K$ chosen with pr. $1/|K|$ by Gen; (ii) $\forall m \in M, c \in C$, $\exists! k \in K$ s.t. $Enc_k(m) \mapsto c$.

Perfect Secrecy: Assume the eavesdropper knows the probability d. of M. For a scheme to be perfectly secret, observing the ciphertext should have no effect on the adversary's knowledge regarding m. Def: An encryption scheme (Gen, Enc, Dec) with message space M is perfectly secret if $\forall$ pr. distribution for M, every $m \in M$, and every ciphertext $c \in C$ for which $P(C=c) > 0$: $P[M=m \mid C=c] = P[M=m]$

A scheme is (t, ε)-secure if $\forall$ adversary running for time at most t, succeeds in breaking the scheme with P at most ε.

The RSA problem: Given a positive integer n that is a product of two distinct primes p, q., a positive integer e s.t $\gcd(e, (p-1)(q-1)) = 1$ and an integer c, find an integer m s.t. $m^e \equiv c \pmod n$. This problem is about finding e^{th} roots modulo a composite integer n. The conditions imposed on the problem parameters n, e, ensure that for each integer $c \in \{0,1,\ldots,n-1\}$, $\exists!$ $m \in \{0,1\cdots,n-1\}$ s.t $m^e \equiv c \pmod n$. Equivalently, the function $f: \mathbb{Z}_n \to \mathbb{Z}_n$, $f := f(m) = m^e \bmod n$ is a permutation

(key can be public or secret.)

Symmetric-key-enc. $\pi = (Gen, Enc, Dec)$ $k \leftarrow Gen$, $c \leftarrow Enc_k(m)$, $m = Dec_k(c)$

Public-key-enc. $(pk, sk) \leftarrow Gen$, $c \leftarrow Enc_{pk}(m)$, $m = Dec_{sk}(c)$

Post-Quantum-Security. Computational Security. Polynomial-time Algorithm: $A(x)$, $\forall x$ of length n, running time $O(n^c)$ for some constant c. $\forall$ probabilistic poly-time (PPT) A, $Enc(m_0) \overset{c}{\approx} Enc(m_1)$. Security Parameter: $k \leftarrow Gen(1^\lambda)$, λ: sec. parameter

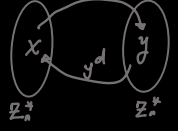
(i) Adversary runs in time $p(\lambda)$; negligible(λ) $< \frac{1}{\lambda^c}$ $\forall$ constant c. Ex: $\lambda = 128$, Best A to break the scheme takes time $\sim 2^\lambda$

2^{128} CPU cycles.

$2^{(a)}\ 19426^{(b)} \bmod 392643$ $\quad b = 2^0 + 2^1 + 2^7 + \ldots$ (n terms) $\quad \therefore a^b = a^{\cdot} \cdot a^{2^{\cdot}} \cdot a^{2^{7}} \cdot \ldots$ How to compute $a^b \bmod N$ for a, b, N of n bits $\longrightarrow O(n)$

Time complexity of gcd(a,b) for a, b of n bits: O(n). if $\gcd(a,N) = 1$, a & N are coprime $\implies \exists b$ s.t $a \cdot b \equiv 1 \pmod N$: a is invertible modulo N, b is its inverse denoted as a^{-1}. $\mathbb{Z}_n^* := \{a \mid a \in [1, N-1], \gcd(a,N) = 1\}$. Euler's phi function $\phi(N) := |\mathbb{Z}_n^*|$

Euler's theorem: $\forall a, N$ where $\gcd(a,N) = 1$, $a^{\phi(N)} \equiv 1 \pmod N$. If $d = e^{-1} \bmod \phi(N)$ then $(a^d)^e \equiv a \bmod N$



Given a modulus N, and an integer $e > 2$ relatively prime to $\phi(N)$, the exponentiation to the eth power modulo N is a permutation. $\therefore$ We can define $[y^{1/e} \bmod N]$ $\forall y \in \mathbb{Z}_N^*$ as the unique element of $\mathbb{Z}_N^*$ that yields y when raised to the e-th power modulo N. $\therefore$ $x = y^{1/e} \bmod N$ iff $x^e = y \bmod N$. The RSA problem is to compute $[y^{1/e} \bmod N]$ for a modulus N of unknown factorization. Let GenRSA be a probabilistic polynomial-time algorithm that, on input 1^n, outputs a modulus N that is, the product of 2 n-bit primes, as well as integers e, $d > 0$ with $\gcd(e, \phi(N)) = 1$ and $ed = 1 \bmod \phi(N)$

