Signature Bouquets: Immutability for Aggregated/Condensed Signatures

Einar Mykletun, Maithili Narasimha, Gene Tsudik
Computer Science Department
School of Information and Computer Science
University of California, Irvine
{mykletun, mnarasim, gts}@ics.uci.edu

Abstract

Database outsourcing is a popular industry trend which involves organizations delegating their data management needs to an external service provider. In this model, a service provider hosts its clients' databases and offers mechanisms for clients to create, store, update and access (query) their databases. Since a service provider is almost never fully trusted, security and privacy of outsourced data are important concerns.

This paper focuses on integrity and authenticity issues in outsourced databases. Whenever someone queries a hosted database, the returned results must be demonstrably authentic: the querier needs to establish – in an efficient manner – that both integrity and authenticity (with respect to the actual data owner) are assured. To this end, some recent work [19] examined two relevant signature schemes: one based on a condensed variant of batch RSA [3] and the other – on aggregated signature scheme by Boneh, et al. [4]

In this paper, we introduce the notion of **immutability** for aggregated signature schemes. Immutability refers to the difficulty of computing new valid aggregated signatures from a set of other aggregated signatures. This is an important feature, particularly for outsourced databases, as lack thereof would enable a frequent querier to eventually amass enough aggregated signatures to answer other (un-posed) queries, thus becoming a *de facto* service provider. Since the schemes considered in [19] do not offer immutability, we propose several practical methods to achieve it.

1 Introduction

Database outsourcing is a prominent example of the more general commercial trend of outsourcing non-core competencies. In the Outsourced Database (ODB) Model, a third-party database service provider offers adequate software, hardware and network resources to host its clients' databases as well as mechanisms to efficiently create, update and access outsourced data.

The ODB model poses numerous research challenges which influence overall performance, usability and scalability. One of the biggest challenges is the security of hosted data. A *client* stores its data (which is usually a critical asset) at an external, and potentially untrusted, database service provider. It is thus important to secure outsourced data from potential attacks not only by malicious outsiders but also from the service provider itself.

The two pillars of data security are privacy and integrity. (We use the term integrity in a somewhat broad sense, encompassing both *data integrity* and *authentication of origin*.) The need for data privacy in the ODB model has been recognized and addressed, to some degree, in prior work by Hacigümüş, et al. [13]. The central problem in the context of privacy is allowing a client to efficiently query its own data hosted by a third-party service provider (referred to as simply "server" from here on) while revealing to the latter neither the actual query nor the data over which the query is executed.

Other relevant prior work [9, 14] examined integrity issues in outsourced databases and suggested some limited solutions.[1] Recently, more general techniques were investigated in a paper by Mykletun, et al. [19] which proposed using two signature schemes suitable for querying outsourced databases: one based on a variant of RSA and the other – on the aggregated signature scheme by Boneh, et al. [4]. Essentially, these schemes enable bandwidth- and computation-efficient integrity verification for any possible query reply. However, techniques proposed in [19] (as well as in [9]) are mutable, i.e., any entity in possession of multiple authentic query replies can derive other, equally authentic query replies.

We view mutability not as a flaw of the underlying signature schemes but rather as an issue with their specific application in the ODB model. In this paper, we focus on providing a feature that we term *immutability* for aggregated signature schemes.

Contributions: This work makes several contributions. First, it defines (albeit, informally) the new notion of immutability for aggregated signatures which is, at some level, equivalent to resistance of aggregated signature schemes to adaptive attacks. Second, it demonstrates some simple add-on techniques for schemes considered in [19]. These techniques provide, at a little additional cost, immutability for the respective underlying signature schemes.

Organization: In section 2, we describe the ODB model in more detail. Section 3 motivates the need for immutable aggregated signature schemes. Next, section 4 describes the variant of RSA that allows aggregation of signatures by a single signer and the aggregated signature scheme by Boneh et al. [4] which allows aggregation of signatures by multiple signers. Section 5 then presents some techniques to achieve *immutability* for these two schemes. Section 6 discusses the overhead associated with the proposed techniques, followed by section 7 which overviews relevant prior work. The paper concludes with the summary of results in section 8.

2 System Model

The ODB model is an example of the well-known Client-Server paradigm. In ODB, a *Database Service Provider* (which we refer to as a server) has the infrastructure to host outsourced databases and provides efficient mechanisms for remote clients to create, store, update and query their databases.

Clients are assumed to trust the server to faithfully maintain outsourced data. Specifically, the server is relied upon for the replication, backup and availability of outsourced databases. However, the server is assumed not to be trusted with the integrity of the actual database contents. This lack of trust is crucial as it brings up new security issues and serves as the chief motivation for our work. Specifically, we want to prevent the server from making unauthorized modifications to the data stored in the database.

Depending on the types of clients involved, we distinguish among three flavors of the ODB model:

1. **Unified Client:** a database is owned by a single client which is also the only entity querying the same database. This is the simplest ODB scenario with relatively few security challenges (in terms of integrity).

2. **Multi-Querier:** a database is owned by a single client but multiple *queriers* are allowed to query the hosted database. This scenario is very similar to authentic third-party publication [9].

3. **Multi-Owner:** a database is jointly owned by multiple clients and multiple queriers are allowed to query the hosted database. This scenario is typical in many organizational settings where multiple

[1] See section 7 for the discussion of this and other related work.

users/entities are allowed to own a subset of records within the same database. (Consider, for example, a sales database where each salesperson owns all records for the transactions that she performed.)

Since the integrity issues in the Unified Client scenario are few and easily handled with standard textbook techniques, in the remainder of this paper, we focus on the Multi-Querier and Multi-Owner scenarios.

