Private Information Retrieval

PIR

S. Nemati

Department of Computer Science
University of Virginia

December 15, 2014

PIR and OT

Oblivious Transfer

PIR and OT

Oblivious Transfer

- sender who initially has n secrets

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

- sender who initially has n bit

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

- sender who initially has n bit
- receiver who initially holds an index $1 \leqslant i \leqslant n$

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

- sender who initially has n bit
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows ith bit

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

- sender who initially has n bit
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows ith bit
- the sender has no information about the index i

PIR and OT

Oblivious Transfer

- sender who initially has n secrets
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows just ith bit
- the sender has no information about the index i
- complexity is not important

Single DB PIR

- sender who initially has n bit
- receiver who initially holds an index $1 \leqslant i \leqslant n$
- At the end of the protocol the receiver knows ith bit
- the sender has no information about the index i
- complexity is important

single-DB

- TDP $\Rightarrow$1-DB $(n - o(n))$ -bit PIR

[1]Homomorphic Encryption Scheme

single-DB

- TDP $\Rightarrow$1-DB $(n - o(n))$ -bit PIR
- ($n - o(n))$ -bit PIR $\Rightarrow$ OT

[1]Homomorphic Encryption Scheme

single-DB

- TDP $\Rightarrow$1-DB $(n - o(n))$ -bit PIR
- ($n - o(n))$ -bit PIR $\Rightarrow$ OT
- OT $\Rightarrow$ One-Way function

[1] Homomorphic Encryption Scheme

single-DB

- TDP $\Rightarrow$1-DB $(n - o(n))$ -bit PIR
- ($n - o(n)$) -bit PIR $\Rightarrow$ OT
- OT $\Rightarrow$ One-Way function
- One-Way function $\Rightarrow$ 2-DB (n^{ϵ})

[1] Homomorphic Encryption Scheme

single-DB

- TDP $\Rightarrow$1-DB $(n - o(n))$ -bit PIR
- ($n - o(n)$) -bit PIR $\Rightarrow$ OT
- OT $\Rightarrow$ One-Way function
- One-Way function $\Rightarrow$ 2-DB (n^{ϵ})
- HES[1] $\Rightarrow$1-DB(n^{ϵ})-bit PIR

[1] Homomorphic Encryption Scheme

Definition

- We model a database as an $n - bit$ string $x = x_1, x_2, \cdots, x_n$ together with a computational agent

Definition

- We model a database as an $n-bit$ string $x = x_1, x_2, \cdots, x_n$ together with a computational agent
- agent is Turing Machine (Algorithm)

Definition

- A $1-round\ k-DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

Definition

- A $1 - round\ k - DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

  0.There are k copies of the database which all have $x = x_1, x_2, \cdots, x_n$.

Definition

- A $1 - round\ k - DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

  0. There are k copies of the database which all have $x = x_1, x_2, \cdots, x_n$.
  1. Alice wants to know x_i.

Definition

- A $1-round$ $k-DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

  0. There are k copies of the database which all have $x = x_1, x_2, \cdots, x_n$.

  1. Alice wants to know x_i.

  2. Alice flips coins and, based on the coin flips and i, computes (query) strings $q_1, q_2, \cdots, q_k$.

Definition

- A $1-round\ k-DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

  0.There are k copies of the database which all have $x = x_1, x_2, \cdots, x_n$.

  1. Alice wants to know x_i.

  2. Alice flips coins and, based on the coin flips and i, computes (query) strings $q_1, q_2, \cdots, q_k$.

  3. For all j $1 \leqslant j \leqslant k$, DB_jsends back a (answer) string $ANS_j\ (q_j)$.

Definition

- A $1 - round\ k - DB$ Information Retrieval Scheme With $x \in \{0,1\}^n$ and k databases has the following form:

  0.There are k copies of the database which all have $x = x_1, x_2, \cdots, x_n$.

