Verifiable Computation over Large Database with Incremental Updates

Xiaofeng Chen[1,4], Jin Li[2,4], Jian Weng[3], Jianfeng Ma[1], and Wenjing Lou[4]

[1] State Key Laboratory of Integrated Service Networks (ISN),
Xidian University, Xi'an 710071, P.R. China
xfchen@xidian.edu.cn, jfma@mail.xidian.edu.cn

[2] School of Computer Science and Educational Software,
Guangzhou University, Guangzhou 510006, P.R. China
jinli71@gmail.com

[3] Department of Computer Science,
Jinan University, Guangzhou 510632, P.R. China
cryptjweng@gmail.com

[4] Department of Computer Science,
Virginia Polytechnic Institute and State University, USA
wjlou@vt.edu

Abstract. The notion of verifiable database (VDB) enables a resource-constrained client to securely outsource a very large database to an untrusted server so that it could later retrieve a database record and update a record by assigning a new value. Also, any attempt by the server to tamper with the data will be detected by the client. When the database undergoes frequent while small modifications, the client must re-compute and update the encrypted version (ciphertext) on the server at all times. For very large data, it is extremely expensive for the resources-constrained client to perform both operations from scratch. In this paper, we formalize the notion of verifiable database with incremental updates (Inc-VDB). Besides, we propose a general Inc-VDB framework by incorporating the primitive of vector commitment and the encrypt-then-incremental MAC mode of encryption. We also present a concrete Inc-VDB scheme based on the computational Diffie-Hellman (CDH) assumption. Furthermore, we prove that our construction can achieve the desired security properties.

Keywords: Verifiable Database, Incremental Cryptography, Outsourcing Computations, Vector Commitment.

1 Introduction

With the availability of cloud services, the techniques for securely outsourcing the prohibitively expensive computations are getting widespread attentions in the scientific community [1–3, 18, 19]. That is, the clients with resource-constraint devices can outsource the heavy computation workloads into the untrusted cloud servers and enjoy the unlimited computing resources in a pay-per-use manner.

Since the cloud servers may return an invalid result in some cases (e.g., the servers might contain a software bug that will fail on a constant number of invocation), one crucial requirement of outsourcing computation is that the client has the ability to verify the validity of computation result efficiently.

The primitive of verifiable computation has been well studied by plenty of researchers in the past decades [4, 8, 9, 20, 21, 23–25]. Most of the prior work focused on generic solutions for arbitrary function (encoded as a Boolean circuit). Though in general the problem of verifiable computation has been theoretically solved, the proposed solutions are still much inefficient for real-world applications. Therefore, it is still meaningful to seek for efficient protocols for verifiable computation of specific functions.

Benabbas, Gennaro and Vahlis [12] first proposed the notion of verifiable database (VDB), which is extremely useful to solve the problem in the context of verifiable outsourcing storage. Assume that a resource constrained client would like to store a very large database on a server so that it could later retrieve a database record and update a record by assigning a new value. If the server attempts to tamper with the database, it will be detected by the client with an overwhelming probability. Besides, the computation and storage resources invested by the client must not depend on the size of the database (except for an initial setup phase).

For the case of static database, we can construct VDB based on simple solutions using message authentication codes or digital signatures. That is, the client signs each database record before sending it to the server, and the server is requested to output the record together with its valid signature. The solution does not work if the client performs updates on the database. As noted in [12], the main technical difficulty is that the client must have a mechanism to revoke the signatures given to the server for the previous values. Otherwise, the malicious server can utilize the previous (while valid) database records and corresponding signatures to responde the current query of the client. This is called the Backward Substitution updates (BSU) attack on VDB. In order to solve this issue, the client should keep track of every change locally. However, this totally contradicts the goal of outsourcing, i.e., the client should use much less resources than those needed to store the database locally.

This problem has been addressed by works on accumulators [15, 16, 29] and authentication data structures [27, 28, 30, 31]. However, it seems that the previous solutions based on the two techniques either rely on non-constant size assumptions (such as q-Strong Diffie-Hellman assumption), or require expensive operations such as generation of primes and expensive "re-shuffling" procedures. Benabbas, Gennaro and Vahlis [12] presented the first practical verifiable computation scheme for high degree polynomial functions and used it to design an efficient VDB scheme. The construction relies on a constant size assumption in bilinear groups of composite order, while does not support public verifiability (i.e., only the owner of the database can verify the correctness of the proofs). Very recently, Catalano and Fiore [13] proposed an elegant solution to build

VDB from a primitive named vector commitment. The concrete construction relies on standard constant-size assumption and supports public verifiability.

