Private Information Retrieval with a Trusted Hardware Unit – Revisited

Łukasz Krzywiecki, Mirosław Kutyłowski, Hubert Misztela, Tomasz Strumiński

Wrocław University of Technology

INSCRYPT 2010, Shanghai, 23.10.2010

Model - keeping a database in a cloud

Problem

keeping a database in a cloud

Securing a database kept in a cloud

- everything encrypted
- encryption mode hides occurrence of the same ciphertexts
- encryption method prevents modifications (even blind ones)
- ...

Problem

keeping a database in a cloud

Securing a database kept in a cloud

- everything encrypted
- encryption mode hides occurrence of the same ciphertexts
- encryption method prevents modifications (even blind ones)
- ...

Naïve solution

- each record encrypted
- the records stored in their original positions

Information leak
naïve solution

Assumptions

- Alice’s customer data updated after each transaction (e.g. loyalty programs)
- the cloud in keeping a log of operations made in the database (time+location)

Information leak

naïve solution

Assumptions

- Alice’s customer data updated after each transaction (e.g. loyalty programs)
- the cloud in keeping a log of operations made in the database (time+location)

Attack - finding transactions times of Alice

1. persuade Alice to make a transaction
2. locate the location of the record of Alice, find in the log file the previous updates to the same record.

Threat
traffic analysis

Encryption is not enough

- the cloud can derive sensitive information from encrypted database even if encryption is perfect,
- access pattern is a valuable source of information

Dilemma

- one has to hide the access pattern
- but in order to read or write one has to access a given location!

Private information retrieval (PIR) problem

- the user is fetching some information from a database
- the adversary is the database administrator and can see the data transmitted
- ... but cannot say what has been fetched

- Chor, Kushilevitz, Goldreich, Sudan: Private information retrieval. IEEE FOCS 1995,
- Ostrovsky, Skeith: A Survey of Single-Database Private Information Retrieval: Techniques and Applications. PKC 2007

PIR

Problems

- the solutions are computation and communication intensive.
  intuition: in order to hide what are you fetching you need to hide the information in a large stream of bits.
- ..but very clever methods has been designed reducing communication volume.
- still: we want not only to retrieve data but also modify it!

Architecture

secure hardware module

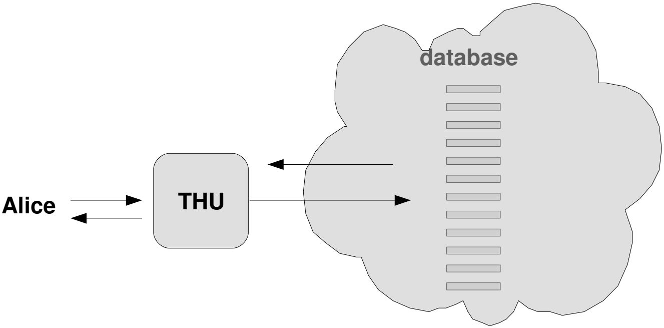
Wang, S., Ding, X., Deng, R. H., Bao, F.: Private Information Retrieval Using Trusted Hardware.
ESORICS 2006. LNCS 4189

Assumptions

- the cloud itself is a curious but passive adversary (any attempt to change the contents of the database means end of the business)
- the database owner uses a trusted hardware unit as an interface with the cloud.

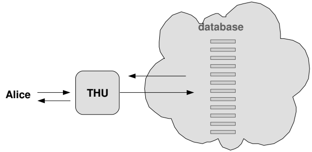
Architecture

secure hardware module



Architecture

secure hardware module



Assumptions

The trusted hardware unit: :

- performs cryptographic operations in behalf of the database owner,
- uses a cache memory

The cloud is keeping an encrypted (and re-encrypted) database, so it never learns the information stored in the database.

Yang, Ding, Deng, Bao Scheme – ideal world

YDDB Scheme
idea

Yang, Y., Ding, X., Deng, R. H., Bao, F.: An Efficient PIR Construction Using Trusted Hardware. ISC 2008

General framework

- fetch the records to the cache of THU
- ... until it becomes full

YDDB Scheme

idea

Yang, Y., Ding, X., Deng, R. H., Bao, F.: An Efficient PIR Construction Using Trusted Hardware. ISC 2008

General framework

- fetch the records to the cache of THU
- ... until it becomes full
- then flash the data from cache back to the cloud

YDDB Scheme
idea

Yang, Y., Ding, X., Deng, R. H., Bao, F.: An Efficient PIR Construction Using Trusted Hardware. ISC 2008

