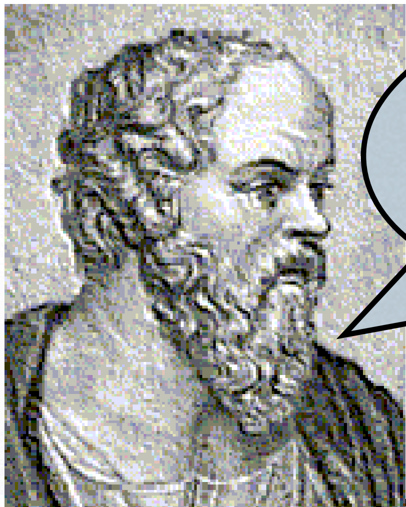
PIR: *crypto design perspective*

Aggelos Kiayias
University of Connecticut

aggelos@cse.uconn.edu
http://www.cse.uconn.edu/~akiayias

Efficiency & Algorithms



Three levels of consideration

- *Polynomial-time vs. Non-polynomial-time* : The inherent complexity of problem. The absolute boundary of efficient computation.
- *Exact time/space/communication-complexity function*: good data structures / clever all around design/ art of computer programming.
- *Benchmarks* : the bottom-line/ hardware - software coupling / compiler optimization.

Life and Times of a Problem

- Definition / Motivation.
- First solution/ Feasibility/ Polynomial-time.
- More solutions... Diversity. Alternate settings. Exact complexity functions.
- First implementations.
- Fine tunings. More implementations. Benchmarks.

A Crypto Design Exclusive

Party A performs a number of crypto operations “per **X**” of its input.

- “Per-**bit**” vs. “Per-**block**”
  - *Per-bit is easier to design and argue the security of.*
  - **HOWEVER : complexity suffers a multiplicative factor**.

Observe

input length n

security parameter k

crypto - op complexity $f(k)$

"Per-**bit**" vs. "Per-**block**"

$\Theta(n \cdot f(k))$ $\Theta(n + f(k))$

Retrospective

- First *provably secure* public-key cryptosystem: [GM82] : **per-bit** primitive.
- First *provably secure* digital signature: [GMR88] : **per-bit** primitive.
- First zero-knowledge proof: [GMR85] : **per-bit** primitive.

Development

- **None** *of the previous schemes is in use.*
- **Still,** they were seminal works that pointed to the right direction.
- **Now,** 20 years later we have: finely tuned benchmarked and secure **per-block** cryptographic primitives implemented in every computer.

What about PIR?

- First (single-server) PIR:
  [KO97] : a **per-bit** primitive.
- First (single-server) poly-log PIR:
  [CMS99] : a **per-bit** primitive.
- A **Per-bit** to **Per-block** transformation is possible for both the above protocols.

Communication Rate

- More suitable for judging communication complexity of block PIR protocols.
- *What is the* ***communication rate*** *for each bit that is PIR transfered?*
- Observe : all "per-bit" protocols transformed to "per-block" have *<u>vanishing rates</u>* in the size of the database.

We need **constant rate** protocols ----- native "Per block" constructions

Be harsh on PIR protocols!

- PIR has a characteristic that many previous cryptographic primitives do not have:
  - PK-encryption, digital signatures, zk-proofs etc. are *essentially solving the impossible* thus even per-bit primitives can be useful!
  - PIR can be solved by transferring the database. duh!

Achieving Constant Rate

- Gentry-Ramzan PIR (ICALP 2005):
  - Transmission Rate : ~1/4
- Lipmaa PIR (ISC 2005) original rate : ~1/logn
  - *New optimized version* rate ~1

Where is the catch?

- Transmission rate still an asymptotic parameter. What about the constants?
- What about time complexity?
- What about benchmarks on real inputs?

Towards PIR Implementations

- Optimized version of Lipmaa's PIR has superb communication complexity :
  e.g., for `1MB` PIR transfer the communication can be merely `1.56 MB!`

- Time-complexity for server can be very taxing:

  - [GR05] one modular exponentiation *with huge exponent*. (proportional to the database)

  - [Lip05] many modular exponentiations with regular size exponents *but over huge groups*!
    (e.g., 20000 bit)

Let’s Crunch

Use optimized [GR05] PIR for blocks and estimate implementation costs for a hypothetical database.

Caveat : the following numbers are rough estimates that are NOT based on an implementation. They are subject to change once an implementation is at hand.

Results

- Database consists of **`2048`** entries of documents each **`64Kbytes`** long.
- Required communication for a PIR read : ~ **`256Kbytes`**.
- Client computation-time : ~ **`95 seconds`**.
  extrapolation from Powerbook G4 `1.3 GHz` openssl benchmarks.
- Server computation-time ~ **`45 seconds`**.
  extrapolation from Sun fire T2000 `1.2 GHz 8`core openssl benchmarks.
- Sending the whole database (`128MB`) at **`350 KB/sec`** bandwidth : **`374 seconds`**.

the above assume 1024-bit moduli

Details

- [GR05] has a heavy toll on the client.
  Understanding the underlying intractability assumption may lead to substantial improvements (or substantial degradation if the assumption crumbles).

- Optimized version of [Lip05] has better com. complexity and superior client side computation.

  Server side computation blows up though.

Directions

- *Improve* on [GR05][Lip05].
- *Focus on related primitives:* Reduction of Block-PIR to Secure Multivariate Polynomial evaluation from [Kiayias-Yung ICALP '02].
- *Design PIRs based on alternative assumptions:* avoid modular exponentiations and other expensive operations.

Conclusion

- Practical PIR?
  - not there yet but we are maybe just seeing the first glimpses of it.
- *My prediction based on history and the recent works just described*: upcoming cryptography research focusing on the right direction will beat the problem soon.
- Support crypto research.