The data records often contain some sensitive information that should not be exposed to the cloud server. Therefore, the client should encrypt the database and store the encrypted version on the server. In some scenarios, the data (plaintext) of client undergoes frequent while small modifications and the client must *re-compute* and *update* the encrypted version (ciphertext) on the server at all times [5, 6]. For very large data, it is extremely expensive for the resources-constrained client to re-compute and update the ciphertext from scratch each time. Therefore, it is meaningful to seek for efficient constructions for VDB with incremental updates (Inc-VDB, for short). Loosely speaking, Inc-VDB means that re-computing and updating the ciphertext in VDB are both incremental algorithms, i.e., the client can efficiently perform both operations with previous values, rather than from scratch.

Bellare, Goldreich, and Goldwasser [5, 6] introduced the notion of incremental cryptography to design cryptographic algorithms whose output can be updated very efficiently when the underlying input changes. For example, if a single block of the data is modified (we can view the data as a sequence of blocks), the client only needs to re-compute the ciphertext on this certain block and the ciphtertext of other blocks remains identical [7, 26]. Nevertheless, we argue that the incremental encryption does not provide a full solution for constructing efficient Inc-VDB schemes. The reasons are two folds: Firstly, previous incremental encryption schemes cannot solve the case of distributed updates on the data. That is, multiple blocks of the plaintext are modified while the modification on each single block is very small. The worst case is that every block of the plaintext is updated while only one bit for each single block is changed. If this case happens, the client must re-compute the whole ciphertext from scratch. Secondly, previous incremental encryption schemes cannot necessarily lead to incremental updates on VDB. That is, the update algorithm of VDB is not incremental and the client still needs to re-compute new updated token from scratch each time. To the best of our knowledge, it seems that there is no research work on constructing efficient Inc-VDB schemes.

1.1 Our Contribution

In this paper, we further study the problem of constructing verifiable database with efficient updates. Our contributions are three folds:

- We first introduce the notion of verifiable database with incremental updates (Inc-VDB). The update algorithm in Inc-VDB is an incremental one, i.e., the client can efficiently compute the new ciphertext and the updated tokens with previous values, rather than from scratch. Thus, Inc-VDB schemes can lead to huge efficiency gain when the database undergoes frequent while small modifications.
- We propose a general Inc-VDB framework by incorporating the primitive of vector commitment [13] and the encrypt-then-incremental MAC mode

of encryption [7]. We also present a concrete Inc-VDB scheme based on the computational Diffie-Hellman (CDH) assumption. Besides, the proposed Inc-VDB scheme supports the public verifiability.

– We first introduce a new property called accountability for VDB schemes. That is, after the client detected the tampering of the server, the client should be able to provide a proof to convince the judge of the facts. All of the existing VDB schemes does not satisfy the property of accountability. We prove that the proposed Inc-VDB scheme satisfies the property of accountability.

1.2 Organization

This paper is organized as follows. Some preliminaries are presented in Section 2. We present the formal definition and security requirements of Inc-VDB in Section 3. We propose a new efficient Inc-VDB framework and a concrete Inc-VDB scheme in Section 4. The security and efficiency analysis of the proposed Inc-VDB scheme are given in Section 5. Finally, concluding remarks will be made in Section 6.

2 Preliminaries

In this section, we first introduce the basic definition and properties of bilinear pairings. We then present the formal definition of VDB.

2.1 Bilinear Pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order p. Let g be a generator of $\mathbb{G}_1$. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties:

1. Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}_1$, and $a, b \in \mathbb{Z}_p^*$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}_1$.

The examples of such groups can be found in supersingular elliptic curves or hyperelliptic curves over finite fields, and the bilinear pairings can be derived from the Weil or Tate pairings. In the following, we introduce the Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$.