General framework

- fetch the records to the cache of THU
- ... until it becomes full
- then flash the data from cache back to the cloud

white and black records

black record : a record that has been already touched by the THU (trusted hardware unit)

white record : a record that has not been touched by the THU at this session

YDDB Scheme

main trick

Problem

an adversary can see if a black or white record is fetched to the cache! an important information revealed

YDDB Scheme
main trick

Problem

an adversary can see if a black or white record is fetched to the cache! an important information revealed

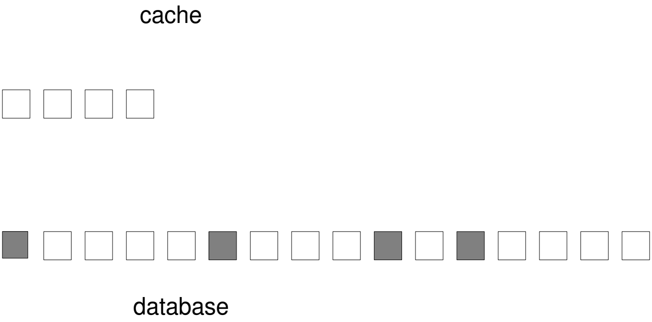
Solution

1. if THU want to fetch a black record, then it ask the cloud for this record as well as for some random white record
2. if THU want to fetch a white record, then it ask the cloud for this record as well as for some random black record

Outcome: always a pair of black and white records is fetched into the cache: the execution becomes oblivious!

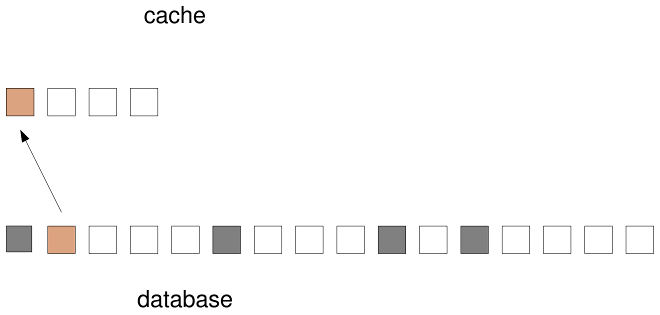
YDDB Scheme
animation

beginning of a phase



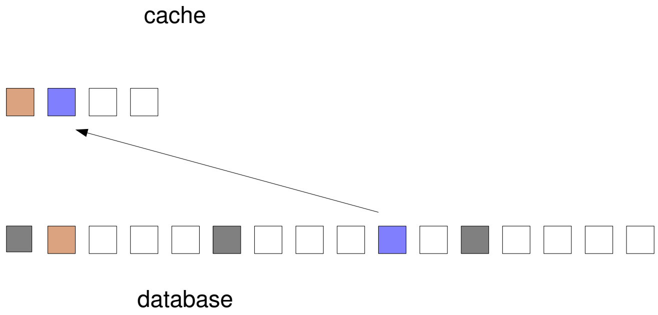
YDDB Scheme
animation

fetching a white record ..



YDDB Scheme
animation

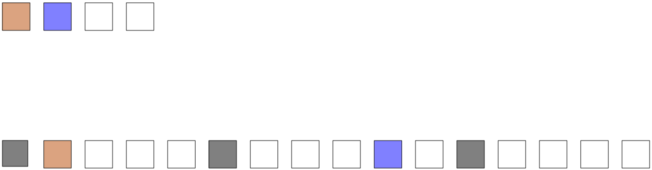
and a black record .



YDDB Scheme
animation

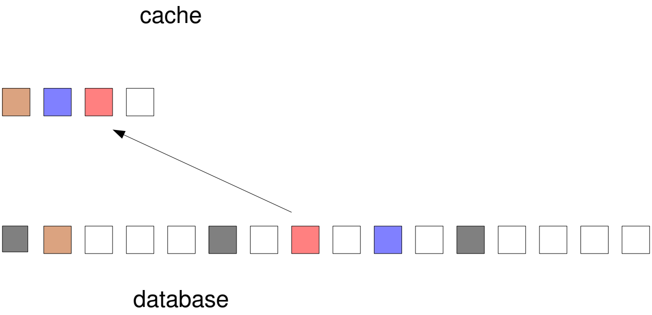
After the first query

cache



database

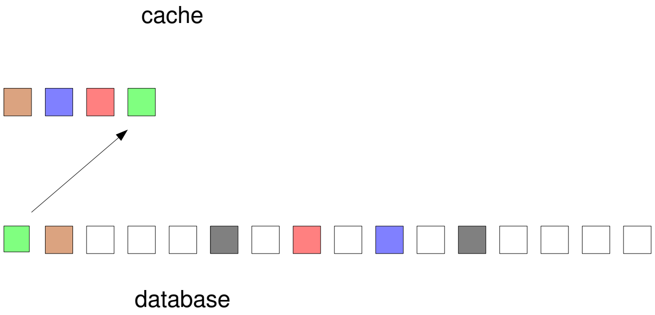
The second query: fetching a white record