Plain RSA encryption: Let GenRSA be a PPT algorithm that on input 1^n RSA key generation. (i) Input: security Parameter 1^n; (ii) Output: N, e, d $(N, p, q) \leftarrow GenModulus(1^n)$, $\phi(N) := (p-1)\cdot(q-1)$, (iii) choose $e > 1$ s.t. $gcd(e, \phi(N)) = 1$; (iv) Compute $d := [e^{-1} \bmod \phi(N)]$; (v) return N, e, d

Let N, e, d be as above, let $c = m^e \bmod N$ for some $m \in \mathbb{Z}_N^*$. RSA encryption relies on the fact that someone who knows d can recover m from c by computing $[c^d \bmod N]$: $c^d = (m^e)^d = m^{ed} = m \bmod N$.

(i) Receiver runs GenRSA ($GenRSA(1^n)$) to obtain N, e, d; (ii) publishes N, e as public key ($\langle N, e \rangle$), keeps d (private key, $\langle N, d \rangle$); (iii) Enc: To encrypt a message $m \in \mathbb{Z}_N^*$ a sender computes $c := [m^e \bmod N]$ ($pk = \langle N, e \rangle$)

(iv) Dec: private key ($sk = \langle N, d \rangle$), and ciphertext $c \in \mathbb{Z}_N^*$: $m := [c^d \bmod N]$

Ex: $GenRSA(N, e, d) = (391, 3, 235)$, $391 = 17 \cdot 23$. $\therefore \phi(391) = 16 \cdot 22 = 352$. Moreover $3 \cdot 235 = 1 \bmod (352)$, $pk = \langle 391, 3 \rangle$, $sk = \langle 391, 235 \rangle$

To encrypt $m = 158 \in \mathbb{Z}_{391}^*$ using $pk = \langle 391, 3 \rangle$ $m \longrightarrow c := [158^3 \bmod 391] = 295$. (ciphert.), to decrypt: $m = [295^{235} \bmod 391] = 158$

RSA assumption implies that, if $m \sim Uniform\{\mathbb{Z}_N^*\}$, then an eavesdropper given N, e, c, cannot recover m in its entirety.

Since RSA is deterministic, if an attacker knows that $m \in B$, then the attacker can determine m from $c = [m^e \bmod N]$ in time $O(B)$

If the message is a uniform n-bit key and so $B = 2^n$, $\exists$ attack that recovers m in $O(\sqrt{B})$

ALGORITHM 12.28
An attack on plain RSA encryption

Input: Public key $\langle N, e \rangle$; ciphertext c; bound 2^n
Output: $m < 2^n$ such that $m^e = c \bmod N$

set $T := 2^{\alpha n}$
for $r = 1$ to T:
  $x_r := [c/r^e \bmod N]$
sort the pairs $\{(r, x_r)\}_{r=1}^T$ by their second component
for $s = 1$ to T:
  if $[s^e \bmod N] \stackrel{?}{=} x_r$ for some r
    return $[r \cdot s \bmod N]$

CONSTRUCTION 12.30

Let GenRSA be as before, and let ℓ be a function with $\ell(n) < 2n$. Define a public-key encryption scheme as follows:

- Gen: on input 1^n, run $\mathsf{GenRSA}(1^n)$ to obtain (N, e, d). Output the public key $pk = \langle N, e \rangle$, and the private key $sk = \langle N, d \rangle$.
- Enc: on input a public key $pk = \langle N, e \rangle$ and a message $m \in \{0,1\}^{\|N\| - \ell(n) - 1}$, choose a uniform string $r \in \{0,1\}^{\ell(n)}$ and interpret $\hat{m} := r \| m$ as an element of $\mathbb{Z}_N$. Output the ciphertext
$$c := [\hat{m}^e \bmod N].$$
- Dec: on input a private key $sk = \langle N, d \rangle$ and a ciphertext $c \in \mathbb{Z}_N^*$, compute
$$\hat{m} := [c^d \bmod N],$$
and output the $\|N\| - \ell(n) - 1$ least-significant bits of $\hat{m}$.

The padded RSA encryption scheme.