Definition 1. *The Computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is defined as follows: given a triple (g, g^x, g^y) for any $x, y \in_R \mathbb{Z}_p$ as inputs, output g^{xy}. We say that the CDH assumption holds in $\mathbb{G}_1$ if for every probabilistic polynomial time algorithm $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that* $\mathsf{Pr}[\mathcal{A}(1^k, g, g^x, g^y) = g^{xy}] \leq \mathsf{negl}(k)$ *for all security parameter k.*

A variant of CDH problem is the Square Computational Diffie-Hellman (Squ-CDH) problem. That is, given (g, g^x) for $x \in_R \mathbb{Z}_p$ as inputs, output g^{x^2}. It has been proved that the Squ-CDH assumption is equivalent to the classical CDH assumption.

2.2 Verifiable Database

Informally, a VDB scheme allows a resource-constraint client to outsource the storage of a very large database to a server in such a way that the client can later retrieve and update the data records from the server. Besides, any attempts to tamper with the data by the dishonest server will be detected when the client queries the database. The formal definition for VDB is given as follows [12, 13]:

Definition 2. *A verifiable database scheme* $\mathsf{VDB}=(\mathsf{Setup}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ *consists of four algorithms defined below.*

- $\mathsf{Setup}(1^k, DB)$*: On input the security parameter* k*, the setup algorithm is run by the client to generate a secret key* SK *to be secretly stored by the client, and a public key* PK *that is distributed to all users (including the client itself) for verifying the proofs.*
- $\mathsf{Query}(\mathsf{PK}, x)$*: On input an index* x*, the query algorithm is run by the server, and returns a pair* $\tau = (v, \pi)$*.*
- $\mathsf{Verify}(\mathsf{PK}/\mathsf{SK}, x, \tau)$*: The public/private verification algorithm outputs a value* v *if* τ *is correct with respect to* x*, and an error* $\perp$ *otherwise.*
- $\mathsf{Update}(\mathsf{SK}, x, v')$*: In the update algorithm, the client firstly generates a token* t'_x *with the secret key* SK *and then sends the pair* (t'_x, v') *to the server. Then, the server uses* v' *to update the database record in index* x*, and* t'_x *to update the public key* PK*.*

Remark 1. There are two different kinds of verifiability for the outputs of the query algorithm, i.e., $\tau = (v, \pi)$. In the Catalano-Fiore's scheme [13], anyone can verify the validity of τ with the public key PK. Therefore, it satisfies the property of public verifiability. However, in some applications, only the client can verify the proofs generated by the server since the secret key of the client is involved in the verification. This is called the private verifiability [12]. Trivially, a verifiable database scheme should support both verifiability for various applications.

3 Verifiable Database with Incremental Updates

3.1 Formal Definition

Without loss of generality, we consider the database DB as a set of tuples (x, m_x) in some appropriate domain, where x is an index and m_x is the corresponding value which can be arbitrary payload sizes. In order to achieve the confidentiality of the data record m_x, the client can use an arbitrary semantically-secure encryption scheme ENC (the key is implicit in the notaton) to encrypt each m_x. Trivially, given the ciphertext $v_x = \mathsf{ENC}(m_x)$, only the client can compute the record m_x. Therefore, we only consider the case of encrypted database (x, v_x). This is also implicitly assumed in the existing academic research.

Informally, verifiable database with incremental updates (Inc-VDB) can be viewed a special case of VDB in which the updated record m'_x is only slightly different from the previous one m_x (note that the corresponding ciphertexts v'_x and v_x may be totally different). The distinct feature of Inc-VDB is that the update algorithm is an incremental one. That is, the client can efficiently compute a new token t'_x from the previous one, rather than re-computing it from scratch (similarly, the server can efficiently update the public key rather than re-computing it from scratch). Trivially, Inc-VDB can lead to huge efficiency gains, especially in the scenario when the database is subject to frequent, small modification. In the following, we present a formal definition for Inc-VDB.

