CSE 410/565 Computer Security
Spring 2022

Lecture 23: Privacy Enhancing Technologies

Department of Computer Science and Engineering
University at Buffalo

Lecture Outline

- Electronic cash
- Anonymous credentials and access control
- Secure voting
- Computation over encrypted data

Electronic Cash

- As we perform many transactions in electronic form, there is a need for electronic money
  - check and credit cards leave trails
  - can we have an equivalent of anonymous cash?
- Properties of cash
  - it is anonymous and untraceable
  - it can be used off-line, not connected to a bank
  - it is transferable
  - it has different denominations, and one can make change with it
  - it can be used only once (or stolen)

Electronic Cash

- Can we design digital cash with similar properties that can be sent through computer networks?
- Version 0
  - a bank gives Alice a digital banknote with the bank's signature
- What problems does this have?

Electronic Cash

- Version 1: adding anonymity
  - Alice prepares a banknote and bank blindly signs it
  - consider blind signing using plain RSA:
    - given $m < n$, signing is $\sigma = m^d \bmod n$ and verification uses $\sigma^e \bmod n$
    - message is first blinded as $m' = m \cdot r^e \bmod n$ using random $r < n$
    - bank signs m' by producing σ'
    - Alice recovers signature on m as $\sigma' \cdot r^{-1} \bmod n$

Electronic Cash

- Version 2: cheating prevention
  - to ensure that Alice correctly forms the banknote to be signed, we a technique called cut-and-choose
  - Alice prepares k banknotes to be signed and blinds them
  - bank chooses one and asks the rest to be opened
  - if all opened $k - 1$ banknotes are well-formed, bank signs the remaining mesage
  - an alternative is to use a zero-knowledge proof of knowledge of coin wellformedness

Electronic Cash

- Version 3: dealing with double spending
  - we can require the bank to keep track of all spent coins
  - each coin will need to be unique and have a serial number
  - serial numbers can be chosen randomly and be unique with very high probability if the space is large enough

Electronic Cash

- Version 4: attributing double spending to the party at fault
  - if Alice is trying to spend the same coin with more than one merchant, we want her to loose anonymity
  - if the merchant is trying to deposit Alice's coin more than once, we want the bank to know that the merchant is cheating
  - this is typically solved by encoding Alice's identity into a coin
    - if a coin is spent once, identity is protected
    - if a coin is spent more than once, identity can be recovered

Cryptocurrencies

- Bitcoin solved the chicken and an egg problem in adopting digital cash by adopting a different model
  - real banks are not a part of the protocol and digital transactions
- Public signing keys are used as users pseudonyms
  - transactions are not linked to real identities, but transactions with the same key are linked to each other
- Blockchain is a mechanism for distributed consensus
  - it offers a distributed unmutable storage
  - previous transactions are recorded and stored in the blockchain
  - the concept of the proof of work holds it all together

Anonymous Access Control

- Anonymous access control can be realized using anonymous credentials and anonyomus authentication
- A user can obtain credentials on certain attributes
  - at the certification time and at the time of authentication an attribute or only partial information about an attribute can be disclosed
  - examples:
    - a user can prove that she is over 21 without revealing the birth date (or anything else)
    - the user can prove that she is a student member and the expiration date is some time in the future

Anonymous Access Control

- To permit anonymous authentication, a user obtains authority's certification in the form of a signature on relevant attributes

- Relevant properties of the attributes can be demonstrated using zero-knowledge proofs of knowledge

- Each time anonymous credentials are used, the user
  - needs to randomize the credentials to prevent linkability
  - prove that the signed values satisfy the access control policy

- One significant issue with using anonymous credentials in a commercial setting is prevention of duplicating user credentials

Anonymous Access Control

- Solutions to the problem of credential duplication include:
  - incorporating sensitive information into each credential the knowledge of which must be shown upon each use
  - restricting the number of simultaneous uses of a credential
  - issuing one-time credentials that can be exchanged for a new token upon each use
- Certain other techniques allow a user to be anonymous with the ability to uncover the user's identity under exceptional circumstances

Secure Elections

- Preserving privacy of voters is crucial for fair outcome of elections
- Secure elections should have at least the following properties:
  1. only registered voters can vote
  2. no person can vote more than once
  3. no one can determine for whom anyone else voted
  4. every voter can make sure that his vote has been counted
  5. no person can duplicate any other person's vote
  6. no person can change any other person's vote undetected

Secure Elections

- Version 1: use encryption
  - each voter encrypts their vote with the tabulating authority's public key and communicates it
  - what desired properties are achieved?
- Version 2: use signatures for authentication
  - each voter signs her vote with her private signing key
  - each voter encrypts her signed vote with the authority's public key and communicates it

