Midterm Review

April 10 2026

Cryptographic primitives

- What is a one-way function (OWF)?
  - Hard to invert: sample x, probability of f(x) = f(x') is negligible for any adversary A(f(x))=x'
- What is a collision-resistant function (CR)?
  - Hard to find collisions: probability of f(x)=f(x') is negligible for any adversary $A(1^{\lambda})=(x, x')$
- What is a pseudo-random function (PRF)?
  - A **keyed** function that looks like a truly random function to anyone without the secret key
  - Sample k, PRFAdv(A,F) is negligible for any adversary A who tries to distinguish F from R
- What is a pseudo-random permutation (PRP)?
  - A **keyed** function that behaves like a random permutation
  - Every input maps to a unique output and is reversible (i.e., a permutation)
  - PRPAdv is defined similarly to PRFAdv

Symmetric encryption

- What is CPA security?
  - $(Enc_\lambda, Dec_\lambda)$ is CPA-secure, if $\forall$ A, $\exists$ negligible $\mu(\cdot)$ s.t., $\forall$ λ, A wins the following game with probability at most ½ + $\mu(\lambda)$:
    - Challenger chooses key k; A **chooses** m and receives Enc(k, m) for polynomial times
    - A **chooses** m_0, m_1; Challenger chooses a random bit
    - A receives Enc(k, m_b) and output b'; A wins if b=b'
- CPA-secure symmetric encryption:
  - F or P: $K \times \{0,1\}^n \rightarrow \{0,1\}^n$, m is nl bits: $m_1||m_2||...||m_l$
  - CTR from PRFs: $c_i = F(k, r+i-1) \oplus m_i$ nonce r should never be reused
  - CBC from PRPs: $c_i = P(k, c_{i-1} \oplus m_i)$ c_0 must be freshly randomly sampled

Diffie Hellman and public key encryption

- How does Diffie Hellman key exchange work?
  - Alice sends g^a mod p. Bob sends g^b mod p. Since (g^a)^b = (g^b)^a mod p, both Alice and Bob end up with the same key
- What is the computational assumption?
  - The discrete log assumption states that it is computational infeasible to compute a, given g^a mod p
- How does the ElGamel encryption scheme work?
  - The public key is y = g^x, and we encrypt by outputting g^k, m*y^k for a random k. Decryption computes m*y^k * (g^k)^-x mod p
- What are the security properties?
  - ElGamel IND-CPA secure, but not IND-CCA secure. This means that an attacker can modify the message (e.g multiply by 2).

Rabin and RSA

- How does RSA work?
  - Two primes p and q are secret factors of a public modulus n. Encryption exponentiates by a public exponent e mod n, and decryption exponentiates by a secret d such that ed = 1 mod totient(n)
- What is the security assumption for RSA?
  - While factoring breaks RSA, formally RSA depends on the RSA assumption
- What is the advantage/disadvantage of Rabin's scheme over RSA?
  - Rabin's scheme is provably secure from the factoring assumption, but outputs multiple answers to decryptions

Learning with Errors

- What is the learning with errors assumption?
  - The learning with errors assumption states that for a matrix A, secret s, and error e, sA + e is computationally indistinguishable from U, the uniformly random distribution
- What are the properties of A, s, and e?
  - A is a public matrix, meaning it can be viewed by the adversary. The m dimension of A represent LWE samples, while the n dimension provides security. Therefore more columns gives more information to the adversary, while more rows provide more mixing.
  - s is the secret key. Larger secrets are more secure
  - e is the error, and is chosen from some error distribution specified as part of the assumption. It should be relatively small for the LWE assumption to be useful

Encryption from LWE

- How do we construct symmetric key encryption with LWE?
  - Since sA+e is computationally indistinguishable from random, we can use it as a one time pad. We add the message m to sA+e, and we multiply m by a large constant so that we can differentiate it from the error
- What is the correctness condition for decryption?
  - The error is bounded by [-B, B] and numbers are in the range [-q/2, q/2]. So let B = (q/2)/2, and we have q > 4B
- How do we construct public key encryption from LWE?
  - We publish many LWE ciphertexts, and the user selects a random combination to hide their message. This is because for IND-CPA security, the scheme must be randomizable where another user would not get the same result -- or else the adversary could compute it too.

