Security and Privacy

Secure Multiparty Computation & Privacy

To remember: Privacy Definition

- The human definition:
  - **Private information**: information about us that we **feel** is personal
  - **Privacy**:
    - Confidential or private information **should not be unnecessarily distributed or publicly known**
    - Our personal or private information **should not be misused** (whatever that means)
- Challenges to formulate this:
  - The same information is classified differently by different people
  - Legitimate use is interpreted differently by different people

Motivation

- **Huge databases** exist in society today
  - Medical data
  - Consumer purchase data
  - Communication and media-related data
  - Data gathered by government agencies
- **This data be utilized**
  - For medical research
  - For improving customer service
  - For homeland security

Motivation – Concrete Example

- Investigation at Stillwater State Correctional Facility, Minnesota [2001]

  - Data mining software was applied to **phone records** from the prison
  - A pattern **linking calls** between **prisoners** and a **recent parolee** was discovered
  - The **calling data** was then **mined again together with records of prisoners' financial accounts**
  - The result: a large drug smuggling ring was discovered

http://www.nytimes.com/2006/02/25/technology/taking-spying-to-higher-level-agencies-look-for-more-ways-to.html

Data Utilization **or** Privacy?

- Data sharing is necessary for **full utilization**:
  - Example: Pooling of information from **different government agencies** can provide a wider picture
    - What is the **health status** of citizens that are supported by **social welfare**?
    - Are there citizens that receive **simultaneous support from different agencies?**
- Challenges Associated:
  - Example: a number of hospitals wish to jointly mine patient data for medical research
    - **Privacy policy and law prevents** hospitals from pooling/sharing
    - Without sharing, **classical data mining solutions cannot be used**
      - Solution is needed for **mining data on the union of independent DBs** without pooling or revealing data
      - They can **share conclusions**, but **more patterns** could be found if data and not just conclusions are shared

Data Utilization **or** Privacy?

- In many cases, due to public outcry, important data mining projects have been halted
  - The Canadian big-brother database
  - The Total Awareness project

Data Utilization or Privacy?

- Canadian Big-brother Database
  - Vast federal DB that pooled **citizen data** from **a number of different government ministries**
  - Officially called Longitudinal Labor Force File, quickly became known as **“big brother” DB**
  - **Goal**: implement governmental research that could improve services received by citizens
  - **Dismantled** due to privacy concerns and public outcry

It may be good that these projects were scrapped in their current form, but it would be better if we could have **our cake and eat it too…**

Privacy Preserving Data Mining (PPDM)

1. Statistical data is released
   - Data is first modified so that
     - Does not compromise anyone's privacy
     - Still possible to obtain meaningful results

2. **Data divided among 2+ different parties**
   - Run data mining algorithm on the **union of the DBs** without allowing any party to view other DBs

Multi-Party Computation

- **Distributed computing** considers the scenario where a number of **distinct**, yet **connected**, **computing devices** (or parties) wish to carry out **a joint computation** of some function.
  - Distributed computing classically deals with **questions of computing** under the threat of **machine crashes** and or **inadvertent faults.**

- **The aim of secure multiparty computation** is to enable parties to carry out such distributed computing tasks in a **secure manner**.
  - Secure multiparty computation is concerned with the possibility of **malicious behaviour** by some adversarial entity

Multi-Party Computation

- In Secure multiparty computation
  - It is assumed that a **protocol execution** may come **under "attack"** by an **external entity**, or even by **a subset of the participating parties**
  - The **aim of this attack** may be to learn **private information** or cause the result of the computation to be **incorrect**.
- Two important requirements on any secure computation protocol are
  - **Privacy**
  - **Correctness**

Multi-Party Computation

- **Privacy requirement**
  - **Nothing should be learned beyond what is absolutely necessary**; more exactly, parties should learn their output and nothing else
- **Correctness requirement**
  - Each party should receive **correct output**
  - Therefore, the adversary must not be able to cause the result of the computation to deviate from the **function that the parties had set out to compute.**

Question: How to construct **efficient secure protocols** for the **multi-party setting** that guarantee both **privacy** and **correctness**?

