Authentication and Integrity in Outsourced Databases

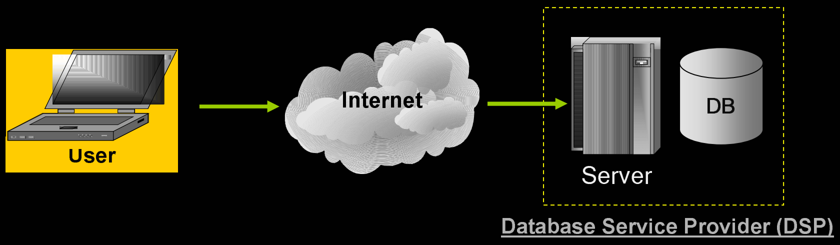
by

Einar Mykletun, Maithili Narasimha, Gene Tsudik
University of California, Irvine

Outline of Talk

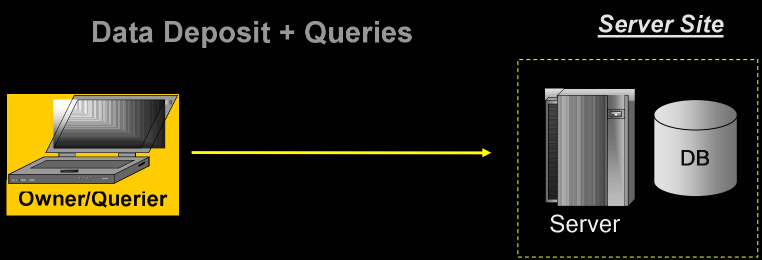
- Outsourced Database Model
- Motivation
- Digital Signature Solutions
- Authenticated Data Structure Solution
- Conclusion & Future Work

Outsourced Database Model (ODB)



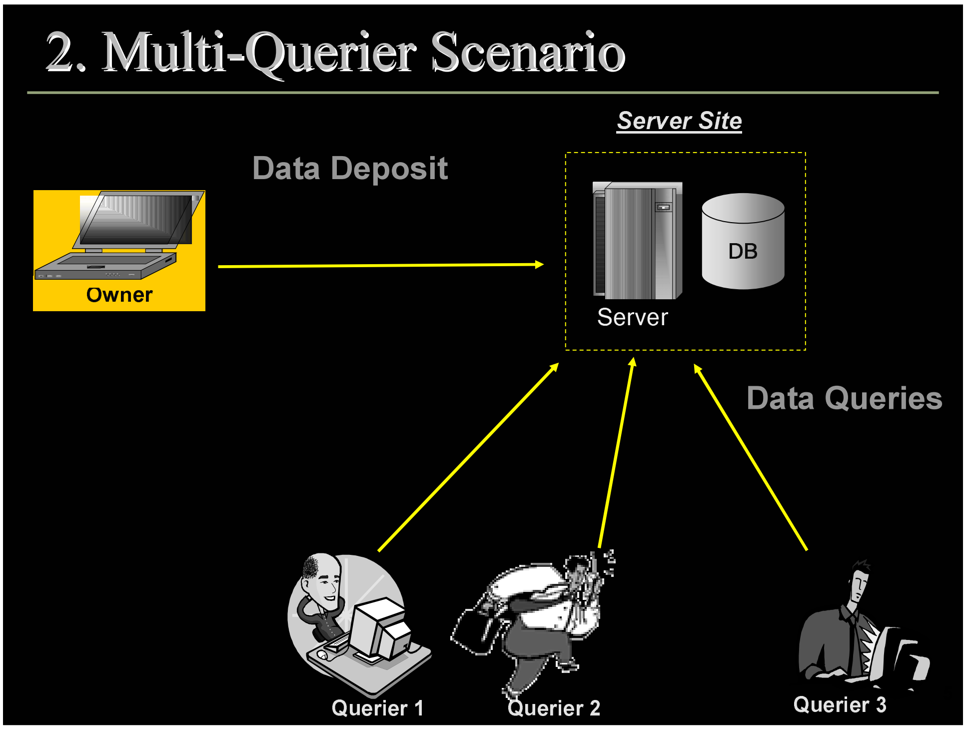
- Data owner wishes to outsource his database to a Service Provider
- The Players
  - Data Owner: deposits, modifies, removes data
  - Server: where the outsourced database is stored and queried upon
  - Client / Querier: entity who queries the database at the server
- Server is not fully trusted
- 3 Flavors of ODB