We assume that a querier may be a device (or an entity) limited in all or some of: computation, communication and storage facilities. A cellphone, a wireless PDA or a computer communicating over a slow dial-up line are all examples of such *anemic* queriers. Limited amount of battery power may be an additional, yet orthogonal, issue.

All of these constraints incentivize new techniques that optimize (i.e., minimize) both communication and computation overhead for the queriers in the ODB model. To this end, the recent work in [19] considered two signature schemes: Condensed-RSA and Aggregated-BGLS both of which allow the server to return to the querier a set of records[2] matching the query predicate along with a single *aggregated* signature. Condensed-RSA is very efficient but only permits aggregation of signatures produced by a single signer. In contrast, Aggregated-BGLS is less efficient but supports aggregation of signatures produced by multiple signers. Hence, Condensed-RSA is more suitable for the Multi-Querier, and Aggregated-BGLS – for the Multi-Owner, scenario.

3 Motivation

Although both techniques explored in [19] are fairly practical, each exhibits a potentially undesirable *mutability* feature. *Mutability* means that anyone in possession of multiple aggregated signatures can derive new and valid (authentic) aggregated signatures which may correspond to un-posed queries. For example, consider a database with two relations *employee* and *department* with the following respective schemas: *employee(empID, name, salary, deptID)* and *department(deptID, managerID)*. We now suppose that two SQL queries: Q1 and Q2 (in Figure 1) are posed. Essentially, Q1 asks for all managers' names and salaries where managers' salary $> 100K$ and Q2 asks for the same information for all managers whose salary $\geq 140K$.

```
Q1.        SELECT e.name, e.salary
           FROM employee e, department d
           WHERE e.empID = d.managerID AND e.salary > 100000

Q2.        SELECT e.name, e.salary
           FROM employee e, department d
           WHERE e.empID = d.managerID AND e.salary ≥ 140000

Q3.        SELECT e.name, e.salary
           FROM employee e, department d
           WHERE e.empID = d.managerID AND
                     e.salary BETWEEN 100000 AND 140000
```

Figure 1: SQL Queries

A querier who posed SQL queries Q1 and Q2 and obtained the corresponding aggregated signatures

[2] In our setting, the client's database is a typical relational database (RDBMS) where data is organized in tables (or relations). Each table has multiple rows and columns. A column represents an attribute of the table and a row (or a record) is an instance of the table

S1 and S2, can compute, on her own, a valid new signature for the un-posed query Q3, also in Figure 1. Q3, in essence, is (Q1 − Q2) i.e., information about all managers who earn more than 100K and less than 140K. The specifics of computing a new signature from a set of existing signatures depend on the underlying aggregated signature scheme, as described in the next section.

We note that the above example is not specific to the use of aggregated signature schemes for integrity purposes in the ODB context. In fact, without aggregated signatures, it suffices to obtain a single authentic query reply in order to construct other such replies. Suppose that, instead, plain record-level signatures were used (e.g., DSA or RSA). In this case, a single SELECT-style query would cause the server to construct a query reply containing a set of records (matching the query predicate), each accompanied by its signature. Clearly, it is then trivial for a querier to construct legitimate and authentic query replies for un-posed queries since she is free to manipulate individual record-level signatures. Furthermore, other more efficient methods, such as the elegant constructs based on Merkle Hash Trees (MHTs) suggested by Devanbu, et al. [9], are equally susceptible to mutability of authentic query replies.[3]

At this point, one might wonder about the dangers of *mutability* in aggregated signature schemes, i.e., whether it is really an undesirable feature. The answer is clearly application dependent; in some settings, it might indeed be desirable to derive, propagate and/or transfer authentic query replies. For example, if queriers are expected, over time, to be able to propagate or answer queries from other queriers (e.g., to reduce load on a busy server), mutability might be an attractive feature. However, in the present setting of outsourced databases, mutability may indeed be undesirable. Recall that, in both Multi-Querier and Multi-Owner ODB scenarios, owner(s) outsource the database to an external server. The server handles all queries and provides each querier the result set matching a query along with an aggregated signature by the original data owner. In other words, the server is the *authorized agent* for re-distributing the information stored in, or derived from, the outsourced database.

It is safe to assume that Multi-Querier and Multi-Owner scenarios involve more queriers than data owners. Consequently, one reason for avoiding mutability is to prevent *unauthorized splitting and re-distribution* of authentic query replies. For example, consider the case of data owners and/or servers who wish to charge a fee for each query over the outsourced database. Consequently, it might be important to prevent queriers from deriving new valid aggregated signatures (new query reply sets are easy to derive) from prior query reply sets and re-selling information that has not been paid for. In this case, we would like the server to be the sole data distributor.

As noted in section 1, we stress that mutability can be a blessing in some application scenarios. For instance, if a server does not get paid on a per query basis, in order to reduce load, it might be beneficial to allow some queriers to themselves process queries from other queriers (i.e., to cache parts of the outsourced database).

4 Aggregated Signature Schemes

In this section, we take a closer look at the two signature schemes considered in [19] and illustrate their respective mutability properties.

[3] In [9], a querier receives from a server a set of records matching a posed query along with a set of of non-leaf nodes of an MHT. The exact composition of this set depends on the type of a query.

4.1 Condensed-RSA

The RSA [20] signature scheme is multiplicatively homomorphic which makes it suitable for combining multiple signatures generated by a single signer into one *condensed* signature.[4] A valid condensed signature signifies to the verifier that each individual signature contained in the condensed signature is valid, i.e., generated by the purported signer. Aggregation of single-signer RSA signatures can be performed incrementally by anyone in possession of individual RSA signatures. By incrementally, we mean that the signatures can be combined in any order and the aggregation need not be carried out in a single operation.