  1. Alice wants to know x_i.

  2. Alice flips coins and, based on the coin flips and i, computes (query) strings $q_1, q_2, \cdots, q_k$.

  3. For all j $1 \leqslant j \leqslant k$, DB_jsends back a (answer) string $ANS_j\ (q_j)$.

  4. Using the value of i, the coin flips, and the $ANS_j(q_j)$, Alice computes x_i

Definition

- The complexity of the PIR scheme is $\sum_{j=1}^{k} |qj| + |ANS_j(q_j)|$.

Definition

- two model of privacy

Definition

- two model of privacy

  DB unbounded

  DB bounded

Definition

- A $k - DB$ PIR Scheme with $x \in \{0,1\}^n$ is an information retrieval scheme such that, after the query is made and answered, the database does not have any information about what i is

Definition

- A $k - DB$ PIR Scheme with $x \in \{0, 1\}^n$ is an information retrieval scheme such that, after the query is made and answered, the database does not have any information about what i is
- they shouldn't can distinguish between transcript of i and j

Theorem

Theorem

there is a $4 - DB$*,* $O(\sqrt{n})$*-bit PIR scheme*

proof

- Each index of the database is represented as an ordered pair (i_1, i_2)

proof

- Each index of the database is represented as an ordered pair (i_1, i_2)
- i_1 and i_2 are written in base $\lceil \sqrt{n} \rceil$

proof

- Each index of the database is represented as an ordered pair (i_1, i_2)
- i_1 and i_2 are written in base $\lceil\sqrt{n}\rceil$

$$\begin{matrix} x_{(1,1)} & \cdots & x_{(1,\sqrt{n})} \\ \vdots & \vdots & \vdots \\ x_{(\sqrt{n},1)} & \cdots & x_{(\sqrt{n},\sqrt{n})} \end{matrix}$$

proof

- Each index of the database is represented as an ordered pair (i_1, i_2)
- i_1 and i_2 are written in base $\lceil \sqrt{n} \rceil$

$$\begin{matrix} x_{(1,1)} & \cdots & x_{(1,\sqrt{n})} \\ \vdots & \vdots & \vdots \\ x_{(\sqrt{n},1)} & \cdots & x_{(\sqrt{n},\sqrt{n})} \end{matrix}$$

- databases are labeled DB_{00} , DB_{01} , DB_{10} and DB_{11}

proof

- Alice wants to know bit $x_{(i_1,i_2)}$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0\ ,\ 1\}^{\sqrt{n}}$

proof

- Alice wants to know bit $x_{(i_1, i_2)}$
- Alice generates σ , $\tau \in \{0 , 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0\ ,\ 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database

proof

- Alice wants to know bit $x_{(i_1, i_2)}$
- Alice generates σ , $\tau \in \{0 , 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'

proof

- Alice wants to know bit $x_{(i_1, i_2)}$
- Alice generates σ , $\tau \in \{0\ ,\ 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'
- D_{00} sends $\oplus_{\sigma(j_1)=1, \tau(j_2)=1} x_{j1,j2}$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0\ ,\ 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'
- D_{00} sends $\oplus_{\sigma(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{01} sends $\oplus_{\sigma(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0 , 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'
- D_{00} sends $\oplus_{\sigma(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{01} sends $\oplus_{\sigma(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$
- D_{10} sends $\oplus_{\sigma'(j_1)=1,\tau(j_2)=1} x_{j1,j2}$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0\ ,\ 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'
- D_{00} sends $\oplus_{\sigma(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{01} sends $\oplus_{\sigma(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$
- D_{10} sends $\oplus_{\sigma'(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{11} sends $\oplus_{\sigma'(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$

proof

- Alice wants to know bit $x_{(i_1,i_2)}$
- Alice generates σ , $\tau \in \{0 , 1\}^{\sqrt{n}}$
- Alice then generates two additional $\sqrt{n}$ bits strings from the first two strings: $\sigma' = \sigma \oplus i_1$ and$\tau' = \tau \oplus i_2$
- Alice sends two strings to each database
- DB_{00}receives σ, τ . DB_{01} receives σandτ' . DB_{10}receives σ'andτ . DB_{11} receives σ'andτ'
- D_{00} sends $\oplus_{\sigma(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{01} sends $\oplus_{\sigma(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$
- D_{10} sends $\oplus_{\sigma'(j_1)=1,\tau(j_2)=1} x_{j1,j2}$
- D_{11} sends $\oplus_{\sigma'(j_1)=1,\tau'(j_2)=1} x_{j1,j2}$
- Alice XORs the four bits

