L15

① Public-Key Cryptosystem.
e_K transmitted via authenticated (~~confidential~~) channel. n parties – n pairs (e_K, d_K).

② 1. A ring where every nonzero element is invertible is called a field. $\mathbb{Z}_p$ is a field.
2. $\phi(n) = |\mathbb{Z}_n^*|$ for every $n \in \mathbb{Z}^+$.
3. $n > 1, \forall \alpha, \gcd(\alpha, n) = 1 : \alpha^{\phi(n)} \equiv 1 (mod\ n)$
4. primitive element modulo $\boldsymbol{p}$ ($\boldsymbol{p}$ is a prime): $\exists \alpha \in \mathbb{Z}_p^*$, s.t. $\mathbb{Z}_p^* = \{\alpha^0, \alpha^1, \dots, \alpha^{p-1}\}$.

③ RSA: Rivest, Shamir, Adleman.
$\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$, $n = pq$,
$\mathcal{K} = \{(n, p, q, a, b) : ab \equiv 1 \pmod{\phi(n)}\}$
$e_K(x) = x^b \bmod n$, $d_K(y) = y^a \bmod n$

④ $\pi(N) = \sum_{p \le N} 1$: the number of primes $\le N$
$\lim_{N\to\infty} \pi(N)/(N/\ln N) = 1$
$\pi(N) > \frac{N}{\ln N}\left(1 + \frac{1}{2\ln N}\right)$ when $N \ge 59$
$\pi(N) < \frac{N}{\ln N}\left(1 + \frac{3}{2\ln N}\right)$ when $N > 1$
Let $\mathbb{P}_\lambda$ be the set of λ-bit primes. Then

$$|\mathbb{P}_\lambda| \ge \frac{2^\lambda}{\lambda \ln 2}\left(\frac{1}{2} + O\left(\frac{1}{\lambda}\right)\right)$$

⑤ Square-and-Multiply (complexity $O(\log b\,(\log n)^2)$)
e.g. 2^{123} mod 5: $123 = (1111011)_2 = 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^6$. [$\mathrm{x} = \mathrm{x}^2 - y \times (x^2 \div Ry)$]

L16

① Group

2. Group:
① Closed $\forall a, b \in \mathbb{Z}_m$. $a + b \in \mathbb{Z}_m$. ⑥ $a \cdot b \in \mathbb{Z}_m$
② Associative $\forall a, b, c \in \mathbb{Z}_m$, $(a+b)+c = a+(b+c)$ ⑦ $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
③ Identity $\forall a \in \mathbb{Z}_m$, $a + 0 = 0 + a = a$ ⑧ $a \cdot 1 = 1 \cdot a = a$
④ Inverse $\forall a \in \mathbb{Z}_m$. $\exists m - a \in \mathbb{Z}_m$ s.t. $a + (m-a) = (m-a) + a = 0$
⑤ Commutative $\forall a, b \in \mathbb{Z}_m$, $a + b = b + a$ ⑨ $a \cdot b = b \cdot a$.
⑩ Distributive Property. $\forall a, b, c \in \mathbb{Z}_m$.
$(a+b) \cdot c = a \cdot c + b \cdot c$, $a \cdot (b + c) = a \cdot b + a \cdot c$.
①-④ : $(\mathbb{Z}_m, +)$ is a group.
①-⑤ : $(\mathbb{Z}_m, +)$ is an Abelian group.
①-⑩ : $(\mathbb{Z}_m, +, \cdot)$ is a ring.

② Subgroup: Let $(G,\cdot)$ be an Abelian group. A non-empty $H \subseteq G$ is called a **subgroup** of G if $(H,\cdot)$ is a group. $(H \le G)$.
Let $(G,\cdot)$ be an Abelian group and let $H \ne \emptyset$ be a subset of G. Then $H \le G$ if and only if $ab^{-1} \in H$ for any $a, b \in H$.

③ Coset: Let $(G,\cdot)$ be an Abelian group and let H be a subgroup of G. For any $g \in G$, the set $gH = \{gh : h \in H\}$ is said to be a **coset** of H in G.
Let $(G,\cdot)$ be an Abelian group and let H be a subgroup of G. We define $G/H = \{gH : g \in G\}$.
$|G/H| = |G|/|H|$

④ Quadratic Residue 二次剩余
Legendre symbol. Suppose that p is an odd prime.

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } a \equiv 0 \pmod p, \\ 1 & \text{if } [a]_p \in \mathbf{QR}(p), \\ -1 & \text{if } [a]_p \in \mathbf{QNR}(p). \end{cases} \qquad \left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p.$$

Jacobi Symbol. $n = p_1^{e_1} \cdots p_k^{e_k}$. $\left(\frac{a}{n}\right) = \prod_{i=1}^k \left(\frac{a}{p_i}\right)^{e_i}$.

① $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
② $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{b}{n}\right)$.
③ If $a \equiv b\ (mod\ n)$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
④ $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$.
⑤ $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$.
⑥ $\left(\frac{m}{n}\right) = (-1)^{\frac{(m-1)(n-1)}{4}}\left(\frac{n}{m}\right)$.

⑤ An integer $n > 1$ is said to be an **Euler Pseudo-Prime to the base** a if $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2}$ $(mod\ n)$.
$\left(\frac{a}{n}\right) = 0$ if and only if $\gcd(a, n) > 1$.