RSA Signature Scheme: We first describe the setup of the standard RSA signature scheme. A party has a public key $pk = (n, e)$ and a secret key $sk = d$, where n is a k-bit modulus formed as a product of two $k/2$-bit primes p and q. Both public and private exponents $e, d \in Z_n^*$ and satisfy $ed \equiv 1 \mod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. The minimum currently recommended k is 1024. The security of the RSA cryptosystem is widely believed to be based on the conjectured intractability of the large integer factorization problem.

In practice, an RSA signature is computed on the hash of an input message. Let $h()$ denote a cryptographically suitable hash function (such as, MD5 or SHA-1) which takes a variable length input m and produces a fixed-length output denoted as $h(m)$. A standard RSA signature on message m is computed as: $\sigma = h(m)^d \pmod{n}$. Verifying a signature involves checking that $\sigma^e \equiv h(m) \mod n$. Both signature generation and verification involve computing one modular exponentiation.

Condensed-RSA Signature Scheme: Given t different messages $\{m_1, ..., m_t\}$ and their corresponding signatures $\{\sigma_1, ..., \sigma_t\}$ generated by the same signer, a Condensed-RSA signature is computed as the product of all t individual signatures:

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

The resulting aggregated (or condensed) signature $\sigma_{1,t}$ is of the same size as a single standard RSA signature. Verifying an aggregated signature requires the verifier to multiply the hashes of all t messages and checking that:

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

Mutability of Condensed RSA: Given two condensed signatures: $\sigma_{1,i}$ on messages $\{m_1, ..., m_i\}$ and $\sigma_{1,j}$ on messages $\{m_1, ..., m_j\}$ where $j < i$, it is possible to obtain a new condensed signature $\sigma_{j+1,i}$ on messages $\{m_{j+1}, ..., m_i\}$ by simply dividing $\sigma_{1,i}$ by $\sigma_{1,j}$ (modulo n)[5].

$$(\sigma_{j+1,i}) \equiv (\sigma_{1,i})/(\sigma_{1,j}) \pmod{n}$$

Similarly, given two condensed signatures $\sigma_{1,i}$ on messages $\{m_1, ..., m_i\}$ and $\sigma_{i+1,j}$ on messages $\{m_{i+1}, ..., m_j\}$, anyone can obtain a new condensed signature $\sigma_{1,j}$ on messages $\{m_1, ..., m_i, m_{i+1}, ..., m_j\}$ by multiplying $\sigma_{1,i}$ and $\sigma_{i+1,j}$ assuming all the messages are distinct.

$$(\sigma_{1,j}) \equiv (\sigma_{1,i}) \times (\sigma_{i+1,j}) \pmod{n}$$

[4] We use the term *condensed* in the context of a single signer and *aggregated* in the context of multiple signers. Clearly, former is a special case of the latter.

[5] Of course, this is possible if $\sigma_{1,j}$ is relatively prime to n

4.2 BGLS

Boneh, et al. in [4] construct an interesting aggregated signature scheme that allows aggregation of signatures generated by multiple signers on different messages into one short signature based on elliptic curves and bilinear mappings. This scheme (BGLS) operates in a Gap Diffie-Hellman group (GDH) – a group where the Decisional Diffie-Hellman problem (DDH) is easy while the Computational Diffie-Hellman problem (CDH) is hard. The first instance of such a group was illustrated in [16]. Prior to describing the BGLS signature scheme, we briefly overview the necessary parameters [4].

- G_1 is a cyclic additive group with generator g_1
- G_2 is a cyclic group with generator g_2
- ψ is a computable isomorphism from G_2 to G_1, with $\psi(g_2) = g_1$
- e is a computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$ as described below

A bilinear mapping $e : G_1 \times G_2 \rightarrow G_T$, where $|G_1| = |G_2| = |G_T|$, satisfies the following two properties.

1. Bilinearity: $\forall p \in G_1, q \in G_2$ and $a, b \in \mathbb{Z}$, $e(ap, bq) = e(p, q)^{ab}$
2. Non-degenerativity: $e(g_1, g_2) \neq 1$

These two properties imply that, for any $p_1, p_2 \in G_1, q \in G_2, e(p_1 + p_2, q) = e(p_1, q) \cdot e(p_2, q)$; and, for any $p, q \in G_2, e(\psi(p), q) = e(\psi(q), p)$.

BGLS Signature Scheme: BGLS requires the use of a full-domain hash function $h() : \{0,1\}^* \rightarrow G_1$. Key generation involves picking a random $x \in \mathbb{Z}_p$, and computing $v = xg_2$. The public key is $v \in G_2$ and the secret key is $x \in \mathbb{Z}_p$. Signing a message m involves computing $H = h(m)$, where $H \in G_1$ and $\sigma = xH$. The signature is σ. To verify a signature one needs to compute $H = h(m)$ and check that $e(\sigma, g_2) = e(H, v)$.

BGLS Aggregated Signature Scheme To aggregate t BGLS signatures one computes the point-addition operation (on the elliptic curve) of the individual signatures as follows: $\sigma_{1,t} = \sum_{i=1}^{t} \sigma_i$, where σ_i corresponds to the signature of message m_i. The aggregated signature $\sigma_{1,t}$ is of the same size as a single BGLS signature, i.e., $|p|$ bits. Similar to Condensed-RSA, the aggregation of signatures can be performed incrementally and by anyone.