Definition 3. *A verifiable database scheme with incremental updates* $\mathsf{Inc\text{-}VDB} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Inc\text{-}Update})$ *consists of four algorithms defined below.*

- $\mathsf{Setup}(1^k, DB)$*: On input the security parameter k, the setup algorithm is run by the client to generate a secret key* SK *to be secretly stored by the client, and a public key* PK *that is distributed to all users (including the client itself) for verifying the proofs.*
- $\mathsf{Query}(\mathsf{PK}, x)$*: On input an index x, the query algorithm is run by the server, and returns a pair* $\tau = (v, \pi)$*.*
- $\mathsf{Verify}(\mathsf{PK}/\mathsf{SK}, x, \tau)$*: The public/private verification algorithm outputs a value v if τ is correct with respect to x, and an error ⊥ otherwise.*
- $\mathsf{Inc\text{-}Update}(\mathsf{SK}, x, v')$*: In the update algorithm, the client utilizes the secret key* SK *to compute a new token* t'_x *from the previous one in an incremental manner rather than computing it from scratch. Then, the client sends the pair* (t'_x, v') *to the server. If the token* t'_x *is valid, the server uses* v' *to update the database record in index x, and* t'_x *to incrementally update the public key* PK*.*

3.2 Security Requirements

In the following, we introduce some security requirements for Inc-VDB. Obviously, Inc-VDB should inherently satisfy three security properties of VDB [12], i.e., security, correctness, and efficiency. Besides, we also introduce a new property named accountability for Inc-VDB.

The first requirement is the **security** of Inc-VDB scheme. Intuitively, an Inc-VDB scheme is secure if a malicious server cannot convince a verifier to accept an invalid output, i.e., $v \neq v_x$ where v_x is the value of database record in the index x. Note that v_x can be either the initial value given by the client in the setup stage or the latest value assigned by the client in the update procedure.

Definition 4. ***(Security)*** *An* $\mathsf{Inc\text{-}VDB}$ *scheme is secure if for any database* $DB \in [q] \times \{0,1\}^*$*, where* $q = poly(k)$*, and for any probabilistic polynomial time (PPT) adversary A, we have*

$$Adv_A(\mathsf{Inc\text{-}VDB}, DB, k) \leq negl(k),$$

where $Adv_A(\mathsf{Inc\text{-}VDB}, DB, k) = Pr[\boldsymbol{Exp}_A^{\mathsf{Inc\text{-}VDB}}(DB, k) = 1]$ *is defined as the advantage of A in the experiment as follows:*

```
Experiment $\boldsymbol{Exp}_A^{\mathsf{Inc\text{-}VDB}}[DB, k]$
    $(PK, SK) \leftarrow \mathsf{Setup}(DB, k)$;
    For $i = 1, \ldots, l = poly(k)$;
        $\boldsymbol{Verify}$ query :
            $(x_i, \tau_i) \leftarrow A(PK, t'_1, \ldots, t'_{i-1})$;
            $v_i \leftarrow \mathsf{Verify}(PK/SK, x_i, \tau_i)$;
        $\boldsymbol{Inc\text{-}Update}$ query :
            $(x_i, v_{x_i}^{(i)}) \leftarrow A(PK, t'_1, \ldots, t'_{i-1})$;
            $t'_i \leftarrow \mathit{Inc\text{-}Update}(SK, x_i, v_{x_i}^{(i)})$;
    $(\hat{x}, \hat{\tau}) \leftarrow A(PK, t'_1, \ldots, t'_l)$;
    $\hat{v} \leftarrow \mathsf{Verify}(PK/SK, \hat{x}, \hat{\tau})$
    If $\hat{v} \neq \perp$ and $\hat{v} \neq v_{\hat{x}}^{(l)}$, output 1; else output 0.
```

In the above experiment, we implicitly assign $\mathsf{PK} \leftarrow \mathsf{PK}_i$ after every update query.

The second requirement is the **correctness** of Inc-VDB scheme. That is, the value and proof generated by the honest server can be always verified successfully and accepted by the client.

Definition 5. *(Correctness) An Inc-VDB scheme is correct if for any database $DB \in [q] \times \{0,1\}^*$, where $q = poly(k)$, and for any valid pair $\tau = (v, \pi)$ generated by an honest server, the output of verification algorithm is always the value v.*

The third requirement is the **efficiency** of Inc-VDB scheme. That is, the client in the verifiable database scheme should not be involved in plenty of expensive computation and storage (except for an initial pre-processing phase)[1].

Definition 6. *(Efficiency) An Inc-VDB scheme is efficient if for any database $DB \in [q] \times \{0,1\}^*$, where $q = poly(k)$, the computation and storage resources invested by the client must be independent of the size of the database DB. Besides, the cryptographic operations performed by the client should be incremental.*

Finally, we introduce a new requirement named **accountability** for Inc-VDB scheme. That is, after the client has detected the tampering of dishonest server, he should provide some evidence to convince a judge of the facts.