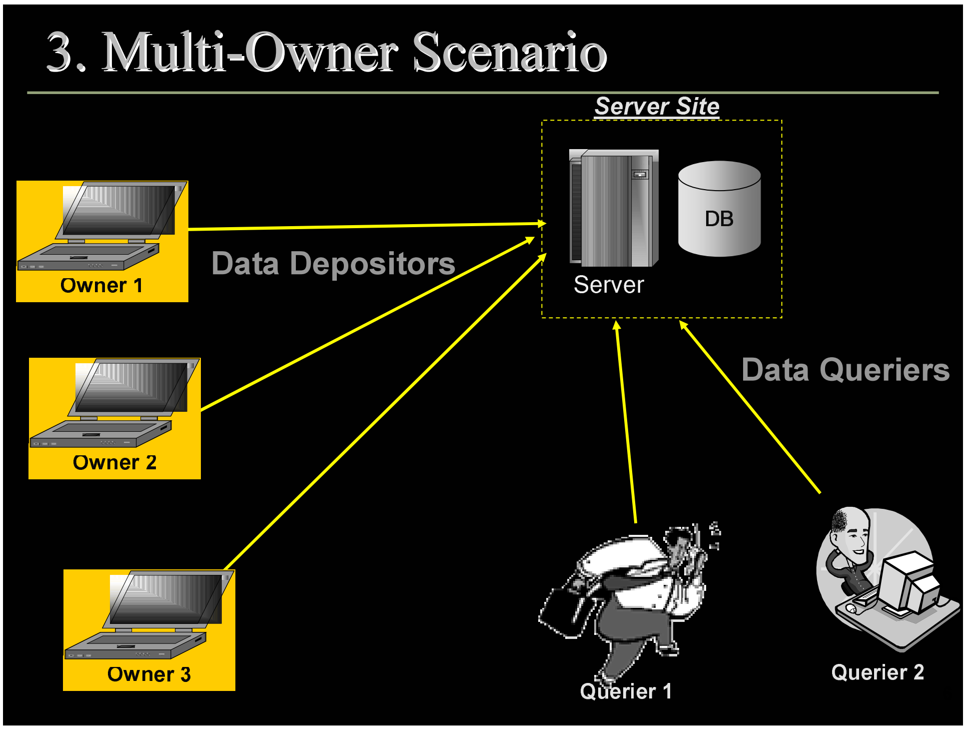
1. Unified Owner Scenario



Note that:

- Querier may be anemic (battery, CPU, storage)
- Querier may have a slow/unreliable link


2. Multi-Querier Scenario
Server Site
Data Deposit
Owner
DB
Server
Data Queries
Querier 1
Querier 2
Querier 3


3. Multi-Owner Scenario
Server Site
Owner 1
Data Depositors
DB
Server
Owner 2
Data Queriers
Owner 3
Querier 1
Querier 2

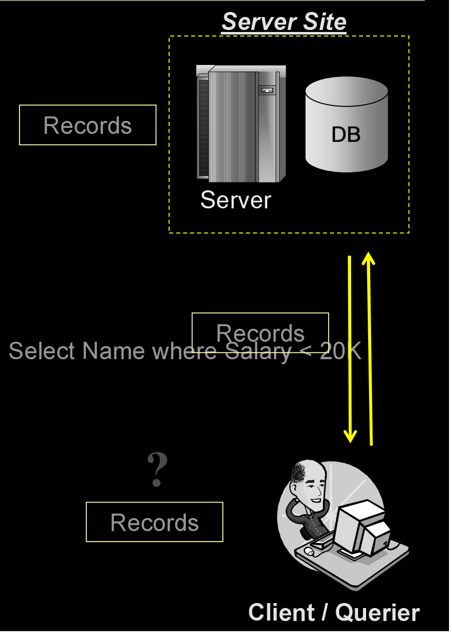
Client queries the Database at the Server

1) Client issues a query
   – "Select Name where Salary < 20K"

2) Server selects appropriate records

3) Server sends records to client

4) Client receives data, wishes to check the authenticity & integrity of the query reply
   – Do these records originate from the data owner?
   – Has anyone modified the records in any way?



