Practical Privacy with **LARGE** Databases

Duane Merrill

November 29, 2007

Overview

Schemes for private database query

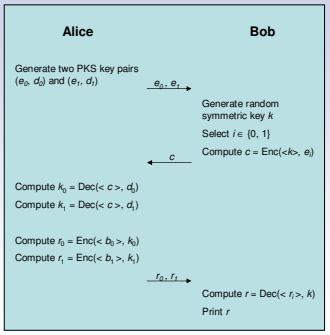
1. Motivational Scenarios
2. The "Gold Standard"
   - Oblivious Transfer (OT)
   - Symmetric Private Information Retrieval (SPIR)
3. My alternatives
   - Security through plausible deniability
   - Security through obscurity

Motivational Scenarios: Sender & Chooser Privacy

- *Pharmaceutical*
  - Drug designers need access to pharmaceutical databases to access properties of chemical compounds.
  - Drug designers don't want to disclose what they're working on
  - Database owners may be unwilling to sell them their entire database.
- *Patent*
  - Researchers with potentially new ideas often need to check for "prior art" on the subject.
  - If the patent database is aware of the queries, the researcher's competitive advantage is jeopardized.
- *Media*
  - Users may wish to purchase videos or music without the vendor knowing what they bought
  - Database owners don't want to expose content other than what's being purchased.

Oblivious Transfer (OT)

- Simplest form: 1-of-2 OT
  - Alice has two bits b_0, b_1
  - Bob gets to choose one
  - Alice learns nothing about which one Bob took
  - Bob learns nothing about Alice's other bit
- Implementable with every known public key cryptosystem



OT Generalizations

Garden varieties:

- 1-of-*N*
- *k*-of-*N*
- Keyword search
- Block retrieval
- Committed
- Private information retrieval (PIR)

Schemes measured by

- Communication complexity
  - Initialization
  - Online (query time)
- Computational complexity
  - Initialization
  - Online (query time)

Lower Bounds

- *O*(Computation) ≥ *O*(Communication)
  - Have to process each bit sent, even if just to copy
  - Computation is often measured in terms of exponentiations w/r/t *N*
- Communication is *clearly* *O*(log *N*)
  - Chooser needs a minimum of log N bits to index his desired datum
- Chooser-privacy makes it O(*N*)

Order *N*? Oh noes!

Bit database = | b_0 | b_1 | b_2 | b_3 | b_4 | b_5 | … | b_{N-1} |

- Chooser privacy
  - For a given configuration, the response message must be the same for all indices i=0, i=1, … , i=N-1
  - Otherwise the server is aware of a difference between choices
- Proof: Assume response message has fewer than N bits
  - 2^N database configurations, not as many possible response messages
  - Two different configurations must share the same response message
  - The configurations will differ at some b_i, and the chooser will not know what to print (received the same message).

But wait….Computational PIR!

- Rely on some intractability assumptions

responses are the same for each choice → responses are equally-as-likely (as far as the server can tell)

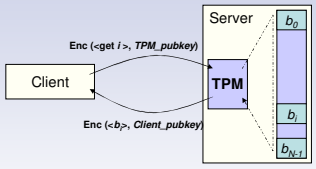
- Client encrypts "*Return me the i^{th} record*"
- Server processes

encrypted-result = *f* (*encrypted-query*, *database*)

- Server recognizes neither the result or the query
- Response messages are now O(polylog(N))
- Must read every record in the database for each query

But wait… Hardware PIR!

- Use "secure co-processor"
  - TPM's to *avoid* detection of copyright infringement!
- TPM is a black box for selection
  - No one can access its memory from the outside.
- Much less computational overhead (PKI crypto)



- Do you trust the TPM?
- Must read every record in the database for each query

But wait… Offline + Preprocessing!

- *Homomorphic* encryption
  - An algebraic operation on the ciphertext has an algebraic effect on the encoded plaintext
  - Often can add and multiply encrypted messages

1. Server encrypts every record
2. Encrypted records shipped offline to choosers
3. Chooser selects desired record, re-encrypts it, sends doubly-encrypted record to server
4. Server homomophically removes its encryption underneath, returns singly-encrypted record
5. Client decrypts record to get plaintext

- No need to run through the database each time
- Client must own entire copy of database (and keep it current)

OT/SPIR Maybe Not the Best Fit for Media Distribution…

- Too many entries
  - iTunes has 6 million track available
  - 3.75MB per track
  - 21TB database
    - Can't iterate over per-query
    - Ship to each customer? (Won't fit on Zune)
- VERY important to monitor download popularity
  - Billboard top 100

Ring Signatures

- User signs a message with their public/private keypair and the public keys of k-1 other people
- No setup or cooperation
- Verifier cannot distinguish with probability greater than 1/k who signed
- Signatures includes the public keys of all "suspects" (thus grows linearly)

