Cryptographic Techniques in Privacy-Preserving Data Mining

Helger Lipmaa

University College London

ECML/PKDD 2006 Tutorial

Outline

Disclaimer

Disclaimer: I am not a data miner.

Privacy-Preserving Data Mining: Motivation

- Goal of DM: to build models of real data
- Problem of DM: real data is too valuable and thus difficult to obtain
- Solution: add privacy. Only information that is really necessary will be published. E.g.,
  - Parties learn only average values of entries
  - Linear classification: parties learn only the classifiers of new data

World I: Data Mining

- Goal: to model data
- Many methods are efficient only with "real data" that has redundancy, good structure etc
  - Data compression, many algorithms of data mining, special methods of machine learning...
  - Random data cannot be compressed and does not have small-sized models
- Conclusion: world I is data dependent
  - Look at the disclaimer

World II: Cryptography

- General goal: secure (confidential, authentic, ...) communication
- Subgoal: to hide properties of data
- For example, oblivious transfer:
  - Alice has input $i \in [n]$, Bob has n strings $D_1, \ldots, D_n$
  - Alice obtains D_i
  - Cryptographic goal: Alice obtains no more information. Bob obtains no information at all
- Since cryptographic algorithms must hide (most of the) data, they must be data independent
  - A few selected additional properties like the length of the input may be leaked if hiding such properties is too expensive

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - Information retrieval. It is a "trivial" task to retrieve the ith element D_i of a database D
  - Oblivious transfer:
    - Database server's computation is $\Omega(|D|)$
    - "Proof": If she does not do any work with the jth database element then she "knows" that $i \neq j$. QED.

Cryptographic PPDM: A Weird Coctail

- Goal: discover a model of the data, but nothing else
  - Both "model" and "nothing else" must be well-defined!
- Simplest example: find out average age of all patients (and nothing else)
- More complex example: publish average age of all patients with symptom X, where X is not public
  - I.e., database owner must not get to know X
- Another example: find 10 most frequent itemsets in the data
- In PPDM, data mining provides objectives, cryptography provides tools

Cryptographic PPDM: Good, Bad and Ugly

- Good: companies and persons may become more willing to participate in data mining
- Bad: already inefficient data mining algorithms become often almost intractable
  - Simpler tasks can still be done
- There is no ugly: it's a nice research area ☺
  - At this moment far from being practical, and thus offers many open problems

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions

Randomization Approach

- Much more popular in the data mining community, see Srikant's SIGKDD innovation award talk in KDD 2006, Gehrke's tutorial in KDD 2006, Xintao Wu's tutorial in ECML/PKDD 2006
- There are significant differences between cryptographic and randomization approaches!
- . . . and they are studied by completely different communities

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Randomization Approach: Short Overview

- Clients have data that is to be published and mined
- It is desired that one can build certain models of the data without violating the privacy of individual records
  - E.g., compute average age before getting to know the age of any one person
  - It is allowed to get to know the average age of say any three persons
- Untrusted publisher model: clients perturb their data and send their perturbed version to miner who mines the results
- Trusted publisher model: clients send original data to a TP, who perturbs it and sends the results to miner who mines the results

Cryptographic Approach: Short Overview

- Assume there are n parties (clients, servers, miners) who all have some private inputs x_i, and they must compute some private outputs $y_i = f_i(\vec{x})$
  - f_i etc are defined by the functionality we want to compute — by data miners
- Build a cryptographic protocol that guarantees that after some rounds, the ith party learns y_i and nothing else— with probability $1 - \epsilon$

Cryptographic vs Randomization Approach: Differences

- Who owns the database:
  - Randomization: randomized data is published, and the miner operates on the perturbed database without contacting any third parties
  - Cryptographic: depends on applications
    - Data is kept by a server, and the miner queries the server
    - Data is shared by several miners, who can only jointly mine it
    - ...

Cryptographic vs Randomization Approach: Differences

- Correctness:
  - Randomization:
    - Client "owns" a perturbed database, and must be able to compute (an approximation to) the desired output from it
  - Cryptographic:
    - Client can usually compute the precise output after interactive communicating with the server

Cryptographic vs Randomization Approach: Differences

- Privacy:
  - Randomization: one can usually only guarantee that the values of individual records are somewhat protected
    - E.g., in Randomized Response Technique, variance depends on the size of the population
    - Interval privacy, k-anonymity, ...
  - Cryptographic: one can guarantee that only the desired output will become known to the client
    - Protect everything as much as possible

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions

Cryptographic vs Randomization Approach: Differences

- Definitional:
  - Randomization: privacy definitions seem to be ad hoc (to a cryptographer)
  - Cryptographic:
    - A lot of effort has been put into formalizing the definitions of privacy, the definitions and their implications are well understood
    - Cryptographic community has invested dozens of man years to come up with correct definitions!

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Cryptographic vs Randomization Approach: Differences

- Efficiency:
  - Randomization: randomizing might be difficult but it is done once by the server; client's work is usually comparable to her work in the non-private case
    - Better efficiency, but privacy depends on data and predicate
  - Cryptographic: privatization overhead every single time when a client needs to obtain some data
    - Better privacy, but efficiency depends on *predicate*

Cryptographic vs Randomization Approach: Differences

- Communities:
  - Randomization: bigger community, people from the data mining community
    - Too many results to even mention...
    - Randomization is an optimization problem: tweak and your algorithm might work for some concrete data
  - Cryptographic: small community
    - Cryptographic approach is seen to be too resource-consuming and thus not worth the research time
    - Some people: Benny Pinkas, Kobby Nissim, Rebecca Wright and students, myself and Sven Laur, ...

Private Information Retrieval

- Alice (client) has index $i \in [n]$, Bob (database server) has database $D = (D_1, \ldots, D_n)$
- Functional goal: Alice obtains D_i, Bob does not have to obtain anything
- Cryptographic privacy goal I: Bob does not obtain any information about i
  - "Private information retrieval"
- Cryptographic privacy goal II: Alice does not obtain any information about D_j for any $j \neq i$
  - PIR + goal II = ("relaxed" secure) oblivious transfer
- Cryptographic security/correctness goal III: the string that Alice obtains is really equal to D_i
  - goal I + II + III = fully secure oblivious transfer

PIR: Computational vs Statistical Client-Privacy

- Privacy can be defined to be statistical or computational
- Statistical client-privacy:
  - Alice's messages that correspond to any two queries i_0 and i_1 come from similar distributions
  - Then even an unbounded adversary cannot distinguish between messages that correspond to any two different queries
    - Even if the queries i_0/i_1 are chosen by the adversary
- Well-known fact: communication of statistically client-private information retrieval with database D is at least $|D|$ bits.
- I.e., the trivial solution — Bob sends to Alice his whole database, Alice retrieves D_i — is also the optimal one

PIR: Computational Client-Privacy (Intuition)

- Computational client-privacy: no computationally bounded Bob can distinguish between the distributions corresponding to any two queries i_0 and i_1
- I.e., the distributions of Alice's messages $A(i_0)$ and $A(i_1)$ corresponding to i_0 and i_1 are computationally indistinguishable

PIR: Formal Definition of Client-Privacy

- Consider the next "game":
  - B picks two indices i_0 and i_1, and sends them to A
  - A picks a random bit $b \in \{0, 1\}$ and sends $A(i_b)$ to B
  - $B(i_0, i_1, A(i_b))$ outputs a bit b'
- B is successful if $b' = b$
- PIR is (ε, τ)-computationally client-private if no τ-time adversary B has better success than $|\varepsilon - 1/2|$
- If B tosses a coin then it has success $1/2$ and thus is a $(0, \tau)$-adversary for some small τ
- IND-CPA security: INDistinguishability against Chosen Plaintext Attacks

OT: Formal Definition of Server-Security

- Difference with client-privacy:
  - Client obtains an output D_i and thus can distinguish between databases D, D' with $D_i \neq D'_i$
    - This must be taken into account
  - We can achieve statistical server-privacy
    - With communication $\Theta(\log |D|)$
  - Since server gets no output, server-privacy=server-security
    - Recall goal III

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions
Private Information Retrieval
Scalar Product Computation

OT: Formal Definition of Server-Security

- Consider the next ideal world with a completely trusted third party T:
  - A sends her input i to T, B sends the database D to T (secretly, authenticatedly)
  - T sends D_i to A (secretly, authenticatedly)
- This clearly models what we want to achieve!
- A protocol is server-secure if:
  - For any attack that A can mount against B in the protocol, there exists an adversary A^* that can mount the same attack against B in the described ideal world
- Technical differences: real world is always asynchronous, but it does not matter here

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Note on Security Definitions

- Security definitions are uniform and modular, and remain the same for most protocols
- The previous definitions work for any two-party protocol where on client's input a and server's input b, client must obtain an output $f(a, b)$ for some f, and server must obtain no output
- Computational client-privacy: client's messages corresponding to any, even chosen-by-server, inputs a and a' must be computationally indistinguishable
- Statistical server-security: consider an ideal world where client gives a to T, server gives b to T and T returns $f(a, b)$ to client. Show that any attacker in real protocol can be used to attack the ideal world with comparable efficiency.

Tool: Additively Homomorphic Public-Key Crypto

- E is a semantically/IND-CPA secure public-key cryptosystem iff
  - Every user has a public key pk and secret key sk
  - Encryption is probabilistic: $c = E_{pk}(m; r)$ for some random bitstring r
  - Decryption is successful: $D_{sk}(E_{pk}(m; r)) = m$
  - Semantical/IND-CPA security: Distributions corresponding to the encryptions of any m_0 and m_1 are computationally indistinguishable

Tool: Additively Homomorphic Public-Key Crypto

- Additionally, E is additively homomorphic iff

$$D_{sk}(E_{pk}(m_1; r_1) \cdot E_{pk}(m_2; r_2)) = m_1 + m_2 \ ,$$

  where plaintexts reside in some finite group $\mathcal{M}$ and ciphertexts reside in some finite group $\mathcal{C}$.
    - Thus also $D_{sk}(E_{pk}(m; r)^a) = am$
- **Fact**: such IND-CPA secure public-key cryptosystems exist and are well-known [Paillier, 1999]
    - There $\mathcal{M} = \mathbb{Z}_N$, $\mathcal{C} = \mathbb{Z}_{N^2}$ for some large composite $N = pq$
    - If you care: $E_{pk}(m; r) = (1 + mN)r^N \mod N^2$
    - **Theorem** Paillier cryptosystem is IND-CPA secure if it is computationally difficult to distinguish the Nth random residues modulo N^2 from random integers modulo N^2

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $D = (D_1, \ldots, D_n)$ where $D_j \in \mathbb{Z}_N$

1. Alice generates a new public/private key pair (pk, sk) for an additively homomorphic secure public-key cryptosystem E
2. Alice generates her message $a \leftarrow E_{pk}(i; *)$ and sends $A(i) \leftarrow (pk, a)$ to Bob. Bob stops if pk is not a valid public key or a is not a valid ciphertext.
3. Bob does for every $j \in \{1, \ldots, n\}$:
    - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(D_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $D_i = D_{sk}(b_i)$

[Aiello, Ishai, Reingold, Eurocrypt 2001]

AIR PIR: Correctness/Security

- Bob does for every $j \in \{1, \dots, n\}$:
  - Set $b_j \leftarrow (a/E_{pk}(j;1))^* \cdot E_{pk}(D_j;*)$
- Since $a = E_{pk}(i;*)$,

$$b_j = (E_{pk}(i;*)/E_{pk}(j;1))^* \cdot E_{pk}(D_j;*)$$

- Because E is additively homomorphic,

$$b_j = (E_{pk}(i-j;*))^* \cdot E_{pk}(D_j;*) = (E_{pk}(* \cdot (i-j);r)) \cdot E_{pk}(D_j;*)$$

  for some r
- If $i = j$ then

$$b_j = E_{pk}(0;r) \cdot E_{pk}(D_j;*) = E_{pk}(D_j;*)$$

  and thus $D_{sk}(b_j) = D_j$
- Thus Alice obtains D_i

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions
Private Information Retrieval
Scalar Product Computation

AIR PIR: Correctness/Security

- Bob does for every $j \in \{1, \dots, n\}$:
  - Set $b_j \leftarrow (a/E_{pk}(j;1))^* \cdot E_{pk}(D_j;*)$
- Since $a = E_{pk}(i;*)$ then

$$b_j = (E_{pk}(i;*)/E_{pk}(j;1))^* \cdot E_{pk}(D_j;*)$$

- Because E is additively homomorphic then

$$b_j = (E_{pk}(i-j;*))^* \cdot E_{pk}(D_j;*) = (E_{pk}(*(i-j);r)) \cdot E_{pk}(D_j;*)$$

  for some r
- If $\gcd(i-j, N) = 1$ then $* \cdot (i-j) = *$ is a random element of $\mathbb{Z}_N$ and thus

$$b_j = E_{pk}(*;r) \cdot E_{pk}(D_j;*) = E_{pk}(*;*) \ ,$$

  and thus $D_{sk}(b_j) = *$, i.e., b_j gives no information about D_j
- Thus Alice obtains D_i and nothing else!

AIR 1-out-of-n PIR: Security Properties

- Alice's query is computationally "IND-CPA" private: Bob sees its encryption, and the cryptosystem is IND-CPA private by assumption
- Bob's database is statistically private: Alice sees an encryption of D_i together with $n-1$ encryptions of random strings
  - We can construct a simulator who, only knowing D_i and nothing else about Bob's database, sends

$$(E_{pk}(*;*),\dots,E_{pk}(*;*),E_{pk}(D_i;*),E_{pk}(*;*),\dots,E_{pk}(*;*))$$

    to Alice.
  - Simulator's output is the same as honest Bob's output and was constructed, only knowing $D_i \Rightarrow$ protocol is statistically private for Bob

AIR PIR: Full Server-Security Proof

Proof.

We must assume that simulator is unbounded (this is ok since the attacker may also be unbounded, and thus simulator may need a lot of time to check his work).Alice sends (pk, a) to Bob. Unbounded simulator finds corresponding sk and computes $i^* \leftarrow D_{sk}(a)$. If there is no such sk or a is not a valid ciphertext then simulator returns "reject".Otherwise, simulator sends i^* to T. Bob sends D to T. T sends D_{i^*} to simulator. Simulator sends

$$(E_{pk}(*;*),\dots,E_{pk}(*;*),E_{pk}(D_i;*),E_{pk}(*;*),\dots,E_{pk}(*;*))$$

to Alice.Clearly in this case, even a malicious Alice sees messages from the same distribution as in the real world. □

AIR PIR: Security Fineprints

- It takes some additional work to ascertain that the protocol is secure if i is chosen maliciously such that for some $j \in [n]$, $\gcd(i - j, N) > 1$.
- We have a relaxed-secure oblivious transfer protocol: privacy of both parties is guaranteed but Alice has no guarantee that b_i decrypts to anything sensible

AIR 1-out-of-n PIR: Efficiency

- Alice's computation: one encryption at first, and one decryption at the end. Good
- Bob's computation: $2n$ encryptions, n exponentiations, etc. Bad but cannot improve to $o(n)$!
- Communication: Alice sends 1 ciphertext, Bob sends n ciphertexts, in total $n + 1$ ciphertexts. Bad, can be improved.
- One encryption $\approx$ one exponentiation
  - On 1024-bit integers, ≈ 512 1024-bit multiplications or $\approx 512^2$ additions

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated
- With a well-constructed protocol, proofs can become straightforward
  - Existing designs can be (hopefully?) explained to non-specialists
- Even for really simple tasks, computational overhead can crash the party

More Efficient PIRs: Computation

- As said previously, Bob must do something with every database element
- However, this something doesn't have to be public-key encryption — and symmetric key encryption (block ciphers, . . .) is often 1000 times faster
- Simple idea [Naor, Pinkas]: every database element is masked by pseudorandom sequence and then transferred to Alice. Alice obtains $\log n$ symmetric keys needed to unmask D_i by doing $\log n$ 1-out-of-2 PIR-s with Bob.
- Needs n symmetric-key operations and $\log n$ public-key encryptions

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob and Bob responds with D_i. I.e., $\log n + length(D_i)$ bits.
- Also in PIR, the communication is lower bounded by $\log n + length(D_i)$ bits.
- [Lipmaa, 2005]: A PIR with communication $O(\log^2 n + length(D_i) \log n)$
- [Gentry, Ramzan, 2005]: communication $O(\log n + length(D_i))$ but much higher Alice-side computation
- Open problem: construct a PIR with sublinear communication $o(n)$ where server does $\ll n$ public-key operations

Private Scalar Product

- Goal: Given Alice's vector $a = (a_1, \dots, a_n)$ and Bob's vector $b = (b_1, \dots, b_n)$, Alice needs to know $a \cdot b = \sum a_i b_i$
- Cryptographic privacy goals: Alice only learns $a \cdot b$, Bob learns nothing
- Scalar product is another subprotocol that is often needed in data mining
  - Finding if a pattern occurs in a transaction is basically a scalar product computation
  - Etc etc
- Many "private" scalar product products have been proposed in the data mining community, but they are (almost) all insecure

GLLM04 Private Scalar Product Protocol

- Assume E is additively homomorphic, $E_{pk}(m_1; r_1)E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 r_2)$
- Alice has $a = (a_1, \ldots, a_n)$, Bob has $b = (b_1, \ldots, b_n)$
- For $i \in \{1, \ldots, n\}$, Alice sends to Bob $A_i \leftarrow E_{pk}(a_i; *)$
- Bob computes $B \leftarrow \prod A_i^{b_i} \cdot E_K(0; *)$ and sends B to Alice
- Alice decrypts B
- Correct: $B = \prod A_i^{b_i} \cdot E_{pk}(0; *) = \prod E_{pk}(a_i; *)^{b_i} \cdot E_{pk}(0; *) = \prod E_{pk}(a_i b_i; \ldots) \cdot E_{pk}(0; *) = E_{pk}(\sum a_i b_i; \ldots) \cdot E_{pk}(0; *) = E_{pk}(\sum a_i b_i; *)$
- Since B is a random encryption of $\sum a_i b_i$, then this protocol is also private
- See [Goethals, Laur, Lipmaa, Mielikäinen 2004] for more

GLLM04: Complexity

1. For $i \in \{1, \ldots, n\}$, Alice sends to Bob $A_i \leftarrow E_{pk}(a_i; *)$
2. Bob computes $B \leftarrow E_K(0; *) \cdot \prod_{i=1}^{n} A_i^{b_i}$ and sends B to Alice
3. Alice decrypts B

Alice does $n + 1$ decryptions
Bob does n exponentiations
One can optimize it significantly, see [GLLM04]

Homomorphic Protocols: SWOT Analysis

- Bad:
  - Applicable mostly only if client's/server's outputs are affine functions of their inputs:
    - E.g., scalar product
  - Some additional functionality can be included:
    - PIR uses a selector function: Client gets back some value if her input is equal to some other specific value
- Good:
  - "Efficient" whenever applicable
  - Security proofs are standard and modular, client's privacy comes directly from the security of the cryptosystem, sender's privacy is also often simply proven
  - Easy to implement (if you have a correct implementation of the cryptosystem)

The Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive
  - E.g., (kernel) adatron algorithm
- Condition $a > 0$ can be checked by using affine operations but it is cumbersome and relatively inefficient
- Thus, in many protocols we need tools that make it possible to efficiently implement non-affine functionalities
- Circuit evaluation: a well-known tool that is efficient whenever the functionality has a small Boolean complexity

Setting: Recap

- Two parties, Alice and Bob, have inputs a and b, correspondingly
- Functionality: Alice learns $A(a, b)$, Bob learns $B(a, b)$
- Neither party learns more in the semihonest model, i.e., when Alice and Bob follow the protocol but try to devise new information from what they see
- Can decompose: First run a protocol where Alice learns $A(a, b)$ and Bob learns nothing, then a second protocol where Bob learns $B(a, b)$.
- Thus we will consider the case where $B(a, b) = \perp$
- Wlog, $A(a, b) : \{0, 1\}^m \times \{0, 1\}^n \to \{0, 1\}$ /* run x protocols in parallel if output is longer */

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions

High level idea

- Every function $A : \{0, 1\}^m \times \{0, 1\}^n \to \{0, 1\}$ can be decomposed as a Boolean circuit
- Idea:
  - Bob garbles the Boolean circuit for A, together with his inputs, and handles the circuit to Alice
  - Alice obtains from Bob the key that corresponds to one possible Alice's input
  - Alice "runs" this circuit on this key
  - Alice obtains from Bob the real output, corresponding to the garbled output
- Bob garbles the circuit, corresponding to his concrete input b
- Alice should not be able to obtain Bob's input b or run the circuit on two different inputs a, a'

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Example

- Millionaire's problem: Who has more toys?
- I.e., $A(a, b) = 1$ iff $a > b$ in $\mathbb{Z}_{2^\ell}$
- Boolean way:

$$(a_{\ell-1} = 1 \wedge b_{\ell-1} = 0) \vee (a_{\ell-1} = b_{\ell-1} \wedge a_{\ell-2} = 1 \wedge b_{\ell-2} = 0) \vee \dots$$

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions

Obtaining The Input Key

- Alice has m inputs a_i.
- Bob generates $2m$ keys K_{i0} and K_{i1}, $\forall i \in [m]$
- For $i \in [m]$, Alice uses an $\binom{2}{1}$-OT to obtain $K_{i\alpha_i}$

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Obtaining The Output Key

- After running the circuit, Alice has exactly one output key K_{out}
- Assume that Bob has before also transferred $E_{K^i_{out}}(answer_i)$ for all possible output keys/corresponding answers

Garbling The Circuit

- Every gate ψ is constructed so that if you know input keys then you get to know output keys
- E.g., $\wedge$ gate:
  - Alice gets to know the key $K^{\psi}_{out,1}$ corresponding to 1 if both his keys correspond to the 1-input keys $K^{\psi}_{1,1}$, $K^{\psi}_{2,1}$ of this gate
  - Otherwise, Alice gets to know the key corresponding to 0
  - Alice should not get to know to what does the new key correspond
- Basic idea: encrypt K^{ψ}_{out} by using K^{ψ}_1, K^{ψ}_2. Store a randomly ordered table table that corresponds to $E_{K^{\psi}_{1,i},K^{\psi}_{2,j}}(K^{\psi}_{out,i\wedge j})$ for $i, j \in \{0, 1\}$
- Call this table a Yao gate
- Alice later tries to decrypt all four values $\Leftarrow$ It is needed that one can detect that $K^{\psi}_{out,i\wedge j}$ is correct

Construction

- Bob creates key pairs for all bits of all inputs and for each "wire" of the circuit
- Given these key pairs, Bob turns gates into Yao gates.
- Bob gives Alice all Yao gates, keys corresponding to his inputs.
- Alice obtains keys corresponding to her inputs.
- Alice computes Yao gate, until she gets the output keys.
- Alice converts output keys to correct answers.

What if Bob cheats?

- Recent research (Katz-Ostrovsky, 2004) etc: it is possible to design two-party protocols, secure in the malicious model, for any "computable" A in five rounds
- However: is it practical?
  - Circuit evaluation is not even practical in semihonest model, except for functions of special type
  - For protocols, seen previously, homomorphic solutions are much more efficient
- Circuit evaluation is practical if the circuit is small: e.g., computing a XOR of two inputs etc.

Secret Sharing: Multi-Party Model

- Sharing a secret X: X is shared between different parties so that only legitimate coalitions of parties can reconstruct it, and any smaller coalition has no information about X
- Well-known, well-studied solutions starting from [Shamir 1979]
- Multi-Party Computation:
  - n parties secretly share their inputs
  - The protocol is executed on shared inputs
  - Intermediate values and output will be shared
  - Only legitimate coalitions can recover the output
- MPC: well-known, well-studied since mid 80-s
- Contemporary solutions quite efficient
- Needs more than two parties: 2/3rd fraction of parties must be honest ☹

Disclaimer
Motivation And Introduction
Some Simple PPDM Algorithms
Circuit Evaluation: Tool For Complex Protocols
Secret Sharing/MPC And Combining Tools
Conclusions

Combining Tools

- Most algorithms are not affine and have a high Boolean complexity
- Many algorithms can be decomposed into smaller pieces, such that some pieces are affine, some have low Boolean complexity
- Solve every piece of the algorithm by using an appropriate tool: homomorphic protocols, circuit evaluation or MPC
- Internal states of the algorithm should not become public and must therefore be secretly shared between different participants
- All more complex cryptographic PPDM protocols have this structure, see [Pinkas, Lindell, Crypto 2000] or [Laur, Lipmaa, Mielikäinen, KDD 2006]

Helger Lipmaa Cryptographic Techniques in Privacy-Preserving Data Mining

Combining Example: Private Kernel Perceptron

Kernel Perceptron

Input: Kernel matrix K, class labels $\vec{y} \in \{-1, 1\}^n$.
Output: A weight vector $\vec{a} \in \mathbb{Z}^n$.

1. Set $\vec{a} \leftarrow \vec{0}$.
2. **repeat**
    1. **for** $i = 1$ **to** n **do**
        1. **if** $y_i \cdot \sum_{j=1}^{n} k_{ij}\alpha_j \leq 0$ **then** $\alpha_i \leftarrow \alpha_i + y_i$
    2. **end for**
3. **until** convergence

Conclusions

- Cryptography and Data-Mining — two different worlds
- Cryptographic PPDM: data itself is not made public, different parties obtain their values by interactively communicating with the database servers
- Security definitions are precise and well-understood
- Security guarantees are very strong: no adversary working in time 2^{80} can violate privacy with probability $\geq 2^{-80}$
- Computational/communication overhead makes many protocols impractical
- Constructing a protocol that is practical enough may require breakthroughs in cryptography and/or data mining

Further work?

- From cryptographic side:
  - Construct faster public-key cryptosystems
  - Superhomomorphic public-key cryptosystems that allow to do more than just add on ciphertexts
  - PIR with $o(n)$ communication and $o(n)$ public-key operations
- From data mining side:
  - Construct privacy-friendly versions of various algorithms that are easy to implement cryptographically
  - E.g.: a version of SVM algorithm that is faster than adatron but privacy-friendly

Questions?

- Slides will be soon available from http://www.adastral.ucl.ac.uk/~helger