Integrity and Authenticity in the ODB model

- Client-Server Protocol
  - Client queries the database
  - Server returns a set of records matching query predicates
  - Client wants to ensure the correctness of the query reply
- Assuming selection-type queries (i.e., no aggregation)
- Need to minimize overhead:
  - Bandwidth, Computation, Storage at the Querier, Owner and Server
- Server not fully trusted. It might try to:
  - Insert fake records
  - Modify existing records
- Data Privacy?
  - Subject of related work at UCI.

Challenge

How to provide efficient authentication and integrity for a potentially large and unpredictable set of records returned?

Integrity Granularity

- The granularity level affects performance
  - Too Fine: high computation overhead
  - Too Broad: high communication overhead

Record ID	Age	Salary
1	27	40K
2	38	50K
3	31	44K

Integrity Granularity

Table

Record ID	Age	Salary
1	27	40K
2	38	50K
3	31	44K

Signature

- Table Level Integrity
  - Large communication overhead
  - Entire table needs to be returned

Integrity Granularity

Record ID	Age	Salary
1	27	40K
2	38	50K
3	31	44K

Attribute

Signature

- **Attribute Level Integrity**
  - No wasted bandwidth
  - Yields a large amount of signatures
  - Costly for the client to verify

Integrity Granularity

Record ID	Age	Salary
1	27	40K
2	38	50K
3	31	44K

Record

Signature

- **Record Level Integrity**
  - Seems to be the optimal choice
  - Server returns matching query records along with integrity checks over entire record
  - Implies that the smallest unit of data returned is a record

Possible Solutions

- MAC's – Message Authentication Codes
  - Compute a MAC over each record, very efficient
  - Uses symmetric key
  - Works in Unified Owner Model: owner = querier
- Digital Signatures
  - Owner signs each record individually
  - RSA, ElGamal Family, BGLS (Boneh, et al.)
- Authenticated Data Structures
  - Build a Merkle Hash Tree based on the database records and have the data owner sign the root
  - Well suited for range queries

Digital Signature Solution

- One Solution (inefficient)
  - Data owner signs each of the deposited records
  - Server returns 1 signature per record
  - For *t* records, the client would need to verify *t* signatures
  - Results in high bandwidth and computational overhead
- Instead, combine multiple individual signatures into one unified signature
  - Verification of unified signature should be equivalent to verifying individual signatures
- We consider 3 signature schemes
  - Condensed RSA, extension of Batch RSA
  - ElGamal Family, feasibility of solution?
  - BGLS (Boneh, et al.), allows for aggregation

Batch Verification of RSA Signatures

- Batching: useful when many signature verifications need to be performed simultaneously
- Reduces verifier's computational overhead
  - By reducing the total number of modular exponentiations
- Fast screening of RSA signatures (Bellare, et al.):
  - Given a batch instance of signatures $\{\sigma_1, \sigma_2 \ldots \sigma_t\}$ on distinct messages $\{m_1, m_2 \ldots m_t\}$

$$\left(\prod_{i=1}^{t} \sigma_i\right)^e \equiv \prod_{i=1}^{t} h(m_i) \pmod{n}$$

where h() is a full domain hash function

Fast Screening of RSA Signatures

- Reduces querier computation but not bandwidth overhead
  - All individual signatures are sent to the querier for verification
- Bandwidth overhead can be overwhelming
  - Consider weak (anemic) queriers
  - Query replies can contain many records
  - Each RSA signature is at least 1024 bits long!

Can we do better?

Condensed RSA

- In condensed RSA, the Server
  - Selects the records matching a posed query
  - Multiplies the corresponding RSA signatures
  - Returns a single aggregated signature to the querier

Server

Given *t* different signatures $\{\sigma_1, \sigma_2 \dots \sigma_t\}$,

Computes condensed signature

$$\sigma_{1,t} = \prod_{i=1}^{t} \sigma_i$$

Sends $\sigma_{1,t}$ to the querier

$\sigma_{1,t}$ →

Querier

Given *t* different messages $\{m_1, m_2 \dots m_t\}$ and $\sigma_{1,t}$

Verifies the condensed signature:

$$(\sigma_{1,t})^e \equiv \prod_{i=1}^{t} h(m_i)(\bmod n)$$

Condensed RSA

- Reduces querier computation costs
  - Querier performs *t-1* multiplications and a single exponentiation
- Achieves constant bandwidth overhead
  - Querier receives a single RSA signature
- Is as secure as Batch Verification of RSA
  - Reduction: if we can break Condensed RSA, then we can break Batch Verification of RSA (construct batch that passes the Fast Screening Test)
- Not directly applicable in the multi-owner model
  - Server would create one condensed signature per signer

