Signature Bouquets: Immutability for Aggregated/Condensed Signatures

Einar Mykletun, Maithili Narasimha, and 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. 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: a condensed variant of batch RSA [3] and an aggregated signature scheme based on bilinear maps [6]
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, since lack thereof enables a frequent querier to eventually amass enough aggregated signatures to answer other (un-posed) queries, thus becoming a *de facto* service provider. Since prior work does not offer immutability, we propose several practical techniques 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 only partially trusted, database service provider. It thus becomes important to secure outsourced data from potential attacks not only by malicious outsiders but also from the service provider itself.

P. Samarati et al. (Eds.): ESORICS 2004, LNCS 3193, pp. 160–176, 2004.

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. [14]. 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 [10][15] examined integrity issues in outsourced databases and suggested some limited solutions[1]. Recently, more general techniques were investigated in [19] where two signature schemes were proposed for efficient integrity and authenticity support in querying outsourced databases. One scheme is a simple variant of batch RSA and the other – the aggregated signature scheme based on bilinear maps [6]. Each scheme enables bandwidth- and computation-efficient integrity verification for any possible query reply. However, as shown below in more detail, the schemes in [19] (as well as those in [10]) 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 two contributions. First, it informally defines the notion of immutability for aggregated signatures which is, at some level, equivalent to adaptive attack resistance for aggregated signature schemes. Second, it demonstrates some simple add-on techniques for schemes considered in [19]. These techniques provide, at a little additional cost (as illustrated in Section 6), immutability for the respective 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. [6] 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 our 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 in-

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

frastructure 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 replication, backup and availability of outsourced databases. However, the server is not assumed 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 [10].
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 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.

[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.