Verification of an aggregate BGLS signature $\sigma_{1,t}$ involves computing the point-addition of all hashes and verifying that:

$$e(\sigma_{1,t}, g_2) = \sum_{i=1}^{t} e(H_i, v_i)$$

Due to the properties of the bilinear maps, we can expand the left hand side of the equation as follows:

$$e(\sigma_{1,t}, g_2) = e(\sum_{i=1}^{t} x_i H_i, g_2) = \sum_{i=1}^{t} e(H_i, g_2)^{x_i} = \sum_{i=1}^{t} e(H_i, x_i g_2) = \prod_{i=1}^{t} e(H_i, v_i)$$

Mutability of Aggregated BGLS: Similar to Condensed-RSA, aggregated BGLS signatures can be manipulated to obtain new and valid signatures that correspond to un-posed query replies. Specifically, it is possible to either (or both) add and subtract available aggregated signatures to obtain new ones.

For example, given 2 aggregated BGLS signatures $\sigma_{1,i}$ on messages $\{m_1, ..., m_i\}$ and $\sigma_{i+1,j}$ on messages $\{m_{i+1}, ..., m_j\}$, if the messages $\{m_1, ..., m_i\}$ and $\{m_{i+1}, ..., m_j\}$ are all distinct (i.e.,the two queries do not overlap), the verifier can obtain a new BGLS signature $\sigma_{1,j}$ on messages $\{m_1, ..., m_i, m_{i+1}, ...m_j\}$ by adding $\sigma_{1,i}$ and $\sigma_{i+1,j}$.

$$(\sigma_{1,j}) \equiv (\sigma_{1,i}) + (\sigma_{i+1,j}) \pmod p$$

5 Immutable Signature Schemes

In this section, we propose extensions that strengthen previously described signature schemes and make them *immutable*.

5.1 Immutable Condensed RSA (IC-RSA)

To make condensed-RSA signatures immutable, we use the technique that can be broadly classified as a zero-knowledge proof of knowledge of signatures. The server, instead of revealing the actual aggregated signature for a posed query, reveals only the proof of knowledge of that signature. We present two variants: one that requires interaction, based on the well-known Guillou-Quisquater scheme, and the other that is non-interactive, based on so-called “signatures of knowledge”.

5.1.1 Interactive Variant

This technique uses the well-known Guillou-Quisquater (GQ) identification scheme [12] which is among the most efficient follow-ons to the original Fiat-Shamir zero-knowledge identification Scheme [2]. The version we present is an interactive protocol between the server (Prover) and the querier (Verifier) that provides the latter with a zero-knowledge proof that the Prover has a valid Condensed-RSA signature corresponding to the records in the query result set.

Basically, as shown in Figure 2, the server returns to the querier the result set along with a *witness*. The querier then sends a random *challenge* to which the server replies with a valid *response*. The *response* together with the *witness* convince the querier of server's knowledge of the Condensed-RSA signature, without revealing any knowledge about the Condensed-RSA signature itself.

The actual protocol is shown, in more detail, in Figure 3. We use the terms *Prover* (P) and *Verifier* (V) instead of Server and Querier, respectively, since the protocol is not specific to the ODB setting [6]. Let $X = \sigma_{1,t} = \prod_{i=1}^{t} \sigma_i \pmod n$ be the condensed-RSA signature computed as shown above. Recall that (e, n) is the public key of the original data-owner which all concerned parties are assumed to possess. Let $M \equiv \prod_{i=1}^{t} h(m_i) \pmod n$ and $X^e = (\sigma_{1,t})^e \equiv M \pmod n$.

In step 0, the querier poses a query (not shown in figure 3). In step 1, the server (prover) replies with the result set for that query as well as a commitment Y. Note that $Y = r^e \pmod n$ where r is a randomly chosen element in $\mathbb{Z}_n^*$ and n is the RSA modulus of the data owner who generated the individual RSA signatures corresponding to the records in the result set [7] and e, the corresponding public exponent. In step 2, the verifier (querier) sends back a challenge v that is chosen randomly from $\{0,1\}^{l(k)}$ where $l(k)$ is the

[6] The original GQ scheme proposed in [12] is *identity-based* since it is used by the Prover to prove his “identity” to the verifier. However, in the current scenario, we present a version that is not id-based and does not require a key generation phase since the server uses the public key of the data owner to prove knowledge of the condensed-RSA signature by that data owner

[7] Recall that Condensed-RSA allows only single signer aggregation

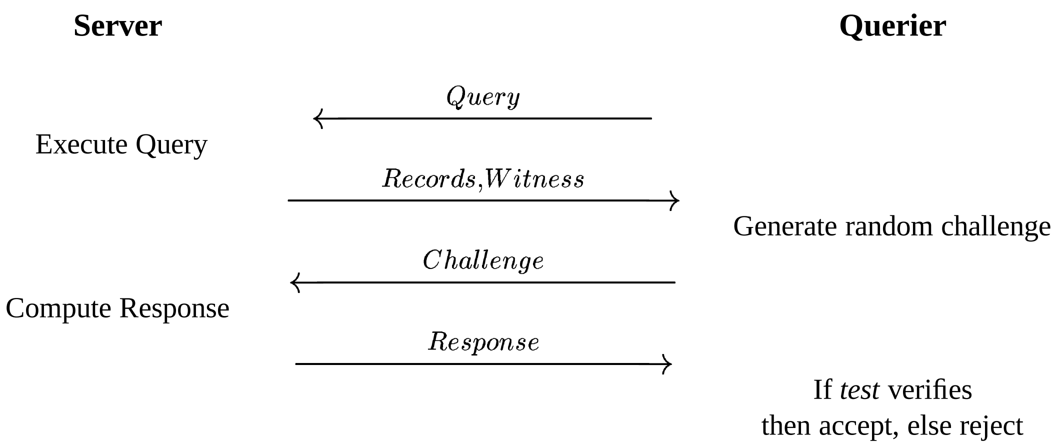

Figure 2: Protocol Overview

bit-length of the public exponent e. In Step 3, server, upon receiving the challenge v, computes the response $z = rX^v \pmod{n}$ where X is the Condensed-RSA signature of the result set. In Step 4, the verifier accepts the proof if $z^e \equiv YM^v \pmod{n}$ where M is the product of (hashes of) all messages in the result set. Note that $z^e \equiv (rX^v)^e \equiv r^e X^{ev} \equiv Y(X^e)^v \equiv YM^v \pmod{n}$. Hence the protocol works.

