CS6290 Course Review

Final Project Due Tonight (Apr 27, 2025)

Please submit:

- The project report
- Other deliveries (if you have): source codes, prototype systems, etc.

Final Examination

(Please refer to ARRO's final notice)

- **Venue:** AC-216
- **Time:** 9:30 – 11:30 AM, May 14, 2025
- **Materials and Aids Permitted:** 1 sheet of notes
    - A4 size
    - Double-sided

Agenda

- ~2h course revision, covering the key concepts we've learned:
  - Module 1: Decentralization, Consensus & Blockchain
  - Module 2: Privacy-Preserving Computation Techniques
  - Module 3: Web Privacy: Tracking Mechanisms & Countermeasures
  - Module 4: Data Privacy: Anonymization vs. Differential Privacy
  - Module 5: Privacy & Security in Machine Learning
- ~1h free Q&A (no tutorial session today)

Warning

- This review-session only highlights a few key concepts
- You should thoroughly review all lecture and tutorial materials (the final exam won't cover concepts in Lecture 4: Ethereum Security)

Module 1: Module 1: Decentralization, Consensus & Blockchain

Let's start with the basics: why build systems without a central authority, and what challenges does this create?

What is a Decentralized System?

- **Centralized:** Controlled by a single entity (e.g., a bank's database, a company server).
  - Pros: Efficiency, clear control.
  - Cons: Single point of failure, censorship risk, potential for unilateral control/abuse.
- **Decentralized:** Control and data are distributed among multiple participants. No single entity has full control.
  - Pros: Increased resilience, censorship resistance, transparency (potentially).
  - Cons: Coordination challenges, potentially slower performance.
- **Examples:** Bitcoin, Ethereum, distributed file storage systems.

The Core Challenge: Reaching Agreement (Consensus)

- In a decentralized system with a shared ledger (a common record of events/transactions), how do participants agree on:
    - Which new transactions/updates are valid?
    - The order in which they occurred?
- Without a central authority to dictate the "truth", we need a **Consensus Mechanism**.

Consensus Mechanisms: The Engine of Agreement

- **Goal:** Allow independent nodes to propose and agree on the next valid block/update to the shared ledger, ensuring everyone eventually has the same, consistent view.
- **Key Examples:**
    - **Proof-of-Work (PoW):** Nodes (miners) compete to solve a computationally hard puzzle. The winner proposes the next block.
        - *Principle:* Making block creation costly deters attackers. Requires significant energy.
    - **Proof-of-Stake (PoS):** Nodes (validators) are chosen to propose blocks based on the amount of cryptocurrency they "stake" (lock up) as collateral.
        - *Principle:* Economic incentive – validators lose stake if they cheat. More energy-efficient.

Maintaining Consistency: Forks & Resolution

- **What is a Fork?** When two or more valid blocks are proposed at roughly the same time, creating temporary diverging versions of the history.
- **Why is Resolution Needed?** To ensure the entire network eventually agrees on a single history.
- **Nakamoto Consensus Resolution (PoW): The Longest Chain Rule**
  - Nodes always try to extend the chain with the most **cumulative proof-of-work** (typically the longest one).
  - Over time, one chain attracts more work and outpaces others.
  - Shorter, abandoned forks become "orphaned".
- **Result: Eventual Consistency** – the network converges.

Real-World Consideration: Scalability

- **Scalability:** A system's ability to handle increasing amounts of work (e.g., more users, more transactions).
- **Challenge in Blockchains:** Processing a high volume of transactions quickly can be difficult.
- **Bottlenecks Examples:**
    - **Block Generation Time:** A fixed interval limits how often new transactions are confirmed (e.g., Bitcoin ≈ 10 mins).
    - **Block Size Limit:** Restricts how many transactions fit into each block.
    - **Network Latency:** Time for blocks/transactions to spread across the network.
- **Impact:** Limits **throughput** (transactions per second). Improving scalability is a major area of blockchain research (e.g., Layer 2, PoS variations, Sharding).

