Final Exam Review Session

CS 171

April 30, 2024

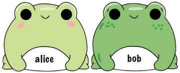


Table of Contents

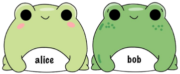
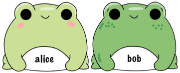


Table of Contents



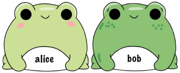
IBE: Syntax

(Similar high-level syntax and properties as other encryption schemes we've seen earlier like SKE/PKE)

- $Setup(1^\lambda) \to (msk, mpk)$.
- $KeyGen(msk, \mathbf{ID}) \to \mathbf{sk_{ID}}$
- $Enc(mpk, \mathbf{ID}, m) \to ct$
- $Dec(\mathbf{sk_{ID}}, ct) \to m$

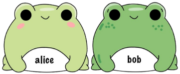
Properties:

- Correctness: $Dec(\mathbf{sk_{ID}}, Enc(mpk, \mathbf{ID}, m)) \to m$
- CPA Security – slightly different game compared to CPA security in SKE/PKE



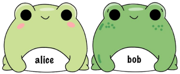
IBE: CPA Security Game

1. Challenger runs $Setup(1^\lambda) \rightarrow (msk, mpk)$ and sends mpk to the adversary.
2. **Keygen Queries: Phase 1** Adversary sends ID to the challenger and gets back $sk_{ID} \leftarrow KeyGen(msk, ID)$ corresponding to the ID.
3. **Challenge phase:** Adversary sends a $ID*$ that was not queried as well as messages $m_0 \neq m_1$.
4. Challenger picks $b \leftarrow \{0, 1\}$ and returns $c_b \leftarrow Enc(mpk, ID^*, m_b)$.
5. **Keygen Queries: Phase 2** Adversary sends ID to the challenger and gets back $sk_{ID} \leftarrow KeyGen(msk, ID)$ corresponding to the ID (ID^* *not allowed*).
6. Adversary outputs a guess b' for b.



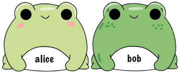
IBE: Tips

- The adversary has the power to choose which ID to use for the challenge phase, unlike in SKE/PKE, where the public key for encryption is fixed at the very beginning.
- KeyGen does what is designed to be hard to do in SKE/PKE – it computes a secret key for an ID given a public key (*How? Using additional secret information msk*).
- For questions: Most reductions will look similar to CPA security of SKE/PKE – make sure the adversaries receive the right answers to queries and that the ciphertext distribution is correct.
- Additional complexity: Need to take care of KeyGen queries.



IBE: Practice problem

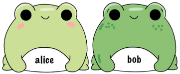
Show that IBE implies PKE, i.e., given a CPA-secure IBE scheme (S, K, E, D), construct a CPA-secure PKE scheme (Gen, Enc, Dec).



IBE: Practice problem

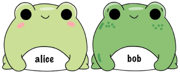
Show that IBE implies PKE, i.e., given a CPA-secure IBE scheme (S, K, E, D), construct a CPA-secure PKE scheme (Gen, Enc, Dec).

- $Gen(1^\lambda)$: Run $S(1^\lambda) \to (msk, mpk)$ and return $sk = msk$, $pk = mpk$.
- $Enc(pk, m)$: Sample a random ID and run $E(mpk, ID, m) \to ct$. Output (ID, ct) as the ciphertext.
- $Dec(sk, (ID, ct))$: First, derive sk_{ID} for the ID and then run $Dec(sk_{ID}, ct) \to m$.



IBE: Practice problem - Properties

Correctness: follows from correctness of IBE.



IBE: Practice problem - Properties

Correctness: follows from correctness of IBE.

CPA security: Suppose PKE was not CPA-secure. Let $\mathcal{A}$ be an adversary that wins in the CPA game for PKE. We'll build an adversary B to break CPA security of IBE.