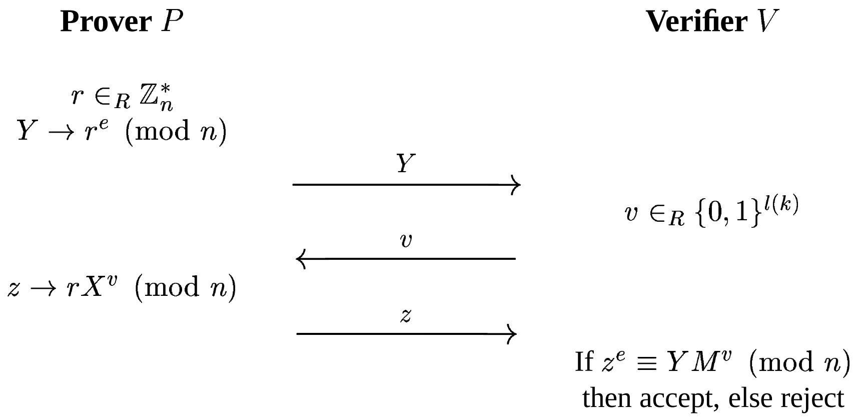

Figure 3: IC-RSA GQ-based Interactive Technique

Security Considerations: GQ is RSA-based; the protocol is known to be honest-verifier zero-knowledge and is secure against impersonation under passive attacks, assuming RSA is one-way [12].
Forgery: The public exponent e defines the security level, i.e., a cheating prover can convince the verifier, and thus defeat the protocol with probability $1/e$, by correctly guessing the value of the *challenge v a priori*. Therefore, the bit-length of v should be large enough. Note that the above protocol can be run multiple times for commensurably lower probability of successful forgery. In general, if it is run t times, the probability of forgery is e^{-t}.
Security Assumptions: The security of the protocol is based on the hardness of the RSA problem (i.e., computing e-th roots mod a composite integer n which is formed as a product of two large primes.)

5.1.2 Non-Interactive Immutable Condensed-RSA

The second, non-interactive, variant uses the technique of *Signatures of Knowledge* first popularized by Camenisch and Stadler in [6]. Specifically, we use the so-called **SKROOTLOG** primitive which can be used to prove knowledge of an e-th root of the discrete logarithm of a value to a given base. Before presenting the details, we briefly describe how this technique is used in our scenario. Conceptually, the server reveals all records matching the query as well as a signature of knowledge for the actual condensed-RSA signature corresponding to these records. A querier verifies by checking the SKROOTLOG proof. However, since the querier never actually gets the condensed signature, she can not exploit the mutability of Condensed-RSA to derive new signatures. In general, the querier can not derive proofs for any other queries by using proofs for any number of previously posed queries.

SKROOTLOG Details: Let $G =< g >$ be a cyclic group of order n. An e-th root of the discrete logarithm of $y \in G$ to the base g is an integer α satisfying $g^{(\alpha^e)} = y$ if such a α exists. If the factorization of n is unknown, for instance if n is an RSA modulus, computing e-th roots in $\mathbb{Z}_n^*$, is assumed to be infeasible. A signature of knowledge of an e-th root of the discrete logarithm of y to the base g is denoted $SKROOTLOG[\alpha : y = g^{\alpha^e}](m)$

Below, we briefly outline an efficient version of SKROOTLOG proposed in [6] which is applicable when the public exponent e is a small value (for instance, this efficient SKROOTLOG version is applicable when the value of e is set to 3).

Definition 1 *If e is small, it is possible to show the proof of knowledge of the e-th root of the discrete log of $y = g^{\alpha^e}$ to the base g by computing the following $e - 1$ values:*

$$y_1 = g^{\alpha}, y_2 = g^{\alpha^2}, \ldots, y_{e-1} = g^{\alpha^{e-1}}$$

and showing the signature of knowledge:

$$U = SKREP[\alpha : y_1 = g^{\alpha} \wedge y_2 = y_1^{\alpha} \wedge \ldots \wedge y = y_{e-1}^{\alpha}]$$

that the discrete logarithms between two subsequent values in the list g, y_1, …, y_{e-1} are all equal (to α) and known (to the prover).

Below, we give the formal definition of $SKREP$ as in [6]:

Definition 2 *A signature of the knowledge of representations of $y_1, \ldots, y_w$ with respect to bases $g_1, \ldots, g_v$ on the message m is defined as follows:*

$$SKREP\left[(\alpha_1, \ldots, \alpha_u) : \left(y_1 = \prod_{j=1}^{l_1} g_{b_{1j}}^{\alpha_{e_{1j}}}\right) \wedge \ldots \wedge \left(y_w = \prod_{j=1}^{l_w} g_{b_{wj}}^{\alpha_{e_{wj}}}\right)\right](m)$$

where the indices $e_{ij} \in \{1, \ldots, u\}$ refer to the elements $\alpha_1, \ldots, \alpha_u$ and the indices $b_{ij} \in \{1, \ldots, v\}$ refer to the base elements $g_1, \ldots, g_v$.
The signature consists of an $(u + 1)$ tuple $(c, s_1, \ldots, s_u) \in \{0,1\}^k \times \mathbb{Z}_n^*$ satisfying the equation

$$c = \mathcal{H}\left(m||y_1||\ldots||y_w||g_1||\ldots||g_v||\{\{e_{ij}, b_{ij}\}_{j=1}^{l_i}\}_{i=1}^{w}||y_1^c \prod_{j=1}^{l_1} g_{b_{1j}}^{s_{e_{1j}}}||\ldots||y_w^c \prod_{j=1}^{l_w} g_{b_{wj}}^{s_{e_{wj}}}\right)$$