$$(\sigma_{1,t})^e \equiv \prod_{i=1}^{t} h(m_i)(\bmod n)$$

Batching ElGamal Family signatures

- Signatures are efficient to generate
  - Pre-compute values
- Different signers can share system parameters
  - Would be applicable for the Multi-Owner model
- Batch verification is possible but
  - Currently known secure methods require "small-exponent test"
  - The verifier is required to exponentiate (with small exponent) each component *signature* before verifying the batch
  - Without this test, an adversary can create a batch instance which satisfies the verification criterion without possessing valid individual signatures

Unfortunately, no secure way to aggregate ElGamal type signatures !

Aggregated signature scheme by Boneh et al.

- Signature Scheme based on Elliptic Curve, Bilinear Mapping, GAP-DH group
  - GAP-DH Group: Decisional-DH is easy, Computational-DH conjectured to be hard
- Signatures by multiple signers on different messages can be combined into one short signature.
  - Applicable for the multi-owner ODB model
- Bilinear Mapping

Let G_1, G_2 be multiplicative cyclic groups of prime order *p*
e is a computable bilinear map, *e*, such that

for all *p* in G_1, *q* in G_2, and *a*,*b* in Z_p, $e(p^a,q^b) = e(p,q)^{ab}$

Aggregated signature scheme by Boneh et al.

Key Generation:

Pick a generator g and a random $x \in Z_p$ and compute $v = g^x \pmod{p}$

v is the public key and x is the secret key.

Signing:

Let $h = H(m)$ be the hash of the message

$\sigma = h^x \pmod{p}$

Aggregation:

To aggregate t signatures, compute the product $\sigma_{1,t} = \prod_{i=1}^{t} \sigma_i$

Verification:

Compute the product of the hashes and verify where e() is a computable bilinear mapping $e(\sigma_{1,t}, g) = \prod_{i=1}^{t} e(h_i, v_i)$

$$e(\sigma_{1,t}, g) = e(\prod_{i=1}^{t} h_i^{x_i}, g) = \prod_{i=1}^{t} e(h_i, g)^{x_i} = \prod_{i=1}^{t} e(h_i, g^{x_i}) = \prod_{i=1}^{t} e(h_i, v_i)$$

Aggregated signature scheme by Boneh et al.

- Applicable to all three ODB flavors
  - Constant bandwidth overhead
- In case of the Unified-owner and Multi-querier models
  - Querier computation involves $t-1$ multiplications and two bilinear mappings.
- In case of the Multi-owner scenario
  - Querier computation involves $k+1$ bilinear mappings (k = number of signers) and
  - ($k*t - 1$) multiplications where t is number of signatures by each user
- Bilinear mappings are expensive to compute.
  - Computing a single bilinear mapping in a Field F_p where $|p|$ is 512 bits on a P3-977MHz takes 31 mSec!

Cost Comparisons

Querier computation costs:

(P3-977MHz, Time in mSec)

		Condensed-RSA	Batch ElGamal	BGLS
Sign	1 signature	6.82	3.82	**3.54**
Verify	1 signature	**0.16**	8.52	62
	1 signer, 1000 sigs	**44.12**	1623.59	184.88
	10 signers, 100 sigs each	**45.16**	1655.86	463.88
	10 signers, 1000 sigs each	**441.1**	16203.5	1570.8

Parameters used (bit size):

RSA: n = 1024 bits, e = 17
DSA: p = 1024 bits and q = 160 bits
BGLS: Field F_p with p = 512 bits

Cost Comparisons

Querier bandwidth overhead:

(Unit: bits)

	Condensed-RSA	Batch ElGamal	BGLS
1 signature	1024	1184	**512**
1 signer, 1000 sigs	1024	1184 K	**512**
10 signers, 100 sigs each	10240	1184 K	**512**
10 signers, 1000 sigs each	10240	11840 K	**512**

Features of the 3 Signature Schemes

1. Condensed RSA
   – Can aggregate but cannot combine signatures by distinct signers
   – Querier computation as well as bandwidth overhead is linear in the number of signers

