# Fundamentals of Cryptography: Final

Wednesday Jan 7, 2-4PM

**Problem 1 (2pt)** Which of the following assumptions implies digital signature.

(a) RSA assumption.
(b) DDH assumption.
(c) LWE assumption.
(d) DCR assumption (Pailliar).

**Problem 2 (2pt)** Consider a malicious-verifier zero-knowledge proof protocol $\Pi$ for NP problems. Protocol $\Pi$ is a sigma-protocol, that is, it has a 3-move structure: the prover commits, the verifier chooses a random challenge, and the prover response. Say the soundness error of the protocol is $\Theta(1)$. Which of the following must be a malicious-verifier zero-knowledge proof protocol with negligible soundness error.

(a) $\lambda$ parallel repetition of $\Pi$.
(b) The Fiat-Shamir heuristic of (a).
(c) $\lambda$ sequential repetition of $\Pi$.
(d) The Fiat-Shamir heuristic of (c).

To enable Fiat-Shamir, we additionally assume that $\Pi$ is public-coin, i.e., the verifier's randomness is his challenge. The hash function is modeled as a random oracle.

**Problem 3 (3pt)** Write down the construction of your favorite digital signature scheme and the name of the computational assumption it depends on.

**Problem 4 (3pt) Garbled Circuits** You should state how to garble a boolean circuit. The solution is not unique.

Given the circuit $C$, for each wire $i \in [n]$, the garbling algorithm generates two random $L_{i,0}, L_{i,1}$ as follows: fill the blank. Output $L_{1,0}, L_{1,1}, \ldots, L_{n_{\text{in}},0}, L_{n_{\text{in}},1}$ as the input labels. And output the garbled circuit $\tilde{C}$ as follows

- For each $i \in \{n_{\text{in}}+1, \ldots, n\}$, generate and output a table as follows: fill the blank. Say the gate function is $g : \{0,1\} \times \{0,1\} \to \{0,1\}$, and the gate takes wires $j_1, j_2$ as inputs.
- For each output wire $i \in \{n - n_{\text{out}} + 1, \ldots, n\}$, output fill the blank.

**Formalization of a circuit (for problem 4).** A circuit has $n$ wires, including $n_{\text{in}}$ input wires, $n_{\text{out}}$ output wires and $n - n_{\text{in}} - n_{\text{out}}$ intermediate wires. W.l.o.g., the wires are indexed by $1, \ldots, n$. Given the input $x$, the value of the $i$-th wire, denoted by $v_i$, and the output of the circuit are determined as follows. For a boolean circuit, the input is in $\{0,1\}^{n_{\text{in}}}$. For an arithmetic circuit over $\mathcal{R}$, the input is in $\mathcal{R} \in \{0,1\}^{n_{\text{in}}}$.

- For each $i \le n_{\text{in}}$, the $i$-th wire is the $i$-th input wire, so $v_i = x_i$.
- For each $i > n_{\text{in}}$, the $i$-th wire is the output of a gate. Say the gate function is $g$, and the gate takes wires $j_1, \ldots, j_t$ as inputs ($j_1 \le \cdots \le j_t < i$). Then $v_i = g(v_{j_1}, \ldots, v_{j_t})$.
- The output of the circuit is $(v_{n-n_{\text{out}}+1}, \ldots, v_n)$.

**Solve any 4 of Problem 5, 6, 7, 8, 9.**

**Problem 5 (5pt) $(N-1)$-out-of-$N$ Oblivious Transfer** Let $\Pi$ be a secure 2-message 1-out-of-2 oblivious transfer protocol. For simplicity, we focus on semi-honest security in this problem.

In $(N-1)$-out-of-$N$ OT, the sender has $N$ inputs $m_1, \ldots, m_N \in \{0,1\}^\ell$; the receiver chooses an index $i \in [N]$ and receives all $m_j$ s.t. $j \neq i$. The sender should learn nothing about $i$ and the receiver should learn nothing about $m_i$.

**Part A.** Construct a $(2^n - 1)$-out-of-$2^n$ oblivious transfer protocol based on $\Pi$. Your protocol should make only black-box use of $\Pi$, and should not rely on any other assumption.

**Part B.** Say $\Pi$ is a "rate-1" protocol. Its communication cost is highly optimized. If the two messages are $\ell$-bit long, the total communication complexity of $\Pi$ is $\ell + \text{poly}(\lambda)$.