- IBE challenger runs $S(1^\lambda) \rightarrow (msk, mpk)$ and gives mpk to B. B sends it to A as pk.
- A outputs two challenge messages m_0, m_1.
- B samples a random ID and sends (ID, m_0, m_1) to the IBE challenger.
- The IBE challenger chooses random $b = 0/1$ and returns $c = E(mpk, ID, m_b)$.
- B sends (ID, c) to A and outputs whatever A outputs.

We did not need to make any keygen queries!

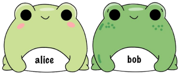
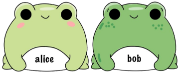


Table of Contents



Groups: Syntax

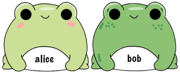
A group G is a set with a binary operation $\cdot$ satisfying the following properties:

Closure $\forall g, h \in G$, we have that $g \cdot h \in G$.

Identity existence $\exists i \in G$ such that $\forall g \in G$, $g \cdot i = g = i \cdot g$.

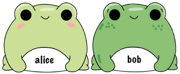
Inverse existence $\forall g \in G$, $\exists h \in G$ such that $g \cdot h = i = h \cdot g$.

Associativity $\forall g_1, g_2, g_3 \in G$, we have that $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$.



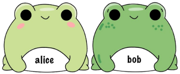
Groups: Properties

1. Let G be a finite group with order m, Then:
    - for any element $g \in G$, we have $g^m = 1$.
    - for any element $g \in G$ and integer x, $g^x = g^{x \bmod m}$.
2. A group G is *cyclic* if $\exists g \in G$ such that $\{g^1, \ldots, g^m\} = G$.
    - If G is a group of prime order p, then G is cyclic and every element except the identity is a generator of G.



The Discrete-Log Problem

1. Let $\mathcal{G}(1^n)$ be a PPT algorithm generating the description of a cyclic group of order q ($q = |G| \approx 2^n$) and a generator g.
2. Note that:
   - We can represent each group element with a unique bit representation of size $\log_2(n)$.
   - The group operation (addition) can be performed in time poly(n).
   - Sampling a group element uniformly at random can be performed in time poly(n) (given randomness).
3. I.e., we can sample a random element $x \in \mathbb{Z}_q$ and compute g^x in time poly(n).



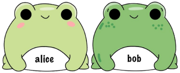
The Discrete-Log Game

$\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain (G, g, q).
2. Sample uniform $h \in G$.
3. $\mathcal{A}$ is given (G, g, q, h) and it outputs x.
4. Output 1 if $g^x = h$ and 0 otherwise.

We say that the Discrete-Log Problem is hard relative to $\mathcal{G}$ if $\forall$ PPT adversaries $\mathcal{A}$, $\exists$ function $\mathsf{negl}(\cdot)$ such that

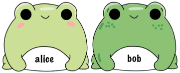
$$|\Pr[\mathrm{DLog}_{\mathcal{A},\mathcal{G}}(n) = 1]| \leq \mathsf{negl}(n).$$



The Diffie-Hellman Problems

Two main forms:

1. *Computational* Diffie-Hellman Problem (CDH): given g^a and g^b, adversary needs to *compute* g^{ab} to win the game.
2. *Decisional* Diffie-Hellman Problem (DDH): given g^a and g^b, adversary needs to *distinguish* g^{ab} from a random group element to win the game.



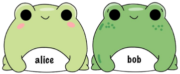
The Computational Diffie-Hellman Game

$\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain (G, g, q).
2. Sample uniform $a, b \in \mathbb{Z}_q^*$.
3. $\mathcal{A}$ is given (G, g, q, g^a, g^b) and it outputs h.
4. Output 1 if $g^{ab} = h$ and 0 otherwise.