2. BGLS
   – Can aggregate signatures from different users
   – Querier computation overhead linear in the number of signers
   – Expensive bilinear mapping operation

3. Batch ElGamal
   – Can combine signatures by distinct users and batch verify but cannot aggregate
   – Querier computation as well as bandwidth overhead linear in the number of signatures

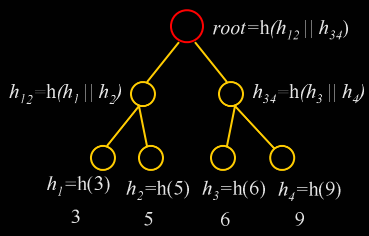
Authenticated Data Structures

- Search Directed Acyclic Graphs (Search DAG's)
  – Devanbu, Stubblebine, et al.
  – Any data structure that can be modeled as a Search DAG can be converted an authenticated structure
  – Includes: binary tree, b-tree, skip list, range tree, …
  – Idea: Create verification object that proves integrity of results

- Merkle Hash Trees
  – Originally used for authenticated public key distribution and one-time signatures
  – Certificate Revocation Tree (CRT), Kocher 1998
    - Leaves represent ranges of valid certificates
    - Certification Authorities issues the CRT to directory services
    - Provides a short proof.

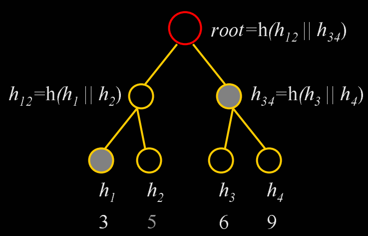
Authenticated Data Structures

- Construction of a Merkle Hash Tree (MHT)
  - Leaf nodes contain hash of their element
  - Interior nodes contain hash of concatenation of its children
  - Root node is signed



Authenticated Data Structures

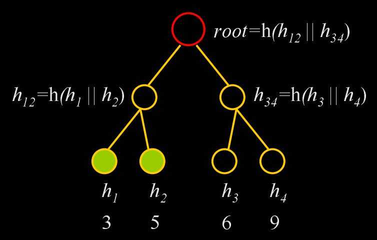
- Construction of a Merkle Hash Tree (MHT)
  - Leaf nodes contain hash of their element
  - Interior nodes contain hash of concatenation of its children
  - Root node is signed
- Querying a MHT (authenticated dictionary)
  - Search for 5
  - Need nodes on the co-path from 5 to root
  - Co-path nodes allows querier to recreate the root



Authenticated Data Structures

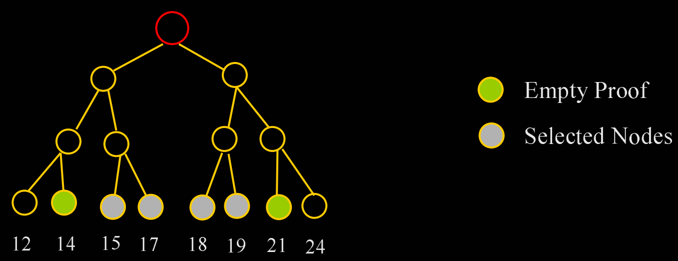
- Merkle Hash Trees in ODB
  - Leaves represent ordered set of database records (sorted by an attribute)
  - Data owner signs root
  - Server returns to the client the nodes necessary to recompute the root node
  - By verifying the root's signature, the client can be confident of the query reply
  - Due to the size of the data sets, the MHT is implemented using a B-tree (reduce I/O costs)

- Empty Proof
  - Prove that a record does not exist
  - Return siblings covering range of searched upon value
  - Search for 4
  - Return co-path's for nodes 3 and 5



Authenticated Data Structures

- Query Completeness
  - By combining empty proofs with typical MHT response we get query completeness
  - Query completeness ensures user of that all records in the appropriate range were accounted for
    - Protects against "lazy server"
  - Range Query
    - Select * where 15K ≤ Salary ≤ 20K
- MHT's in ODB are most useful for range queries
  - One tree for every attribute



In conclusion…

- Summary
  - Compared performance of digital signature schemes in providing integrity & authenticity of query replies
  - No clear winners
- Future Work
  - Query completeness (“lazy” server) with signature schemes – on-going work, some neat results
  - Any other efficient and practical signature scheme that allows multi-signer aggregation?
  - Find solution to batching of ElGamal Family signatures