L17

① Let $n > 1$ be an **odd composite** number. Then the number of $a \in \mathbb{Z}_n^*$ s.t. n is a Euler pseudo-prime to the base a is $\le \phi(n)/2$.

② **Solovay-Strassen** (n) // Time complexity is $O((\log n)^3)$

```
choose a random integer a such that 1 ≤ a ≤ n − 1
   x ← (a/n)
   if x = 0
      then return ("n is composite")
   y ← a^((n−1)/2) (mod n)
   if x ≡ y (mod n)
      then return("n is prime")
      else return ("n is composite")
```

yes-biased Monte Carlo Alg: error probability $\Pr[a \in G(n)] \le |G(n)|/(n-1) < 1/2$, where $G(n) = \{a : a \in \mathbb{Z}_n^*,\ (a/n) \equiv a^{(n-1)/2}\ (mod\ n)\}$.

③ **Random Prime Number Generation with SS Test:**
choose from $[N, 2N]$
Choose an odd integer $n \in [N, 2N]$ uniformly
Run **Solovay-Strassen**(n) m times.
If all executions output "n is prime" then output n
Otherwise, Output "failure"
Error rate $\Pr[\mathbf{a}|\mathbf{b}] = \frac{Pr[\mathbf{b}|\mathbf{a}]\Pr[\mathbf{a}]}{Pr[\mathbf{b}]} \le \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}}$

④ Chinese Remainder Theorem
$n = n_1 \cdots n_k$, 互质, RHS always has a solution. Furthermore, if $b \in \mathbb{Z}$ is a solution, then any solution x must satisfy $x \equiv b (\mathrm{mod}\ n)$.

$$\begin{cases} x \equiv b_1 \pmod{n_1} \\ x \equiv b_2 \pmod{n_2} \\ \vdots \\ x \equiv b_k \pmod{n_k} \end{cases}$$

> Let $N_i = n/n_i$ for every $i \in [k]$. $\gcd(N_i, n_i) = 1$ for every $i \in [k]$. $\exists\ s_i,\ t_i,\ N_i s_i + n_i t_i = 1$.
Let $b = b_1(N_1 s_1) + \cdots + b_k(N_k s_k)$.
Then $b \equiv b_i (\mathrm{mod}\ n_i)$ for every $i \in [k]$.

⑤ **Test**$_L(n)$

```
choose a ← {1,2, …, n − 1} uniformly and at random
if a ∈ L_n
   then return "n is prime"
   else return "n is composite"
```

Fermat(n): $L_n = \{a : a^{n-1} \equiv 1\ (mod\ n)\}$
Carmichael Number n: $a^{n-1} \equiv 1\ (mod\ n)$ when $\gcd(a, n) = 1$. There are ∞ such numbers. Bad.

L18

① Miller-Rabin Test
$L_n = \{a :\ 1 \le a \le n - 1;\ \ a^{2^k m} \equiv 1 (mod\ n);$ $a^{2^{j+1} m} \equiv 1 (mod\ n) \Rightarrow a^{2^j m} \equiv \pm 1 (mod\ n)$ for $0 \le j < k)\}$ a. If n is an odd prime, then $L_n = \{1, 2, \dots, n-1\}$. b. If n is an odd composite and not a prime power, then $|L_n| \le (n-1)/2$.

Miller-Rabin(n) // Time complexity $O((\log n)^3)$

```
write n − 1 = 2^k m, where m is odd
choose a random integer a such that 1 ≤ a ≤ n − 1
b ← a^m mod n
if b ≡ 1 (mod n)
   then return("n is prime")
for i ← 0 to k − 1
      ⎧ if b ≡ −1 (mod n)
   do ⎨    then return ("n is prime")
      ⎩ else b ← b^2 mod n
return("n is composite")
```

yes-biased, error rate $\Pr[a \in L_n] \le |L_n|/(n-1) < 1/2$ (can be improved to $1/4$)

② Pollard $p{-}1$ Algorithm
Scenario: $n = pq$ and the prime power divisors of $p - 1$ are all small. i.e. There exists $B > 0$ such that $p - 1 = p_1^{e_1} \cdots p_\ell^{e_\ell}$ and for all $i \in [\ell]$, $p_i^{e_i} \le B$.
Pollard $p - 1(n, B)$ $O(B \log B\,(\log n)^2 + (\log n)^3)$

```
a ← 2
for j ← 2 to B
   do a ← a^j mod n
d ← gcd(a − 1, n)
if 1 < d < n
   then return (d)
   else return ("failure")
```

safe prime: $p = 2p' + 1$.

③ Pollard Rho Algorithm
Scenario: $n = pq$ and $\min\{p, q\}$ is small.
Birthday paradox: $Q \approx \sqrt{2M \cdot \ln\frac{1}{1-\epsilon}} = 1.17\sqrt{M}$.
Pollard Rho Factoring Algorithm(n, x_1)

```
external: f
x ← x_1
x' ← f(x) mod n
p ← gcd(x − x', n)
while p = 1
      ⎧ x ← f(x) mod n
   do ⎨ x' ← f(x') mod n
      ⎪ x' ← f(x') mod n
      ⎩ p ← gcd(x − x', n)
if p = n
   then return ("failure")
   else return (p)
```

$O(\sqrt{p}) = O(n^{1/4})$ - $\min\{p, q\}$ is large enough.