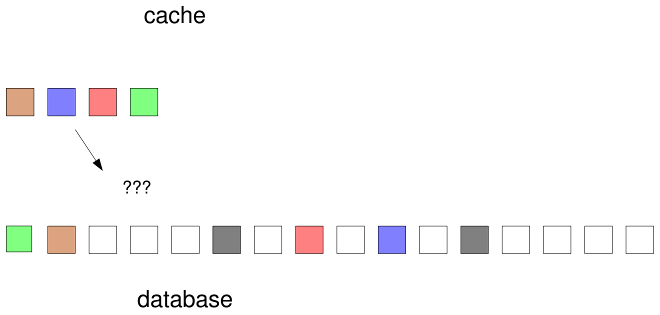
YDDB Scheme

animation

The second query: fetching a black record



Starting to flush the cache



YDDB Scheme

animation - flushing the cache

After flushing (with mixing)

YDDB Scheme
flushing the cache

Mixing black elements

- when the cache is flushed, then all black elements (including white elements written into the cache) are re-encrypted and mixed at random
- they are written on the positions of black records, rewriting the old contents

How to mix black elements
limitations

Bottleneck

- if the mixing is done in the cloud, the adversary can see everything and effectively there is no mixing
- if the mixing is done through THU, then each record has to be fetched into cache
- it is impossible to keep all black elements in the cache memory

Problem

How to permute the elements at random with a small cache?

Mixing assumptions

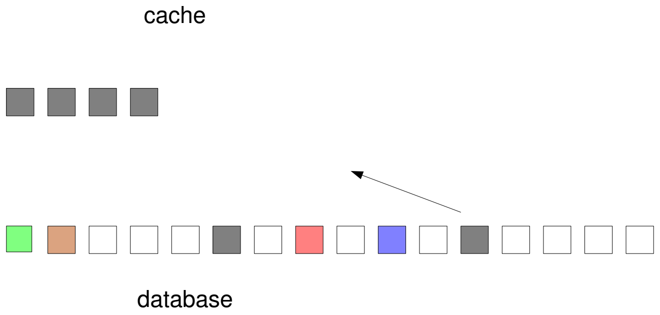
efficiency issues

Data flow

efficiency each record from the cloud read in only once, (less than once impossible - the adversary would see that some elements do not move)

storage when a record is read in into the cache, one record (re-encrypted) is written immediately to the cloud
(otherwise there would be memory overflow in the cache!)