8.18 Algorithm ElGamal public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key (p, α, α^a).
   (b) Represent the message as an integer m in the range $\{0, 1, \ldots, p-1\}$.
   (c) Select a random integer k, $1 \le k \le p-2$.
   (d) Compute $\gamma = \alpha^k \bmod p$ and $\delta = m \cdot (\alpha^a)^k \bmod p$.
   (e) Send the ciphertext $c = (\gamma, \delta)$ to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use the private key a to compute $\gamma^{p-1-a} \bmod p$ (note: $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$).
   (b) Recover m by computing $(\gamma^{-a}) \cdot \delta \bmod p$.

Proof that decryption works. The decryption of Algorithm 8.18 allows recovery of original plaintext because
$$\gamma^{-a} \cdot \delta \equiv \alpha^{-ak} m \alpha^{ak} \equiv m \pmod p.$$

8.19 Example (*ElGamal encryption with artificially small parameters*)
Key generation. Entity A selects the prime $p = 2357$ and a generator $\alpha = 2$ of $\mathbb{Z}_{2357}^*$. A chooses the private key $a = 1751$ and computes
$$\alpha^a \bmod p = 2^{1751} \bmod 2357 = 1185.$$
A's public key is $(p = 2357, \alpha = 2, \alpha^a = 1185)$.
Encryption. To encrypt a message $m = 2035$, B selects a random integer $k = 1520$ and computes
$$\gamma = 2^{1520} \bmod 2357 = 1430$$
and
$$\delta = 2035 \cdot 1185^{1520} \bmod 2357 = 697.$$
B sends $\gamma = 1430$ and $\delta = 697$ to A.
Decryption. To decrypt, A computes
$$\gamma^{p-1-a} = 1430^{605} \bmod 2357 = 872,$$
and recovers m by computing
$$m = 872 \cdot 697 \bmod 2357 = 2035. \quad \square$$

Theorem: Let $p(x)$ be a polynomial of degree e. Then in time $poly(\|N\|, e)$ one can find all m s.t. $p(m) = 0 \bmod N$ and $|m| \le N^{1/e}$

Diffie-Hellman problem (DHP): given a prime p, a generator α of $\mathbb{Z}_p^*$, and elements $\alpha^a \bmod p$ and $\alpha^b \bmod p$, find $\alpha^{ab} \bmod p$.

(GDHP): given a finite cyclic group G, a generator α of G, and group elements α^a, α^b, find α^{ab}

Cyclic groups and generators: Let G be a finite group of order m. For arbitrary $g \in G$ consider the set $\langle g \rangle \stackrel{def}{=} \{g^0, g^1, \ldots\}$.

Corollary: If G is a group of prime order p, then G is cyclic. Furthermore, all elements of G except the identity are generators of G

Chosen-Plaintext-Attack (CPA) security: non-deterministic encryption algorithm

$(pk, sk) \leftarrow Gen(1^n)$

$\phi(N) = |\mathbb{Z}_N^*|$, Ex: $N = pq$ (p, q primes) $\phi(N) = (p-1)(q-1)$

Factoring Assumption: Generate two n-bit primes $p \cdot q$, $p \neq q$, let $N = pq$. Given N, it is computationally hard to find p & q. $O(2^n)$

$\phi(N) = (p-1)(q-1)$, choose e s.t. $gcd(e, \phi(N)) = 1$. Compute $d = e^{-1} \bmod \phi(N)$

Given (N, e) & random $y \leftarrow \mathbb{Z}_N^*$, it's comp. hard to find x s.t. $x^e \equiv y \pmod N$

RSA cannot be CPA secure because it is deterministic

Def: a Group is a set G along with a binary operation $\circ$ with properties

(i) closure: $\forall g, h \in G$, $g \circ h \in G$

(ii) $\exists!$ identity: $\exists e \in G$ s.t. $\forall g \in G$ $e \circ g = g \circ e = g$

(iii) $\forall g \in G$, $\exists h \in G$ s.t. $g \circ h = h \circ g = e$

(iv) Associativity: $\forall g_1, g_2, g_3 \in G$ $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$

if $\forall g, h, \in G$, $g \circ h = h \circ g$ G is abelian

$g^m = g \circ g \circ g \ldots$, $g^0 = 1 \leftarrow e$; $g^{-m} := (g^{-1})^m$, $g^{m_1} \cdot g^{m_2} = g^{m_1 + m_2}$, $(g^{m_1})^{m_2} = g^{m_1 m_2}$. Def: for a finite group of order q ($|G|$). (g elements) $\forall g \in G$, $\langle g \rangle := \{g^0, g^1, \ldots, g^{q-1}\}$, G is a cyclic groups if $\exists g \in G$ s.t $\langle g \rangle = G$, g is a generator of G

Ex: $\mathbb{Z}_p^*$ for p prime is a cyclic group of order $p-1$. Ex: $p = 7$, $\langle 3 \rangle = \{1, 3, 2, 6, 4, 5\}$ $\therefore$ 3 is a generator.