Multi-Party Computation - Example

- Election protocol
  - **Privacy requirement**: ensuring that no coalition of parties **learns anything** about the individual votes of other parties
  - **Correctness requirement**: ensuring that no coalition of parties **can influence the outcome of the election**

Multi-Party Computation - Example

- Auction protocol
  - **Privacy requirement**: ensuring that only the winning bid is revealed
  - **Correctness requirement**: that the highest bidder is indeed the party to win

Approach

To achieve secure multiparty computation (SMC)

Heuristic Approach to Security

1. Build a protocol
2. Try to break the protocol
3. Fix the break
4. Return to (2)

Q: Why doesn’t this work with privacy?

Once an individual's privacy is breached, there is no way that the clock can be turned back.

Heuristic Approach – Dangers

- Hackers will do anything to exploit a weakness – if one exists, it may well be found
- Real adversaries **won't tell** you that they have broken the protocol
  - You **can never be really sure that the protocol is secure**

Security **cannot** be checked empirically

Another Heuristic Tactic

- Design a protocol
- Provide a **list of attacks that (provably) cannot be carried out on the protocol**
- **Reason** that the list is **complete**

- **Problem:** often, the list is **not** complete…

A Rigorous Approach

- The history of computer security shows that the heuristic approach is likely to fail
    - Security is very tricky and often **anti-intuitive**

- In a Rigorous Approach
    - Provide an exact problem definition
        - Adversarial power
        - Meaning of security (privacy)
    - Prove that the protocol is secure
        - Often by reduction to an assumed hard problem

Secure Computation and Privacy

- **Secure computation:**
  - Assume that there is a **function** that **all parties wish to compute**
  - **Secure computation** shows how to compute that function in the **safest way possible**
  - In particular, it guarantees **minimal information leakage** (e.g., the output only)

- **Concerns:**
  - Does the function output itself reveal "sensitive information", or
  - **Should** the parties agree to compute this function? (secure computation protocols are not concerned about whether or not a party should compute the function)

Secure Multiparty Computation

- A set of parties with **private inputs**
- Parties wish to jointly **compute a function of their inputs** and make sure that certain security properties (like privacy and correctness) are preserved
- **SMC properties must be ensured even if some of the parties maliciously attack the protocol (deliberate misbehavior)**

Secure Multiparty Computation

- In SMC model:
  - An **adversarial entity** **controls** some sub-set of the parties and wishes to **attack the protocol execution**.
  - The parties under the control of the adversary are called **corrupted**, and follow the adversary's instructions.
  - **Secure protocols should withstand any adversarial attack**

Secure Multiparty Computation

- **Corruption strategy**:
  - **Static corruption model**: the adversary is given a fixed set of parties whom it controls.
    - **Honest parties** remain honest throughout, **while corrupted parties** remain corrupted.
  - **Adaptive corruption model**: rather than having a fixed set of corrupted parties, adaptive adversaries are given the capability of corrupting parties during the computation.
    - The choice of who to corrupt and when can be arbitrarily decided by the adversary or may depend on its view of the execution.
    - Once a party is corrupted, it remains corrupted from that point on.
  - **Proactive corruption model:** Considers the possibility that parties are corrupted for a certain period of time only.
    - Honest parties may become corrupted throughout the computation, but corrupted parties may also become honest.

Secure Multiparty Computation

- **Allowed adversarial behavior:**
  - **Semi-honest adversaries (**weak adversarial model**)**:
    - The adversary **obtains the internal state** of all the corrupted parties and attempts to use this to learn information that should remain private.
    - But, the **corrupted parties correctly follow the protocol specification.**
    - Semi-honest adversaries are also called "**honest-but-curious**" and "**passive**".
  - **Malicious adversaries**:
    - The corrupted parties deviate from the protocol specification
    - Malicious adversaries are also called "**active**".