We say that the CDH Problem is hard relative to $\mathcal{G}$ if $\forall$ PPT adversaries $\mathcal{A}$, $\exists$ function $\mathrm{negl}(\cdot)$ such that

$$|\Pr[\mathrm{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1]| \leq \mathrm{negl}(n).$$



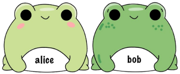
The Decisional Diffie-Hellman Game

$\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain (G, g, q).
2. Sample uniform $a, b, r \in \mathbb{Z}_q^*$. Sample a uniform bit $c \in \{0, 1\}$.
3. $\mathcal{A}$ is given $(G, g, q, g^a, g^b, g^{ab+cr})$ and it outputs c'.
4. Output 1 if $c = c'$ and 0 otherwise.

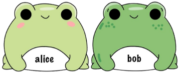
We say that the DDH Problem is hard relative to $\mathcal{G}$ if $\forall$ PPT adversaries $\mathcal{A}$, $\exists$ function $\mathsf{negl}(\cdot)$ such that

$$|\Pr[\mathrm{DDH}_{\mathcal{A},\mathcal{G}}(n) = 1]| \leq \frac{1}{2} + \mathsf{negl}(n).$$



Bilinear Groups

1. "Groups where CDH is hard, but DDH is easy"
2. Consider a group G of prime order q and generator g:
3. We get a pairing operation e such that:
   - $e : G \times G \to G_T$
   - If g is a generator of G then $e(g, g)$ is a generator of G_T
   - $\forall a, b \in \mathbb{Z}_q^*,\ e(g^a, g^b) = e(g, g)^{ab}$
4. Intuition:
   - DDH is easy because if A, B, C is a DDH tuple, we can check $e(A, B) = e(g, C)$
   - CDH is hard because... no attacks are known.



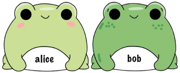
The Decisional *Bilinear* Diffie-Hellman Game

$\mathrm{DBDH}_{\mathcal{A},\mathcal{G}}(n)$

1. Run $\mathcal{G}(1^n)$ to obtain $(G, G_T, g, q, e(\cdot,\cdot))$.
2. Sample uniform $a, b, c, r \in \mathbb{Z}_q^*$. Sample a uniform bit $\beta \in \{0,1\}$.
3. $\mathcal{A}$ is given $(G, G_T, g, q, g^a, g^b, g^c, e(g,g)^{abc+\beta r})$ and it outputs β'.
4. Output 1 if $\beta = \beta'$ and 0 otherwise.

We say that the DBDH Problem is hard relative to $\mathcal{G}$ if $\forall$ PPT adversaries $\mathcal{A}$, $\exists$ function negl($\cdot$) such that

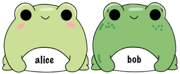
$$|\Pr[\mathrm{DBDH}_{\mathcal{A},\mathcal{G}}(n) = 1] - \frac{1}{2}| \leq \mathsf{negl}(n).$$



Relationships Between (Hard) Problems

From Weakest (Easiest) to Strongest (Hardest):

$$\mathrm{DDH} \implies \mathrm{CDH} \implies \mathrm{DLog} \implies \mathrm{CRHF} \implies \mathrm{OWF}$$
$$\mathrm{CDH} \implies \mathrm{DBDH}$$



Relationships Between (Hard) Problems Continued

$\mathrm{CDH} \implies \mathrm{DLog}$:

1. Want to show that if computing x from g^x in G was easy, then so is computing g^{ab} from g^a and g^b in G.
2. Given (G, g, q, g^a, g^b), run $\mathcal{A}_{\mathrm{Dlog}}$ on g^a to get a. Compute $(g^b)^a = g^{ab}$.
3. This approach wins with the same probability that $\mathcal{A}_{\mathrm{Dlog}}$ solves the Dlog instance (non-negl).

$\mathrm{DDH} \implies \mathrm{CDH}$:

1. Want to show that if computing g^{ab} from g^a and g^b in G was easy, then so is distinguishing DDH triples.
2. Given $(G, g, q, g^a, g^b, g^{ab+cr})$, run $\mathcal{A}_{\mathrm{CDH}}$ on g^a and g^b to get g^{ab} and check if it equals g^{ab+cr}.
3. This approach wins the DDH game with non-negl $- \frac{1}{q} =$ non-negl probability.