$SKREP$ can be computed easily if the u-tuple $(\alpha_1, \dots, \alpha_u)$ is known. Prover first chooses $r_i \in_R \mathbb{Z}_n$ for $i = 1, \dots, u$, computes c as

$$c = \mathcal{H}\left(m||y_1|| \dots ||y_w||g_1|| \dots ||g_v||\{\{e_{ij}, b_{ij}\}_{j=1}^{l_i}\}_{i=1}^{w}|| \prod_{j=1}^{l_1} g_{b_{1j}}^{r_{e_{1j}}} || \dots ||y_w^c \prod_{j=1}^{l_w} g_{b_{wj}}^{r_{e_{wj}}} \right)$$

and then sets $s_i = r_i - c\alpha_i \pmod{n}$ for $i = 1, \dots, u$

Non-Interactive IC-RSA: The server executing a client query is required to perform the following:

1. select records that match the query predicate;
2. fetch the signatures corresponding to these records;
3. aggregate the signatures (by multiplying them modulo n, as mentioned above) to obtain the condensed RSA signature σ;
4. send the individual records $\mathcal{M} = \{m_1, \dots, m_t\}$ back to the querier along with a proof of knowledge of σ which is essentially a SKROOTLOG proof showing that the server knows the e-th root of g^{σ^e}. In other words, SKROOTLOG proof shows that the server knows the e-th root of $g^{\prod m_i}$. In order to show this, the server sends g^{σ}, g^{σ^2} and[8] the SKREP proof computed as above.

Security Considerations: In practice, the efficient version of the SKROOTLOG proof which was described in the previous section cannot be used as is. This is because the values $y_1 \dots y_{e-1}$ that are required for the SKREP proofs leak additional information about the secret. Hence a randomized version that is proposed in [6] needs to be used. The interactive protocol corresponding to the above definition of SKROOTLOG is proven honest-verifier zero-knowledge in [5]. For brevity, we skip the details of this discussion and refer interested readers to [6]. However, we would like to note that the security of the SKROOTLOG protocol is based on the difficulty of the discrete logarithm problem and the RSA problem. In addition, SKREP is based on the security of Schnorr signature scheme. The Non-Interactive IC-RSA, which in essence is the SKROOTLOG primitive, is therefore honest-verifier zero-knowledge [5]. This implies that the querier who is given only the proof of the condensed RSA signature, can not derive new signatures. In addition, the querier can not derive new proofs for any other queries by using proofs for any number of previously posed queries.

5.1.3 Discussion

In this section, we compare the two techniques presented above.

- **Initialization and Parameter Generation:** Non-Interactive technique (SKROOTLOG based) requires an elaborate parameter generation phase at the server. For each data owner whose RSA public key is (n, e), the server needs to generate a large prime $p = j * n + 1$ (where n is the RSA modulus and j is some integer) and an element $g \in \mathbb{Z}_p^*$ such that order of g is n. On the other hand, Interactive (GQ based) technique requires no additional parameter generation at the server since the server only requires to have knowledge of each data owner's RSA public key (n, e).

[8] Note that the server need not send $g^{\sigma^3} = g^{\prod m_i}$ explicitly since the querier can compute this value knowing g and the individual m_i-s

- **Verifiability:** In the Non-Interactive technique, the SKROOTLOG proof provided by the server is universally verifiable (or in other words, the proof is self authenticating and hence transferrable). On the other hand, the Interactive (GQ-based) technique provides guarantees only to the interactive verifier who poses the challenge and the proof of knowledge in this case is non-transferrable. This is perhaps the biggest difference between the two techniques.
- **Communication Rounds:** Since SKROOTLOG based technique requires no interaction with the verifier for the proof, it requires no additional rounds of communication. In other words, the server executes the query and returns the result set as well as the proof of knowledge of the corresponding unified Condensed-RSA signature. On the other hand, the Interactive technique requires two additional rounds of communication with the verifier.

5.2 Immutable BGLS

The extension to aggregated BGLS to achieve immutability is very simple: The server computes its own signature on the whole query reply and aggregates it with the aggregated BGLS signature of the owners. In other words, for a given query whose result includes messages $\{m_1, m_2, \ldots, m_k\}$, the server computes $\mathcal{H} = h(m_1 || m_2 ... || m_k)$ where $||$ denotes concatenation, and signs this hash $\mathcal{H}$ using its own private key x_s to obtain $x_s\mathcal{H}$ and computes:

$$\sigma = \sigma_{1,t} + x_s\mathcal{H}$$

where $\sigma_{1,t}$ is the aggregated BGLS signature of t messages obtained as described above.