Secure Multiparty Computation

- Complexity: computational complexity of the adversary
  - **Polynomial time:** The adversary is allowed to run in (probabilistic) polynomial time (and sometimes, expected polynomial time).
  - **Computationally unbounded:** In this model, the adversary has no computational limits whatsoever.
- In order to formally claim and prove that a protocol is secure, a **precise definition of security** for multiparty computation is required.

Secure Multiparty Computation - Properties

- **Privacy**: No party should learn **anything more than its prescribed output**. In particular, the only information that should be learned about other parties' inputs is what can be derived from the output itself (e.g., highest bidder in auction example).

- **Correctness**: Each party is **guaranteed that the output that it receives is correct**. (e.g., the party with the highest bid is guaranteed to win, and no party including the auctioneer can influence this.)

Secure Multiparty Computation - Properties

- **Independence of Inputs**: Corrupted parties must choose their inputs independently of the honest parties' inputs. (e.g., bids are kept secret and parties must fix their bids independently of others).
- **Guaranteed Output Delivery**: Corrupted parties should not be able to prevent honest parties from receiving their output. In other words, the adversary should not be able to disrupt the computation by carrying out a "**denial of service**" attack.
- **Fairness**: Corrupted parties should receive their outputs if and only if the honest parties also receive their outputs.

The above list does not constitute a definition of security, but rather a set of requirements that should hold for any secure protocol.

Analysis of Security Concerns

- Option 1: **Analyze security concerns for each specific problem**
  - **Example: Elections** (privacy and correctness only)
    - An adversary may wish to learn the votes of others –> to prevent this, require **privacy**
    - An adversary may wish to win the elections without having most votes –> to prevent this, require **correctness**
  - **Example: Auction**
    - An adversary may wish to learn the bids of all parties -> to prevent this, require **privacy**
    - An adversary may wish to win with a lower bid than the highest -> to prevent this, require **correctness**
    - The adversary may also wish to ensure that it always gives the highest bid, to prevent this, require **independence of inputs, as in election**

Analysis of Security Concerns

- Challenges:
  - How do we know that all concerns are covered?
  - Security Definitions **are application dependent** and need to be redefined from scratch for each task

Analysis of Security Concerns

- Option 2: The **real/ideal model** paradigm for defining security:
  - Consider an **"ideal world"** in which an **external trusted** (and incorruptible) **party** is willing to help the parties carry out their computation.
  - In such a world, the parties can simply send **their inputs over perfectly private channels to the trusted party**, which then computes the desired function and passes each party its prescribed output.

Analysis of Security Concerns

- The **real/ideal model** paradigm for defining security:
  - **Ideal model:** parties send inputs to a trusted party, who computes the function for them
    - Since **no** attacks can be carried out in the **ideal model**, security is implied (all properties like privacy and correctness)
  - **Real model: there is no external party that can be trusted** by all parties. Thus, parties run a protocol (among themselves) with no trusted help
  - A secure protocol (in real model) should **emulate** the so-called **"ideal world".**

- In this model, the security of a protocol is established by **comparing the outputs of the adversary and honest parties in a real protocol execution to their outputs in an ideal computation.**

Example 1: Private Dating

Alice and Bob meet at a pub

- If both of them want to date together – they will find out
- If Alice doesn’t want to date – she won’t learn his intentions
- If Bob doesn’t want to date – he won’t learn her intentions

Example 1: Private Dating

Alice and Bob meet at a pub

- If both of them want to date together – they will find out
- If Alice doesn't want to date – she won't learn his intentions
- If Bob doesn't want to date – he won't learn her intentions

Solution: use a trusted bartender

Example 2: Private Auction

Many parties wish to execute a private auction

- The highest bid wins
- Only the highest bid (and bidder) is revealed

Example 2: Private Auction

Many parties wish to execute a private auction

- The highest bid wins
- Only the highest bid (and bidder) is revealed

Solution: use a trusted auctioneer

Example 3: Private Set Intersection

Intelligence agencies holds lists of potential terrorists

- They would like to compute the intersection
- Any other information must remain secret