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 are posed, as shown in Figure 1. The first (Q1) asks for names and salaries of all managers with salary $> 100K$ and the second (Q2) asks for the same information for all managers with 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
```

Fig. 1. SQL Queries

A querier who previously posed queries Q1 and Q2 and obtained corresponding replies along with their aggregated signatures S1 and S2, can compute, on her own, a valid new signature for the un-posed query Q3, as shown in Figure 1. In essence, the reply to Q3 is $(Q1 - Q2)$, i.e., information about all managers who earn between 100K and 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. If, instead of aggregated signatures, plain record-level signatures were used (e.g., DSA or RSA), a single SELECT-style query would cause the server to compose a query reply containing a set of records matching the query predicate, each accompanied by its signature. The querier can then easily construct legitimate and authentic query replies for un-posed queries, since she is free to manipulate individual record-level signatures. Furthermore, other methods, such as the constructs based on Merkle Hash Trees (MHTs) suggested by Devanbu, et al. [10], are equally susceptible to mutability of authentic query replies. (In [10], a querier obtains 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.) We now consider a few concrete scenarios where mutability is undesirable.

Paid Database Services: Mutability is undesirable when the data owner wants to offer *paid database services* in association with the server. (This clearly applies only to the Multi-Querier and Multi-Owner ODB scenarios.) In essence, a server can be viewed as an *authorized re-distribution agent* for the information contained in, or derived from, the outsourced database. 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 owner and/or server 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 from prior query reply sets and re-selling information that has not been paid for.

To make the example more specific, consider an on-line authorized music distributor with a large database of songs, each digitally signed by the artist. Suppose that the distributor only wishes to sell complete albums (compilations) and not individual songs. The distributor (server) can then simply aggregate the signatures of individual tracks to provide its clients a unified proof of authenticity and integrity for the entire album. In this case, signature aggregation gives the distributor the means to *mix and match* the songs to make various compilations.

One concern that would arise in this scenario (due to the mutability of the underlying aggregated signature scheme) is that clients could potentially start their own music distribution services with the goal of reselling individual songs at higher cost per song than that of the original distributor (who only sells complete albums).

Content Access Control: Consider the ODB scenario where the owner wants the server to enforce a certain content access control mechanism: for each client (or a group of clients) access is restricted to a specific subset of the database. A client who poses a query only gets back the data that she is entitled to see based on her specific *privileges*. If the database is a collection of individually signed records, the server can aggregate individual record signatures to construct a single proof of authenticity and integrity for the entire query reply.

Two colluding clients (each with different access control privileges) can share their respective query replies. If the aggregated signature scheme is mutable, the two clients, by further aggregating the two query replies into a single quantity, can convince others that they have more privileges than they really have. In other words, a client can *combine* two aggregated signatures to produce a new and authentic aggregated signature that can act as proof that she has higher access privileges. This can have undesirable implications.

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.

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[3]. A valid condensed signature assures its 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 $sk = (n, d)$, 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 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 strong hash function (such as, 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:

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

Security of Condensed-RSA: [19] describes the security of Condensed-RSA by demonstrating that it is at least as secure as Batch verification of RSA [3]. Batch verification of RSA signatures was shown to be secure (in [3]) under the assumption that RSA is a collection of one-way functions. The proof assumes

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

that the individual RSA signatures are generated using a full-domain hash function (FDH) in place of a standard hash function (such as SHA-1), as described in [5]. An FDH is a hash function which takes arbitrary length input and produces an output that is an element of Z_n^*, i.e., $H_{FDH} : \{0,1\}^* \rightarrow Z_n^*$

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).

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

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 $\{m_1, ..., m_i, m_{i+1}, ..., m_j\}$ (assuming all messages are distinct) by multiplying $\sigma_{1,i}$ and $\sigma_{i+1,j}$:

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

4.2 BGLS

Boneh, et al. in [6] construct an interesting signature scheme that allows incremental aggregation of signatures generated by multiple signers on different messages into one short signature based on elliptic curves and bilinear maps. 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 [17]. Before describing BGLS, we briefly overview the necessary parameters:

- G_1 is a cyclic additive group with generator g_1
- G_2 is a cyclic multiplicative group
- e is a computable bilinear map $e : G_1 \times G_1 \rightarrow G_2$ as described below