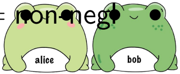
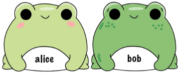


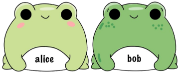
Table of Contents



Signatures: Syntax

- $\textbf{Gen}(1^n)$: Outputs public key and secret key pair (pk, sk).
- $\textbf{Sign}_{sk}(m)$: Outputs a signature σ on the message m.
- $\textbf{Vrfy}_{pk}(m, \sigma)$: Outputs $0/1$.

Correctness: For all n, except for negligible choices of (pk, sk), it holds that for all m, $\textbf{Vrfy}_{pk}(m, \textbf{Sign}_{sk}(m)) = 1$.



Signatures: Unforgeability Security Game

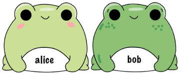
The task of the adversary is essentially to *forge* a valid signature, which successfully verifies, without having the secret key.

$\textbf{Forge}_{A,\Pi}(1^n)$

1. Sample $(pk, sk) \leftarrow \textbf{Gen}(1^n)$.
2. Let (m^*, σ^*) be the output of $\textbf{Sign}_{sk}(\cdot)$ by adversary $A(pk)$. Let M be the list of queries A makes.
3. Output 1 if $\textbf{Vrfy}_{pk}(m^*, \sigma^*) = 1 \wedge m^* \notin M$ and 0 otherwise.

$\Pi = (\textbf{Gen}, \textbf{Sign}, \textbf{Vrfy})$ is existentially unforgeable under adaptive chosen message attack if $\forall$ probabilistic polynomial time (PPT) adversary A, it holds that:

$$\Pr[\text{Forge}_{A,\Pi} = 1] \leq \text{negl}(n)$$

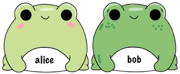


Signatures: Practice Problem, Spring 2021 Final

Let (Gen, Sign, Vrfy) **be a perfectly correct secure digital signature scheme**. Perfect correctness states that for any message m,

$$\Pr_{r_{\mathsf{Gen}}, r_{\mathsf{Sign}} \leftarrow \{0,1\}^n, (vk, sk) := \mathsf{Gen}(1^n; r_{\mathsf{Gen}})} [\mathsf{Vrfy}(vk, m, \mathsf{Sign}(sk, m; r_{\mathsf{Sign}})) = 1] = 1,$$

where r_{Gen} are the random coins used by Gen and r_{Sign} are the random coins used by Sign. **Define** $f(x)$ to output the verification key vk output by $\mathsf{Gen}(1^n; x)$. **Show that** f is a one-way function.



Signatures: Practice Problem Solution

If there exists a probabilistic polynomial time (PPT) A that can invert f with non-negligible probability, then we can construct a PPT B that breaks the security of the signature scheme:

1. B gets pk from its challenger and forwards it to A.
2. A outputs x' such that $f(x') = pk$.
3. B computes $(pk, sk') := \text{Gen}(1^n; x')$.
4. B picks an arbitrary message m and computes $\sigma \leftarrow \text{Sign}_{sk'}(m)$.
5. Since (pk, sk') is generated from Gen, σ is a valid signature for m with respect to pk. Hence B breaks the security of the signature scheme with non-negligible probability.

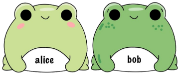
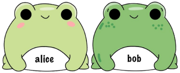


Table of Contents



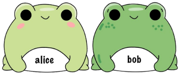
Commitment Scheme Syntax

1. $\mathsf{Gen}(1^n) \to \mathsf{params}$

2. $\mathsf{Commit}(\mathsf{params}, m; r) = \mathsf{com}$
   - $\mathcal{M}$ is the message space, and $m \in \mathcal{M}$.
   - Other notation: $\mathsf{Commit}(\mathsf{params}, m) \to \mathsf{com}$

3. Open: Committer publishes m and proves that com is a commitment to m. The verifier decides whether to accept or reject the proof.