Starting to flush the cache



PIR with Trusted Hardware Unit

Model

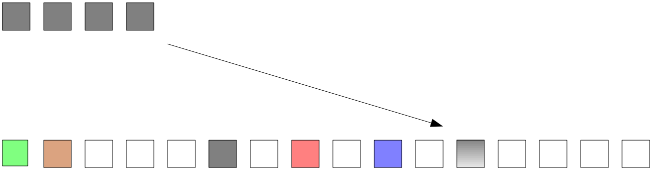
YDDB scheme – ideal world

Perfect security

Implementation problems

Starting to flush the cache

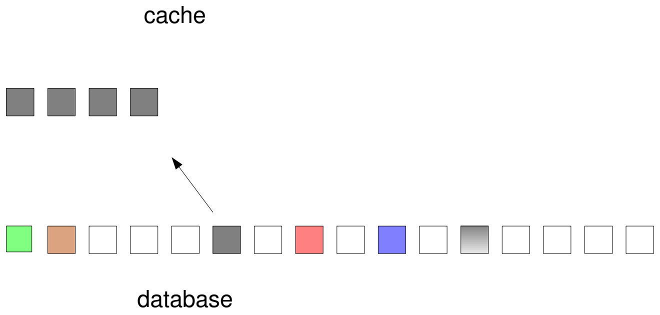
cache



database

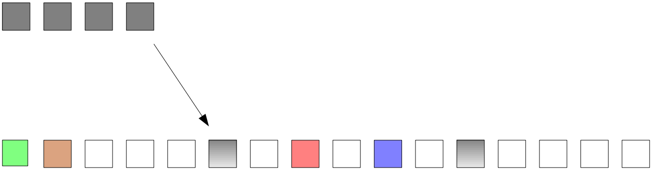
Mixing
step

the second record



the second record

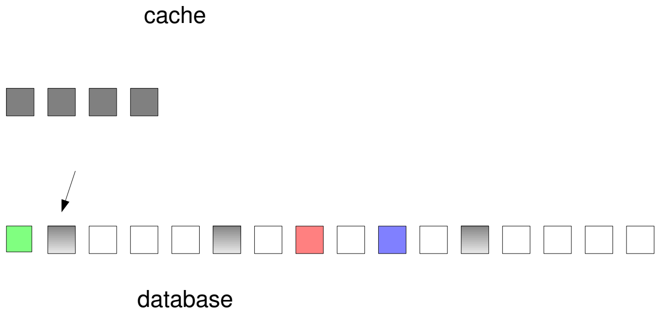
cache



database

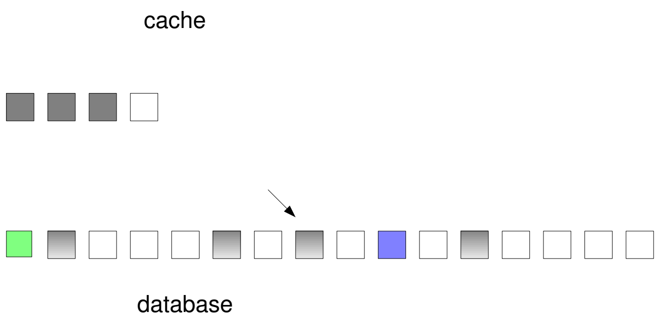
Mixing
step

cleaning the cache



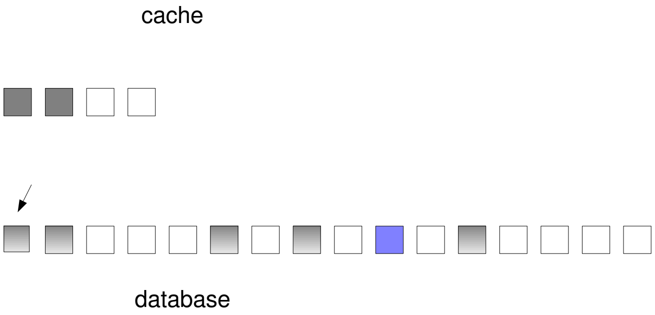
Mixing
step

cleaning the cache



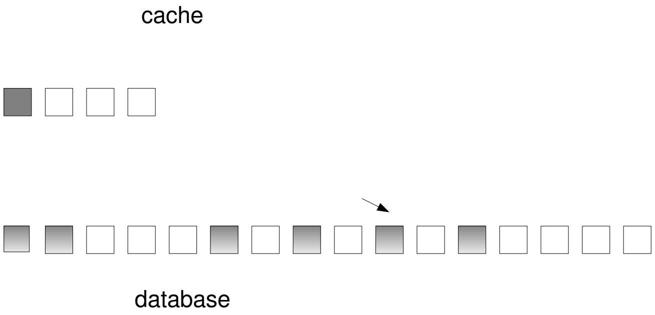
Mixing
step

cleaning the cache



Mixing
step

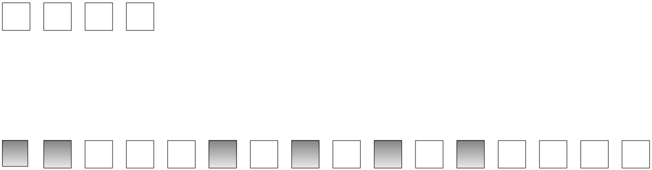
cleaning the cache



Mixing
step

cache cleaned

cache



database

Mixing bottleneck

Lemma

Consider a shuffling procedure that starts with k elements and m black records in the database, and such that:

- each black position is read exactly once,
- the black positions are read in some fixed predefined order,
- after reading a black record, some black record (may be the same) is immediately written into the same position.

The number of permutations on the set of $m + k$ positions that can be generated by this procedure is bounded by

$$k^m \cdot k!$$

Mixing bottleneck

Corollary

The ratio of the number of permutations possible to generate to the number of all permutations on $m + k$ positions is at most

$$\frac{k^m \cdot k!}{(m+k)!} \leq \frac{k^m \cdot k^{k+0.5} \cdot e^{-k+1/12k}}{(m+k)^{m+k+0.5} \cdot e^{-m-k+1/(12(m+k)+1)}}$$

$$\approx (\frac{k}{m+k})^{m+k+0.5} \cdot e^m .$$

In particular, if $5k < m$, then the above fraction is lower than 2^{-m}.