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

1. Bilinearity: $\forall P, Q \in G_1$ and $a, b \in \mathbb{Z}$, $e(aP, bQ) = e(P, Q)^{ab}$
2. Non-degenerativity: $e(g_1, g_1) \neq 1$

These two properties imply that, for any $P_1, P_2, Q \in G_1, e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$; and, for any $P, Q \in G_1, e(\psi(P), Q) = e(\psi(Q), P)$.

BGLS Signature Scheme: BGLS requires the use of a full-domain hash function $h() : \{0,1\}^* \rightarrow G_1$ that maps binary strings to non-zero points in G_1. Key generation involves picking a random $x \in \mathbb{Z}_p$, and computing $v = xg_1$. The public key is $v \in G_1$ 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 $e(\sigma, g_1) = e(H, v)$.

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:

$$e(\sigma_{1,t}, g_1) = \prod_{i=1}^{t} e(H_i, v_i)$$

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

$$e(\sigma_{1,t}, g_1) = e(\sum_{i=1}^{t} x_i H_i, g_1) = \prod_{i=1}^{t} e(H_i, g_1)^{x_i} = \prod_{i=1}^{t} e(H_i, x_i g_1) = \prod_{i=1}^{t} e(H_i, v_i)$$

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".

Interactive Variant. This technique uses the well-known Guillou-Quisquater (GQ) identification scheme [13] 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, 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 Figure 2. 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[4]. 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 2). 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[5] 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 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 $z \neq 0$ and $z^e \equiv YM^v \pmod n$ where M is the product of (hashes of) all messages in the result set. Checking $z \neq 0$ precludes a malicious server from succeeding by choosing $r = 0$. 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.

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 [13]. Subsequently, it is also proven secure against impersonation under active attacks in [4].
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 (and, therefore, e since $v \in_R \{0,1\}^{l(k)}$ where $l(k)$ is the bit-length of e)

[4] The original GQ scheme proposed in [13] 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.

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

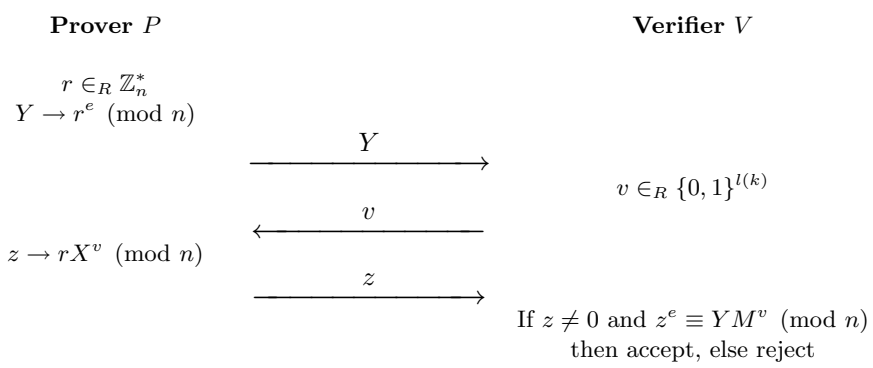

Fig. 2. IC-RSA GQ-based Interactive Technique

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.)

Non-interactive Immutable Condensed-RSA. The second, non-interactive, variant uses the technique of *Signatures of Knowledge* first popularized by Camenisch and Stadler in [8]. 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 [8] 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 [8]:

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 $(\alpha_1, \ldots, \alpha_u)$ is known. Prover first chooses $r_i \in_R \mathbb{Z}_n$ for $i = 1, \ldots, u$, computes c as

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

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

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 $\mathcal{M} = \{m_1, \ldots, m_t\}$ 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[6] the SKREP proof computed as above.

[6] 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

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 $y_1 \ldots y_{e-1}$ that are required for the SKREP proofs leak additional information about the secret. Hence a randomized version that is proposed in [8] needs to be used. The interactive protocol corresponding to the above definition of SKROOTLOG is proven honest-verifier zero-knowledge in [7]. For brevity, we skip the details of this discussion and refer interested readers to [8]. 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 [7]. 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.

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).
- **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 transferable). 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 (iBGLS)

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

$\mathcal{H} = h(m_1||m_2...||m_k||$ other information[7]) 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 iBGLS signature on the entire result set. Due to this simple extension, it is no longer feasible for anybody to manipulate the existing iBGLS signatures to obtain new and authentic ones or get any information about individual component BGLS signatures. Verification of an iBGLS 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: $e(\sigma, g_1) = \prod_{i=1}^{t} e(H_i, v_i).e(\mathcal{H}, v_s)$ 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:

$$\begin{aligned} e(\sigma, g_1) &= e(\textstyle\sum_{i=1}^{t} x_i H_i + x_s\mathcal{H}, g_1) \\ &= e(\textstyle\sum_{i=1}^{t} x_i H_i, g_1).e(x_s\mathcal{H}, g_1) \\ &= \textstyle\prod_{i=1}^{t} e(H_i, g_1)^{x_i}.e(\mathcal{H}, g_1)^{x_s} \\ &= \textstyle\prod_{i=1}^{t} e(H_i, x_i g_1).e(\mathcal{H}, x_s g_1) \\ &= \textstyle\prod_{i=1}^{t} e(H_i, v_i).e(\mathcal{H}, v_s) \end{aligned}$$

Security Considerations: iBGLS 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 iBGLS signatures.

Let r_i denote database record i. An iBGLS 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

[7] BGLS aggregation is secure only when the messages signed are all distinct. Therefore, if the server signed only the result set, it can potentially create a problem if the result set to a particular query contained a single record (message). Note that if the result set contained only one message then the owner's message as well as the server's message would be the same since server computes its signature on the entire result set. In order to avoid such situations, we require that the server add some additional information to the message that he signs. For example, the server may include query and/or querier specific information, timestamp etc.

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.