Beyond Simple Ledgers: Smart Contracts

- **Concept:** Self-executing contracts with the terms of the agreement directly written into code. They run on a blockchain network.
- **Functionality:**
  - Automate processes when predefined conditions are met.
  - Enforce rules transparently.
  - Examples: Decentralized finance (DeFi), supply chain tracking, automated verification checks.
- **Execution Cost (e.g., "Gas" in Ethereum):**
  - Users pay fees to execute transactions or smart contract functions.
  - Compensates node operators for computational resources.
  - Prevents network spam and resource exhaustion.

Module 1 Recap: Key Concepts

- Decentralization avoids single points of failure but requires **consensus**.
- **PoW** (computation) and **PoS** (stake) are major consensus approaches.
- **Fork resolution** (e.g., longest chain rule) ensures eventual consistency.
- **Scalability** (throughput) is a critical challenge.
- **Smart contracts** enable automated, programmable logic on blockchains.
- **Gas/Fees** manage resource usage on public platforms.

Module 2: Privacy-Preserving Computation Techniques

How can multiple parties collaborate using their data without revealing it to each other?

The Need for Privacy-Preserving Computation

- **Problem:** Often, valuable insights can be gained by combining data from different sources (e.g., hospitals, companies, individuals).
- **Constraint:** This data is often sensitive and cannot be shared directly due to privacy regulations, business confidentiality, or personal concerns.
- **Goal:** Enable computation on combined data **without revealing the private inputs**.
- **Key Technologies:** Secure Multi-Party Computation (MPC) and Zero-Knowledge Proofs (ZKP).

What is Secure Multi-Party Computation (MPC)?

Goal: Compute a function on multiple parties' private data *without* them revealing their data to each other.

Example: Alice has `X`, Bob has `Y`. They want to compute `f(X, Y)` (e.g., `X + Y`, `Avg(X,Y)`) without Alice learning `Y` or Bob learning `X`.

Core Idea: Transform the data and the computation so no single party sees another's raw input.

How does MPC work? (Conceptual Process)

1. **Input Sharing/Transformation:**
    - Parties process their private inputs using techniques like:
        - **Secret Sharing:** Input is split into "shares", distributed among parties. No single share reveals the input.
        - **Homomorphic Encryption:** Input is encrypted. Computations can be done directly on the encrypted data.
    - *Key Point: Raw private data is never directly exchanged.*

How does MPC work? (Conceptual Process)

2. **Secure Computation:**
    - Parties jointly perform calculations on the *transformed* (shared or encrypted) data following a specific protocol.
    - The protocol ensures calculations are correct without revealing intermediate values linked to original inputs.
3. **Output Reconstruction:**
    - Parties combine their results (or decrypt the final encrypted result) to get the final output of `f(X, Y)`.
    - Ideally, only the final result is learned.

MPC: Key Goal Achieved

- **Input Privacy:** Throughout the process (sharing, computation, reconstruction), no party learns the raw private inputs of others.
- **Correctness:** The protocol ensures the final output is the same as if the computation were done centrally with all raw data (assuming no cheating/failures, which protocols also address).

What are Zero-Knowledge Proofs (ZKP)?

Goal: A **Prover** wants to convince a **Verifier** that a statement is true, *without* revealing *why* it's true or any secret information related to the statement.

Example: Alice (Prover) wants to prove to Bob (Verifier) that she knows the solution to a puzzle, without showing Bob the solution itself.

Core Idea: Use a protocol involving challenges and responses that demonstrate knowledge or property adherence probabilistically or definitively, without leaking the core secret.

How do ZKPs work? (Conceptual Process)

1. **Commitment/Claim:** The Prover makes a claim or commits to possessing certain knowledge (the "witness").
2. **Challenge-Response (often iterative):**
    - The Verifier issues challenges based on the claim.
    - The Prover provides responses based on their secret knowledge (witness) and the challenge.
    - *Crucially, the responses don't reveal the witness itself.*
3. **Verification:** The Verifier checks if the Prover's responses are consistent with the claim according to the protocol rules. If they are (after enough rounds, if iterative), the Verifier is convinced.

Key Properties of ZKPs

- **Completeness:** If the Prover's statement *is true* (and they have the witness), they can always convince an honest Verifier.
- **Soundness:** If the Prover's statement *is false* (or they lack the witness), they have only a negligible chance of fooling an honest Verifier.
- **Zero-Knowledge:** The Verifier learns *nothing* beyond the fact that the statement is true. The proof reveals no information about the secret witness itself.