Secure Elections

- Version 3: use blinding for anonymity
  - we can use blind signatures for anonymity, but also need to prevent misbehavior
  - each voter prepares k sets of messages with a large random number and a valid vote for each candidate
  - each voter blinds each message individually and asks to get them signed
  - the authorithy picks one of them randomly and asks $k - 1$ of them to be opened
  - if all opened messages are formed correctly, the remaining one is signed
  - the user picks a candidates and submits the corresponding signed message as the vote

Secure Elections

- The next version achieves all desired properties and the two additional properties:

  7. if a voter finds that his vote is miscounted, he can correct the problem without compromising secrecy of his ballot

  8. a voter can redecide and cast a new vote within allocated time frame

Secure Elections

- Version 4: use anonymous IDs and one-time encryption keys
  - the tabulating authority publishes a list of eligible voters who intend to vote
  - each voter receives a unique number I not known to the authority
  - each voter generates public-key encryption pair (pk, sk) and sends $\mathsf{Enc}_{pk}(I; v)$, where v is the vote
  - the authority publishes received $\mathsf{Enc}_{pk}(I; v)$
  - each voter sends I, sk
  - the authority decrypts votes and at the end of election publishes results: for each vote v the list of all $\mathsf{Enc}_{pk}(I; v)$ that contained it

Secure Elections

- Version 4 (cont.):
  - if a voter sees error, she objects by sending $I, \mathsf{Enc}_{pk}(I; v), sk$
  - if voter wants to change vote from v to v', he sends $I, \mathsf{Enc}_{pk}(I; v'), sk$
- Malicious authorities still have some powers
  - forging votes for people who intended to vote, but didn't
  - neglecting to count Alice's vote and claiming that Alice never voted

Homomorphic Encryption

- Homomorphic encryption is a special type of encryption that, given ciphertexts, permits computation on the underlying plaintexts

$$\mathsf{Enc}_k(m_1) \otimes \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 \oplus m_2)$$

- Different types of homomorphic encryption are known:
  - partially homomorphic encryption
    - supports a single operation on ciphertexts
    - additively homomorphic encryption
      $\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 + m_2)$
    - multiplicatively homomorphic encryption
      $\mathsf{Enc}_k(m_1) \cdot \mathsf{Enc}_k(m_2) = \mathsf{Enc}_k(m_1 \cdot m_2)$

Homomorphic Encryption

- Different types of homomorphic encryption
  - fully homomorphic encryption (FHE)
    - supports two operations on ciphertexts: addition and multiplication
    - allows for any functionality to be evaluated on encrypted data
- Homomorphic encryption enables computation on encrypted data and results in efficient protocols for certain problems

Secure Multi-Party Computation

- More generally, secure multi-party computation allows for any desired function f to be securely evaluated on private data without revealing it
  - a number of parties hold private inputs $x_1, \ldots, x_n$
  - we evalute $f(x_1, \ldots, x_n)$ to obtain one or more outputs $y_1, \ldots$
  - each output y_i is revealed to a party or parties entitled to learning it
  - no other information about any x_i is available to any participant
    - more precisely, given your x_i and the output, you may deduce something about other x_is
    - but no additional information is revealed during the computation
  - this should hold even if a number of participants conspire against others and combine their information

Secure Multi-Party Computation

- To model security, we compare a real protocol execution with an ideal execution
  - in the ideal setting, no interaction takes place
    - the computation is performed by trusted party that received all inputs and computes outputs
  - showing security consists of demonstrating that real protocol execution can be simulated by querynig the trusted party in the ideal setting
  - this implies that messages transmitted by the protocol reveal no information about inputs
    - i.e., a participant cannot tell whether an intermediate message was simulated or computed using actual data

Working with Private Values

- One way to protect a value is by splitting it into multiple shares
  - this is called secret sharing
  - given value s, generate shares $s_1, s_2, \ldots, s_n$
  - each s_i is stored in a different place and doesn't reveal the secret
  - access to enough shares allows for s to be reconstructed, but individual shares don't reveal anything
  - specifically, (n, t) threshold secret sharing means
    - a secret s is divided into n shares
    - access to t shares allows for s to be reconstructed
    - access to $\leq t - 1$ shares reveals nothing about s

Secret Sharing

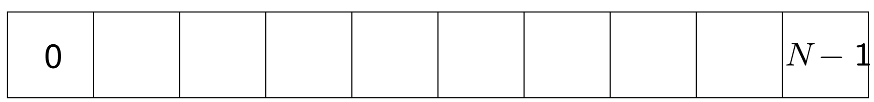
- Example: (2, 2) additive secret sharing
  - additive means we use addition to produce shares
  - our secret is $0 \le x < N$
  - choose random r from $\mathbb{Z}_N$ and set the first share $x_1 = r$
  - compute the second share $x_2 = (x - r) \bmod N$
  - to reconstruct, compute $x = x_1 + x_2 \bmod N$
  - for example
    - let $N = 10$ and $x = 3$
    - suppose we choose random $x_1 = 5$
    - we compute $x_2 = 3 - 5 \bmod 10 = 8$