L19

① **Dixon's Random Squares Algorithm**
$\mathcal{B} = \{p_1, \dots, p_b\}$
$c = b + 4$
Choose c integers $z_1, z_2, \dots, z_c$ such that for every $j = 1, 2, \dots, c$

$$z_j^2 \equiv p_1^{\alpha_{1j}} \times \dots \times p_b^{\alpha_{bj}} \pmod n$$

$a_j = (\alpha_{1j}, \dots, \alpha_{bj})$ for all $j \in [c]$
Find a subset $J \subseteq \{1, 2, \dots, c\}$ such that

$$(t_1, t_2, \dots, t_b) = \sum_{j \in J} a_j \equiv (0, 0, \dots, 0) \quad (mod\ 2)$$

$x = \prod_{j \in J} z_j \ \ mod\ n$
$y = p_1^{t_1/2} \times p_2^{t_2/2} \times \cdots \times p_b^{t_b/2}\ \ mod\ n$
Output $\gcd(x \pm y, n)$
Failure probability: $\Pr[x \equiv \pm y\ (mod\ n)] \le \frac{1}{2}$
Complexity: $O\left(e^{(1+o(1))\sqrt{\ln n \ln\ln n}}\right)$

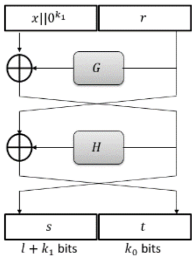
② Wiener's Low Decryption Exponent Attack
Scenario: $n = pq$, $K = (n, p, q, a, b)$ and a is small: $3a < n^{1/4}$ and $q < p < 2q$
$ab - t\phi(n) = 1$; $\left|\frac{b}{n} - \frac{t}{a}\right| < \frac{1}{3a^2}$
Wiener's Algorithm (n, b)

```
(q_1, …, q_m; r_m) ← Euclidean algorithm(b, n)
c_0 ← 1;  c_1 ← q_1;  d_0 ← 0;  d_1 ← 1
for j ← 2 to m
      ⎧ c_j ← q_j c_{j−1} + c_{j−2}
      ⎪ d_j ← q_j d_{j−1} + d_{j−2}
      ⎪ n' ← (d_j b − 1)/c_j
      ⎪  comment: n' = φ(n) if c_j/d_j is the correct convergent
   do ⎨ if n' is an integer
      ⎪           ⎧ let p and q be the roots of the equation
      ⎪      then ⎨    x^2 − (n − n' + 1)x + n = 0
      ⎪           ⎪ if p and q are positive integers less than n
      ⎩           ⎩    then return (p, q)
return ("failure")
```

L20

① Adversarial Goals: Total break (sk); Partial break; Distinguishability of ciphertexts (p>1/2)

② Semantic Security: cannot distinguish ciphertexts.
RSA Optimal Asymmetric Encryption Padding



③ Let $(G,\cdot)$ be an Abelian group. If there is an $\alpha \in G$ such that $G = \langle\alpha\rangle$, then G is a **cyclic group** and α is a **generator** of G.

④ **ElGamal Cryptosystem**
$\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{C} = \mathbb{Z}_p^*$, $\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod p\}$
Encryption: $y_1 = \alpha^k \bmod p$; $y_2 = x\beta^k \bmod p$;
$e_K(x, k) = (y_1, y_2)$.
Decryption: $d_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$.

L21

① Algorithms for the Discrete Logarithm Problem
Exhaustive Search; Lookup Table.

② Shanks' Algorithm - **Shanks**(G, n, α, β) // $O(m \log m)$ time; $O(m)$ Space

1. $m \leftarrow \lceil\sqrt{n}\rceil$
2. for $j \leftarrow 0$ **to** $m - 1$
  do compute α^{mj}
3. Sort the m ordered pairs (j, α^{mj}) with respect to their second coordinates,
  obtaining a list L_1
4. for $i \leftarrow 0$ **to** $m - 1$
  do compute $\beta\alpha^{-i}$
5. Sort the m ordered pairs $(i, \beta\alpha^{-i})$ with respect to their second coordinates, obtaining a list L_2
6. Find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
7. $\log_\alpha \beta \leftarrow (mj + i) \bmod n$

③ Pollard Rho Algorithm
Pollard-Rho(G, n, α, β) // $O(\sqrt{n})$ iterations

```
main
   define the partition G = S_1 ∪ S_2 ∪ S_3
   (x, a, b) ← f(1,0,0)
   (x', a', b') ← f(x, a, b)
   while x ≠ x'
         ⎧ (x, a, b) ← f(x, a, b)
      do ⎨ (x', a', b') ← f(x', a', b')
         ⎩ (x', a', b') ← f(x', a', b')
   if gcd(b' − b, n) ≠ 1
      then return ("failure")
      else return (a − a')(b' − b)^(−1) mod n
```

$$f(x, a, b) = \begin{cases} (\beta x,\ a, b + 1) & \text{if } x \in S_1 \\ (x^2, 2a, 2b) & \text{if } x \in S_2 \\ (\alpha x,\ a + 1, b) & \text{if } x \in S_3 \end{cases}$$