   Canonical Opening Procedure:
   - Committer publishes (m, r).
   - Verifier checks whether $\mathsf{com} = \mathsf{Commit}(\mathsf{params}, m; r)$. If so, they accept; if not, they reject.



Hiding Definition

The definition of hiding resembles CPA security.

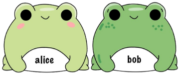
Hiding-Game$(n, \mathcal{A})$:

1. The challenger samples params $\leftarrow$ **Gen**(1^n) and sends params to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ outputs two messages $m_0, m_1 \in \mathcal{M}$.
3. The challenger samples $b \leftarrow \{0, 1\}$ and computes:

$$\text{com}^* \leftarrow \text{Commit}(\text{params}, m_b)$$

   They send com^* to $\mathcal{A}$.

4. $\mathcal{A}$ outputs a guess b' for b. The output of the game is 1 if $b' = b$ and 0 otherwise.



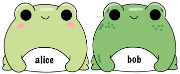
Hiding Definition

The commitment scheme is **computationally hiding** (a.k.a. **hiding**) if for any PPT adversary $\mathcal{A}$,

$$\Pr[\text{Hiding-Game}(n, \mathcal{A}) \to 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

The commitment scheme is **statistically hiding** if for any adversary $\mathcal{A}$ *running in unbounded time*,

$$\Pr[\text{Hiding-Game}(n, \mathcal{A}) \to 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$



Binding Definition

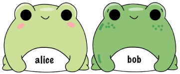
The definition of binding resembles collision-resistance.

Binding-Game$(n, \mathcal{A})$:

1. The challenger samples params $\leftarrow$ **Gen**(1^n) and sends params to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ outputs two pairs (m_0, r_0) and (m_1, r_1), where $m_0, m_1 \in \mathcal{M}$.
3. The output of the game is 1 if $m_0 \neq m_1$, and

$$\text{Commit}(\text{params}, m_0; r_0) = \text{Commit}(\text{params}, m_1; r_1)$$

Otherwise, the output of the game is 0.



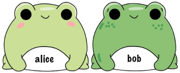
Binding Definition

The commitment scheme satisfies **computational binding** (a.k.a. **binding**) if for any PPT adversary $\mathcal{A}$,

$$\Pr[\text{Binding-Game}(n, \mathcal{A}) \to 1] \leq \text{negl}(n)$$

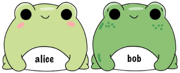
The commitment scheme satisfies **statistical binding** if for any adversary $\mathcal{A}$ *running in unbounded time*,

$$\Pr[\text{Binding-Game}(n, \mathcal{A}) \to 1] \leq \text{negl}(n)$$



Notes

- By default, "hiding" refers to computational hiding, and "binding" refers to computational binding.
- No commitment scheme can be both statistically hiding and statistically binding.



Commitment Scheme Practice Problem[1]

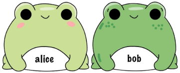
The following construction uses a PRG to construct a commitment scheme.

Let $G : \{0,1\}^n \rightarrow \{0,1\}^{3n}$ be a PRG. Let $m \in \{0,1\} = \mathcal{M}$.

1. $\text{Gen}(1^n)$: Sample $s \leftarrow \{0,1\}^{3n}$ and output params $= s$.
2. $\text{Commit}(\text{params}, m; r)$: Let $r \leftarrow \{0,1\}^n$. Compute

$$\text{com} = G(r) \oplus (m \cdot s)$$

Prove that this construction satisfies computational hiding and statistical binding.



[1] Adapted from the fall 2019 final exam, question 2.2.