MI5

FBI

Example 3: Private Set Intersection

Intelligence agencies holds lists of potential terrorists

- They would like to compute the intersection
- Any other information must remain secret

Solution: use a trusted party

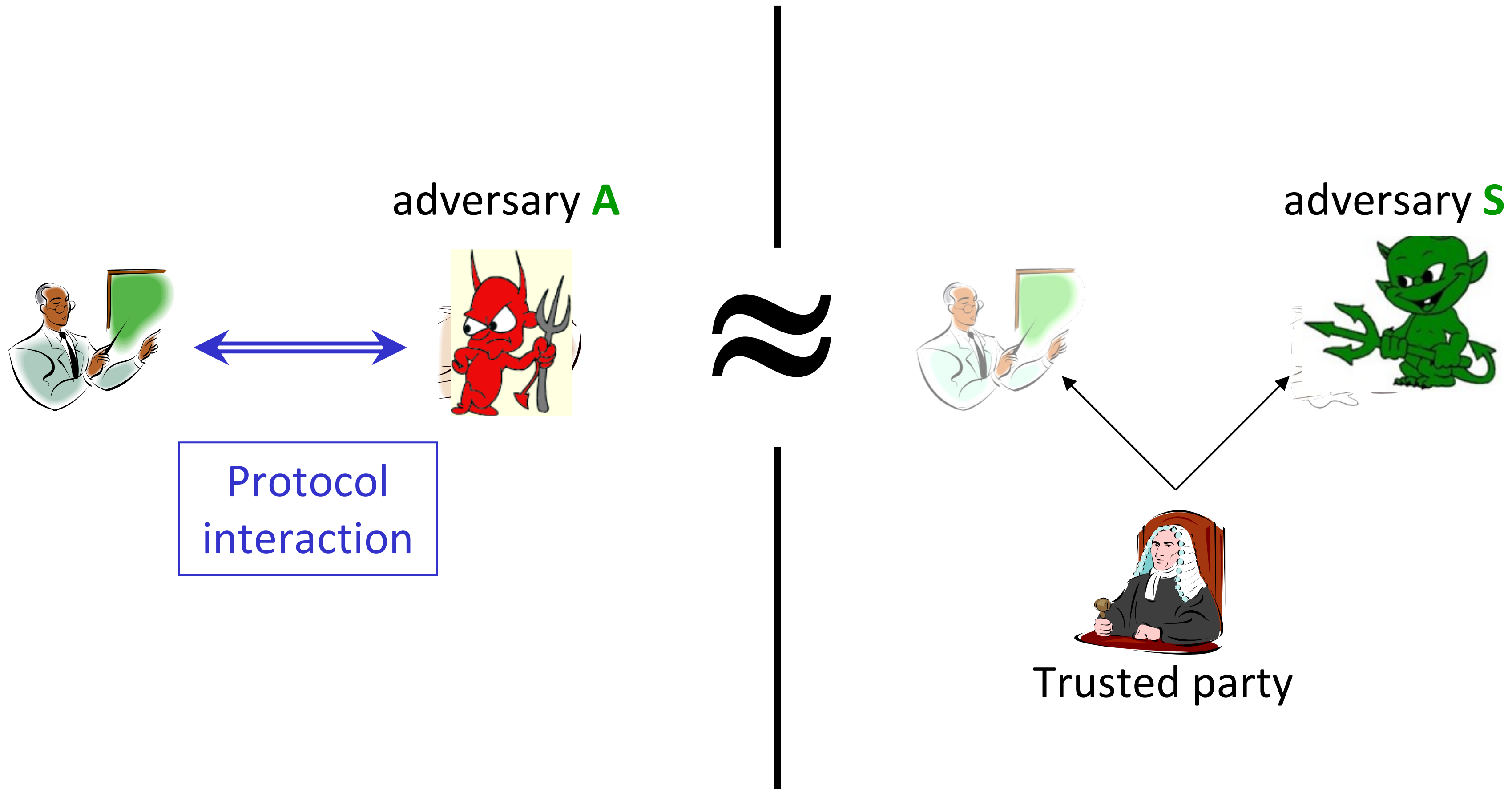
Trust me

Mossad

MI5

FBI

Security Definition - ideal/real simulation paradigm



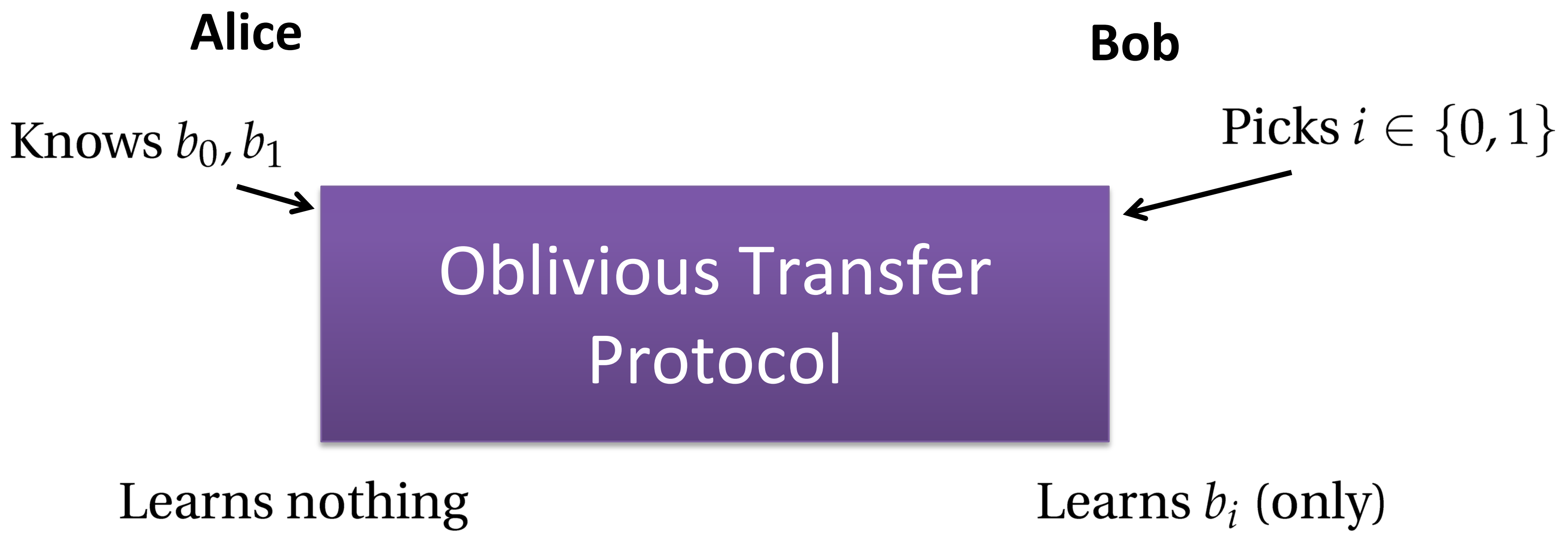
REAL **IDEAL**

Overview of SMC Primitives

- Oblivious transfer (OT)
  - Share information without knowing what information was shared
- Bit commitment (BC)
  - How to flip a coin over the telephone?
- Zero Knowledge (ZK)
  - Prove a statement without disclosure
- Private set intersection (PSI)
  - Determine shared cellphone contacts without knowing contacts that are not shared

Oblivious Transfer (OT)

- A sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece (if any) has been transferred.