④ Pohlig-Hellman Algorithm
Scenario: $n = p_1^{c_1} p_2^{c_2} \cdots p_k^{c_k}$ and $\max\{p_1, p_2, \dots, p_k\}$ is small. and $|\langle\alpha\rangle| = n$.
Pohlig-Hellman$(G, n, \alpha, \beta, q, c)$ //complexity: $O(c\sqrt{q})$ (q is p_i here).

```
j ← 0
β_j ← β          //Remark: β_0 = β
while j ≤ c − 1
      ⎧ δ ← β_j^(n/q^(j+1))
      ⎪ find i such that δ = α^(in/q)
   do ⎨ a_j ← i
      ⎪ β_(j+1) ← β_j α^(−a_j q^j)
      ⎩ j ← j + 1
return (a_0, ⋯, a_(c−1))
```

L22

① The Index Calculus Method
Scenario: $G = \mathbb{Z}_p^*$ and α is a primitive element modulo p. // $n = p - 1$
Factor base $\mathcal{B} = \{p_1, p_2, \dots, p_B\}$: a set of small primes
Step 1: find the discrete logarithms of the small primes, i.e., $\{\log_\alpha p_i\}_{i=1}^B$
We have C equations in the B unknowns $\{\log_\alpha p_i\}_{i=1}^B$
$x_i \equiv a_{1i} \log_\alpha p_1 + a_{2i} \log_\alpha p_2 + \cdots + a_{Bi} \log_\alpha p_B$ $(mod\ p - 1)$, $i = 1, 2, \dots, C$
Step 2: Factorize a random element $\beta\alpha^s\ mod\ p$ over the factor base.
$\log_\alpha \beta = c_1 \log_\alpha p_1 + c_2 \log_\alpha p_2 + \cdots + c_B \log_\alpha p_B - s$ $mod\ (p-1)$
Time complexity: $O\left(e^{(1+o(1))\sqrt{\ln p \ln\ln p}}\right)$

② ElGamal cryptosystem based $\mathbb{Z}_p^*$ is not semantically secure. $(x/p) = (x\beta^k/p)\cdot(\beta^k/p)$
Solution: always choose x such that (x/p) =1.

③ Computational Diffie-Hellman Problem: (CDH Problem): Find $\delta \in \langle\alpha\rangle$ such that $\log_\alpha \delta \equiv \log_\alpha \beta \log_\alpha \gamma\ (mod\ n)$.
Decision Diffie-Hellman Problem: (DDH Problem): Is it the case that $\log_\alpha \delta \equiv \log_\alpha \beta \log_\alpha \gamma\ (mod\ n)$
(1) DDH is reducible to CDH; (2) CDH is reducible to Dlog.

④ RSA Signature: $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$, $\mathcal{K} = \{(n,p,q,a,b): n = pq,\ p \text{ and } q \text{ are primes},\ ab \equiv 1(\text{mod } \phi(n))\}$. $pk = (n,b)$, $sk = (n,a)$
$\mathbf{sig}_K(x) = x^a \bmod n$, $\mathbf{ver}_K(x,y) = \text{true}$ iff $x \equiv y^b(\text{mod } n)$. easy to produce a forgery. Choose $y \in \mathbb{Z}_n$, compute $x = y^b\ mod\ n$; output (x,y).

L23

① Attack Model: key-only attack (KOA); known message attack (KMA); chosen message attack (CMA). Adversarial Goals: total break; selective forgery (uncontrolled x); existential forgery.

② Hash-and-Sign Paradigm: If$(\mathcal{P},\mathcal{A},\mathcal{K},\mathcal{S},\mathcal{V})$ is existentially unforgeable under the chosen message attack (EUF-CMA) and h is collision-resistant, then the hash-and-sign scheme is EUF-CMA.

③ ElGamal Signature
$\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$, $\mathcal{K} = \{(p,\alpha,a,\beta): \beta \equiv \alpha^a(\text{mod } p)\}$, pk=(p,α,β), sk=a.
$\mathbf{sig}_K(x,k) = (\gamma,\delta)$: $k \leftarrow \mathbb{Z}_{p-1}^*$, compute $\gamma = \alpha^k \bmod p$ and $\delta = (x - a\gamma)k^{-1} \bmod (p-1)$
Verification: If $\beta^\gamma\gamma^\delta \equiv \alpha^x(\text{mod } p)$, output true.
Attack: Given $pk = (p,\alpha,\beta)$, choose (γ,δ,x).
choose $i,j \in \{0,1,\dots,p-2\}$
$\gamma = \alpha^i\beta^j\ mod\ p$; $\delta = -\gamma j^{-1}\ mod\ (p-1)$; $x = i\delta\ mod\ (p-1)$
Given $pk = (p,\alpha,\beta)$, a valid signed message (x,γ,δ), compute (x',λ,μ)
Choose $h,i,j \in \{0,1,\dots,p-2\}$ such that $\gcd(h\gamma - j\delta,\ p-1) = 1$
$\lambda = \gamma^h\alpha^i\beta^j\ mod\ p$; $\mu = \delta\lambda(h\gamma - j\delta)^{-1}\ mod\ (p-1)$
$x' = \lambda(hx + i\delta)(h\gamma - j\delta)^{-1}\ mod\ (p-1)$
Countermeasure: Hash-and-Sign

④ Schnorr Signature
$\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$, $\mathcal{K} = \{(p,q,\alpha,a,\beta): \beta \equiv \alpha^a(\text{mod } p)\}$; $0 \le a < q$.
sig_K (x,k)=(γ,δ): $k \leftarrow \{1,2,\dots,q-1\}$, $\gamma = h(x||\alpha^k\ mod\ p)$, $\delta = k + a\gamma\ mod\ q$
Verification: If $h(x||\alpha^\delta\beta^{-\gamma} \bmod p) = \gamma$, true.