Construct a 2-message $(2^n - 1)$-out-of-$2^n$ oblivious transfer protocol based on $\Pi$. Your protocol should make only black-box use of $\Pi$, and should not rely on any other assumption. When all the $2^n$ messages are $\ell$-bit long, the total communication complexity of your protocol should be at most $2^n \ell + \text{poly}(n, \lambda)$.

**Problem 6 (5pt) $k$-Lin Assumptions** This problem consider a generalization of Diffie-Hellman assumption(s).

A generation algorithm $\mathsf{Gen}(1^\lambda)$ samples a prime $p \le 2^{O(\lambda)}$, a $p$-order group $\mathcal{G}$ and a generator $g \in \mathcal{G}$. We say the sampled group is $k$-Lin hard if

**CLIN = Computational $k$-Lin** For any p.p.t. algorithm $\mathcal{A}$,

$$\Pr_{a_1,\ldots,a_k,x \in \mathbb{Z}_p}\left[\mathcal{A}(p, \mathcal{G}, g, g^{a_1}, g^{a_1 x_1}, \ldots, g^{a_k}, g^{a_k x_k}) = g^{\sum_i x_i}\right] \le \text{negl}(\lambda)$$

**DLIN = Decisional $k$-Lin** For any p.p.t. algorithm $\mathcal{D}$,

$$\begin{aligned}\Pr_{a_1,\ldots,a_k,x \in \mathbb{Z}_p}&\left[\mathcal{D}(p, \mathcal{G}, g, g^{a_1}, g^{a_1 x_1}, \ldots, g^{a_k}, g^{a_k x_k}, g^{\sum_i x_i}) = 1\right] \\ &- \Pr_{a_1,\ldots,a_k,x,u \in \mathbb{Z}_p}\left[\mathcal{D}(p, \mathcal{G}, g, g^{a_1}, g^{a_1 x_1}, \ldots, g^{a_k}, g^{a_k x_k}, g^{u}) = 1\right] \le \text{negl}(\lambda)\end{aligned}$$

Let $k$ be any constant greater than 1. What is the relation between CLIN, DLIN, CDH and DDH? Sort the strength of these four assumptions in order. And prove your answer.

**Problem 7 (5pt) RSA as hard as factorization** Recall the RSA construction. The generation algorithm samples two $\lambda$-bit primes $p, q$, sets $N = pq$, outputs public key $e$, secret key $d$ such that $ed \equiv 1 \mod \phi(N)$.

Prove that the following two problems are as hard as each other.

- *Factorization:* Given $N$ sampled by $\mathsf{Gen}(1^\lambda)$, find $p, q$.
- *RSA Key Recovery Attack:* Given $N, e$ sampled by $\mathsf{Gen}(1^\lambda)$, find $d$.

**Part A.** Practical RSA algorithm usually picks a small public key $e$. So let us assume $e \leq \text{poly}(\lambda)$.

**Part B.** Solve the general case.

Hint: Find a number $x \in \mathbb{Z}_N$ such that $x^2 \equiv 1 \mod N$ but $x \neq \pm 1$.

**Problem 8 (5pt) HSS** In the last lecture, we discussed Homomorphic Secret Sharing (HSS). You task is to evaluate the following candidate HSS construction.

The construction is based on Pailliar encryption. $N = pq$ is the product of two safe primes $p = 2p' + 1, q = 2q' + 1$. Let $\Delta = p'q'$. Quadratic residue modulus $N^2$ has a nice property, there exists a bijection (also isomorphism)

$$\pi : \mathbb{Z}_\Delta \times \mathbb{Z}_N \to \mathbb{QR}_{N^2}, \qquad \pi(s, m) = g^s(1+N)^m,$$

where $g$ is a generator of the hard group.

Homomorphic secret sharing requires the multiplication between additive sharing and encryption. The task can be formalized as follows.

- For simplicity, only consider binary date $x, y \in \{0, 1\}$.
- Alice and Bob jointly hold additve sharing of $[x\Delta]$.
  Alice holds $[x\Delta]_A$, Bob holds $[x\Delta]_B$, such that $[x\Delta]_A = [x\Delta]_B + x\Delta$.
- Both Alice and Bob have ciphertext $c = \mathsf{Enc}(y) = g^r(1+N)^y$.
- They want to locally (= no communication) computes additive sharing of $[xy\Delta]$.
- Recursive multiplication allows Alice and Bob locally compute additive sharing of any NC1 program output.