Oblivious: Alice doesn't learn which secret Bob obtains
Transfer: Bob learns one of Alice's secrets

OT Semi-Honest

1-out-of-2 Oblivious Transfer (OT)

- Inputs
  - Sender has two messages m_0 and m_1
  - Receiver has a single bit $\sigma \in \{0,1\}$
- Outputs
  - Sender receives nothing
  - Receiver obtains m_σ and learns nothing of $m_{1-\sigma}$

Semi-Honest OT

- Let (G,E,D) be a public-key encryption scheme
  - G is a key-generation algorithm $(pk,sk) \leftarrow G$
  - Encryption: $c = E_{pk}(m)$
  - Decryption: $m = D_{sk}(c)$

- Assume that a public-key can be sampled without knowledge of its secret key:
  - Oblivious key generation: $pk \leftarrow OG$
  - (El-Gamal encryption has this property)

Semi-Honest OT Protocol

Protocol for Oblivious Transfer

- Receiver (with input σ):
  1. Receiver chooses one key-pair (pk,sk) and one public-key pk’ (obliviously of secret-key).
  2. Receiver sets pk_σ = pk, $pk_{1-\sigma}$ = pk’
  3. Note: receiver can decrypt for pk_σ but not for $pk_{1-\sigma}$
  4. Receiver sends pk_0,pk_1 to sender
- Sender (with input m_0,m_1):
  1. Sends receiver $c_0=E_{pk0}(m_0)$, $c_1=E_{pk1}(m_1)$
- Receiver:
  1. Decrypts c_σ using sk and obtains m_σ.

OT Security Proof

- Intuition:
  - Sender’s view consists only of two public keys pk_0 and pk_1. Therefore, it doesn’t learn anything about that value of σ.
  - The receiver only knows one secret-key and so can only learn one **message**

- Note: this assumes semi-honest behavior. **A malicious receiver can choose two keys together with their secret keys.**

OT Generalization

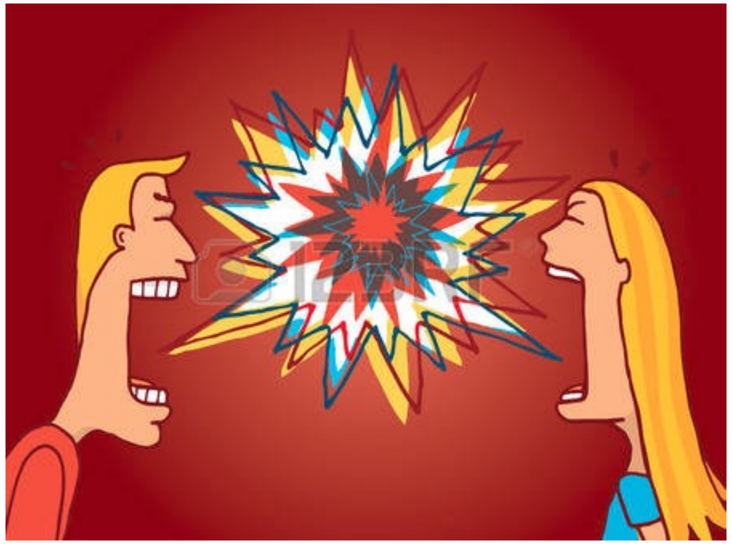
- Can define 1-out-of-k oblivious transfer

- Protocol remains the same:
  - Choose k-1 public keys for which the secret key is unknown
  - Choose 1 public-key and secret-key pair

- Application: Secure Key Exchange

Problem: Flipping a Coin Over the Phone

- Alice and Bob got divorced and no longer trust each other
- They want to decide over the phone who gets the car by flipping a coin



Problem: Flipping a Coin Over the Phone

Suppose Alice and Bob want to resolve some dispute via coin flipping.

- If they are physically in the same place:
  - Alice "calls" the coin flip
  - Bob flips the coin
  - If Alice's call is correct, she wins, otherwise Bob wins
- If Alice and Bob are not in the same place:
  - Once Alice has "called" the coin flip, Bob can stipulate the flip "results" to be whatever is most desirable for him.
  - Similarly, if Alice doesn't announce her "call" to Bob, after Bob flips the coin and announces the result, Alice can report that she called whatever result is most desirable for her.

Problem: Flipping a Coin Over the Phone

- Similar to Fair cake-cutting:
  - You have a large piece of cake that you want to split between 2 persons
  - Only 1 of the 2 persons can cut it
  - How to split it evenly, if both users are potentially malicious?