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 iBGLS signature that verifies. In other words, the iBGLS 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; $MTP(t) \leftarrow t$ Map-To-Point operations which are $h() : \{0,1\}^* \rightarrow G_1$ that map binary strings to non-zero points in G_1; $SPM(t) \leftarrow t$ Scalar-Point-Multiplications.

Table 1. Cost comparison of techniques for Immutability

	Computation		Communication
	at Client	at Server	
GQ	$Mult^1(n) + Exp_e^2(n)$	$Mult^1(n) + Exp_e^2(n)$	2
SKROOTLOG	$Exp_n^4(p') + Mult^3(p')$	$Exp_n^4(p') + Exp_2^1(n)$ $+Mult^1(n)$	0
iBGLS	$BM(1)$	$MTP(1) + SPM(1)$	0

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-3 800 MHz processor with 1GB memory. The results for BGLS are obtained by using the

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

		Condensed-RSA		BGLS
		$e = 3$	$e = 2^{30} + 1$	
Sign	1 signature	6.82	6.82	12.0
Verify	1 signature	0.14	0.56	77.4
	t = 1000 sigs, k = 1 signer	44.12	45.531	12085.4
	t = 100 sigs, k = 10 signers	45.16	50.31	12320.2

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 (3) gives the time required to generate and verify an immutable signature under the two different RSA-based techniques as well as the BGLS extension. In this table, we only measure the **overhead** associated with the immutability extensions. Therefore, the costs do not include the original condensed-RSA or BGLS costs. In addition, we also do not count the communication delays introduced by the protocols, particularly in the case of GQ which is multi-round protocol.

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

Technique Used	Computation at Client	Computation at Server	Total Overhead
GQ	0.309	0.309	0.618
SKROOTLOG	48.88	46.489	89.369
iBGLS	37.2	12.0	49.2

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).

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 [16]. In our work, we used a very similar system model.

Private Information Retrieval (PIR) [9, 11] deals with the exact matching problem and has been explored extensively in the cryptographic literature. However, most of current PIR techniques aim for very strong security bounds and,

consequently, remain unsuitable for practical purposes. Song et al. [21] develop a more pragmatic scheme to search on data encrypted using a secret symmetric key. In summary, searching on encrypted data is becoming an increasingly popular research topic with such recent interesting results as [21, 12]. 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 [14] explore how different types of SQL queries can be executed over encrypted data Specifically, they support *range* searches and *joins* in addition to exact-match queries.

On a more related topic, [15] 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. The work of [10] explores the applicability of Merkle Hash Tree-s (MHT-s) as a technique for providing authenticity and integrity in third-party data publication settings. The use of authenticated data structures for providing data integrity in general has been studied extensively in [18].

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]. We constructed add-on techniques to provide immutability for these schemes. We also performed a detailed comparison and performance analysis of the proposed techniques.

Acknowledgments

The authors would like to thank Claude Castelluccia, Stanislaw Jarecki and Moti Yung for helpful comments and discussions. We thank Nitesh Saxena and Jeong Hyun Yi for providing timing measurements of BGLS-related crypto operations. Finally, the authors would like to thank the anonymous reviewers for their insightful comments.

References

1. MIRACL Library. http://indigo.ie/˜mscott
2. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. Advances in Cryptology - Crypto (1987) 186–194
3. Bellare, M., Garay, J., Rabin, T.: Fast Batch Verification for Modular Exponentiation and Digital Signatures. Eurocrypt, volume 1403 (1998) pages 191–2048
4. Bellare, M., Palacio, A.: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. Advances in Cryptology - Crypto (1992) 162–177

5. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. ACM Press (1993) 62–73
6. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. Eurocrypt (1993)
7. Camenisch, J.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. Vol. 2 of ETH-Series in Information Security an Cryptography, ISBN 3-89649-286-1, Hartung-Gorre Verlag, Konstanz (1998)
8. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups. Advances in Cryptology - Crypto (1997) 410–424
9. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. Journal of ACM (1998) 965–981
10. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.: Authentic third-party data publication. 14th IFIP Working Conference in Database Security (2000) 101–112
11. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting Data Privacy in Private Information Retrieval Schemes 30th Annual Symposium on Theory of Computing (STOC) ACM Press (1998)
12. Goh, E.: Secure Indexes for Efficient Searching on Encrypted Compressed Data. Cryptology ePrint Archive, Report 2003/216 (2003)
13. Guillou L., Quisquater, J.: A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge. Advances in Cryptology - Crypto (1998) 216–231
14. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. ACM SIGMOD Conference on Management of Data (2002) 216–227
15. Hacigümüş, H., Iyer, B., Mehrotra, S.: Encrypted Database Integrity in Database Service Provider Model. International Workshop on Certification and Security in E-Services (2002)
16. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Providing Database as a Service. International Conference on Data Engineering (2002)
17. Joux, A., Nguyen, K.: Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003 (2001)
18. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.: A General Model for authenticated data structures. Algorithmica, Volume 39 (2004)
19. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and Integrity in Outsourced Databases. ISOC Symposium on Network and Distributed Systems Security (2004) 205–214
20. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM (1978) 120–126
21. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. IEEE Symposium on Security and Privacy (2000) 44–55