L24

① The Digital Signature Algorithm (DSA) like Schnorr
$\mathbf{sig}_K(x,k) = (\gamma,\delta)$: $\gamma = (\alpha^k \bmod p) \bmod q$, $\delta = (\text{SHA3-224}(x) + a\gamma)k^{-1}\ mod\ q$
Verification: $e_1 = \text{SHA3-224}(x)\delta^{-1} \bmod q$, $e_2 = \gamma\delta^{-1} \bmod q$. If $(\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma$, true.

② Public-Key Infrastructure (PKI); Certification Authority (CA). Generate sig and ver for users.
$\mathbf{Cert}(\text{Alice}) = ID(\text{Alice})||\mathbf{ver}_{\text{Alice}}||s$, $s = \mathbf{sig}_{\text{CA}}(ID(\text{Alice})||\mathbf{ver}_{\text{Alice}})$; Verify with $\mathbf{ver}_{CA}(ID(\text{Alice})||\mathbf{ver}_{\text{Alice}}, s) = \text{true}$.

③ Sign-then-Encrypt (sender: Alice; receiver: Bob):
Alice: $y = \mathbf{sig}_{\text{Alice}}(x, ID(\text{Bob}))$; $z = e_{\text{Bob}}(x, y, ID(\text{Alice}))$.
Bob: $(x, y, ID(\text{Alice})) = d_{\text{Bob}}(z)$.
Bob: ?$\mathbf{ver}_{\text{Alice}}\big((x, ID(\text{Bob})), y\big) = \text{true}$.
Encrypt-then-Sign (sender: Alice; receiver: Bob):
Alice: $z = e_{\text{Bob}}(x, ID(\text{Alice}))$; $y = \mathbf{sig}_{\text{Alice}}(z, ID(\text{Bob}))$
Alice: sends $(z, y, ID(\text{Alice}))$ to Bob.
Bob: If $\mathbf{ver}_{\text{Alice}}\big((z, ID(\text{Bob})), y\big) = \text{true}$, $(x', ID') = d_{\text{Bob}}(z)$.
Bob: ? $ID' =$ the 3rd entry of $(z, y, ID(\text{Alice}))$.

L25

① Passwords: Online Attacks / Offline Attacks
Hash the passwords; Salt: fingerprint= $h(userid||salt)$ (avoid precompute / same passwords); Key Stretching (hash 10000 times).

② Challenge-and-Response
Secret-Key Setting
Bob chooses a random challenge, r, which he sends to Alice.
Alice computes $y = MAC_K(ID(Alice)\|r)$ and sends y to Bob.
Bob computes $y' = MAC_K(ID(Alice)\|r)$.
If $y' = y$, then Bob "accepts"; otherwise, Bob "rejects."

③ Attack Model: Passive information-gathering model; Active information-gathering model (temporary access to the oracle $MAC_K(\cdot)$ and responds to challenges).
Active adversary: $\mathcal{A}$ creates a message and places it into the channel; $\mathcal{A}$ changes a message in the channel; $\mathcal{A}$ diverts a message in the channel so it is sent to someone other than the intended receiver.

④ Secure Mutual Challenge-and-Response
Bob: sends a random challenge r_1 to Alice.
Alice: picks a random challenge r_2; Computes $y_1 = MAC_K(ID(Alice)\|r_1\|r_2)$; sends (y_1, r_2) to Bob.
Bob: computes $y_1' = MAC_K(ID(Alice)\|r_1\|r_2)$. If $y_1' = y_1$, then accepts; otherwise, rejects; Somputes $y_2 = MAC_K(ID(Bob)\|r_2)$ and sends y_2 to Alice.
Alice: computes $y_2' = MAC_K(ID(Bob)\|r_2)$. If $y_2' = y_2$, accepts; o.w., rejects

⑤ Public-Key Setting
Bob: picks a random challenge r_1; sends $(\mathbf{Cert}(B),\ r_1)$ to Alice.
Alice: picks a random challenge r_2, computes $y_1 = \mathbf{sig}_A(ID(B)\|r_1\|r_2)$; Sends $(\mathbf{Cert}(A), r_2, y_1)$ to Bob.
Bob: verifies $\mathbf{ver}_A$ with $\mathbf{Cert}(Alice)$; If $\mathbf{ver}_A(ID(B)\|r_1\|r_2, y_1) = true$, accepts; otherwise, rejects; Computes $y_2 = \mathbf{sig}_B(ID(A)\|r_2)$; Sends y_2 to Alice.
Alice: verifies $\mathbf{ver}_B$ with $\mathbf{Cert}(Bob)$; If $\mathbf{ver}_B(ID(A)\|r_2, y_2) = true$, accepts; otherwise, rejects.