Now, a valid and authentic query reply comprises of the records in the result set along with an authentic Immutable-BGLS signature on the entire result set. Due to this simple extension, it is no longer feasible for anybody to manipulate the existing Immutable-BGLS signatures to obtain new and authentic ones or get any information about individual component BGLS signatures. Verification of an immutable BGLS signature σ involves computing the individual hashes H_i-s of each message as well as computing the hash of the concatenation of all the messages $\mathcal{H}$ and verifying the following equality: $e(\sigma, g_2) = \sum_{i=1}^{t} e(h_i, v_i) + e(\mathcal{H}, v_s)$ where v_s is the server's public key. Due to the properties of the bilinear mapping, we can expand the left hand side of the equation as follows:

$$e(\sigma, g_2) = e(\textstyle\sum_{i=1}^{t} x_i H_i + x_s\mathcal{H}, g_2) = e(\sum_{i=1}^{t} x_i H_i, g_2) + e(x_s\mathcal{H}, g_2) =$$
$$\textstyle\sum_{i=1}^{t} e(H_i, g_2)^{x_i} + e(\mathcal{H}, g_2)^{x_s} = \sum_{i=1}^{t} e(H_i, x_i g_2) + e(\mathcal{H}, x_s g_2) = \sum_{i=1}^{t} e(H_i, v_i) + e(\mathcal{H}, v_s)$$

Security Considerations: Immutable-BGLS is a direct application of the original aggregate BGLS signature scheme (see section 4.2). The security of BGLS relies upon a Gap-Diffie Hellman group setting and specifically requires that each message included in the aggregated signature be unique. Below we argue, informally, the security of our construction of Immutable-BGLS signatures.

Let r_i denote database record i. An Immutable-BGLS signature σ is then constructed as follows: $\sigma = \sum_{i=1}^{t+1} x_i h(m_i)$, where messages $m_1, m_2, ..., m_t$ correspond to the selected records $r_1, r_2, ..., r_t$ and $m_{t+1} = (r_1 || r_2 || ... || r_t)$, i.e., the concatenation of the these records. $x_1, x_2, ..., x_t$ are the data owner's private keys and x_{t+1} is the server's key. We then claim that these $t + 1$ messages are distinct. Each database record r_i contains a unique record identifier, resulting in messages $m_1, m_2, ..., m_t$ being distinct. m_{t+1} will be unique for each record set returned as it consists exactly of the selected records, and moreover, it will be different than any m_j where $j < t + 1$. Therefore, all $t + 1$ messages are distinct.

Immutability Considerations: The immutability property of the above scheme relies upon the inability of an adversary to forge the server's signature. This, in turn, implies that such an adversary cannot use an aggregate BGLS signature to generate a new Immutable-BGLS signature that verifies. In other words, the Immutable-BGLS signature construction is resistant to mutations assuming that BGLS signatures are unforgeable.

6 Performance Analysis

In this section, we present and discuss the experimental results for immutable signature schemes. We mainly consider the overheads introduced by the extensions we made to the Condensed-RSA and BGLS signature schemes to achieve immutability. Since these signature schemes were not implemented in their entirety, we provide rough estimates on the running costs by showing the number of additional basic cryptographic operations (such as modular exponentiations and multiplications) required by the extensions and also point out any additional communication overheads. We then present the actual cost (time) required to carry out these additional operations.

Table 1 enlists the computational as well as communication overheads associated with the various techniques we propose in this paper. We use the following notation to describe the various basic cryptographic operations: $Mult^t(k) \leftarrow t$ modular multiplications with modulus of size $|k|$; $Exp_l^t(k) \leftarrow t$ modular exponentiations with modulus of size $|k|$ and exponent of size $|l|$; $BM(t) \leftarrow t$ bilinear mappings

	Computation at Client	Computation at Server	Communication
GQ Identification	$Mult^1(n) + Exp_e^2(n)$	$Mult^1(n) + Exp_e^2(n)$	2
SKROOTLOG	$Exp_h^3(p) + Exp_n^3(p)$	$Exp_n^4(p) + Exp_2^1(n) + Mult^1(n)$	0
BGLS Extension	$BM(1)$	$Mult^1(p) + Exp_q^1(p)$	0

Table 1: Cost comparison of techniques for Immutability

Table 2 gives the actual time required to generate a single signature and also the time required to verify a single signature, multiple signatures by a single signer, and multiple signatures by multiple signers in both condensed RSA and BGLS schemes. We set the RSA public exponent e to 3 for SKROOTLOG and set $e = (2^{30} + 1)$ for GQ. Note that it is essential to have a large e for GQ since e in this case is also the security parameter. Further, we have used a 1024 bit modulus n and the Chinese Remainder Theorem to speed up the signing procedure. All computations were carried out on an Intel Pentium-4 2.8 GHz processor with 1GB memory. The results for BGLS are obtained by using the MIRACL library [1] and the elliptic curve defined by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_p$ where p is a 512 bit prime and q is a 160 prime factor of $p - 1$. In the table k denotes the total number of signers and t denotes the number of signatures generated by each signer.

The next table gives the time required to generate and verify an immutable signature under the two different RSA-based techniques as well as the BGLS extension. Here we do not count the communication delays introduced by the protocols, particularly in the case of GQ which is multi-round protocol.

We also would like to mention at this point that SKROOTLOG, in addition to the above mentioned costs, also incurs additional setup costs. These costs are necessary to set the parameters prime p and element g of order n. Further, it is also worth noting that since condensed-RSA only enables single-signer aggregation, it is necessary for the server to set up multiple sets of parameters: one for each signer. (In other words, since n-s of distinct owners are different, it becomes necessary to find distinct pairs (p, g) for each n).

		Condensed-RSA		BGLS
		$e = 3$	$e = 2^{30} + 1$	
Sign	1 signature	6.82	6.82	9.939
Verify	1 signature	0.14	0.56	63.681
	t = 1000 sigs, k = 1 signer	29.11	29.531	186.56
	t = 100 sigs, k = 10 signers	30.11	34.31	463.88
	t = 1000 sigs, k = 10 signers	291.11	295.31	1570.8

Table 2: Cost comparison (time in msec): verification and signing

Technique Used	Computation at Client	Computation at Server	Total
GQ	0.309	0.309	0.618
SKROOTLOG	41.88	47.489	89.369
Immutable BGLS	34.492	11.939	46.431

Table 3: Cost comparison (time in msec): Immutable Signature Schemes

7 Related Work

Database security has been studied extensively within the database as well as cryptographic research communities. Specifically, the problem of data privacy for outsourced databases has been investigated by many. Hacigümüş, et al. examined various challenges associated with providing database as a service in [15]. In our work, we used a very similar system model.