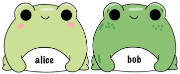
Commitment Scheme Practice Problem: Hiding

Theorem

The scheme is computationally hiding.

Proof:

1. <u>Intuition:</u> This follows from the PRG security of G.
2. <u>Overview:</u> Assume toward contradiction that there exists a PPT adversary $\mathcal{A}$ that can break hiding. Then we will use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the PRG security of G. This is a contradiction because $\mathcal{B}$ is a secure PRG. Therefore, there is not actually a PPT adversary $\mathcal{A}$ that can break hiding, so the commitment scheme is computationally hiding.



Commitment Scheme Practice Problem: Hiding

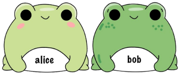
Construction of $\mathcal{B}$:

1.
    1. *Pseudorandom Case:* The PRG challenger samples $r \leftarrow \{0,1\}^n$ and sends $g = G(r)$ to $\mathcal{B}$.
    2. *Truly Random Case:* The PRG challenger samples $g \leftarrow \{0,1\}^{3n}$ and sends g to $\mathcal{B}$.
2. $\mathcal{B}$ chooses $m_0 = 0$ and $m_1 = 1$ and then samples $b \leftarrow \{0,1\}$.
3. $\mathcal{B}$ computes

$$\text{com}^* = g \oplus (m_b \cdot s)$$

   and sends com^* to $\mathcal{A}$.

4. $\mathcal{A}$ outputs a guess b' for b. $\mathcal{B}$ checks whether $b = b'$. If so, $\mathcal{B}$ outputs 0. If not, $\mathcal{B}$ outputs 1.



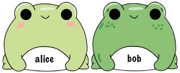
Commitment Scheme Practice Problem: Hiding

1. *Pseudorandom Case:* If $g = G(r)$ for some random $r \leftarrow \{0,1\}^n$, then $\mathcal{B}$ simulates the hiding security game for the commitment scheme. In this case,

$$\Pr[b = b'] = \Pr[\text{Hiding-Game}(n, \mathcal{A}) \to 1] \geq \frac{1}{2} + \text{non-negl}(n)$$

2. *Truly Random Case:* If $g \leftarrow \{0,1\}^{3n}$, then com^* is independent of b. com^* is basically a one-time pad ciphertext. In this case:

$$\Pr[b = b'] = \frac{1}{2}$$

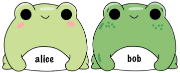


Commitment Scheme Practice Problem: Hiding

In summary, $\mathcal{B}$ breaks the PRG security of G because:

$$\begin{aligned}\Pr[\mathcal{B} \to 0 | \text{Pseudorandom Case}] - \Pr[\mathcal{B} \to 0 | \text{Truly Random Case}] \\ &\geq \frac{1}{2} + \text{non-negl}(n) - \frac{1}{2} \\ &\geq \text{non-negl}(n)\end{aligned}$$

Q.E.D.



Commitment Scheme Practice Problem: Binding

Theorem

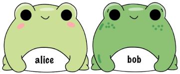
The scheme is statistically binding.

Proof:

1. If the adversary can break binding, then they can find two openings $(0, r_0)$ and $(1, r_1)$ such that

$$G(r_0) = G(r_1) \oplus s$$

2. This is only possible if there exist values $(r_0, r_1) \in \{0,1\}^n \times \{0,1\}^n$ such that $G(r_0) \oplus G(r_1) = s$.