⑥ Schnorr Identification Scheme
t: a security parameter such that $q > 2^t$; private key: $a \in \{0,1,\dots,q-1\}$; public key: $v = \alpha^{-a}\ mod\ p$.
Alice: chooses a random number $k \in \{0,1,\dots,q-1\}$; computes $\gamma = \alpha^k \bmod p$; sends $(\mathbf{Cert}(Alice),\gamma)$ to Bob.
Bob: verifies v on the certificate $\mathbf{Cert}(Alice)$; sends a random challenge $r \in \{1,2,\dots,2^t\}$ to Alice.
Alice: sends the response $y = k + ar \bmod q$ to Bob.
Bob: If $\gamma \equiv \alpha^y v^r(\text{mod } p)$, accepts; otherwise, rejects.

L26

① **The Computational Composite Quadratic Residues Problem (CCQR):**
Instance: (n,x), where $n = pq$ is the product of two primes $p,q \equiv 3\ (mod\ 4)$ and $x \in \mathbb{Z}_n^*$ is an integer such that $(x/n) = 1$.
Question: Find $y \in \mathbb{Z}_n^*$ such that $y^2 \equiv x(mod\ n)$ or $y^2 \equiv -x\ (mod\ n)$

② **Feige-Fiat-Shamir Identification Scheme**
$n = pq$, $p \equiv q \equiv 3\ (mod\ 4)$; $S_1, S_2, \dots, S_k \in \mathbb{Z}_n^*$ for $k = \log\log n$; $I_j = \pm 1/S_j^2\ mod\ n$ for all $j \in [k]$, where $+,-$ are chosen randomly.
Alice's public key: $\mathbf{I} = (I_1, I_2, \dots, I_k)$; private key: $\mathbf{S} = (S_1, S_2, \dots, S_k)$
Repeat the following steps $t = \log_2 n$ times:
Alice: chooses a random value $R \in \mathbb{Z}_n$; computes $X = \pm R^2 \bmod n$ with the sign chosen randomly; sends X to Bob.
Bob: sends a random challenge $\mathbf{E} = (E_1, \dots, E_k) \in \{0,1\}^k$ to Alice.
Alice: sends the response $Y = R\prod_{\{j:E_j=1\}} S_j\ mod\ n$ to Bob.
Bob: If $X = \pm Y^2 \prod_{\{j:E_j=1\}} I_j\ mod\ n$, accepts; otherwise, rejects.

③ Schnorr Identification and Feige-Fiat-Shamir are both Complete (Correct) + Sound, Zero-Knowledge from honest and dishonest verifiers.

④ Secure Computation: input x_i; communicate securely; public function $f(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n)$; curious party P_i.

⑤ Private Information Retrieval (PIR): If there is only one server and the perfect secrecy of i is required, then the communication cost must be $O(n)$.

⑥ A k-server PIR protocol is a triple $(\mathcal{Q},\mathcal{A},\mathcal{R})$, where $\mathcal{Q}$ is a probabilistic **querying algorithm**. It takes i as input and outputs k queries $q_1, q_2, \dots, q_k$ and a reconstruction key rk. $\mathcal{A}$ is an **answering algorithm**. It takes (x, q_j) as input and outputs an answer a_j. $\mathcal{R}$ is a **reconstructing algorithm**. It takes $(i, rk, a_1, a_2, \dots, a_k)$ as input and outputs x_i.

L27

① Covering Code Protocol
Communication Cost: $O(n^{1/3})$
8-Server PIR: Send $\{S_1^u \times S_2^v \times S_3^w : (u,v,w) \in \{0,1\}^3\}$ to 8 servers.
$n = l^3$; $\omega: [n] \to [l]^3$ is a bijection; $y = \big(y_{(\alpha,\beta,\gamma)}\big)$ s.t. $y_{\omega(j)} = x_j$ for all $j \in [n]$ The 8 servers:
$\mathbb{S}_{(0,0,0)}, \mathbb{S}_{(0,0,1)}, \mathbb{S}_{(0,1,0)}, \mathbb{S}_{(0,1,1)}, \mathbb{S}_{(1,0,0)}, \mathbb{S}_{(1,0,1)}, \mathbb{S}_{(1,1,0)}, \mathbb{S}_{(1,1,1)}$
$\mathcal{Q}(i)$: suppose that $\omega(i) = (i_1, i_2, i_3) \in [l]^3$
Choose $S_1^0, S_2^0, S_3^0 \subseteq [l]$ uniformly and at random
Set $S_1^1 = S_1^0 \oplus \{i_1\}; S_2^1 = S_2^0 \oplus \{i_2\}; S_3^1 = S_3^0 \oplus \{i_3\}$
Set $q_{(u,v,w)} = (S_1^u, S_2^v, S_3^w)$ for all $(u,v,w) \in \{0,1\}^3$; set $rk =\perp$
Output $\{(q_{(u,v,w)}) : (u,v,w) \in \{0,1\}^3\}$ and rk.
$\mathcal{A}(x, q_{(u,v,w)})$: output $a_{(u,v,w)} = \sum_{(\alpha,\beta,\gamma)\in S_1^u\times S_2^v\times S_3^w} y_{(\alpha,\beta,\gamma)}$ //done by $\mathbb{S}_{(u,v,w)}$
$\mathcal{R}(i, rk, \{a_{(u,v,w)}\})$: output $\sum_{(u,v,w)\in\{0,1\}^3} a_{(u,v,w)}$
2-Server PIR: Send $S_1^0 \times S_2^0 \times S_3^0$ and $S_1^1 \times S_2^1 \times S_3^1$to $\mathbb{S}_{(0,0,0)}$ and $\mathbb{S}_{(1,1,1)}$.
$n = l^3$; $\omega: [n] \to [l]^3$ is a bijection; $y = \big(y_{(\alpha,\beta,\gamma)}\big)$ s.t. $y_{\omega(j)} = x_j$ for all $j \in [n]$
The 2 servers: $\mathbb{S}_{(0,0,0)}$, $\mathbb{S}_{(1,1,1)}$
$\mathcal{Q}(i)$: suppose that $\omega(i) = (i_1, i_2, i_3) \in [l]^3$
Choose $S_1^0, S_2^0, S_3^0 \subseteq [l]$ uniformly and at random
Set $S_1^1 = S_1^0 \oplus \{i_1\}; S_2^1 = S_2^0 \oplus \{i_2\}; S_3^1 = S_3^0 \oplus \{i_3\}$
Set $q_{(u,v,w)} = (S_1^u, S_2^v, S_3^w)$ for all $(u,v,w) \in \{(0,0,0),(1,1,1)\}$; set $rk =\perp$
Output $\{q_{(0,0,0)}, q_{(1,1,1)}\}$ and rk.
$\mathcal{A}(x, q_{(0,0,0)})$: output $a_{(0,0,0)}, A_{(1,0,0)}, A_{(0,1,0)}, A_{(0,0,1)}$
$\mathcal{A}(x, q_{(1,1,1)})$: output $a_{(1,1,1)}, A_{(0,1,1)}, A_{(1,0,1)}, A_{(1,1,0)}$
$\mathcal{R}(i, rk, \{a_{(u,v,w)}\})$: output $\sum_{(u,v,w)\in\{0,1\}^3} a_{(u,v,w)}$
$a_{(1,0,0)}, a_{(0,1,0)}, a_{(0,0,1)}$ are extracted from $A_{(1,0,0)}, A_{(0,1,0)}, A_{(0,0,1)}$
$a_{(0,1,1)}, a_{(1,0,1)}, a_{(1,1,0)}$ are extracted from $A_{(0,1,1)}, A_{(1,0,1)}, A_{(1,1,0)}$