$(G, q, g) \leftarrow G(1^\lambda)$ Cyclic group G of order q with generator g. **Discrete log assumption (DLOG):** $x \leftarrow \mathbb{Z}_q$, compute $h = g^x$. Given (G, q, g, h) it's comp. hard to find x.

Computational Diffie-Hellman: $x, y \leftarrow \mathbb{Z}_q$, compute $h_1 = g^x$, $h_2 = g^y$. Given (G, q, g, h_1, h_2), it's computationally hard to find g^{xy}.

Decisional Diffie-Hellman: $x, y, z \leftarrow \mathbb{Z}_q$, compute $h_1 = g^x$, $h_2 = g^y$. Given (G, q, g, h_1, h_2) it's comp. hard to distinguish $(g^x, g^y, g^{xy}) \stackrel{c}{=} (g^x, g^y, g^z)$.

ElGamal Encryption: $Gen(1^\lambda)$ (can be reused): $(G, q, g) \leftarrow G(1^\lambda)$, $x \leftarrow \mathbb{Z}_q$, compute $h = g^x$, $pk = (G, g, q, h)$, $sk = x$

$Enc_{pk}(m)$: $m \in G$, $y \leftarrow \mathbb{Z}_q$, $c = \langle g^y, h^y \cdot m \rangle$. $Dec_{sk}(c)$: $c = \langle c_1, c_2 \rangle$ $m = \frac{c_2}{c_1^x} = \frac{g^{xy} \cdot m}{g^{xy}} = m$

Theorem: It is impossible to construct secure key exchange from SKE in a black box way.

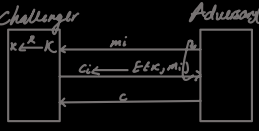
Diffie-Hellmen key exchange: Alice $(G, q, g) \leftarrow G(1^\lambda)$, $x \leftarrow \mathbb{Z}_q$, computes $h_A = g^x$ → (G, g, q, h_A) → Bob $y \leftarrow \mathbb{Z}_q$, computes $h_B = g^y$ → h_B → Alice. $K = h_B^x$ $K = h_A^y$

Message Authentication Code (MAC) scheme: $\Pi = (Gen, Mac, Vrfy)$, $k \leftarrow Gen(1^\lambda)$, $t = Mac_k(m)$, $0/1 := Vrfy_k(m, t)$

Digital Signature Scheme $\Pi = (Gen, Sign, Vrfy)$: $(pk, sk) \leftarrow Gen(1^\lambda)$, $\sigma \leftarrow Sign_{sk}(m)$; $0/1 := Vrfy_{pk}(m, \sigma)$

Authenticated Encryption: $(\mathcal{E})$ (i) $\mathcal{E}$ must be CPA secure; (ii) $\mathcal{E}$ must provide ciphertext integrity. **Ciphertext Integrity:**

Let $\mathcal{E} = (E, D)$ be a cipher defined over (K, M, C) and adversary A. (i) Challenger chooses random $k \xleftarrow{R} K$



(ii) A queries challenger several times. For $i = 1, 2, \ldots$ the i-th query consists of a message $m_i \in M$. Challenger computes $c_i \xleftarrow{R} E(k, m_i)$ and gives c_i to A; (iii) Eventually, A outputs a candidate ciphertext $c \in C$ that is not among the ciphertext it was given, i.e. $c \notin \{c_1, c_2, \ldots\}$. A wins if c is a valid ciphertext under k, that is, $D(k, c) \neq$ reject.

A's advantage with respect to $\mathcal{E}$: $CIadv[A, \mathcal{E}]$ (prob. A wins). A is a Q-query adversary if A issues at most Q encryption queries.

$\mathcal{E} = (E, D)$ provides c. integrity if $CI\,adv$ is negligible.

