Privacy Enhancing Technologies
CSE 701 Fall 2017

Lecture 1: Secure Computation and Outsourcing

Department of Computer Science and Engineering
University at Buffalo

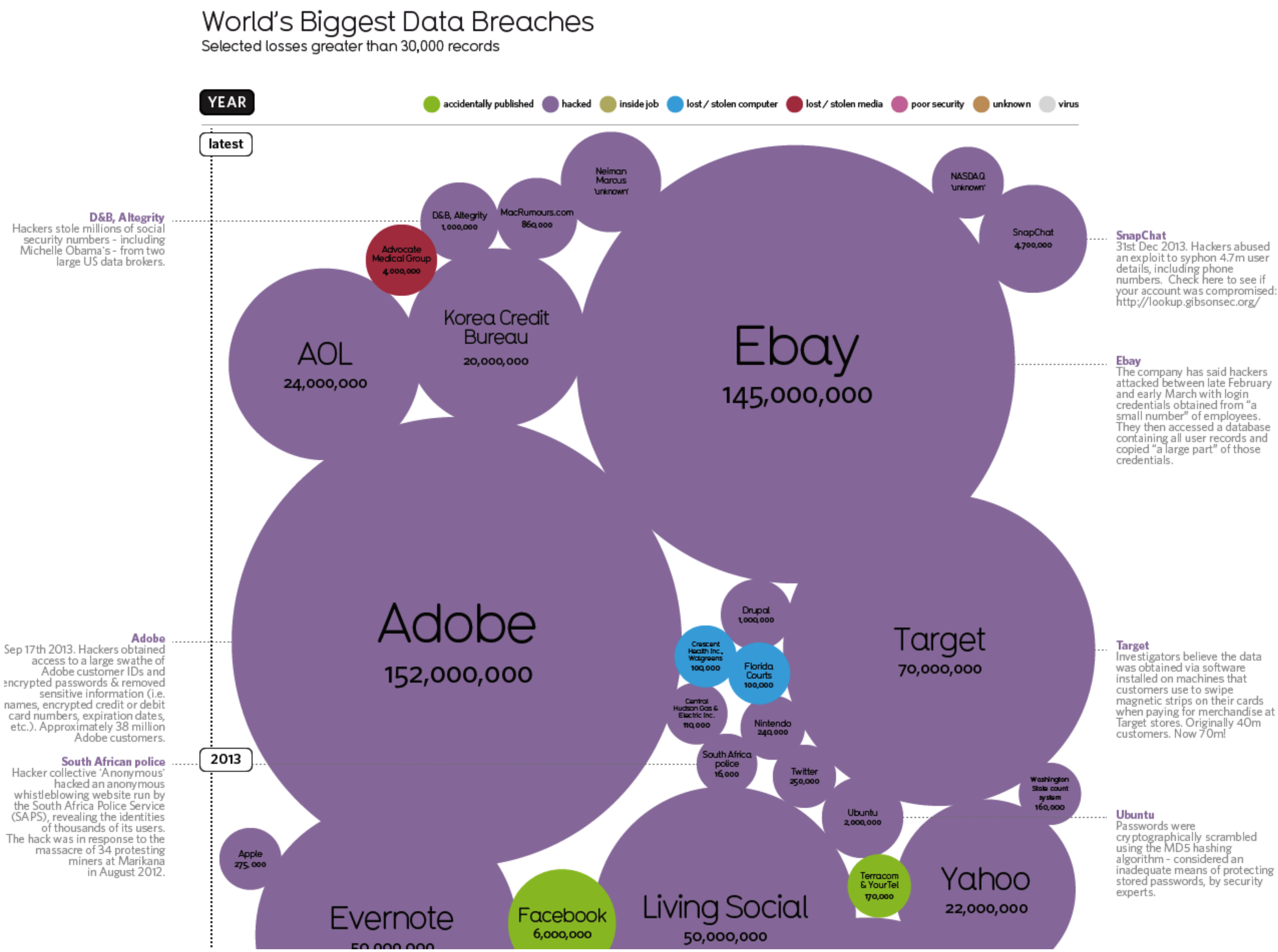
Data Privacy

- Why do we talk about protecting data privacy?