Private Information Retrieval (PIR) [8, 10] deals with the exact matching problem and has been explored extensively in the cryptographic literature. PIR is primarily concerned with *private* retrieval of parts of data stored at an external server, such that no partial information about the query is leaked to the server. PIR techniques support searching based on either the physical location [8] of the data or using keywords [7]. However, most of current PIR techniques aim for very strong security bounds and, consequently, remain unsuitable for practical purposes. Concretely, PIR schemes typically require either multiple non-colluding servers or multiple rounds of communication.

Song et al. [21] develop a more pragmatic scheme to search on data encrypted using a secret symmetric key. Their scheme requires a single server and offers fairly low computational complexity. In summary, searching on encrypted data is becoming an increasingly popular research topic with such recent interesting results as [21, 11]. However, the aforementioned schemes only support exact-match queries, i.e., the server returns data matching either a given address or a given keyword. Hacigümüş, et al. in [13] explore how different types of SQL queries can be executed over encrypted data and provide detailed query processing and optimization techniques. Specifically, they support *range* searches and *joins* in addition to exact-match queries.

[14] investigated integrity issues in the ODB model: data encryption is used in combination with manipulation detection codes to provide integrity. As mentioned earlier [19] mainly focuses on the use of digital signatures in order to facilitate efficient integrity assessment. Specifically, [19] constructs a simple RSA-based aggregation technique based on fast batch verification [3] of RSA signatures. Another solution explored in [19] uses aggregated signatures proposed by Boneh, et al. [4]. This scheme is based on bilinear maps and provides an elegant mechanism to aggregate n individual signatures on n messages (by $t \leq n$ signers) into one single signature.

[9] explores the applicability of Merkle Hash Tree-s (MHT-s) as a technique for providing authenticity and integrity in third-party data publication settings. MHT-s are created by letting the tree leaf nodes represent the ordered set of records of a relation, sorted by a selected attribute. With the help of a MHT, a server can answer a client's query by returning the tree nodes necessary to prove the existence of the selected records. MHT-s are especially well suited for range queries due to the few nodes needing to be returned and the relatively cheap computations required by the client (1 modular exponentiation to verify the MHT's root signature and computation of hash values linear in the number of records returned). Furthermore, this solution provides *query completeness* for queries involving a single attribute in the predicate, namely proof that the server has included all applicable records in its query response. Merkle first introduced MHT-s in [18] and they were initially used for the purpose of one-time signatures and authenticated public key distribution. The use of authenticated data structures for providing data integrity in general has been studied extensively in [17].

8 Conclusions

In this paper, we introduced the notion of **immutability** for aggregated signature schemes. Some aggregated signature schemes suitable for providing data integrity and origin authentication for outsourced databases were considered recently in [19]. In this paper, we extended this work to provide several practical mechanisms to achieve immutability for these schemes. We provided a detailed comparison and performance analysis of the proposed techniques.

References

[1] MIRACL Library. `http://indigo.ie/~mscott`.

[2] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology – CRYPTO '86*, number 263 in Lecture Notes in Computer Science, pages 186–194, Santa Barbara, CA, USA, 1987. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[3] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *Eurocrypt 1998*, volume 1403, pages 191–204, 1998.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT '2003*, Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2003.

[5] J. Camenisch. Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. Vol. 2 of ETH-Series in Information Security and Cryptography. ISBN 3-89649-286-1, Hartung-Gorre Verlag, Konstanz, 1998.

[6] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology – CRYPTO '97*, number 1294 in Lecture Notes in Computer Science, pages 410–424. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1997.

[7] B. Chor, N. Gilboa, and M. Naor. Private Information Retrieval by Keywords. Technical Report TR CS0917, Department of Computer Science, Technion, 1997.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. *Journal of ACM*, 45(6):965–981, Nov. 1998.

[9] P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic third-party data publication. In *14th IFIP 11.3 Working Conference in Database Security*, pages 101–112, 2000.

[10] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. In *30th Annual Symposium on Theory of Computing (STOC)*, Dallas, TX, USA, 1998. ACM Press.

[11] E.-J. Goh. Secure indexes for efficient searching on encrypted compressed data. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[12] L. Guillou and J. J. Quisquater. A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, number 403 in Lecture Notes in Computer Science, Santa Barbara, CA, USA, 1988. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany.

[13] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing SQL over Encrypted Data in the Database-Service-Provider Model. In *ACM SIGMOD Conference on Management of Data*, pages 216–227. ACM Press, June 2002.

[14] H. Hacigümüş, B. Iyer, and S. Mehrotra. Encrypted Database Integrity in Database Service Provider Model. In *International Workshop on Certification and Security in E-Services (CSES'02 IFIP WCC)*, 2002.

[15] H. Hacigümüş, B. Iyer, and S. Mehrotra. Providing Database as a Service. In *International Conference on Data Engineering*, March 2002.

[16] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. In *Cryptology ePring Archive*, number Report 2001/003, 2001.

[17] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1), Jan. 2004.

[18] R. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Research in Security and Privacy*, 1980.

[19] E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and Integrity in Outsourced Databases. In *ISOC Symposium on Network and Distributed Systems Security (NDSS'04)*, 2004.

[20] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[21] D. Song, D. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *2000 IEEE Symposium on Security and Privacy*, May 2000.