proof

$$\begin{matrix} x_{(1,1)} & \cdots & x_{(1,\sqrt{n})} \\ \vdots & x_{(i,j)} & \vdots \\ x_{(\sqrt{n},1)} & \cdots & x_{(\sqrt{n},\sqrt{n})} \end{matrix}$$

proof

$$\begin{matrix} x_{(1,1)} & \cdots & x_{(1,\sqrt{n})} \\ \vdots & x_{(i,j)} & \vdots \\ x_{(\sqrt{n},1)} & \cdots & x_{(\sqrt{n},\sqrt{n})} \end{matrix}$$

- Since $x_{i1,i2}$ is the only bit that appeared an odd number of times, the result is $x_{i1,i2}$

proof

$$\begin{matrix} x_{(1,1)} & \cdots & x_{(1,\sqrt{n})} \\ \vdots & x_{(i,j)} & \vdots \\ x_{(\sqrt{n},1)} & \cdots & x_{(\sqrt{n},\sqrt{n})} \end{matrix}$$

- Since $x_{i1,i2}$ is the only bit that appeared an odd number of times, the result is $x_{i1,i2}$
- Note that the number of bits sent is $8\sqrt{n} + 4$

Theorem

Theorem

For all $k \in N$ *there is a* k*-DB,* $o((k \log k) n^{\frac{1}{\log k}})$ *-bit PIR scheme.*

Theorem

Theorem

For all $k \in N$ *there is a k-DB,* $o((k \log k)n^{\frac{1}{\log k}})$ *-bit PIR scheme.*

Theorem

For all $k \in N$ *there is a k-DB,* $o((k \log k)n^{\frac{1}{\log k + \log \log k}})$ *-bit PIR scheme*

Theorem

Theorem

For all $k \in N$ there is a k-DB, $o((k \log k)n^{\frac{1}{\log k}})$ -bit PIR scheme.

Theorem

For all $k \in N$ there is a k-DB, $o((k \log k)n^{\frac{1}{\log k + \log \log k}})$ -bit PIR scheme

Theorem

For all k there is a k-DB $O(k^3(n^{\frac{1}{2k-1}}))$ -bit scheme.

Theorem

Theorem

For all $k \in N$ *there is a k-DB,* $o((k \log k) n^{\frac{1}{\log k}})$ *-bit PIR scheme.*

Theorem

For all $k \in N$ *there is a k-DB,* $o((k \log k) n^{\frac{1}{\log k + \log \log k}})$ *-bit PIR scheme*

Theorem

For all k there is a k-DB $O(k^3(n^{\frac{1}{2k-1}}))$ *-bit scheme.*

Theorem

One-way Functions Imply $O(n^{\epsilon})$ *2-DB PIRs*

- Definition: Let $z, m \in N$. Assume z is relatively prime to m. The number z is a Quadratic Residue mod m if there exists a number a such that $a^2 \equiv z \mod m$.

- Definition: Let $z, m \in N$. Assume z is relatively prime to m. The number z is a Quadratic Residue mod m if there exists a number a such that $a^2 \equiv z \mod m$.
- Definition: The Quadratic Residue Problem is, given (z,m), determine if z is a quadratic residue mod m

- Definition: Let $z, m \in N$. Assume z is relatively prime to m. The number z is a Quadratic Residue mod m if there exists a number a such that $a^2 \equiv z \mod m$.
- Definition: The Quadratic Residue Problem is, given (z,m), determine if z is a quadratic residue mod m

Theorem

Assume that the quadratic residue problem is 'hard' for m the product of two primes and $|m| \geqslant n^{\delta}$. Then there exists a 1-DB, $O(n^{\frac{1}{2}+\delta})$ -bit PIR scheme

Thank you