Commitment Scheme Practice Problem: Binding

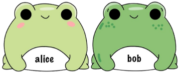
1. Let T be the set of all the values that $G(r_0) \oplus G(r_1)$ can take:

$$T = \{t \in \{0,1\}^{3n} : \exists (r_0, r_1) \in \{0,1\}^n \times \{0,1\}^n \text{ s.t. } t = G(r_0) \oplus G(r_1)\}$$

2. $|T| \leq 2^{2n}$ because there are at most 2^{2n} values of (r_0, r_1).
3. Finally, s is sampled uniformly at random from $\{0,1\}^{3n}$. Therefore,

$$\Pr[s \in T] = \frac{|T|}{2^{3n}} \leq \frac{2^{2n}}{2^{3n}} = 2^{-n} = \mathrm{negl}(n)$$

4. If $s \notin T$, then no adversary, even a computationally unbounded one, can break binding.



Commitment Scheme Practice Problem: Binding

Over the randomness of s, the probability that a computationally unbounded adversary can break binding is $\leq 2^{-n} = \text{negl}(n)$. Therefore, the commitment scheme satisfies statistical binding.

Q.E.D.

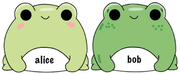
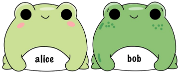
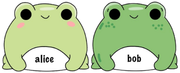


Table of Contents



Secret Sharing: Concept

- A (t, n) threshold secret sharing scheme allows one to split a secret s into n pieces so that one will need at least t shares to reconstruct s.
- A dealer takes s as input and uses a sharing algorithm to split the secret s into parts $s_1, \ldots, s_n$ to be given to parties $P_1, \ldots, P_n$.
- **Correctness:** Any t parties can reconstruct s.
- **Security:** No collusion of $< t$ parties can reconstruct s.

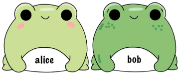


Secret-Sharing: Definition

A (t, n)-secret sharing scheme (**Share**, **Reconstruct**) is defined as follows.

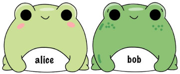
- **Share**(s): On input a secret s it outputs shares $s_1, \ldots, s_n$.
- **Reconstruct**$(\{s_i\}_{i \in T})$: Outputs s or $\perp$.
- **Correctness**: For any T such that $|T| \geq t$ and secret s we have that **Reconstruct**$(\{s_i\}_{i \in T}) = s$.
- **Security**: For any T such that $|T| < t$, secrets s, s' and adversary A we have that $p = p'$ where

$$p = \Pr[A(\{s_i\}_{i \in T}) = 1 \mid (s_1, \ldots, s_n) \leftarrow \textbf{Share}(s)],$$
$$p' = \Pr[A(\{s'_i\}_{i \in T}) = 1 \mid (s'_1, \ldots, s'_n) \leftarrow \textbf{Share}(s')].$$



Secret-Sharing: Practice Problem

How can you secret-share among n parties and reconstruct using only a threshold t of n?



Main Idea: Remember polynomial interpolation from CS 70? This is literally that. To share $s \in \mathbb{Z}_q$: choose a random degree $t-1$ polynomial $p(x)$ such that $p(0) = s$. Give out the shares $(p(1), \ldots, p(n))$.

- Given t shares, we can reconstruct $p(x)$, and can then recover $p(0)$.

Sharing:

- Given a secret $s \in \mathbb{Z}_q$, choose $p(x) = s + a_1 x + \cdots + a_{t-1} x^{t-1}$, where a_i's are chosen randomly in $\mathbb{Z}_q$. Give out the shares $(p(1), \ldots, p(n))$.

Reconstruct:

- Given t values $(i_1, p(i_1)), \ldots, (i_t, p(i_t))$, reconstruct p and output $p(0)$.

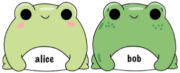
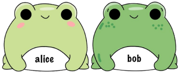


Table of Contents



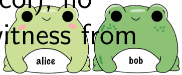
Proof systems: Syntax

A proof system is an interactive protocol between a Prover and Verifier. Prover wants to convince Verifier of the truth of some statement.

- Prover has access to the instance x and witness w such that $C(x, w) = 1$.
- Verifier only has the instance x and outputs $0/1$ at the end of the interaction depending on if it is convinced by the prover.