Hash Functions: (i) **Collision Resistance:** It should be hard to compute any collision $x \neq x'$ s.t. $H(x) = H(x')$; **Second-preimage Resistance:** It should be hard computing any collision involving x: hard to compute $x' = x$ s.t. $H(x) = H(x')$

Brute Force Attacks:

A_{CR}: $O(q^2/n^2)$

for $i = 1, \ldots$

$x_i \leftarrow \{0,1\}^m$

$y_i := H(x_i)$

if $\exists j < i$ s.t. $x_i \neq x_j, y_i = y_j$

return (x_i, x_j)

$A_{2pi}(x)$: $O(2^n)$

while True:

$x' \leftarrow \{0,1\}^m$

$y' := H(x')$

If $y' = H(x)$: return x'.

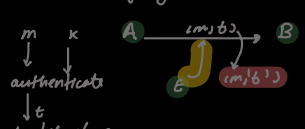
Def: **A digital signature scheme** consists of 3 probabilistic polynomial-time algorithms

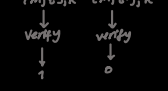
(Gen) (i) key-gen: input security parameter 1^n and outputs pair of keys (pk, sk). s.t. $|pk| > n$.

(Sign) (ii) Signing: input a private key (sk) and message (m) from some message space. Outputs signature σ: $\sigma \leftarrow Sign_{sk}(m)$

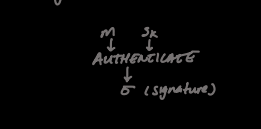
(Ver) (iii) Verification: input public key (pk), message (m), signature (σ). Outputs a bit b. If $b = 1$ (valid) $b = 0$, invalid; $b := Vrfy_{pk}(m, \sigma)$

Message Integrity:





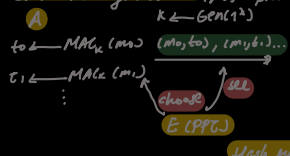
Digital Signature



; Message Auth. Code (MAC) scheme: $\Pi(Gen, Mac, Vrfy)$

$k \leftarrow Gen(1^\lambda)$, $t \leftarrow Mac_k(m)$, $0/1 := Vrfy_k(m, t)$

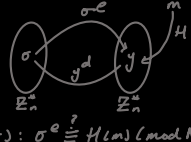
Chosen Message Attack (PPT) (new message) (A): produce (m^*, t^*), $m^* \neq m_i \; \forall i$. $P[Vrfy_k(m^*, t^*) = 1] \leq 2^{-\lambda}$. Strongly Secure: $(m^*, t^*) \neq (m_i, t_i) \; \forall i$



Hash function in RSA Encryption

$H: \{0,1\}^* \longrightarrow \mathbb{Z}_N^*$: public deterministic function.

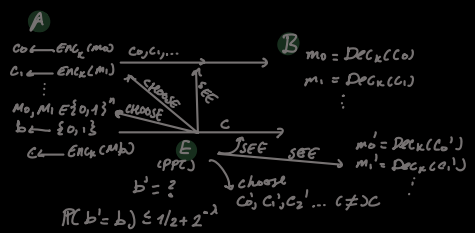
$sk = d$, $pk = \langle N, e \rangle$. $Sign_{sk}(m) = H(m)^d \bmod N$, $Vrfy_{pk}(m, \sigma)$: $\sigma^e \stackrel{?}{=} H(m) \pmod N$



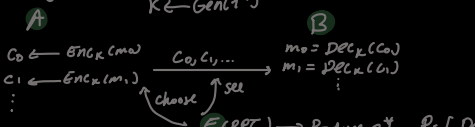
Given (N, e) a random $y \leftarrow \mathbb{Z}_n^*$, it's computationally hard to find x s.t. $x^e \equiv y \bmod (N)$

Authenticated Encryption (AE): CCA-secure & unforgeable. CPA-secure SKE + CMA-secure MAC $\Rightarrow$ AE $\Rightarrow$ CCA-secure & Unforgeable (i) Encrypt-and-Mac; (ii) Mac-then-encrypt; (iii) Encrypt-then-Mac

Chosen Ciphertext Attack (CCA) Security: (secrecy) ; **Unforgeability**



$Pr[b' = b] \leq 1/2 + 2^{-\lambda}$



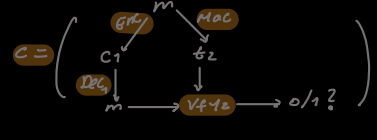
$E^{(PPT)} \Rightarrow$ Produce c^*, $Pr[Dec_K(c^*) = m^b, m^* \neq m_i \ \forall i] \leq 2^{-\lambda}$

Encrypt-and-Mac: Given a CPA-secure SKE scheme $\Pi_1 = (Gen_1, Enc_1, Dec_1)$ & a strongly CMA-secure MAC scheme $\Pi_2 = (Gen_2, Mac_2, Vfy_2)$ construct an AE scheme $\Pi = (Gen, Enc, Dec)$