Example Application: Combining MPC & ZKP: Scenario

Why combine them?

- MPC protects inputs during computation, but assumes inputs are *correctly formed* or *honestly provided*.
- ZKP can *verify* properties of inputs *before* they enter the MPC, without revealing the inputs.

Scenario: Multiple hospitals want to compute average patient recovery time (MPC) but need to ensure each hospital:
* Only includes valid patient records (e.g., times are positive numbers).
* Submits data consistent with a prior commitment (e.g., number of patients).

MPC: Considerations & Limitations

- **Input Privacy vs. Output Privacy:**
  - MPC primarily protects **inputs**.
  - The **output** itself is revealed and might leak information.
  - *Example:* If only two parties compute an average, knowing the average and your own input reveals the other party's input.
- **Mitigation Strategies:**
  - Use techniques like Differential Privacy on the output.
  - Agree only on aggregated/binned results instead of precise values.
  - Ensure enough participants to dilute information leakage.

ZKP: Considerations & Limitations

- **Proof Scope:**
  - A ZKP proves *exactly* the statement formulated.
  - It doesn't inherently prove related things *not* explicitly included in the statement.
  - *Example:* Proving your list contains valid entries doesn't prove you included *all* relevant entries (omission is hard to prove directly).
- **Mitigation Strategies:**
  - Careful protocol design: Define precisely what needs proving.
  - External/Procedural Controls: Use pre-commitments (e.g., publicly declare dataset size before proving properties).

General Considerations (Both MPC & ZKP)

- **Complexity:** Implementing and running these protocols can be complex.
- **Performance:** Can be computationally intensive and require significant communication overhead compared to traditional methods. Performance depends heavily on the specific protocol choice and scale.
- **Assumptions:** Different protocols rely on different cryptographic assumptions (e.g., hardness of factoring, discrete logarithms).

Module 2 Recap: Key Concepts

- **MPC** enables joint computation on private inputs, protecting the inputs themselves.
- **ZKP** allows proving statements true without revealing underlying information, ensuring integrity.

Module 3: Web Privacy: Tracking Mechanisms & Countermeasures

How does our activity get tracked online, and what can be done about it?

The Basic Web Interaction

- **Client-Server Model:** Your **browser** (client) requests web pages and resources from **web servers**.
- **Domains:** Websites are identified by domain names (e.g., `example.com`). Resources (images, scripts) can be loaded from the *same* domain or *different* domains.
- **State Management:** HTTP (the web protocol) is stateless. **Cookies** were invented to help websites remember information about users across requests (e.g., login status, preferences).

Cookies: The Core Tracking Mechanism

- **What are they?** Small text files stored by your browser, associated with a specific domain. The browser sends the cookie back to the domain on subsequent requests.
- **First-Party Cookie:** Set by the domain you are **currently visiting** (e.g., `news.com` sets a cookie while you are browsing `news.com`).
  - *Purpose:* Site functionality (login, cart, settings). Generally expected/needed.
- **Third-Party Cookie:** Set by a domain **different** from the one you are visiting (e.g., `adnetwork.com` sets a cookie via an ad shown on `news.com`).
  - *Purpose:* Cross-site tracking, ad targeting, analytics across different websites.

How Third-Party Cookies Enable Cross-Site Tracking

1. **Visit Site A:** An embedded resource (ad, tracker) from `tracker.com` sets a cookie with a unique ID (`xyz`). `tracker.com` notes your visit to Site A.
2. **Visit Site B:** Another embedded resource from `tracker.com` is present. Your browser sends the cookie (`id=xyz`) back to `tracker.com` .
3. **Linking:** `tracker.com` now knows the **same browser (ID xyz)** visited both Site A and Site B, building a profile of your interests across the web.

Limitations of Cookie Blocking & The Rise of Fingerprinting

- **Problem:** Users started blocking third-party cookies.
- **Industry Response: Browser Fingerprinting**.
- **Concept:** Identify users by combining various characteristics of their browser and device configuration, creating a "fingerprint".
- **Attributes Used:**
    - Browser type & version (User Agent)
    - Operating System
    - Installed fonts & plugins
    - Screen resolution & color depth
    - Language settings
    - ...
- **Result:** The combination can be highly unique, allowing tracking **without** cookies.