L28

① Homomorphic Encryption. Additively homomorphic: $y_1 = e_K(x_1, k_1)$, $y_2 = e_K(x_2, k_2)$, $y = \sigma(y_1, y_2)$ is a ciphertext of $x_1 + x_2$.

② Paillier's Cryptosystem
$N = pq$, p and q are distinct odd primes and $\gcd(N, \phi(N)) = 1$.
$\mathcal{P} = \mathbb{Z}_N, \mathcal{C} = \mathbb{Z}_{N^2}^*$, $\mathcal{K} = \{(N, g, p, q)\} = \{(N, 1+N, p, q)\}$
$pk = (N, g)$; $sk = (p, q)$ or $sk = \phi(N)$
Encryption: For every $K = (N, g, p, q) \in \mathcal{K}$, $x \in \mathcal{P}$, and $r \in \mathbb{Z}_N^*$,
$$e_K(x, r) = g^x r^N \bmod N^2$$
Decryption: For every $K = (N, g, p, q) \in \mathcal{K}$, $y \in \mathcal{C}$,
$$d_K(y) = \left(\frac{\left(y^{\phi(N)} \text{mod } N^2\right) - 1}{N}\right) \times (\phi(N)^{-1} \text{mod } N) \bmod N$$
$\sigma(y_1, y_2) = y_1 \cdot y_2\ mod\ N^2$

③ Security: The Nth Residue Problem. **Instance:** (N, y), $N = pq$ is the product of two odd primes p, q; and $y \in \mathbb{Z}_{N^2}^*$; **Question:** Is there an integer $z \in \mathbb{Z}_{N^2}^*$ such that $y = z^N \bmod N^2$. Paillier's secure under CPA if the Nth residue problem is difficult.

④ HE-PIR Communication: $O(\sqrt{n})$ elements of $\mathbb{Z}_{N^2}^*$
Querying Algorithm: $q \leftarrow \mathcal{Q}(i)$, where $\omega(j) = (u, v) \in [l]^2$
Choose $K = (N, g, p, q)$ for Paillier's cryptosystem
Compute a ciphertext for the vth unit vector $(0,0,\dots,0,1,0,\dots,0)$
If $j \neq v$, $y_j = r_j^N \bmod N^2$
If $j = v$, choose $r_j \leftarrow \mathbb{Z}_N^*$, let $y_j = gr_j^N \bmod N^2$
Output $q = (y_1, y_2, \dots, y_l)$ and $rk = \phi(N)$
Answering Algorithm: $a \leftarrow \mathcal{A}(x, q)$
For every $s \in [l]$, compute $a_s = (y_1)^{X_{(s,1)}}(y_2)^{X_{(s,2)}} \cdots (y_l)^{X_{(s,l)}}\ mod\ N^2$
Output $a = (a_1, a_2, \dots, a_l)$
Reconstructing Algorithm: $\mathcal{R}(i, rk, a)$; $rk = \phi(N)$
$a_u = (y_1)^{X_{(u,1)}}(y_2)^{X_{(u,2)}} \cdots (y_l)^{X_{(u,l)}}\ mod\ N^2$
$= g^{X_{(u,v)}}((r_1)^{X_{(u,1)}}(r_2)^{X_{(u,2)}} \cdots (r_l)^{X_{(u,l)}})^N\ mod\ N^2$
$x_i = d_K(a_u)$