Scheme A: Plausible Deniability

- Assumptions
  - Confidential bi-directional channels
- Protocol
  - Client *A* generates <get *i* > message, signs with *A_pubkey* and *Server_pubkey*.
  - Server authenticates client *A* and returns b_i
- Comments
  - No setup
  - No knowledge of other users
  - Communication and computation cost is $O(\log N)$
    - Signing requires 1 modular exponentiation, 3 multiplications
    - Verification requires 4 modular multiplications
  - *Designated verifier* scheme
    - *A* is positively authenticated to the *Server*
    - *Server* cannot prove to anyone else that *A* ordered a particular song

Scheme B: K-anonymity

- Assumptions
  - Anonymous bi-directional channels
  - Open billing (unlimited downloads)
- Protocol
  - *A* generates < get *i* > message, signs with *A_pubkey* and the public keys of *K-1* other users
  - Server authenticates client *A* and the *K*-1 other users, and returns b_i
- Comments
  - Requires knowledge of other users
  - Users can choose their *K*
    - Other users can "50%-slander" you
  - Communication and computation cost is $O(\log N + K)$
    - Signing and verification each require +2 modular multiplications per non-signer
  - Can be made to work without anonymous channels (symmetric keys and *K*-broadcast)
  - Can't bill/reward user activity

Blind Signatures

- Blinding Protocol
  - Alice "blinds" her message

    $<m'> = \text{blind}(<m>)$
  - <m'> is indistinguishable from random
  - Alice has Bob sign <m'> resulting in <s'>
  - Alice removes the blinding factor at her leisure, yielding <s> as if Bob had signed <m> directly
- Easily implementable with RSA

Scheme C: PG-Anonymity

- Assumptions
  - Anonymous bi-directional channels
- Protocol
  - Client *A* generates a random symmetric key k_A
  - Client *A* generates a blinded voucher $v_A = <$"get *i*", $k_A>$
  - Client *A* creates $v_A' = Blind(v_A)$
  - Client *A* signs $<v_A'>$ and sends both to *Server*
  - Server authenticates client *A* and creates a signature s_A' of $<v_A'>$
  - Server signs $<s_A'>$ and sends both to Client *A*
  - Client A retrieves $<s_A>$ by unblinding $<s_A'>$
  - Client A waits a while…
  - Client A sends $<v_A, s_A>$ to Server
  - Server verifies that it signed the voucher v_A and returns $\text{Enc}(b_i, k_A)$

"Pretty-Good" Anonymity

- General:
  - Redemption phase is (almost) completely anonymous
  - Individuals can be charged/rewarded by their download tallies
- More details:
  - Secret key *k* acts as a nonce to prevent double-spending
    - Server checks for (*i*,*k*) repeats during redemption
    - Server checks for voucher request *v'* repeats to avoid false-negatives

Can we guess who made a particular redemption?

- One voucher out, one redemption in…
  - 100% linkability
- 10 vouchers out, redemptions coming in…
  - 1/10 = 10% regardless of redemption slot
- 5 vouchers out (blue), 1 redemption in, 6 vouchers out (red)…
  - First redemption (from 5 unclaimed)
    - 1/5 = 20% for the blue vouchers
  - Second redemption (from 10 unclaimed)
    - (4/5)(1/10) = 8% for blue vouchers
    - (1/10) = 10% for red vouchers
  - Third redemption (from 9 unclaimed)
    - (4/5)(9/10)(1/9) = 8% for blue vouchers
    - (9/10)(1/9) = 10% for red vouchers
  - Fourth redemption (from 8 unclaimed)
    - (4/5)(9/10)(8/9)(1/8) = 8% for blue vouchers
    - (9/10)(8/9)(1/8) = 10% for red vouchers

Pretty Hairy

- New vouchers reduce linkability for existing vouchers
- Draining redemptions maintain linkability for existing vouchers
- Linkability is most dicey right at redemption time
- Multiple outstanding requests increase linkability
  - But by a proportional factor: same as OT/PIR

Equilibrium

- Linkability is at most 1/(*backlog*)
  - E.g., keep the system with a backlog of ~1000 unclaimed vouchers for ~1000-anonymity
  - Might bootstrap with 1000 users, where everyone orders a song before anyone redeems. (Or a Pepsi bottle-cap giveaway.)
- Towards a steady-state equilibrium:
  - Discrete time steps, redeem with probability *p*
    - Geometric distribution
  - Actual linkability is at most *p*, but *p* is unknown to the Server
- Clients can change their redemption parameter *p* to suit the cost-benefit tradeoff of anonymity vs. immediate-gratification
- Can get a rough-estimate of anonymity using a 3rd party back-log metering system and then selecting *p* appropriately.

Conclusions

- Oblivious Transfer and PIR are cool… but probably not commercially-viable
- Presented three approaches that may be suitable when e-cash isn't
  - Some authorization needed
- Still working on analysis, potentially simulation

Questions

?

iTunes

- ~6 million songs
- 200 million users
- ~3 billion downloads to-date
- Pepsi had a 100-million download giveaway
- (Feb 2006) 3+ million downloads a week
  - 300+ downloads a minute
  - Pipelining a backlog of 10,000 vouchers takes ~30 minutes