Countermeasures Against Tracking

- **Basic Browser Settings:**
  - Blocking 3rd-party cookies (helps, but doesn't stop fingerprinting).
  - Clearing cookies regularly.
- **Privacy-Focused Browsers:** Brave, Firefox (with Enhanced Tracking Protection), DuckDuckGo browser often have stronger default protections.
- **Browser Extensions:**
  - **Content Blockers** (e.g., uBlock Origin): Block requests to known tracking domains/scripts using filter lists.
  - **Behavioral Blockers** (e.g., Privacy Badger): Learn which domains seem to be tracking you across sites and block them.
- **Other:** VPNs (mask IP address), Tor Browser (anonymizes traffic).

Module 3 Recap: Key Concepts

- **Cookies** enable state; **third-party cookies** enable cross-site tracking.
- **Browser Fingerprinting** uses device configuration for cookie-less tracking.
- The **context** (1st vs 3rd party) depends on the site being visited vs the domain setting/reading the cookie/script.
- **Countermeasures** range from browser settings to advanced tools that block tracking requests and scripts.
- This is an ongoing "arms race" between trackers and privacy tools/browsers.

Module 4: Data Privacy: Anonymization vs. Differential Privacy

How can we analyze sensitive data about people while protecting their privacy?

The Core Privacy Challenge with Data

- **Personal Data:** Information relating to an identifiable individual.
- **Sensitive Data:** Subset of personal data needing extra protection (e.g., health, beliefs, finances).
- **Goal:** Enable useful analysis (research, statistics) on datasets containing personal/sensitive info.
- **Risk:** Analysis or data release might inadvertently reveal information about specific individuals.
- **Two Main Strategies:**
    i. **Anonymization:** Modify the data itself to remove identifying links.
    ii. **Formal Privacy:** Add constraints/noise to the *analysis process* to protect individuals mathematically.

Data Anonymization: The k-Anonymity Approach

- **Quasi-Identifiers (QIs):** Attributes that are not unique on their own but can identify individuals when combined (e.g., ZIP Code, Birth Date, Gender).
- **k-Anonymity Goal:** Ensure that for any combination of QIs in the dataset, there are at least `k` individuals sharing that combination. Makes individuals "hide in a crowd" of size `k`.
- **Techniques:**
  - **Generalization:** Replace specific values with broader ones (Age `28` -> `20-29`).
  - **Suppression:** Hide specific values (`*`).

Visualizing k-Anonymity (k=3 Example)

Original Data:

ZIP	Age	Condition
94117	28	Flu
94117	29	Bronchitis
94117	21	Flu
90210	35	HBP
90210	32	HBP
90210	38	HBP

Visualizing k-Anonymity (k=3 Example)

k=3 Anonymized (QIs: ZIP, Age):

ZIP	Age Range	Condition
94117	20-29	Flu
94117	20-29	Bronchitis
94117	20-29	Flu
90210	30-39	HBP
90210	30-39	HBP
90210	30-39	HBP

*Each row is indistinguishable from at least 2 others based on (ZIP, Age Range).

When considering the Condition attribute, it is not 3-Anonymous.

Limitations of k-Anonymity

- **Homogeneity Attack:** In the example above, if you know someone lives in 90210 and is 30-39, you *know* they have HBP, because *everyone* in that group does. k-anonymity is met, but privacy is breached for the sensitive attribute!
- **Background Knowledge Attack:** Attacker might know extra info (e.g., "My neighbor Bob is in the dataset and is ~35"). This can help narrow possibilities even in a k-anonymous dataset.
- **Curse of Dimensionality:** Hard to achieve k-anonymity with many QIs without excessive generalization/suppression, destroying data utility.
- *(Advanced: l-diversity, t-closeness try to address homogeneity but have own issues)*.

Formal Privacy: Differential Privacy (DP)

- **Different Philosophy:** Instead of modifying the data structure, DP defines a mathematical property of the *analysis algorithm* or *query mechanism*.
- **Core Promise:** The output of a DP analysis is **statistically indistinguishable** whether any particular individual's data was included in the input dataset or not.
- **Mechanism:** Carefully calibrated random noise is added to the results of computations (counts, sums, averages, ML model weights, etc.).
- **Privacy Parameter (ε, epsilon):** Controls the level of privacy. Lower ε = more noise, stronger privacy; Higher ε = less noise, weaker privacy.