L29

① The 1-out-of-2 Oblivious Transfer Problem: $f\big((x_0, x_1), i\big) = (\perp, x_i)$
Even-Goldreich-Lempel OT (honest players)
Bob: Choose (pk_i, sk_i) and pk_{1-i}; send (pk_0, pk_1) to Alice
Alice: Compute $y_0 = e_{pk_0}(x_0)$, $y_1 = e_{pk_1}(x_1)$; send (y_0, y_1) to Bob
Bob: Compute $x_i = d_{sk_i}(y_i)$.
Bellare-Micali OT
p: a prime; q: a prime factor of $p-1$; $\alpha \in \mathbb{Z}_p^*$ has order q; $G = \langle\alpha\rangle$
$h: G \to \{0,1\}^n$, a cryptographic hash function
Alice's input: $x_0, x_1 \in \{0,1\}^n$; Bob's input: $i \in \{0,1\}$
Alice: choose a group element $c \leftarrow G$ uniformly and at random;
Bob: choose $k \leftarrow \mathbb{Z}_q$; send $pk_i = \alpha^k$, $pk_{1-i} = c/\alpha^k$ to Alice
Alice: choose $r_0, r_1 \leftarrow \mathbb{Z}_q$; send the following ciphertexts to Bob
$y_0 = (\alpha^{r_0}, h((pk_0)^{r_0}) \oplus x_0)$; $y_1 = (\alpha^{r_1}, h((pk_1)^{r_1}) \oplus x_1)$;
Bob: For $y_i = (a, b)$, output $x_i = b \oplus h(a^k)$

② Yao's Garbled Circuit (computationally secure)
(1) Alice: $f \rightarrow$ Boolean Circuit BC(f)
(2) Alice: BC(f) →Garbled Circuit C (f)
Special Symmetric-Key Cryptosystem for Constructing GC $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$
Elusive Range: without knowing $K \in \mathcal{K}$, it is difficult to find a $y \in \mathcal{C}$ such that y is a ciphertext of encrypting some $x \in \mathcal{P}$ using K.
Efficiently Verifiable Range: Given $K \in \mathcal{K}$ and any y, it is easy to decide whether y is a ciphertext of encrypting some $x \in \mathcal{P}$ using K

$e_{K_u^0}\big(e_{K_v^0}(K_w^0)\big)$
$e_{K_u^0}\big(e_{K_v^1}(K_w^0)\big)$
$e_{K_u^1}\big(e_{K_v^0}(K_w^0)\big)$
$e_{K_u^1}\big(e_{K_v^1}(K_w^1)\big)$

(3) Alice: Send C (f) and Input Labels to Bob
(4) Bob: Collect input labels from Alice (OT)
(5) Bob: Evaluate the Garbled Circuit
(6) Alice: Decide the Output

L30

① Secret Sharing Based Protocol (information-theoretically secure) (confidentiality of inputs)
A SIMPLE SOLUTION: Each party share its input among the 4 parties.
Suppose that $x_1, x_2, x_3, x_4 \in \mathbb{Z}_p$.
Each party represents its input as the sum of 4 random numbers in $\mathbb{Z}_p$
Each party announces the sum of its 4 shares
Example: Alice will announces $y_1 = x_{11} + x_{21} + x_{31} + x_{41}$
Each party outputs $y = y_1 + y_2 + y_3 + y_4$

② Delegation of Computation
The Problem: There is a big matrix $F = \big(F_{ij}\big)_{n\times n}$;
Alice wants to learn $y = Fx$ for any vector $x = (x_1, x_2, \dots, x_n)^\top$.
Idea: Alice precompute a key vk for future verifications.
Suppose that F is a matrix over $\mathbb{Z}_q$, where q is a large prime
Let $r = (r_1, r_2, \dots, r_n) \in \mathbb{Z}_q^n$ be a random vector. Let $s = (s_1, s_2, \dots, s_n) = rF$.
Let $vk = (r, s)$ be a verification key.
For any $x = (x_1, x_2, \dots, x_n)^\top \in \mathbb{Z}_q^n$, if $y = (y_1, y_2, \dots, y_n) = Fx$, then
$$r \cdot y = r(Fx) = (rF)x = s \cdot x.$$
Given (F, x), Bob needs to return $y = Fx$ to Alice.
Alice verifies the equation $r \cdot y = s \cdot x$
Question: The verification require a private key $vk = (r, s)$.
The Protocol: Alice prepares vk; Bob computes Fx; Alice verifies
Alice: choose a random vector $r = (r_1, r_2, \dots, r_n) \in \mathbb{Z}_q^n$
Alice: compute $s = (s_1, s_2, \dots, s_n) = rF$
Alice: Compute $vk = (\alpha^{r_1}, \alpha^{r_2}, \dots, \alpha^{r_n}, \alpha^{s_1}, \alpha^{s_2}, \dots, \alpha^{s_n})$
Bob: Compute $y = (y_1, y_2, \dots, y_n) = Fx$ and send y to Alice
Alice: If $\prod_{i=1}^n (\alpha^{r_i})^{y_i} = \prod_{j=1}^n (\alpha^{s_j})^{x_j}$, output y; otherwise, output $\perp$.
The vk can be precomputed and used for many different x! The cost can be amortized!