(i) $Gen(1^\lambda)$: $k_1 \leftarrow Gen_1(1^\lambda)$; $k_2 \leftarrow Gen_2(1^\lambda)$; output $K = (k_1, k_2)$

(ii) $Enc_K(m)$: $c_1 \leftarrow Enc_1(k_1, m)$; $t_2 \leftarrow Mac_2(k_2, m)$; output $= c = (c_1, t_2)$

(iii) $Dec_K(c)$: $c = (c_1, t_2)$; $m := Dec_1(k_1, c_1)$; $b := Vfy_2(k_2, (m, t_2))$; If $b=1$, output m



Encrypt-then-Mac:

(i) $Gen(1^\lambda)$: $k_1 \leftarrow Gen_1(1^\lambda)$; $k_2 \leftarrow Gen_2(1^\lambda)$; output $K = (k_1, k_2)$

(ii) $Enc_K(m)$: $c_1 \leftarrow Enc_1(k_1, m)$; $t_2 \leftarrow Mac_2(k_2, c_1)$; output $c = (c_1, t_2)$

(iii) $Dec_K(c)$: $c = (c_1, t_2)$; $m := Dec_1(k_1, c_1)$; $b := Vfy_2(k_2, (c_1, t_2))$; $b=1 \rightarrow m$.

CCA-secure.

Mac-then-encrypt:

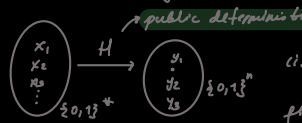
(i) $Gen(1^\lambda)$: $k_1 \leftarrow Gen_1(1^\lambda)$; $k_2 \leftarrow Gen_2(1^\lambda)$; output $K = (k_1, k_2)$

(ii) $Enc_K(m)$: $t_2 \leftarrow Mac_2(k_2, m)$; $c_1 \leftarrow Enc_1(k_1, m \| t_2)$; output $= c = c_1$

(iii) $Dec_K(c)$: $c = c_1$; $m \| t_2 := Dec_1(k_1, c_1)$; $b := Vfy_2(k_2, (m, t_2))$; $b=1$, output m

Hash-Function: $H: \{0,1\}^* \rightarrow \{0,1\}^n$ $\rightarrow \theta(2)$ $\lambda = 128$ $n = 256$

public deterministic



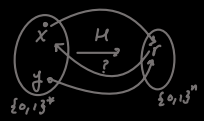
(i) **Collision-Resistant Hash-Function (CRHF):** Computationally hard to find $x, y \in \{0,1\}^*$ s.t. $x \neq y$, $H(x) = H(y)$ (collision)

Probability of collision: **Birthday Paradox.**

Random Oracle Model: $H: \{0,1\}^* \rightarrow \{0,1\}^n$ Deterministic.

$r \xleftarrow{u} \{0,1\}^n$

$H(x) := r$



Hash and sign: Say we have a signature scheme for messages of length l, and wish to sign a (longer) message $m \in \{0,1\}^*$. Rather than signing m, we can use a hash function (H) to hash m to length $l(\lambda)$ and then sign the resulting digest.

Let $\Pi = (Gen, Sign, Vfy)$ be a signature scheme for messages of length $l(n)$, and let $\Pi_H = (Gen_H, H)$ be a hash function with output length $l(n)$. Construct signature scheme $\Pi' = (Gen', Sign', Vfy')$ as follows:

(i) Gen': on input 1^n, run $Gen(1^n)$ to obtain $\langle pk, sk \rangle$ and run $Gen_H(1^n)$ to obtain s; the public key is $\langle pk, s \rangle$ and private key is $\langle sk, s \rangle$.