The cloud gains a lot of information. It is impossible to permute the black elements at random with a small cache in a short time.

Disclaimer

YDDB Scheme

- apparently, the YDDB scheme works according to the assumption that during mixing each record is read in exactly once and immediately some record is written into output position
- the pseudo-code published is not executable
- ... but whatever the authors meant

from adversary's point of view the permutation of black elements is not random.

Perfect security

Lesson learnt

Correlation between probability distributions

There is a strong correlation between the probability distributions of black elements after flushing the cache for the time t and $t + 1$.
Does it mean that the scheme has a security flaw?

NO!

Probability and adversary model

Access pattern until flushing at moment t

- all read and write operations performed before
- values of all elements decrypted by the user

corresponds to games for CCA and CPA security

Question

What is the probability distribution for permutations of black elements after flushing at moment t, conditioned a given access pattern A until moment t?

Main Theorem

Theorem

Probability distribution π_t of permutation of black elements at the end of epoch t conditioned on the access pattern A observed up to the end of epoch t is uniform in the set of all permutations over black records.

Remark

- The meaning is: even if the adversary can see something, any usable knowledge is immediately destroyed.
- Correlations do not destroy uniformness (which is counterintuitive).
- Theorem holds for any reasonable strategy of flushing the cache.

Simplest strategy to flush the cache

Algorithm

call the black elements in the cache - cache records
and the black elements not in the cache – touched records

Simplest strategy to flush the cache

Algorithm

call the black elements in the cache - cache records
and the black elements not in the cache – touched records

- choose the locations for the cache records, uniformly at random

Simplest strategy to flush the cache

Algorithm

call the black elements in the cache - cache records
and the black elements not in the cache – touched records

- choose the locations for the cache records, uniformly at random
- choose any ordering of positions of black records in the database, so that the positions of black records that are cache records come at the end

Simplest strategy to flush the cache

Algorithm

call the black elements in the cache - cache records
and the black elements not in the cache – touched records

- choose the locations for the cache records, uniformly at random
- choose any ordering of positions of black records in the database, so that the positions of black records that are cache records come at the end
- read in the records holding untouched records in their ordering, each time returning a re-encrypted
  - cache record, if this position was chosen for a cache record,
  - a touched record according to policy FIFO, otherwise.

Simplest strategy to flush the cache

Algorithm

call the black elements in the cache - cache records
and the black elements not in the cache – touched records

- choose the locations for the cache records, uniformly at random
- choose any ordering of positions of black records in the database, so that the positions of black records that are cache records come at the end
- read in the records holding untouched records in their ordering, each time returning a re-encrypted
  - cache record, if this position was chosen for a cache record,
  - a touched record according to policy FIFO, otherwise.
- on the remaining positions in the same way but without reading

Proof idea

Observations

- the records that have been read during the current epoch are written into random black positions – so the problem may occur only for the remaining elements,

Proof idea

Observations

- the records that have been read during the current epoch are written into random black positions – so the problem may occur only for the remaining elements,
- let us consider the event E that some specific set of positions are taken by cache elements conditioned by E there is a unique mapping of untouched elements from the positions before flushing and after flushing,

Proof idea

Observations

- the records that have been read during the current epoch are written into random black positions – so the problem may occur only for the remaining elements,
- let us consider the event E that some specific set of positions are taken by cache elements conditioned by E there is a unique mapping of untouched elements from the positions before flushing and after flushing,
- at the beginning of flushing their permutation was a random variable with a uniform distribution.

Implementation problems

Too early to celebrate

What is a problem?

- if we have to fetch a black record, how to fetch a white record at random?
- if we have to fetch a white record, how to fetch a black record at random?
- How the THU has to know that a query concerns a black record?

Solution

data structures stored by the cloud.
But exploring these data structures may betray some information on queries!

Data structures from YDDB

YDDB A beautiful design of data structures.

YDDB However, no security proof in the sense of probability distributions conditioned by the observations on access pattern.

this paper some examples showing that the probability distributions are not perfect. Still the conclusions about non-uniformity in some special situations.

Challenge

A difficult question:

Is it possible at all to build the data structures in a way that probability distributions over permutations remain uniform?

Or at least close to uniform according to some measure like total variation distance?

Thanks for your attention!

Contact data

1 Miroslaw.Kutylowski@pwr.wroc.pl

2 http://kutylowski.im.pwr.wroc.pl

3 +48 71 3202109, fax: +48 71 320 2105