Data Privacy

- Larger and larger volumes of data are being collected about individuals
  - one's shopping behavior, geo location and moving patterns, interests and hobbies, exercise patterns, etc.
- Even intended analysis and use of data is scary, but it is also prone to abuse
  - information about individuals collected by an entity can be legitimately sold to others
  - large datasets with sensitive information are an attractive target for insider abuse
  - data breaches are more common than what we know

Data Breaches



Data Protection

- There are many different ways to protect private, proprietary, classified or otherwise sensitive information
  - this course will cover some of such techniques
- Protection techniques include:
  - computing on private data without revealing the data
  - anonymous communication and authentication
  - applications that provide anonymity (e-cash, voting, etc.)
- Standard techniques of protecting data at rest or in transit are not covered by this course

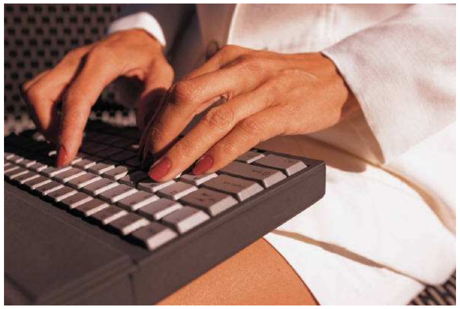
Secure Multi-Party Computation

- Secure multi-party computation allows two or more individuals to jointly evaluate a function on their respective private data
  - security guarantees allow for no unintended information leakage
  - only output of the computation (and any information deduced from the output and its private input) can be known to a participant

Example Secure Two-Party Computation

- Two millionaires Alice and Bob would like to determine who is richer without revealing their worth to each other

Alice
private x



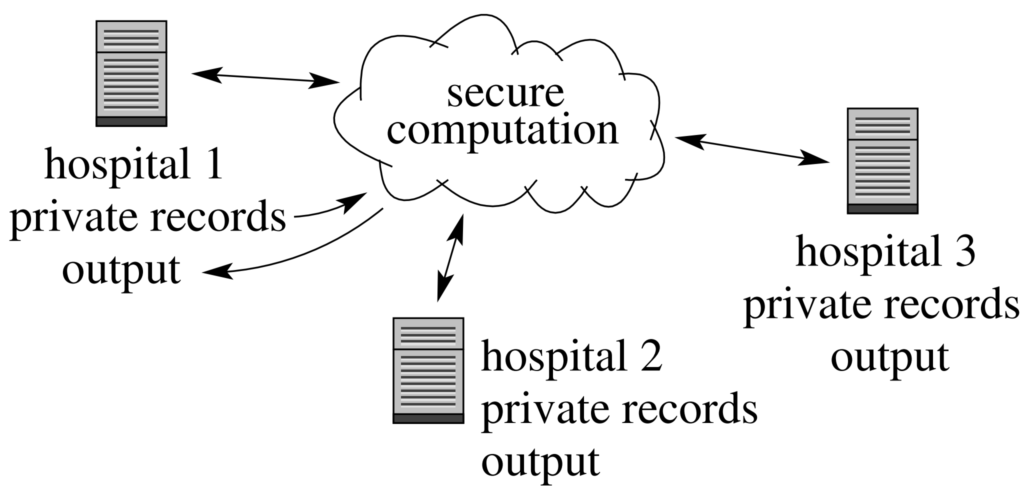
$\longrightarrow$
$\longleftarrow$
$\longrightarrow$
$\longleftarrow$
$\longrightarrow$

Bob
private y

output
$x < y$

Example Secure Multi-Party Computation

- A number of local hospitals would like to jointly determine the most effective treatment to a rare disease



Secure Multi-Party Computation

- Regardless of the setup, the same strong security guarantees are expected:
  - suppose there is an ideal third party that the participants trust with their data
  - they send their data to the trusted third party (TTP) and receive the output
  - then a multi-party protocol is secure if adversarial participants learn no more information than in the case of ideal TTP
  - this is formalized through a simulation paradigm

Security of SMC

- There are two standard ways of modeling participants in SMC
  - a semi-honest participant complies with the prescribed computation, but might attempt to learn additional information about other participants' data from the messages it receives
    - it is also called honest-but-curious or passive
  - a malicious participant can arbitrarily deviate from the protocol's execution in the attempt to learn unauthorized information about other participants' data
    - it is also called active
- There is a third type of adversarial model with covert participants who can act maliciously, but do not wish to be caught