Definition 7. *(Accountability) An Inc-VDB scheme is account if for any database $DB \in [q] \times \{0,1\}^*$, where $q = poly(k)$, the client can provide a proof for this misbehavior if the dishonest server has tampered with the database.*

[1] In some scenarios, the client is allowed to invest a one-time expensive computational effort. This is known as the amortized model of outsourcing computations [22].

4 Inc-VDB Framework from Vector Commitment

In this section, we present an efficient Inc-VDB framework from vector commitment and the incremental encrypt-then-MAC mode of encryption. Besides, we propose a concrete Inc-VDB scheme based on the CDH assumption.

4.1 High Description

Catalano and Fiore presented an elegant construction for building a general VDB framework from vector commitment [13]. The main idea is as follows: Let C be the vector commitment on the database. Given a query on index x by the client, the server provide the value v_x and the opening of commitment as a proof that v_x has not been tampered with. During the update phase, the client computes a new ciphertext v'_x and a token t'_x and then sends them to the server. Finally, the server updates the database and the corresponding public key with the pair (t'_x, v'_x). We also use the vector commitment to construct incremental VDB schemes. However, the main difference is that the client in our construction does not compute the updated ciphertext v'_x and the corresponding (updated) commitment C' in the token t'_x. The main trick is that we use a special incremental encryption to generate the ciphertext v'_x. More precisely, we define $v'_x = (v_x, P_x)$, where $P_x = (p_1, p_2, \cdots, p_\omega)$ denotes the bit positions where m'_x and m_x have different values, i.e, $m'_x[p_i] \neq m_x[p_i]$ for $1 \leq i \leq \omega$. Trivially, given $v'_x = (v_x, P_x)$, the client firstly decrypts v_x to obtain m_x, and then perform the bit flipping operation on the positions of P_x to obtain m'_x. Since the bit flipping operation is extremely fast, the computation overhead of decrypting v'_x is almost the same as that of decrypting v_x. Moreover, it requires much less storage since $|P_x| << |v'_x|$ (note that we only consider the case of incremental updates). Besides, we argue that the incremental encryption scheme (ENC, P) is more suitable for discrete and uniform update on the data record (note that previous incremental encryption schemes mainly focus on local updates, e.g., updates on a single block of the data).

Note that the secret key of the client should be involved in the update algorithm. That is, only the client is allowed to update the database. In order to achieve this goal, we utilize the encrypt-then-incremental MAC mode of encryption [7], i.e., an incremental encryption together with an incremental MAC of the ciphertext (the encrypt-then-MAC approach [11]). Trivially, we could use an incremental signature scheme to substitute the incremental MAC. In our concrete construction, we adopt the (incremental) BLS signature scheme [10]. For every update, the client first verify the current BLS signature on the commitment C_R and all the current modifications $(P_x^{(1)}, \cdots, P_x^{(T)})$ of the data record v_x, where $P_x^{(i)}$ denotes the modification in the i-th update for $1 \leq i \leq T$. This ensures that the current database is not tampered with by the server. If the verification holds, the client then sends a new modification $P_x^{(T+1)}$ and the corresponding (incremental) BLS signature to the server.

Since we also use the signature to achieve the integrity of the database, it is essential to invoke the previous signatures given to the server. Our trick is that we introduce a counter T_x to denote the update times of each index x. Also, the server computes a BLS signature σ on all counters T_x for $1 \le x \le q$. After an update on the record v_x is accomplished, let $T_x \leftarrow T_x + 1$. Then, the server computes an incremental BLS signature on the updated counters (note that only the value of T_x is slightly modified). Given a previous signature σ on the count T_x, the client can reject it by providing a new signature σ' on the latest counter T'_x since $T_x < T'_x$. Note that the server cannot deny his signature, therefore this is a proof that the server is dishonest when a dispute occurred.

4.2 A Concrete Inc-VDB Scheme

In this section, we propose a concrete Inc-VDB scheme based on the CDH assumption.