(ii) $Sign'$: on input a private key $\langle sk, s \rangle$ and a message $m \in \{0,1\}^*$, output $\sigma \leftarrow Sign_{sk}(H^s(m))$

(iii) Vfy': on input a public key $\langle pk, s \rangle$, a message $m \in \{0,1\}^*$, and signature σ, output 1 iff $Vfy_{pk}(H^s(m), \sigma) \stackrel{?}{=} 1$

Constructions for H.F: (i) **(MD5)** output length 128-bit, best-known attack 2^{16}; collision found in 2004

(ii) Secure Hash Functions: **(SHA-0)** $\rightarrow$ 160-bits, best known attack 2^{39}; (iii) **SHA-1**: output l. 160-bits, best known attack 2^{63}, collision found in 2017:

(iv) **SHA-2:** output l. 224, 256, 384, 512-bit; **SHA-3:** o.l. 224, 256, 384, 512-bit. (released in 2015)

Application of Hash-Function:

(i) **HMAC** (Merkle • Damgård)

$K \leftarrow \{0,1\}^\lambda$, SHA-2

$Mac_K(m) = H(K \| m)$, $Mac_K(m) = H(K \| H(K \| m))$, $Hmac_K(m) = H(K_1 \| H(K_2 \| m))$

$Vfy_K(m, t)$: $t = H(K \| m)$

$K_1 = K \oplus OPAD$ $K_2 = K \oplus IPAD$ $\rightarrow$ (Inner) (Outter) pad

(ii) **HASH-AND-SIGN:** $Sign_{sk}(H(m))$, RSA signature: $Sign_{sk}(m) = H(m)^d \bmod N$

PUBLIC ; (iii) **Password-based Authentication:** username, H(password)

(iv) **Deduplication**: $H(D_1) \to h_1$, $H(D_2) \to h_2$ — unique identifier. If $h_1 \neq h_2 \Rightarrow D_1 \neq D_2$ (due to collision-resistance). If $h_1 = h_2 \Rightarrow D_1 = D_2$

(vi) **Virus Scan**: $H(F) \stackrel{?}{=} H(F')$; (vii) **Video Deduplication** $H(V_1) \stackrel{?}{=} H(V_2)$

(HKDF) (KEY DERIVATION FUNCTION): HMAC(g^{ab}, salt): Pseudorandom Generator (PRG): random seed (λ, $\{0,1\}^\lambda$) —G (public, deterministic)→ "looks random" ($poly(\lambda)$, truly random)

SKE Scheme: $K \leftarrow \{0,1\}^\lambda$, $Enc_K(m) = H(K \| m) \to$ Cannot be decrypted.

Pseudorandom Function (PRF): Keyed Function $F: \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$; $F(K, x) \to y$ (key, input, output; (PPT))

$K \leftarrow \{0,1\}^\lambda$ F_K: $\{0,1\}^n \to \{0,1\}^m$ — 2^λ possible F_K's

$f \leftarrow \{F \mid F: \{0,1\}^n \to \{0,1\}^m\}$ $\exists (2^m)^{2^n}$ possible maps ("f"), By not knowing K, F_K is indistinguishable from a random f.

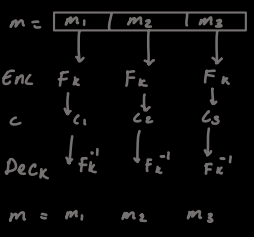
If F_K and f bijective: **Pseudorandom Permutation (PRP)**: $\exists 2^\lambda F_K$'s, $\exists 2^n!$ f's.

Block Cipher: $F: \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ Assumed to be a pseudorandom permutation (PRP)

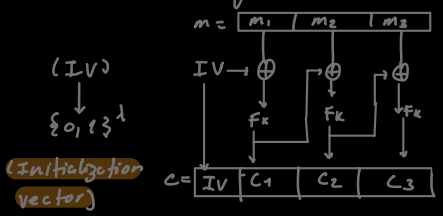
(i) Construction: $\lambda = 128/196/256$, $n = 128$, Standardized by NIST in 2001

(ii) Construct a SKE scheme from F for arbitrary-length messages: $K \{0,1\}^\lambda$, $Enc_K(m)$, $Dec_K(m)$. Goal: $\in PA$. $|m| = \alpha \cdot \lambda$ (BLOCKS)

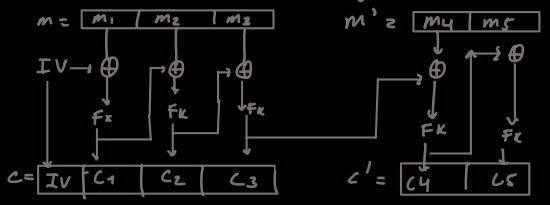
Electronic Code Book (ECB) Mode



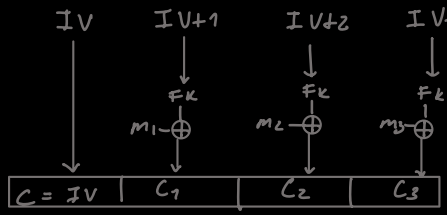
Cipher Block Chaining (CBC) Mode:



Chained Cipher Block Chaining (CBC) Mode:

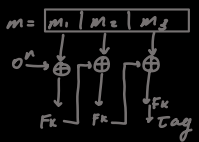


Counter (CTR) Mode



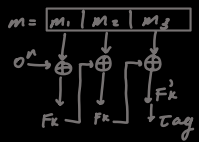
Does not need F_K to be bijective

CBC-MAC:

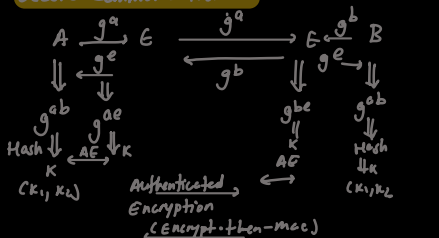


Encrypt last block

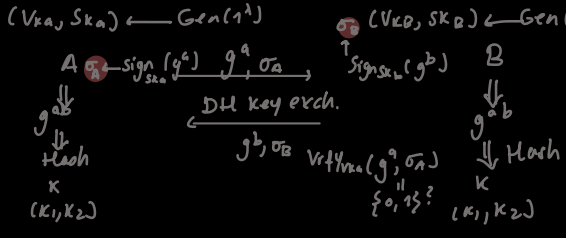
CBC-MAC:



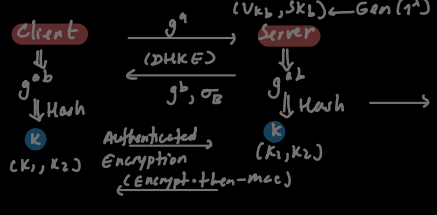
Secure Communication:



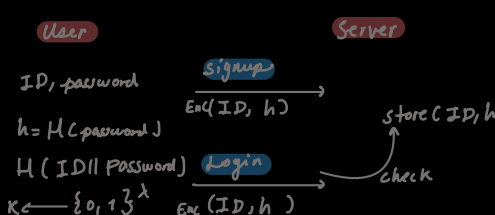
Authenticated key Exchange:



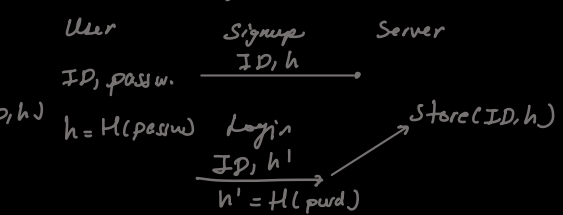
One-sided Secure Auth.:



Password-Based Auth.:



Online-Dictionary Attack:



Offline Dictionary Attack

User: ID, pwd; $h = H(pwd)$

Signup: ID, h

Server: store (ID, h); $h' = H(pwd')$ (preprocessing)

Salting

User: ID, pwd; $h = H(pwd \| salt)$; $h \leftarrow H(pwd \| salt)$

Signup: ID

Server: $salt \leftarrow \{0,1\}^{s=128}$

salt

h

store $(ID_i, salt_i, h_i)$

Login

salt

h

$H(pwd \| salt)$

Salt & Pepper:

User: Id, pwd; $h = H(pwd \| Id)$

Signup: salt, h

Server: $salt \{0,1\}^{128}$; $pepper \{0,1\}^{8}$; $h^* = H(h \| pepper)$; store (Id, salt, h^*)

Login: ID; salt; h; $h = H(pwd \| Id)$

$H(h \| pepper)$

Two-Factor Authentication (2FA):

User: Id, pwd; $h = H(pwd \| salt)$

Signup: ID, salt, h

Server: $salt \{0,1\}^{128}$; $pepper \{0,1\}^{8}$; $h^* = H(h \| pepper)$; store (Id, salt, p^*), phone #

1) SMS

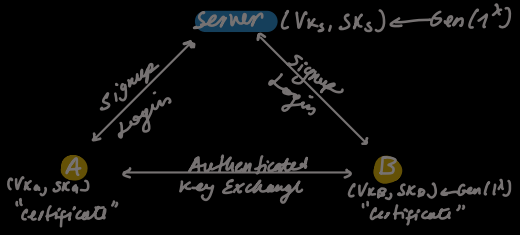
2) APP-gen-code

ID; salt; h; xxxxxx; h^*

F_k(timestamp); PRF

F_k(timestamp); PRF/PRP AES

Secure Authentication:



Two-sided Auth-Key-Exchange: