Cryptographic Techniques in Privacy-Preserving Data Mining

Helger Lipmaa

University College London

Estonian Theory Days, 28.01.2007, Tutorial

Outline

1 Motivation And Introduction

Outline

1. Motivation And Introduction

2. Some Simple PPDM Algorithms
   - Private Information Retrieval
   - Scalar Product Computation

Outline

1. Motivation And Introduction
2. Some Simple PPDM Algorithms
   - Private Information Retrieval
   - Scalar Product Computation
3. Circuit Evaluation: Tool For Complex Protocols

Outline

1. Motivation And Introduction
2. Some Simple PPDM Algorithms
   - Private Information Retrieval
   - Scalar Product Computation
3. Circuit Evaluation: Tool For Complex Protocols
4. Secret Sharing/MPC And Combining Tools

Outline

1. Motivation And Introduction
2. Some Simple PPDM Algorithms
   - Private Information Retrieval
   - Scalar Product Computation
3. Circuit Evaluation: Tool For Complex Protocols
4. Secret Sharing/MPC And Combining Tools
5. Conclusions

Presentation History

- This tutorial is based on three earlier tutorials in
  - ECML/PKDD 2006, the leading European ML/DM conference
  - Inscrypt 2006, a new data security conference
  - University of Bristol, 2007 — for a mixed audience of cryptographers (a majority) and one data miner

Presentation History

- This tutorial is based on three earlier tutorials in
  - ECML/PKDD 2006, the leading European ML/DM conference
  - Inscrypt 2006, a new data security conference
  - University of Bristol, 2007 — for a mixed audience of cryptographers (a majority) and one data miner
- The ECML/PKDD 2006 tutorial was aimed for data miners, and thus spelled out a lot of small cryptographic details that a cryptographer knows by heart. On the other hand, I assumed that the audience knows ML/DM.

Presentation History

- This tutorial is based on three earlier tutorials in
  - ECML/PKDD 2006, the leading European ML/DM conference
  - Inscrypt 2006, a new data security conference
  - University of Bristol, 2007 — for a mixed audience of cryptographers (a majority) and one data miner
- The ECML/PKDD 2006 tutorial was aimed for data miners, and thus spelled out a lot of small cryptographic details that a cryptographer knows by heart. On the other hand, I assumed that the audience knows ML/DM.
- The current slides still spell out a lot of trivial cryptographic details but I will skip many of them

Presentation History

- This tutorial is based on three earlier tutorials in
  - ECML/PKDD 2006, the leading European ML/DM conference
  - Inscrypt 2006, a new data security conference
  - University of Bristol, 2007 — for a mixed audience of cryptographers (a majority) and one data miner
- The ECML/PKDD 2006 tutorial was aimed for data miners, and thus spelled out a lot of small cryptographic details that a cryptographer knows by heart. On the other hand, I assumed that the audience knows ML/DM.
- The current slides still spell out a lot of trivial cryptographic details but I will skip many of them
- Unless you object!

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, . . .

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, . . .
- 02.2004. . . 07.2007: 3.5 year grant on PPDM from Finnish Academy of Sciences, for Sven's PhD studies (Sven still there)

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, ...
- 02.2004...07.2007: 3.5 year grant on PPDM from Finnish Academy of Sciences, for Sven's PhD studies (Sven still there)
- 01.2006...12.2007: 2 year grant on PPDM from Estonian Science Foundation

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, . . .
- 02.2004. . . 07.2007: 3.5 year grant on PPDM from Finnish Academy of Sciences, for Sven's PhD studies (Sven still there)
- 01.2006. . . 12.2007: 2 year grant on PPDM from Estonian Science Foundation
- Soon applying for a grant in the UK

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, ...
- 02.2004...07.2007: 3.5 year grant on PPDM from Finnish Academy of Sciences, for Sven's PhD studies (Sven still there)
- 01.2006...12.2007: 2 year grant on PPDM from Estonian Science Foundation
- Soon applying for a grant in the UK
  - Interest from BT (British Telecom), possibility to hire new postdocs/PhD students

Research History

- Spring 2003: Sven Laur visits me in Finland for a semester, joint seminars with Heikki Mannila, ...
- 02.2004...07.2007: 3.5 year grant on PPDM from Finnish Academy of Sciences, for Sven's PhD studies (Sven still there)
- 01.2006...12.2007: 2 year grant on PPDM from Estonian Science Foundation
- Soon applying for a grant in the UK
  - Interest from BT (British Telecom), possibility to hire new postdocs/PhD students
- Most of the research is a joint work with Sven Laur

Privacy-Preserving Data Mining: Motivation

- Goal of DM: to build models of real data

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

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

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
- Many industrial/... applications

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
- Many industrial/... applications
  - Medical databases: mining necessary to design new drugs/..., but many privacy issues — sharing data may even be illegal

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
- Many industrial/... applications
    - Medical databases: mining necessary to design new drugs/..., but many privacy issues — sharing data may even be illegal
    - Loyal customers: pooling databases helps to provide better services. Many privacy issues

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
- Many industrial/... applications
  - Medical databases: mining necessary to design new drugs/..., but many privacy issues — sharing data may even be illegal
  - Loyal customers: pooling databases helps to provide better services. Many privacy issues
  - ...

World I: Data Mining

- Goal: to model data

World I: Data Mining

- Goal: to model data
- Typical task: given a database of transactions, find most frequent patterns

World I: Data Mining

- Goal: to model data
- Typical task: given a database of transactions, find most frequent patterns
- Many methods are efficient only with "real data" that has redundancy, good structure etc

World I: Data Mining

- Goal: to model data
- Typical task: given a database of transactions, find most frequent patterns
- 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

World I: Data Mining

- Goal: to model data
- Typical task: given a database of transactions, find most frequent patterns
- 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
- Having real data to test your algorithms with is important

World I: Data Mining

- Goal: to model data
- Typical task: given a database of transactions, find most frequent patterns
- 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
- Having real data to test your algorithms with is important
- Data representation is important

World I: Data Mining

- Conclusion: world I is data dependent

World II: Cryptography

- General goal: secure (confidential, authentic, . . .) communication

World II: Cryptography

- General goal: secure (confidential, authentic, . . .) communication
- Subgoal: to hide properties of data

World II: Cryptography

- General goal: secure (confidential, authentic, . . .) communication
- Subgoal: to hide properties of data
- Since cryptographic algorithms must hide (most of the) data, they must be data independent

World II: Cryptography

- General goal: secure (confidential, authentic, ...) communication
- Subgoal: to hide properties of data
- 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
- For example, oblivious transfer:
  - Alice has input $i \in [n]$, Bob has n strings $\mathcal{D}_1, \ldots, \mathcal{D}_n$

World II: Cryptography

- General goal: secure (confidential, authentic, ...) communication
- Subgoal: to hide properties of data
- 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
- For example, oblivious transfer:
  - Alice has input $i \in [n]$, Bob has n strings $\mathcal{D}_1, \ldots, \mathcal{D}_n$
  - Alice obtains $\mathcal{D}_i$

World II: Cryptography

- General goal: secure (confidential, authentic, ...) communication
- Subgoal: to hide properties of data
- 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
- For example, oblivious transfer:
  - Alice has input $i \in [n]$, Bob has n strings $\mathcal{D}_1, \dots, \mathcal{D}_n$
  - Alice obtains $\mathcal{D}_i$
  - Cryptographic goal: Alice obtains no more information. Bob obtains no information at all

World II: Cryptography

- General goal: secure (confidential, authentic, ...) communication
- Subgoal: to hide properties of data
- 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
- For example, oblivious transfer:
  - Alice has input $i \in [n]$, Bob has n strings $\mathcal{D}_1, \dots, \mathcal{D}_n$
  - Alice obtains $\mathcal{D}_i$
  - Cryptographic goal: Alice obtains no more information. Bob obtains no information at all
- OT: everything but $\mathcal{D}_i$ (and n) should be private

World II: Cryptography

- Cryptography is usually inefficient with large amount of data

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$
  - Oblivious transfer:
    - Database server's computation is $\Omega(|\mathcal{D}|)$

World II: Cryptography

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

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$
  - Oblivious transfer:
    - Database server's computation is $\Omega(|\mathcal{D}|)$
    - "Proof": If she does not do any work with the jth database element then she "knows" that $i \neq j$. QED.
    - Of course, the constant in Ω is important!

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$
  - Oblivious transfer:
    - Database server's computation is $\Omega(|\mathcal{D}|)$
    - "Proof": If she does not do any work with the jth database element then she "knows" that $i \neq j$. QED.
    - Of course, the constant in Ω is important!
    - $|\mathcal{D}|$ public-key operations is 1000 times slower than $|\mathcal{D}|$ secret-key operations

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$
  - Oblivious transfer:
    - Database server's computation is $\Omega(|\mathcal{D}|)$
    - "Proof": If she does not do any work with the jth database element then she "knows" that $i \neq j$. QED.
    - Of course, the constant in Ω is important!
    - $|\mathcal{D}|$ public-key operations is 1000 times slower than $|\mathcal{D}|$ secret-key operations
    - In addition, one can do the majority of the work "offline"

World II: Cryptography

- Cryptography is usually inefficient with large amount of data
- Example:
  - It is a "trivial" task to retrieve the ith element $\mathcal{D}_i$ of a database $\mathcal{D}$
  - Oblivious transfer:
    - Database server's computation is $\Omega(|\mathcal{D}|)$
    - "Proof": If she does not do any work with the jth database element then she "knows" that $i \neq j$. QED.
    - Of course, the constant in Ω is important!
    - $|\mathcal{D}|$ public-key operations is 1000 times slower than $|\mathcal{D}|$ secret-key operations
    - In addition, one can do the majority of the work "offline"
    - Total work is still linear!

Cryptographic PPDM: A Weird Coctail

- Goal: discover a model of the data, but nothing else

Cryptographic PPDM: A Weird Coctail

- Goal: discover a model of the data, but nothing else
    - Both "model" and "nothing else" must be well-defined!

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)

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

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

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

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
- Find a model of DNA sequences for patients who have AIDS and are over 40

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
- Find a model of DNA sequences for patients who have AIDS and are over 40
- In PPDM, data mining provides objectives, cryptography provides tools (traditionally!)

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 ☺

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

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
  - Many of the open problems are really-really tough — is it good, bad or ugly?

Randomization Approach

- Extremely 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

Randomization Approach

- Extremely 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!

Randomization Approach

- Extremely 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

Randomization Approach: Short Overview

- Clients have data that is to be published and mined

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

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

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

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

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 $y_i = f_i(\vec{x})$

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

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

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:

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 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

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

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

- 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:

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 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

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 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

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

Cryptographic vs Randomization Approach: Differences

- Definitional:

Cryptographic vs Randomization Approach: Differences

- Definitional:
    - Randomization: privacy definitions seem to be ad hoc (to a cryptographer)

Cryptographic vs Randomization Approach: Differences

- Definitional:
  - Randomization: privacy definitions seem to be ad hoc (to a cryptographer)
  - Cryptographic:

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 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!

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
  - Cryptographic: privatization overhead every single time when a client needs to obtain some data

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:

Cryptographic vs Randomization Approach: Differences

- Communities:
  - Randomization: bigger community, people from the data mining community

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 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 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 $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_n)$
- Functional goal: Alice obtains $\mathcal{D}_i$, Bob does not have to obtain anything
- Cryptographic privacy goal I: Bob does not obtain any information about i
    - "Private information retrieval"

Private Information Retrieval

- Alice (client) has index $i \in [n]$, Bob (database server) has database $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$
- Functional goal: Alice obtains $\mathcal{D}_i$, 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 $\mathcal{D}_j$ for any $j \neq i$
  - PIR + goal II = ("relaxed" secure) oblivious transfer

Private Information Retrieval

- Alice (client) has index $i \in [n]$, Bob (database server) has database $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_n)$
- Functional goal: Alice obtains $\mathcal{D}_i$, 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 $\mathcal{D}_j$ 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 $\mathcal{D}_i$
  - goal I + II + III = secure oblivious transfer

PIR: Computational vs Statistical Client-Privacy

- Privacy can be defined to be statistical or computational

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

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 $\mathcal{D}$ is at least $|\mathcal{D}|$ bits.

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 $\mathcal{D}$ is at least $|\mathcal{D}|$ bits.
  - I.e., the trivial solution — Bob sends to Alice his whole database, Alice retrieves $\mathcal{D}_i$ — 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":

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'

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 $|2\varepsilon - 1|$

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 $|2\varepsilon - 1|$
- If B tosses a coin then it has success $1/2$ and thus is a $(0, \tau)$-adversary for some small τ

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 $|2\varepsilon - 1|$
- 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:

OT: Formal Definition of Server-Security

- Difference with client-privacy:
  - Client obtains an output $\mathcal{D}_i$ and thus can distinguish between databases $\mathcal{D}, \mathcal{D}'$ with $\mathcal{D}_i \neq \mathcal{D}'_i$

OT: Formal Definition of Server-Security

- Difference with client-privacy:
  - Client obtains an output $\mathcal{D}_i$ and thus can distinguish between databases $\mathcal{D}, \mathcal{D}'$ with $\mathcal{D}_i \neq \mathcal{D}'_i$
    - This must be taken into account

OT: Formal Definition of Server-Security

- Difference with client-privacy:
  - Client obtains an output $\mathcal{D}_i$ and thus can distinguish between databases $\mathcal{D}, \mathcal{D}'$ with $\mathcal{D}_i \neq \mathcal{D}'_i$
    - This must be taken into account
  - We can achieve statistical server-privacy

OT: Formal Definition of Server-Security

- Difference with client-privacy:
  - Client obtains an output $\mathcal{D}_i$ and thus can distinguish between databases $\mathcal{D}, \mathcal{D}'$ with $\mathcal{D}_i \neq \mathcal{D}'_i$
    - This must be taken into account
  - We can achieve statistical server-privacy
    - With communication $\Theta(\log |\mathcal{D}|)$

OT: Formal Definition of Server-Security

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

OT: Formal Definition of Server-Security

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

OT: Formal Definition of Server-Security

- Consider the next ideal world with a completely trusted third party T:

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 $\mathcal{D}$ to T (secretly, authenticatedly)

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 $\mathcal{D}$ to T (secretly, authenticatedly)
  - T sends $\mathcal{D}_i$ to A (secretly, authenticatedly)

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 $\mathcal{D}$ to T (secretly, authenticatedly)
  - T sends $\mathcal{D}_i$ to A (secretly, authenticatedly)
- This clearly models what we want to achieve!

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 $\mathcal{D}$ to T (secretly, authenticatedly)
  - T sends $\mathcal{D}_i$ to A (secretly, authenticatedly)
- This clearly models what we want to achieve!
- A protocol is server-secure if:

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 $\mathcal{D}$ to T (secretly, authenticatedly)
  - T sends $\mathcal{D}_i$ 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

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 $\mathcal{D}$ to T (secretly, authenticatedly)
  - T sends $\mathcal{D}_i$ 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

Note on Security Definitions

- Security definitions are uniform and modular, and remain the same for most protocols

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

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

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

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

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

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: $D_{sk}(E_{pk}(m; r)) = m$

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

- 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

Tool: Additively Homomorphic Public-Key Crypto

- Additionally, E is additively homomorphic iff

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

where plaintexts reside in some finite group $\mathcal{M}$ and ciphertexts reside in some finite group $\mathcal{C}$.

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 $D_{sk}(E_{pk}(m; r)^a) = am$

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 $N = pq$

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: $E_{pk}(m; r) = (1 + mN)r^N \mod N^2$

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 $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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.

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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 $j \in \{1, \ldots, n\}$:

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *)$

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

Simple PIR

Inputs: Alice has query $i \in [n]$, Bob has $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_n)$ where $\mathcal{D}_j \in \mathbb{Z}_N$

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 $j \in \{1, \dots, n\}$:
   - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i - j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \dots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

See [AIR01]

AIR PIR: Correctness/Security

- Bob does for every $j \in \{1, \ldots, n\}$:

AIR PIR: Correctness/Security

- Bob does for every $j \in \{1, \ldots, n\}$:
  - Set $b_j \leftarrow (a/E_{pk}(j;1))^* \cdot E_{pk}(\mathcal{D}_j;*)$

AIR PIR: Correctness/Security

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

AIR PIR: Correctness/Security

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

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

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

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

for some r

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

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

  for some r
- If $i = j$ then

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

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

  for some r
- If $i = j$ then

$$b_j = E_{pk}(0;r) \cdot E_{pk}(\mathcal{D}_j;*) = E_{pk}(\mathcal{D}_j;*)$$

  and thus $D_{sk}(b_i) = \mathcal{D}_i$

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

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

  for some r
- If $i = j$ then

$$b_j = E_{pk}(0;r) \cdot E_{pk}(\mathcal{D}_j;*) = E_{pk}(\mathcal{D}_j;*)$$

  and thus $D_{sk}(b_i) = \mathcal{D}_i$
- Thus Alice obtains $\mathcal{D}_i$

AIR PIR: Correctness/Security

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

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

- Because E is additively homomorphic then

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

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

$$b_j = E_{pk}(*; r) \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*; *) \ ,$$

  and thus $D_{sk}(b_j) = *$, i.e., b_j gives no information about $\mathcal{D}_j$
- Thus Alice obtains $\mathcal{D}_i$ 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

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 $\mathcal{D}_i$ together with $n - 1$ encryptions of random strings

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 $\mathcal{D}_i$ together with $n-1$ encryptions of random strings
  - We can construct a simulator who, only knowing $\mathcal{D}_i$ and nothing else about Bob's database, sends

$$(E_{pk}(*;*), \ldots, E_{pk}(*;*), E_{pk}(\mathcal{D}_i;*), E_{pk}(*;*), \ldots, E_{pk}(*;*))$$

to Alice.

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 $\mathcal{D}_i$ together with $n-1$ encryptions of random strings
  - We can construct a simulator who, only knowing $\mathcal{D}_i$ and nothing else about Bob's database, sends

$$(E_{pk}(*;*),\dots,E_{pk}(*;*),E_{pk}(\mathcal{D}_i;*),E_{pk}(*;*),\dots,E_{pk}(*;*))$$

    to Alice.
  - Simulator's output is the same as honest Bob's output and was constructed, only knowing $\mathcal{D}_i$

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 $\mathcal{D}_i$ together with $n-1$ encryptions of random strings
  - We can construct a simulator who, only knowing $\mathcal{D}_i$ and nothing else about Bob's database, sends

$$(E_{pk}(*;*), \dots, E_{pk}(*;*), E_{pk}(\mathcal{D}_i;*), E_{pk}(*;*), \dots, E_{pk}(*;*))$$

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

AIR PIR: Full Server-Security Proof

Proof.

We must assume that simulator is unbounded (this is ok since malicious Alice can also be unbounded, and thus simulator may need a lot of time to check her work).

AIR PIR: Full Server-Security Proof

Proof.

We must assume that simulator is unbounded (this is ok since malicious Alice can also be unbounded, and thus simulator may need a lot of time to check her 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".

AIR PIR: Full Server-Security Proof

Proof.

We must assume that simulator is unbounded (this is ok since malicious Alice can also be unbounded, and thus simulator may need a lot of time to check her 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 $\mathcal{D}$ to T. T sends $\mathcal{D}_{i^*}$ to simulator. Simulator sends

$$(E_{pk}(*;*), \dots, E_{pk}(*;*), E_{pk}(\mathcal{D}_i;*), E_{pk}(*;*), \dots, E_{pk}(*;*))$$

to Alice.

AIR PIR: Full Server-Security Proof

Proof.

We must assume that simulator is unbounded (this is ok since malicious Alice can also be unbounded, and thus simulator may need a lot of time to check her 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 $\mathcal{D}$ to T. T sends $\mathcal{D}_{i^*}$ to simulator. Simulator sends

$$(E_{pk}(*;*), \dots, E_{pk}(*;*), E_{pk}(\mathcal{D}_i;*), E_{pk}(*;*), \dots, E_{pk}(*;*))$$

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

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.

AIR 1-out-of-n PIR: Efficiency

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 $j \in \{1, \ldots, n\}$:

AIR 1-out-of-n PIR: Efficiency

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *)$

AIR 1-out-of-n PIR: Efficiency

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 $j \in \{1, \dots, n\}$:
   - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i - j) + \mathcal{D}_j; *)$

AIR 1-out-of-n PIR: Efficiency

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i - j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

AIR 1-out-of-n PIR: Efficiency

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 $j \in \{1, \dots, n\}$:
   - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \dots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- Alice's computation: one encryption at first, and one decryption at the end.

AIR 1-out-of-n PIR: Efficiency

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- Alice's computation: one encryption at first, and one decryption at the end. Good

AIR 1-out-of-n PIR: Efficiency

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 $j \in \{1, \dots, n\}$:
   - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \dots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- Alice's computation: one encryption at first, and one decryption at the end. Good
- Bob's computation: $2n$ encryptions, n exponentiations, etc.

AIR 1-out-of-n PIR: Efficiency

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 $j \in \{1, \dots, n\}$:
   - Set $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i - j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \dots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- 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 $o(n)$!

AIR 1-out-of-n PIR: Efficiency

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i - j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- 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

AIR 1-out-of-n PIR: Efficiency

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 $b_j \leftarrow (a/E_{pk}(j; 1))^* \cdot E_{pk}(\mathcal{D}_j; *) = E_{pk}(*(i-j) + \mathcal{D}_j; *)$
4. Bob sends $(b_1, \ldots, b_n)$ to Alice, Alice decrypts b_i and obtains thus $\mathcal{D}_i = D_{sk}(b_i)$

- 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

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated
  - Bear in mind that PIR is the simplest possible PPDM algorithm!

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated
  - Bear in mind that PIR is the simplest possible PPDM algorithm!
- With a well-constructed protocol, proofs can become straightforward

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated
  - Bear in mind that PIR is the simplest possible PPDM algorithm!
- With a well-constructed protocol, proofs can become straightforward
  - Existing designs can be (hopefully?) explained to non-specialists

AIR PIR: Lessons

- It is possible to design provably secure PPDM algorithms
- Design is often complicated
  - Bear in mind that PIR is the simplest possible PPDM algorithm!
- 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

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

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
- Trivial PIR: Bob transfers the database to Alice. Good performance, linear communication, no privacy for Bob

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
- Trivial PIR: Bob transfers the database to Alice. Good performance, linear communication, no privacy for Bob
- [NP99] showed how to transfer any PIR to OT:

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
- Trivial PIR: Bob transfers the database to Alice. Good performance, linear communication, no privacy for Bob
- [NP99] showed how to transfer any PIR to OT:
  - Every database element is masked by $\log n$ pseudorandom sequences and then the PIR is applied to the masked database. Alice additionally obtains the concrete $\log n$ pseudorandom sequences needed to unmask $\mathcal{D}_i$ by doing $\log n$ 1-out-of-2 OT-s with Bob.

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
- Trivial PIR: Bob transfers the database to Alice. Good performance, linear communication, no privacy for Bob
- [NP99] showed how to transfer any PIR to OT:
  - Every database element is masked by $\log n$ pseudorandom sequences and then the PIR is applied to the masked database. Alice additionally obtains the concrete $\log n$ pseudorandom sequences needed to unmask $\mathcal{D}_i$ by doing $\log n$ 1-out-of-2 OT-s with Bob.
  - Needs n symmetric-key operations and $\log n$ public-key encryptions in addition to the computation of PIR.

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
- Trivial PIR: Bob transfers the database to Alice. Good performance, linear communication, no privacy for Bob
- [NP99] showed how to transfer any PIR to OT:
  - Every database element is masked by $\log n$ pseudorandom sequences and then the PIR is applied to the masked database. Alice additionally obtains the concrete $\log n$ pseudorandom sequences needed to unmask $\mathcal{D}_i$ by doing $\log n$ 1-out-of-2 OT-s with Bob.
  - Needs n symmetric-key operations and $\log n$ public-key encryptions in addition to the computation of PIR.

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob, and Bob responds with $\mathcal{D}_i$. I.e., $\log n + length(\mathcal{D}_i)$ bits.

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob, and Bob responds with $\mathcal{D}_i$. I.e., $\log n + length(\mathcal{D}_i)$ bits.
- Thus in PIR, the communication is also lower-bounded by $\log n + length(\mathcal{D}_i)$ bits.

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob, and Bob responds with $\mathcal{D}_i$. I.e., $\log n + length(\mathcal{D}_i)$ bits.
- Thus in PIR, the communication is also lower-bounded by $\log n + length(\mathcal{D}_i)$ bits.
- [Lip05]: A PIR with communication $O(\log^2 n + length(\mathcal{D}_i) \cdot \log n)$

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob, and Bob responds with $\mathcal{D}_i$. I.e., $\log n + length(\mathcal{D}_i)$ bits.
- Thus in PIR, the communication is also lower-bounded by $\log n + length(\mathcal{D}_i)$ bits.
- [Lip05]: A PIR with communication $O(\log^2 n + length(\mathcal{D}_i) \cdot \log n)$
- [GR05]: communication $O(\log n + length(\mathcal{D}_i))$ but much higher Alice-side computation

More Efficient PIRs: Communication

- In non-private information retrieval, Alice sends i to Bob, and Bob responds with $\mathcal{D}_i$. I.e., $\log n + \mathit{length}(\mathcal{D}_i)$ bits.
- Thus in PIR, the communication is also lower-bounded by $\log n + \mathit{length}(\mathcal{D}_i)$ bits.
- [Lip05]: A PIR with communication $O(\log^2 n + \mathit{length}(\mathcal{D}_i) \cdot \log n)$
- [GR05]: communication $O(\log n + \mathit{length}(\mathcal{D}_i))$ 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 $a = (a_1, \ldots, a_n)$ and Bob's vector $b = (b_1, \ldots, b_n)$, Alice needs to know $a \cdot b = \sum a_i b_i$

Private Scalar Product

- Goal: Given Alice's vector $a = (a_1, \ldots, a_n)$ and Bob's vector $b = (b_1, \ldots, b_n)$, Alice needs to know $a \cdot b = \sum a_i b_i$
- Cryptographic privacy goals: Alice only learns $a \cdot b$, Bob learns nothing

Private Scalar Product

- Goal: Given Alice's vector $a = (a_1, \ldots, a_n)$ and Bob's vector $b = (b_1, \ldots, b_n)$, 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

Private Scalar Product

- Goal: Given Alice's vector $a = (a_1, \ldots, a_n)$ and Bob's vector $b = (b_1, \ldots, b_n)$, 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

Private Scalar Product

- Goal: Given Alice's vector $a = (a_1, \ldots, a_n)$ and Bob's vector $b = (b_1, \ldots, b_n)$, 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

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, $E_{pk}(m_1; r_1)E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 r_2)$

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 $b = (b_1, \ldots, b_n)$

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 $A_i \leftarrow E_{pk}(a_i; *)$

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

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

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: $B = \prod A_i^{b_i} \cdot E_{pk}(0; *) = \prod E_{pk}(a_i; *)^{b_i} \cdot E_{pk}(0; *) = \prod E_{pk}(a_i b_i; \ldots) \cdot E_{pk}(0; *) = E_{pk}(\sum a_i b_i; \ldots) \cdot E_{pk}(0; *) = E_{pk}(\sum a_i b_i; *)$

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

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 [GLLM04] for more

GLLM04: Complexity

1. For $i \in \{1, \ldots, n\}$, Alice sends to Bob $A_i \leftarrow E_{pk}(a_i; *)$

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

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

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)

Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive

Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive
  - E.g., (kernel) perceptron algorithm (explained later)

Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive
  - E.g., (kernel) perceptron algorithm (explained later)
- Condition $a > 0$ can be checked by using affine operations but it is cumbersome and relatively inefficient

Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive
  - E.g., (kernel) perceptron algorithm (explained later)
- 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

Need For More Complex Tools

- Take, e.g., an algorithm where some steps are conditional on some value being positive
    - E.g., (kernel) perceptron algorithm (explained later)
- 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

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 ☹

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 [LP00] or [LLM06]

Combining Example: Private Kernel Perceptron

- Classifying data: given a collection of existing data vectors $\vec{y} \in \{-1, 1\}^n$ and their classification to two sets -1 and 1 (good/bad, rich/poor, ...), predict the classification of new data vectors

Combining Example: Private Kernel Perceptron

- Classifying data: given a collection of existing data vectors $\vec{y} \in \{-1, 1\}^n$ and their classification to two sets -1 and 1 (good/bad, rich/poor, ...), predict the classification of new data vectors
- Linear classification: assume vectors $\vec{y}$ are in n-dimensional space and that there exists an $(n-1)$-dimensional hyperplane that divides this space into two halves, the "bad" and the "good" datapoints. Find this hyperplane!