Why is Differential Privacy Often Considered Stronger?

- **Resilience to Background Knowledge:** The privacy guarantee holds regardless of what external information an attacker possesses.
- **Protection Against Diverse Attacks:** Inherently protects against linkage attacks, homogeneity attacks, differencing attacks, etc.
- **Quantifiable Privacy Loss:** Epsilon (ε) provides a measure of privacy loss.
- **Composition:** DP guarantees compose predictably – you can track the total privacy loss (ε) across multiple queries (the "privacy budget").
- **Focus:** Protects individuals while allowing aggregate statistical analysis.

Module 4 Recap: Key Concepts

- **Anonymization** (like k-anonymity) modifies data to obscure individuals but has known vulnerabilities (homogeneity, background knowledge).
- **Differential Privacy** adds noise to analysis results, providing a formal, mathematical guarantee against inferring individual information.
- DP is robust against background knowledge and allows quantifiable privacy loss (ε) and composition.
- The choice involves a **privacy-utility trade-off**. DP often requires careful implementation to maintain usefulness.

Module 5: Privacy & Security in Machine Learning

How do we ensure machine learning systems are private and secure?

What is Machine Learning (Briefly)?

- **Concept:** Algorithms that allow computer systems to "learn" patterns and make predictions or decisions **from data**, without being explicitly programmed for every specific task.
- **Process:**
    1. Collect **Training Data**.
    2. Choose a **Model Architecture**.
    3. **Train** the model (adjusting parameters to minimize errors on training data).
    4. **Deploy** the model to make predictions on new, unseen data.
- **Dependency:** The model's behavior is heavily dependent on the training data.

New Risks Introduced by ML

- **Data is the Asset:** ML models encapsulate patterns learned from potentially sensitive training data.
- **Model Accessibility:** Models are often deployed via APIs, allowing queries.
- **This creates vulnerabilities:**
  - **Privacy Risks:** Inferring sensitive information *about* the training data.
  - **Security Risks:** Attacking the model's *integrity* or *availability*.

Key Privacy Risks in ML

- **Membership Inference:** Can an attacker determine if a specific person's data (e.g., their photo, medical record) was used to train the model?
    - *Why it matters:* Leaks potentially sensitive information (e.g., participation in a study, being an employee).
- **Attribute Inference / Model Inversion:** Can an attacker infer sensitive attributes of the training data (e.g., the likely race or gender distribution) or even reconstruct average-looking training examples?
    - *Why it matters:* Leaks aggregate or individual characteristics.

Mitigating Privacy Risks in ML

- **Differential Privacy:** Apply DP principles *during model training*. Add noise to the learning process (e.g., to gradients) so the final model parameters don't overly depend on any single training record.
- **Federated Learning:** Train the model across multiple decentralized devices (e.g., mobile phones) where the raw data resides. Only aggregated model updates (often anonymized or protected) are sent to a central server. Data stays local. (Note: the gradients can still leak from private information!)
- **Secure Enclaves / MPC:** Perform training within secure hardware or using multi-party computation (more complex).

Key Security Risks in ML

- **Adversarial Examples (Evasion Attacks):** Attackers craft inputs with tiny, human-imperceptible changes specifically designed to fool the model into making an incorrect prediction.
    - *Example:* A slightly altered image of a stop sign classified as a speed limit sign.
    - *Impact:* Undermines model reliability, safety, and trust.
- **Data Poisoning Attacks:** Attackers inject malicious data into the *training set* to compromise the final model's behavior (e.g., create backdoors, degrade performance on specific inputs).
    - *Impact:* Corrupts the model's integrity from within.

Mitigating Security Risks in ML

- **Adversarial Training:** Include adversarial examples (and their correct labels) as part of the training process to make the model more robust against them.
- **Input Sanitization / Validation:** Check model inputs for suspicious patterns before prediction.

Module 5 Recap: Key Concepts

- ML models learn from data, creating new **privacy** (inference), **security** (adversarial, poisoning), and **ethical** (bias) risks.
- **Adversarial Training** helps defend against adversarial example.
- **Data security** is crucial to prevent poisoning.
- Building **Trustworthy AI** means addressing all these dimensions.

Q&A