PIR

- What is PIR?
  - PIR (private information retrieval) is a protocol to retrieve an element of a remote database without revealing which
- How does PIR work?
  - A client generates an encrypted query
  - The server uses some homorphic operations to evaluate the query. LWE is one way to implement this efficiently
  - The client decrypts the result

GSW

- How does homorphic addition work?
  - We add the two matrices elementwise
- What is the error growth?
  - The errors sum together
- How does homomorphic multiplication work?
  - We apply h, which decomposes each entry into binary, and then matrix multiply
- What is the error growth?
  - The error growth multiplicatively by a factor l = n log q, which is the dimension of h
- How does error growth affect usability?
  - Since the error grows multiplicatively, the depth of circuits is limited

FHE

- How does bootstrapping work?
  - We evaluate a decryption circuit (where s is hardcoded) using FHE
- Why is it useful?
  - Bootstrapping produces a new ciphertext with less error, allowing unbounded computation
- What assumptions does it require?
  - Bootstrapping requires the circular security assumption, which means we encrypt the secret key s in an encryption scheme based on s. This is not true in general for IND-CPA schemes.

Interactive and zero knowledge proofs

- How does an interactive proof work?
  - A prover wants to show that a statement is true to a verifier. The verifier makes some challenges that the prover answers.
  - In a doubly efficient proof, both the prover and verifier run in polynomial time
- What are the security properties?
  - Completeness means that if the statement is true and the Prover is honest, the Verifier will be convinced and accept the proof with significant probability.
  - Soundness means that if the statement is false,a cheating Prover should not be able to convince the Verifier to accept he proof, except with a very small probability.
- What is zero knowledge?
  - The verifier learns nothing (other than that the proof validity).

Shamir’s secret sharing

- What is t-out-of-n secret sharing?
  - n parties; any t parties can reconstruct the secret together, but not for (t-1) parties
- How does Shamir’s t-out-of-n secret sharing work?
  - Message m in GF[p]
  - Randomly sample a degree-(t-1) polynomial f where the constant is the message, and each non-constant coefficient is random from GF[p]
  - n shares are f(1), f(2), …, f(n)
- Why is it secure and why is it correct?
  - Let A be the vector of the non-constant coefficients, which is a random vector
  - Security: A tuple of (t-1) shares is constant + a bijective map of A
  - Correctness: We can solve the t unknowns from a tuple of t shares
  - All coefficients, not just the secret m, are reconstructed

MPC

- How does BGW work?
  - Adversary controls t_{adv} parties: we need $t=(t_{adv}+1)$-out-of-n secret sharing
  - For a secret value a, there is an underlying (t-1)-degree polynomial g_a and party i has $g_a(i)$
  - Addition between a and b: party i computes $g_a(i)+g_b(i)$, this is the share of (a+b) for an underlying polynomial of (t-1) degrees - no extra work needed
  - Multiplication between a and public c: party i computes $cg_a(i)$, this is the share of ca for an underlying polynomial of (t-1) degrees - no extra work needed
  - Multiplication between a and b: party i computes $g_a(i) * g_b(i)$, this is the share of (ab) for an underlying polynomial of 2(t-1) degrees: we need degree reduction
    - A 2(t-1)-degree polynomial is decided by (2t-1) points, we can have (2t-1) parties re-share their local product with fresh degree-(t-1) polynomials, and use public coefficients to do a linear recombination of those polynomials into a (t-1)-deg polynomial whose constant term is the product: O(n^2) communications
    - n >= 2t-1

Sumcheck

- What is the setting in the sumcheck protocol?
  - Prover: claims that $\beta=\Sigma_{h1,h2,...,hm\in H} f(h_1, h_2, \ldots, h_m)$
  - Verifier: can use randomness and have oracle access to f
  - Goal: the verifier can verify the statement w/o computing f for all $|H|^m$ variable combinations
- What is the sumcheck protocol?
  - Prover sends $g_1(x)$, the univariate polynomial (of degree d) by summing over all variables over H^{m-1} except x_1; if prover is honest, summing g_1 over H gives β
  - Verifier picks t_1 and prover sends $g_2(x)$, the univariate polynomial (of degree d) by summing over all variables over H^{m-2} while fixing $x_1=t_1$ (the only variable is now x_2); if the prover is honest, summing g_2 over H gives $g_1(t_1)$: forces **consistency** with previous round
  - ……
  - Verifier picks t_{m-1} and prover sends $g_m(x)=f(t_1,t_2,...,t_{m-1},x)$, verifier checks if $\Sigma_{x\in H} g_m(x)=g_{m-1}(t_{m-1})$
  - Final check: verifier samples t_m and checks if $g_m(t_m)=f(t_1,t_2,...,t_{m-1},t_m)$
- Completeness/soundness