Security of SMC in the Semi-Honest Model

- We start modeling security using the semi-honest model
  - Let n be the number of participants in secure computation
  - An adversary $\mathcal{A}$ can corrupt and control $t < n$ of them
  - $\mathcal{A}$ knows all information that the corrupt parties have and receive
  - Security is modeled by building a simulator $S_{\mathcal{A}}$ with access to the TTP that produces $\mathcal{A}$'s view indistinguishable from its view in real protocol execution
    - $S_{\mathcal{A}}$ has $\mathcal{A}$'s information, TTP's output, and must simulate the view of $\mathcal{A}$ and form outputs for all parties correctly

Security of SMC in the Semi-Honest Model

- Formal definition:
  - Let parties $P_1, \ldots, P_n$ engage in a protocol Π that computes function $f(\mathsf{in}_1, \ldots, \mathsf{in}_n) \rightarrow (\mathsf{out}_1, \ldots, \mathsf{out}_n)$, where $\mathsf{in}_i \in \{0,1\}^*$ and $\mathsf{out}_i \in \{0,1\}^*$ denote the input and output of party P_i, respectively.
  - Let $\mathrm{VIEW}_\Pi(P_i)$ denote the view of participant P_i during the execution of protocol Π. That is, P_i's view is formed by its input and internal random coin tosses r_i, as well as messages $m_1, \ldots, m_k$ passed between the parties during protocol execution:

$$\mathrm{VIEW}_\Pi(P_i) = (\mathsf{in}_i, r_i, m_1, \ldots, m_k).$$

  - Let $I = \{P_{i_1}, P_{i_2}, \ldots, P_{i_t}\}$ denote a subset of the participants for $t < n$ and $\mathsf{VIEW}_\Pi(I)$ denote the combined view of participants in I during the execution of protocol Π (i.e., the union of the views of the participants in I).

Security of SMC in the Semi-Honest Model

- Formal definition (cont.):
  - We say that protocol Π is t-private in the presence of semi-honest adversaries if for each coalition of size at most t there exists a probabilistic polynomial time simulator S_I such that

$$S_I(\mathsf{in}_I, f(\mathsf{in}_1, \ldots, \mathsf{in}_n)) \equiv \{\mathrm{VIEW}_\Pi(I), \mathsf{out}_I\},$$

    where $\mathsf{in}_I = \bigcup_{P_i \in I}\{\mathsf{in}_i\}$, $\mathsf{out}_I = \bigcup_{P_i \in I}\{\mathsf{out}_i\}$, and $\equiv$ denotes computational or statistical indistinguishability.

- Computational indistinguishability of two distributions means that the probability that they differ is negligible in the security parameter κ
  - for statistical indistinguishability, the difference must be negligible in the statistical security parameter

Security of SMC in the Malicious Model

- In the malicious model we have the following definition:
  - Let Π be a protocol that computes function $f(\mathsf{in}_1, \ldots, \mathsf{in}_n) \rightarrow (\mathsf{out}_1, \ldots, \mathsf{out}_n)$, with party P_i contributing input $\mathsf{in}_i \in \{0,1\}^*$ and receiving output $\mathsf{out}_i \in \{0,1\}^*$
  - Let $\mathcal{A}$ be an arbitrary algorithm with auxiliary input x and S be an adversary/simulator in the ideal model
  - Let $\mathrm{REAL}_{\Pi,\mathcal{A}(x),I}(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ denote the view of adversary $\mathcal{A}$ controlling parties in I together with the honest parties' outputs after real protocol Π execution
  - Similarly, let $\mathrm{IDEAL}_{f,S(x),I}(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ denote the view of S and outputs of honest parties after ideal execution of function f

Security of SMC in the Malicious Model

- Formal definition (cont.):
  - We say that Π t-securely computes f if for each coalition I of size at most t, every probabilistic adversary $\mathcal{A}$ in the real model, all $\mathsf{in}_i \in \{0,1\}^*$ and $x \in \{0,1\}^*$, there is probabilistic S in the ideal model that runs in time polynomial in $\mathcal{A}$'s runtime and

$$\{\text{IDEAL}_{f,S(x),I}(\mathsf{in}_1,\ldots,\mathsf{in}_n)\} \equiv \{\text{REAL}_{\Pi,\mathcal{A}(x),I}(\mathsf{in}_1,\ldots,\mathsf{in}_n)\}$$

Secure Multi-Party Computation

- The setting can be further generalized to allow for more general setups
- We can distinguish between three groups of participants
  - input parties (data owners) contribute their private input into the computation
  - computational parties securely execute the computation on behalf of all participants
  - output parties (output recipients) receive output from the computational parties at the end of the computation
- The groups can be arbitrarily overlapping

Secure Multi-Party Computation

- The above setup allows for many interesting settings
  - a large number of participating hospitals can choose a subset of them to run the computation on behalf of all of them
  - they can also employ external parties (cloud providers) for running the computation
  - the output can be delivered to a subset of them and/or to other interested parties
- This setup also allows for secure computation outsourcing
  - one or more clients securely outsource their computation to a number of external cloud computing providers

Secure Computation Outsourcing

- In the case of secure computation outsourcing, additional security objectives emerge
  - because the computation is performed by external parties, there are no guarantees that the computation was run correctly (or even run at all)
  - thus, the output recipient would like to be able to verify that the returned result is correct
  - if verification succeeds, the probably that the output is incorrect should be negligible (in the security parameter κ)
  - the verification process should be much faster than running the computation locally
- The details of the security definition may differ depending on the problem formulation

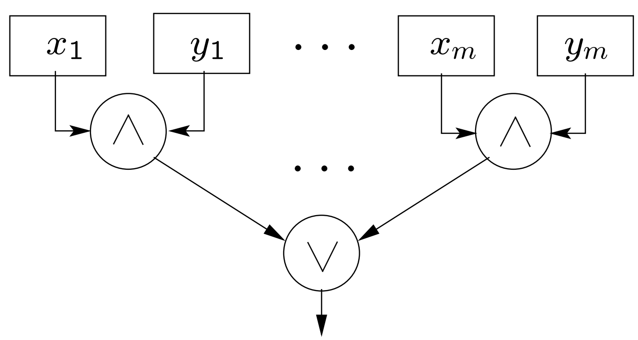
Secure Multi-Party Computation Techniques

- We'll next briefly discuss three major types of secure computation techniques
  - garbled circuit evaluation
    - two-party computation ($n = 2$)
  - linear secret sharing
    - multi-party computation ($n > 2$)
  - homomorphic encryption
    - two- or multi-party computation ($n \geq 2$)

Garbled Circuit Evaluation

- SMC based on garbled circuit evaluation involves two participants: circuit garbler and circuit evaluator

- The function to be computed is represented as a Boolean circuit
  - typically we'll use binary (two input and one output bits) gates and negation gates
  - example:



Garbled Circuit Evaluation

- The garbler takes a Boolean circuit and associates two random labels $\ell_i^0, \ell_i^1 \in \{0,1\}^\kappa$ with each circuit's wire i
  - ℓ_i^0 is associated with value 0 of the wire and ℓ_i^1 with value 1
  - given ℓ_i^b, it is not possible to determine what b is
- The garbler also encodes each gate
  - suppose a binary gate g has input wires i and j and output wire k
  - the garbler uses encryption to enable recovery of $\ell_k^{g(b_i,b_j)}$ given $\ell_i^{b_i}$ and $\ell_j^{b_j}$
- The evaluator obtains appropriate labels for the input wires and evaluates the garbled circuit one gate at a time
  - the evaluator sees labels, but doesn't know their meaning

Garbled Circuit Evaluation

- The evaluator obtains labels for the input wires as follows:
  - the garbler knows its input and simply sends the right labels for its input wires to the evaluator
  - to obtain labels corresponding to its own input, the evaluator engages in the 1-out-of-2 oblivious transfer (OT) with the garbler
    - it allows the evaluator to retrieve one out of two labels for each of its input wires, while the garbler learns nothing
- The basic technique is secure in the presence of semi-honest garbler and malicious evaluator
  - it can be extended to be secure in the malicious model using additional techniques

SMC based on Secret Sharing

- An alternative technique is to use threshold linear secret sharing for secure multi-party computation
  - (n, t)-threshold secret sharing allows secret s to be secret-shared among n parties such that:
    - no coalition of t or fewer parties can recover any information about s
    - $t + 1$ or more shares can be used to efficiently reconstruct s
  - information-theoretic security (i.e., independent of security parameters) is achieved
  - linear secret sharing allows a linear combination of secret-shared values to be computed by each party locally on its shares
    - this includes (integer) addition, subtraction, and multiplication by a known integer

SMC based on Secret Sharing

- Using secret sharing for secure multi-party computation
  - multiplication of secret-shared (integer) values requires interaction and is considered to be a basic building block (one elementary operation)
  - common implementations of multiplication in the semi-honest model require that $t < n/2$
    - e.g., we could use (3, 1), (5, 2), etc. threshold secret sharing
  - examples:
    - let $[x]$ denote that the value of x is protected/secret-shared
    - is $2[x] - 5[y]$ interactive computation? is $2[x][y]$?

SMC based on Secret Sharing

- Implementation of other operations is more complex and is typically composed of elementary operations
  - function representation expressed in terms of additions/subtractions and multiplications is called an arithmetic circuit
- Performance of any function in this framework is then measured in terms of
  - elementary interactive operations
  - sequential interactive operations or rounds

SMC based on Secret Sharing

- SMC based on secret sharing supports the flexible setup with three groups of participants:
  - each data owners secret-shares its private input among the computational parties prior to the computation
  - the computational parties evaluate the function on secret-shared data
  - the computational parties communicate their shares of the result to output recipients who locally reconstruct the output
- A number of techniques are available to strengthen the security guarantees to hold in the malicious model

SMC based on Homomorphic Encryption

- Homomorphic encryption is another technique that allows for securely evaluating general functionalities
  - it is a special type of encryption that, given ciphertexts, permits computation on the underlying plaintexts

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

  - homomorphic encryption enables computation on encrypted data and results in efficient protocols for certain problems

SMC based on Homomorphic Encryption

- Of most significant interest to us is public-key semantically-secure homomorphic encryption
  - a public-key encryption scheme uses a public-private key pair (pk, sk) and consists of three algorithms $\mathsf{Gen}(1^\kappa) \to (pk, sk)$, $\mathsf{Enc}(pk, m) \to c$, and $\mathsf{Dec}(sk, c) \to m \cup \perp$.
  - additional algorithm(s) specify how to use homomorphic properties
  - semantic security means that no information of any kind about plaintexts can be learned from the corresponding ciphertexts
    - this is true even in the presence of adversaries with large capabilities

SMC based on Homomorphic Encryption

- We'll look at two types of public-key homomorphic encryption
- The first type is called partially homomorphic encryption (or just HE for short) and comes with one homomorphic operation
  - of most significant importance to us is the ability to add (integer) values inside ciphertexts
  - we have $\mathsf{Enc}_{pk}(m_1) \cdot \mathsf{Enc}_{pk}(m_2) = \mathsf{Enc}_{pk}(m_1 + m_2)$
  - which in turn implies $\mathsf{Enc}_{pk}(m)^c = \mathsf{Enc}_{pk}(m \cdot c)$
  - Paillier encryption scheme (1999) is a popular cryptosystem of this type

SMC based on Homomorphic Encryption

- To enable secure computation using homomorphic encryption that supports addition, we also need to be able to implement other operations
  - multiplication can be implemented as an interactive protocol between the participants
  - addition/subtraction and multiplication alone are sufficient for supporting any computable function
  - optimized implementations for common operations are available
- Also, we'll often need to use (n, t)-threshold homomorphic encryption
  - similar to secret sharing, the private key is split into n shares
  - $t + 1$ or more shares are needed for decryption
  - Paillier encryption is available in the threshold version for any $t < n$

SMC based on Homomorphic Encryption

- The second type is called fully homomorphic encryption (FHE)
  - it supports two types of operations on ciphertexts: addition and multiplication
  - this type enables any function to be evaluated on encrypted data
  - this is suitable for secure computation outsourcing to a single server
- The drawback of FHE is its speed
  - it is currently not suitable for moderate to large functions or amounts of data

Summary of SMC Techniques

- The three types of SMC techniques described so far can be used to evaluate any function securely
- A large number of custom protocols for specific functions also exist
  - example: private set intersection
  - these can combine the above techniques or use custom approaches
  - the goal of custom protocols is to outperform general solutions
- The same applies to verification of outsourced computation:
  - general approaches are known, but constructions specific to some function target efficiency