Solution: Bit Commitment

- In BC Alice must commit a bit to Bob, such that she cannot change it once she committed, and Bob cannot learn it until she reveals it
  - Alice "calls" the coin flip but only tells Bob a **commitment** to her call,
  - Bob flips the coin and reports the result,
  - Alice reveals what she committed to,
  - Bob verifies that Alice's call matches her commitment,
  - If Alice's revelation matches the coin result Bob reported, Alice wins

Solution: Bit Commitment

- BC incorporates two phases

1. Commit Stage: where Alice commits to a bit by sending Bob a token
    - Committer (Alice) has a bit σ
    - Committer sends Receiver (Bob) a commitment string c (commit to bit σ)
        - E.g., Alice writes σ on a sheet of paper, locks the paper in the safe, and sends the safe to Bob while keeping the key.

2. Reveal Stage: where Alice reveals the bit
    - Committer sends a **decommit message** to receiver
        - E.g., the key to open the safe
    - Receiver uses **decommit message** and c to obtain σ

Solution: Bit Commitment

Security Properties:

- Binding: for every c (commit of σ), there exists only one value σ for which decommitment is accepted
  - Formally: the set of commitment strings to $\sigma = 0$ is **disjoint** from the set of commitment strings to $\sigma = 1$.

- Hiding: the receiver cannot distinguish a commitment string that is to 0 from a commitment string that is to 1.

Bit Commitment Protocol

- Instructive example: committing to a bit using a telephone book
- Commitment using public-key encryption:
  - Committer chooses a key-pair (pk,sk).
  - Committer sends (pk, $c=E_{pk}(\sigma)$) to the receiver.
- Decommitment:
  - Committer sends the secret-key sk to the receiver
  - Receiver verifies that sk is associated with pk and decrypts, obtaining σ.
- Note: the commitment process is randomized. This is essential for any secure commitment. I.e., function of σ and coins r.

Bit Commitment Protocol: Proving Security

- Assumption: given pk, there is exactly one secret key sk that is associated with pk, and this can be efficiently determined (holds for RSA).
- Binding: encryption must have unique decryption. So given correct sk (above assumption), any c can only decrypt to one of 0 or 1.
- Hiding: without knowledge of sk, cannot distinguish encryptions of 0 from encryption of 1 (by definition of security of encryption).

Problem: Prove a statement without disclosure

- One party (a prover) wishes to prove to another party (the verifier) that a given statement is true while the prover avoids conveying any additional information apart from the fact that the statement is indeed true

- This can be done by providing proof, or a small amount of information, that can be verified by the verifier to ensure that the statement is true.

Solution: Zero Knowledge (ZK)

- Zero knowledge: the verifier will learn nothing beyond the fact that the statement is correct
- Soundness: the prover will not be able to convince the verifier of a wrong statement

Zero Knowledge - Illustrative Example

- Prover has two colored cards that he claims are of different color
- The verifier is color blind and wants a proof that the colors are different.
- Idea 1: use a machine to measure the light waves of colors, but then the verifier will learn what the colors are

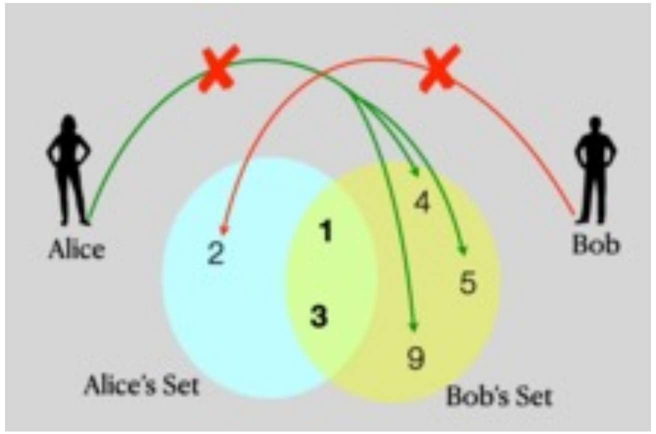
Zero Knowledge - Example

- Protocol:
  1. Verifier writes color1 and color2 on the back of the cards and shows the prover
  2. Verifier holds out one card so that the prover only sees the front
  3. The prover then says whether or not it is color1 or color2

- Soundness: if they are both the same color, the prover will fail with probability ½. By repeating many times, will obtain good soundness bound.