Evaluate the following HSS multiplication algorithm. Tell if the correctness error is negligible or $\Omega(1)$. Prove your answer.

| Alice side | Bob side |
|---|---|
| • Compute $c_A = c^{[x\Delta]_A}$ | • Compute $c_B = c^{[x\Delta]_B}$ |
| • Compute $d_A = c_A \bmod N$ | • Compute $d_B = c_B \bmod N$ |
| • View $c_A, d_A$ as elements in $\mathbb{Z}_{N^2}$. Let $[xy\Delta]_A = \text{DLog}(1+N, c_A/d_A)$. | • View $c_B, d_B$ as elements in $\mathbb{Z}_{N^2}$. Let $[xy\Delta]_B = \text{DLog}(1+N, c_B/d_B)$. |

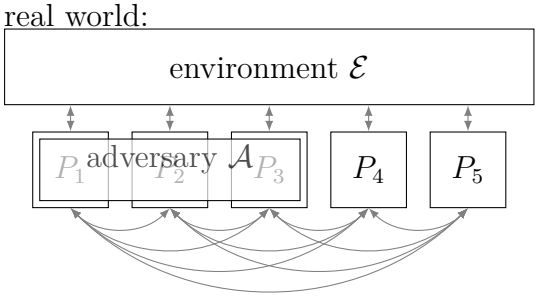
**Problem 9 (5pt) Security with Abort Implies GOD** In this problem, we only consider 2-party functions $f$ that has a unique receiver. That is, only one party (w.l.o.g., the first party $P_1$) receives meaningful output. For any input, the output of the function looks like $f(x_1, x_2) = (y, 0)$.

$\Pi$ is a 2PC protocol for computing $f$. Assume that $\Pi$ is secure with abort. Construct another 2PC protocol $\Pi'$ for computing $f$. $\Pi'$ should be GOD secure. The new protocol should has the same communication cost (number of rounds, message length) as $\Pi$.

You should explicitly state the new protocol $\Pi'$, and the simulator(s) of the new protocol. Explain why your protocol and simulator work.

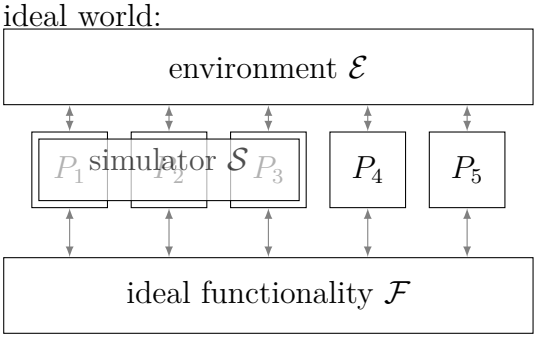
**Malicious Security Definitions** An $n$-party MPC protocol computing a function $f$ is computationally secure against up to $t$ active corruptions, if for any p.p.t. environment $\mathcal{E}$, adversary $\mathcal{A}$, for any set $S \subseteq [n]$ of at most $t$ corrupted parties, there exists a p.p.t. simulator $\mathcal{S}$, such that the view of the environment in the real world and the ideal world are computationally indistinguishable.

real world:




Honest $P_i$ receives input $x_i$ from $\mathcal{E}$, executes the protocol, sends the protocol's output $y_i$ to $\mathcal{E}$.
All corrupted parties are controlled by the adversary. They may interacts arbitrarily with $\mathcal{E}$ and honest parties.

ideal world:




Honest $P_i$ receives input $x_i$ from $\mathcal{E}$, forwards $x_i$ to $\mathcal{F}$, receives $y_i$ from $\mathcal{F}$ sends $y_i$ to $\mathcal{E}$.
All corrupted parties are controlled by the simulator. Each corrupted party $P_i$ sends $x_i$ to $\mathcal{F}$ and receives $y_i$ from $\mathcal{F}$. They may interacts arbitrarily with $\mathcal{E}$.

There are several different level of security definitions, depending on how the ideal functionality is define.

**Full Security = Guaranteed Output Delivery (GOD) Security:** Upon receiving $x_i$ from every party $P_i$, compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, send $y_i$ to $P_i$.

**Security with Abort:** Upon receiving $x_i$ from every party $P_i$, compute $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, send $y_i$ to $P_i$ if $P_i$ is corrupted.

Wait for an signal $b_{\text{abort}} \in \{0, 1\}$ from the adversary (= simulator $\mathcal{S}$). If $b_{\text{abort}} = 1$, send $\perp$ to all honest parties; otherwise send $y_i$ to $P_i$.