Security of Secret Sharing

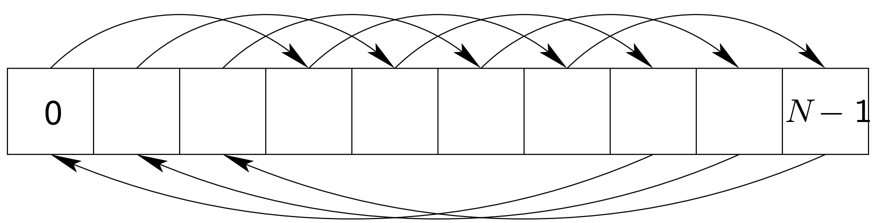
- Unlike encryption, secret sharing is unbreakable
  - secret sharing enjoys information theoretic security and achieves perfect secrecy
  - this goes back to Shannon's work in the 1940s
- Let's examine the two-party secret sharing above

Why Secret Sharing is Secure

- One party holds random r
  - clearly this cannot reveal anything about secret x
- The other party holds $x - r \bmod N$
  - this also doesn't reveal anything about x
  - when we draw random r, all N options are equally likely



  - when we add x to it, all N options are still equally likely



Why Secret Sharing is Secure

- The above means the outcome of protecting one value of x is identical to the outcome of protecting another value of x
  - this means that we learn no information about that value
- The above holds regardless of our computational capabilities
  - encryption requires that the keys and ciphertexts are sufficiently long to maintain security
  - information-theoretic techniques, on the other hand, can be used with arbitrarily small numbers

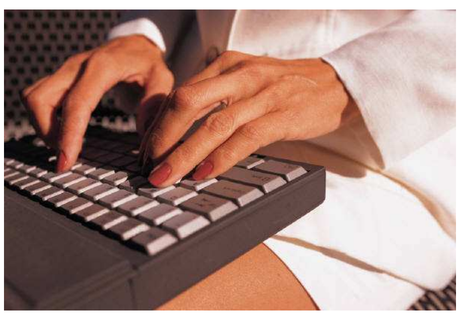
Computing with Secret Shared Values

Most types of secret sharing permit addition to be performed directly on local shares

- Addition $z = x + y$
  - assume (2, 2) additive secret sharing with modulus N
  - party i holds x_i, y_i and computes $z_i = (x_i + y_i) \bmod N$

Alice

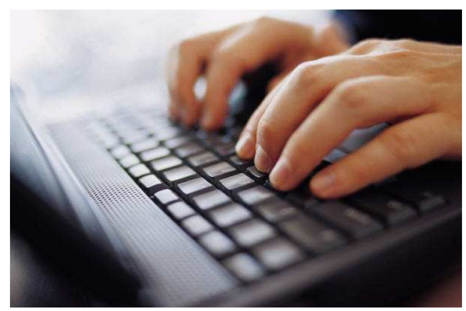
x_1, y_1



$z_1 = (x_1 + y_1) \bmod N$

Bob

x_2, y_2



$z_2 = (x_2 + y_2) \bmod N$

Computing with Secret Shared Values

- Multiplication $x \cdot y$
  - multiplication cannot be computed using only local shares
  - with two shares per value, we need to compute

$$z = x_1y_1 + x_2y_1 + x_1y_2 + x_2y_2 = z_1 + z_2 \pmod{N}$$

  - two terms (x_1y_1 and x_2y_2) can be computed locally, while others require additional tools
    - this requires interaction or tools such as homomorphic encryption
- (Integer) addition and multiplication are sufficient to compute any desired functionality

The Big Picture

- A number of participants would like to perform joint computation on their private inputs
  - secret sharing: each input owner creates shares of its private inputs and communicates a share to each party running the computation
  - secure computation: computation parties evaluate the function on shares one operation at a time
  - once the result is computed, shares are communicated to the output recipients
  - output reconstruction: each output recipient reconstruct its output from the received shares

Secure Multi-Party Computation

- Secure computation can take many forms
  - secure two-party computation
    - e.g., evaluating predisposition to a genetic disease
  - secure multi-party computation
    - e.g., determining the best treatment for a rare condition by multiple hospitals
  - secure computation outsourcing
    - e.g., offloading image processing to a cloud

Summary

- Technical solutions to privacy are numerous
  - in certain applications with want to combine anonymity with accountability
  - in other applications we seek to protect private information
- Work on privacy and anonymity started in early 80s and continues to date
  - efficient constructions for applications such as e-cash, anonymous credentials, etc. are known
  - there is always room for improvement