Problem: Private Set Intersection

- Two Homeland Security agencies want to cooperate to find potential terrorists
  - Intersection of black-lists
- Two persons want to determine their common contacts (e.g., smartphone)
  - Intersection of contact lists



Solution: Private Set Intersection

There are several solutions available:

- a naive hashing solutions where elements are hashed and compared
- the server-aided protocol
- the Diffie-Hellman-based PSI protocol
- the OT-based PSI protocol

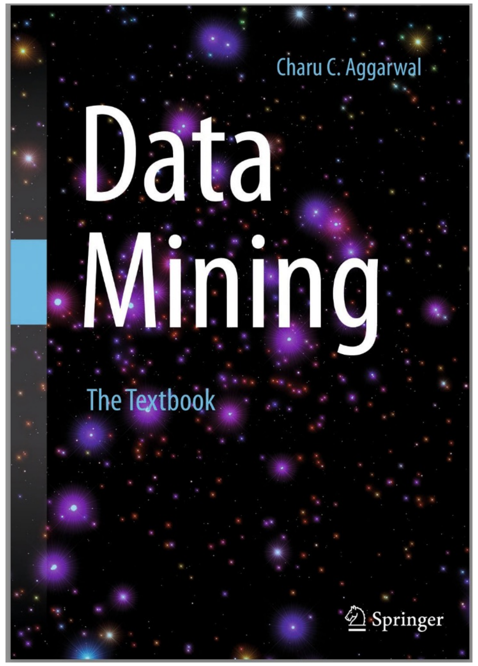
More on this topic in next class …

References

- Secure Multiparty Computation for Privacy-Preserving Data Mining, Y. Lindell and B. Pinkas, Journal of Privacy and Confidentiality 2009
- Computing Cooperatively with People You Don’t Trust slides, David Evans, 2012
- Secure Multiparty Computation: Introduction slides, Ran Cohen, Seminar on Secure Multi-Party Computation 2017

- Privacy, Profiling, Targeted Marketing, and Data Mining (Jaideep Vaidya and Vijay Atluri)
- Privacy-Preserving K-means Clustering over Vertically Partitioned Data, Vaidya and Clifton, KDD’03
- Tools for Privacy Preserving Data Mining, Clifton et al, SIGKDD’02

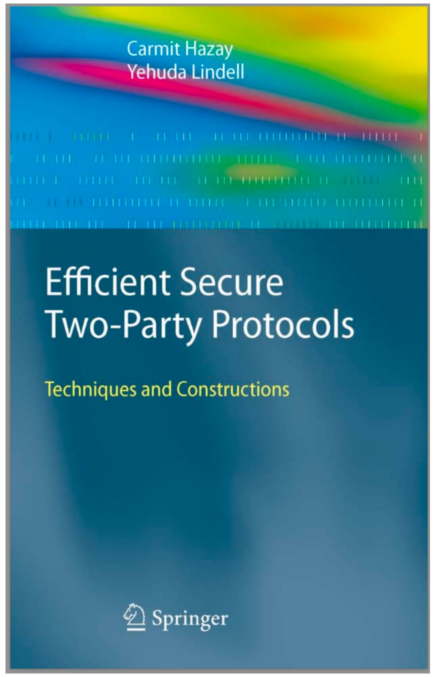
Bibliography


Charu C. Aggarwal

Data Mining

The Textbook

Springer


Carmit Hazay
Yehuda Lindell

Efficient Secure Two-Party Protocols

Techniques and Constructions

Springer