Three main properties:

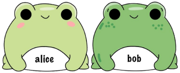
- **Completeness**: If Prover is honest, Verifier always (or with overwhelming probability) outputs 1.
- **Soundness**: If Prover is cheating (i.e., the statement is actually false and no witness exists), Verifier must output 1 only with negligible probability.
- **Zero-Knowledge**: If Prover is honest (follows the protocol), no (cheating) Verifier can gain any information about the witness from the interaction.



Proof systems: Properties and Tips

Soundness: Cheating prover vs Honest verifier

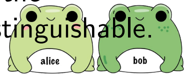
- *Building sound protocols*: Most protocols usually have a randomized step where the verifier sends a random element. Honest provers will always be able to answer for any random element, but a cheating prover will only be able to answer for a very small (read negligible) set of random values – has to hope that the verifier chooses one of those values at random.
- *General proof structure* (to prove soundness): Suppose the statement is false and the verifier accepts the proof (outputs 1) with non-negligible probability. Then, break some assumption / show that the statement is true – which is a contradiction – hence the verifier cannot accept the proof with non-negligible probability. QED.



Proof systems: Zero-Knowledge

Zero-Knowledge: Honest prover vs Cheating verifier

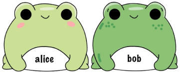
- *Definition*: $\exists Sim$ such that for all V^* and honest prover $P(x, w)$, the view of the verifier in the interaction with $P(x, w)$ and the output of $Sim^{V^*}(x)$ are indistinguishable to any PPT distinguisher.
  - What the verifier sees in a honest interaction can be simulated without knowing the witness, hence contains "zero knowledge" about the witness.
- *Building ZK protocols*: What the verifier sees should not contain any information about the witness – all messages should be blinded with some randomness.
- *General proof structure*: Construct a simulator that generates a transcript of the interaction without the witness. Can run V^* multiple times, can sample things out of order. Then, show that the distributions are either identical or computationally indistinguishable.



Proof systems: Practice problem

Q: Come up with a ZKP for Quadratic Residuosity: Consider a modulus m and a w such that $x = w^2 \mod m$. The instance is (x, m) and the witness is the square root of $x \mod m$.

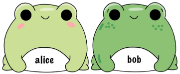
Hint: This is also a three round protocol similar to other protocols you have seen. We only want soundness $1/2$ *– we can use soundness amplification to make it negligible.*



Proof systems: Practice problem - Construction

Construction:

1. The prover samples a random $r \in \mathbb{Z}$ and sends $a = r^2 \mod m$ to the verifier.
2. The verifier samples a random bit $b \leftarrow \{0, 1\}$ and sends it.
3. The prover sends $z = w^b \cdot r \mod m$ to the verifer.
4. Verifier accepts if $z^2 = x^b a \mod m$.



Proof systems: Practice problem - Properties

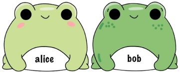
Correctness:

$$z^2 = w^{2b} r^2 = x^b a \mod m$$

Soundness: Suppose there does not exist a square root of x. For the prover to succeed with probability $> 1/2$, the prover should be able to pass the check for both $b = 0$ and $b = 1$ for some choice of first message a. If both checks pass, notice that

$$\begin{aligned} z_1^2 &= a \mod m \\ z_2^2 &= xa \mod m \\ \implies \left(\frac{z_2}{z_1}\right)^2 &= x \mod m \end{aligned}$$

which is a contradiction.



Zero-Knowledge: Idea = Prover can always answer correctly if they know what bit the verifier would pick before they send the first message.
The simulator works like that of Graph Isomorphism (Disc 11).

1. *Sim* samples a random bit b', samples a random $z \mod m$ and computes $a = \frac{z^2}{x^{b'}}$ as the first message.
2. *Sim* runs V^* with a as the first message. If the second message from V^* is the same as b', send z in the third step. Else go to step 1 and start over.

In expectation, *Sim* will need two tries to succeed as V^*'s view is independent of b' after the first message.