Combining Example: Private Kernel Perceptron

- Classifying data: given a collection of existing data vectors $\vec{y} \in \{-1, 1\}^n$ and their classification to two sets -1 and 1 (good/bad, rich/poor, ...), predict the classification of new data vectors
- Linear classification: assume vectors $\vec{y}$ are in n-dimensional space and that there exists an $(n-1)$-dimensional hyperplane that divides this space into two halves, the "bad" and the "good" datapoints. Find this hyperplane!
- Support Vector Machine: a separating hyperplane P that has maximum distance $\min_i d(P, \vec{y}_i)$ from all data vectors

Combining Example: Private Kernel Perceptron

- Most of the datasets are not linearly separable!

Combining Example: Private Kernel Perceptron

- Most of the datasets are not linearly separable!
- Kernel algorithms:

Combining Example: Private Kernel Perceptron

- Most of the datasets are not linearly separable!
- Kernel algorithms:
  - Design an application-specific kernel function K that maps n_1-dimensional vectors into n_2-dimensional space, $n_2 > n_1$, such that the data points will be linearly separable there

Combining Example: Private Kernel Perceptron

- Most of the datasets are not linearly separable!
- Kernel algorithms:
  - Design an application-specific kernel function K that maps n_1-dimensional vectors into n_2-dimensional space, $n_2 > n_1$, such that the data points will be linearly separable there
  - Apply the original algorithm in the n_2-dimensional space
- Kernel perceptron is a concrete well-known kernel linear classifier

Combining Example: Private Kernel Perceptron

- Most of the datasets are not linearly separable!
- Kernel algorithms:
  - Design an application-specific kernel function K that maps n_1-dimensional vectors into n_2-dimensional space, $n_2 > n_1$, such that the data points will be linearly separable there
  - Apply the original algorithm in the n_2-dimensional space
- Kernel perceptron is a concrete well-known kernel linear classifier
- ... not the most efficient one but relatively easy to secure [LLM06]

Combining Example: Private Kernel Perceptron

Kernel Perceptron

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

1. Set $\vec{\alpha} \leftarrow \vec{0}$.
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
4. **return** $\vec{\alpha}$

Or: keep $\vec{\alpha}$ secret and use it to predict new classifiers

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
  - Cryptography with weaker security guarantees
    - E.g., securing standard data structures — structure itself reveals some information about the data, but how much, and how much is acceptable?
- 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

References I

William Aiello, Yuval Ishai, and Omer Reingold.
Priced Oblivious Transfer: How to Sell Digital Goods.
In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer-Verlag.

Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen.
On Private Scalar Product Computation for Privacy-Preserving Data Mining.
In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120, Seoul, Korea, December 2–3, 2004. Springer-Verlag.

Craig Gentry and Zulfikar Ramzan.
Single-Database Private Information Retrieval with Constant Communication Rate.
In Luis Caires, Guiseppe F. Italiano, Luis Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *The 32nd International Colloquium on Automata, Languages and Programming, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815, Lisboa, Portugal, 2005. Springer-Verlag.

Helger Lipmaa.
An Oblivious Transfer Protocol with Log-Squared Communication.
In Jianying Zhou and Javier Lopez, editors, *The 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328, Singapore, September 20–23, 2005. Springer-Verlag.

References II

Sven Laur, Helger Lipmaa, and Taneli Mielikäinen.
Cryptographically Private Support Vector Machines.
In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 618–624, Philadelphia, PA, USA, August 20–23, 2006. ACM.

Yehuda Lindell and Benny Pinkas.
Privacy-Preserving Data Mining.
In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 36–54, Santa Barbara, USA, August 20–24 2000. Springer-Verlag.

Moni Naor and Benny Pinkas.
Oblivious Transfer and Polynomial Evaluation.
In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 245–254, Atlanta, Georgia, USA, May 1–4, 1999. ACM Press.