– $\mathsf{Setup}(1^k, DB)$: Let k be a security parameter. Let the database be $DB = (x, v_x)$ for $1 \le x \le q$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order p equipped with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let g be a generator of $\mathbb{G}_1$. Let $\mathcal{H} : \mathbb{G}_1 \times \{0,1\}^* \to \mathbb{G}_1$ be a cryptographic hash function. Randomly choose q elements $z_i \in_R \mathbb{Z}_p$ and compute $h_i = g^{z_i}$, $h_{i,j} = g^{z_i z_j}$, where $1 \le i, j \le q$ and $i \ne j$. Set $\mathsf{PP} = (p, q, \mathbb{G}_1, \mathbb{G}_2, \mathcal{H}, e, g, \{h_i\}_{1 \le i \le q}, \{h_{i,j}\}_{1 \le i,j \le q, i \ne j})$, and the message space $\mathcal{M} = \mathbb{Z}_p$.
Let $(\alpha, Y = g^\alpha)$ be the secret/public key pair of the client. Let $(\beta, S = g^\beta)$ be the secret/public key pair of the server. Trivially, the validity of Y and S are ensured by the corresponding certificate of a trusted third party, i.e, certificate authority. Let $C_R = \prod_{i=1}^{q} h_i^{v_i}$ be the root commitment on the database record vector $(v_1, v_2, \cdots, v_q)$. For $1 \le x \le q$, let T_x be a counter for index x with the initial value 0 and $H_x^{(0)} = \mathcal{H}(C_R, x, 0)^\alpha$. The server can use the batch verification technique of BLS signatures [14] to ensure the validity of $H_x^{(0)}$ for $1 \le x \le q$, which requires only the workload of two pairings. Then, the server computes a signature $\sigma = \mathcal{H}(C_R, 0, 0, \cdots, 0)^\beta$ on C_R and all initial counters $(0, 0, \cdots, 0)$ (note that all T_x has an initial value 0). Also, set $\mathsf{aux} = \{\mathsf{aux}_1, \cdots, \mathsf{aux}_q\}$, where $\mathsf{aux}_x = (H_x^{(0)}, 0)$ for $1 \le x \le q$.
Define $\mathsf{PK} = (\mathsf{PP}, C_R, \mathsf{aux}, DB)$ and $\mathsf{SK} = \alpha$.

– $\mathsf{Query}(\mathsf{PK}, x)$: Assume that the current public key $\mathsf{PK} = (\mathsf{PP}, C_R, \mathsf{aux}, DB)$. Given a query index x, the server computes $\pi_x = \prod_{1 \le j \le q, j \ne x} h_{x,j}^{v_j}$ and returns the proofs

$$\tau = (v_x, \pi_x, H_x^{(T_x)}, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x).$$

– $\mathsf{Verify}(\mathsf{PK}, x, \tau)$: Parse the proofs $\tau = (v_x, \pi_x, H_x^{(T_x)}, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x)$. If the counter T_x in τ is less than the one in σ that the client stored locally, the client rejects the proofs τ. Otherwise, the client can verify the validity of τ by checking whether the following two equations $e(C_R / h_x^{v_x}, h_x) = e(\pi_x, g)$ and $e(H_x^{(T_x)}, g) = e(\mathcal{H}(C_R, x, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x), Y)$ hold. If the proofs τ

is valid, the verifier accepts it and outputs $v_x^{(T_x)} = (v_x, P_x^{(1)}, \cdots, P_x^{(T_x)})$. Otherwise, outputs an error $\perp$.

– $\mathsf{Inc\text{-}Update}(\mathsf{SK}, x, P_x^{(T_x+1)})$: To update the record of index x, the client firstly retrieves the current record $v_x^{(T_x)}$ from the server. That is, the client obtains $\tau \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x^{(T_x)} \neq \perp$. Then, the client computes the incremental signature

$$t'_x = H_x^{(T_x+1)} = \mathcal{H}(C_R, x, P_x^{(1)}, \cdots, P_x^{(T_x+1)}, T_x + 1)^{\alpha}$$

and then sends $(t'_x, P_x^{(T_x+1)})$ to the server. If t'_x is valid, then the server adds $P_x^{(T_x+1)}$ to the record of index x, and updates aux_x in PK, i.e., $\mathsf{aux}_x \leftarrow (t'_x, P_x^{(1)}, \cdots, P_x^{(T_x+1)}, T_x + 1)$. Also, the server computes an updated incremental signature $\sigma = \mathcal{H}(C_R, T_1, T_2, \cdots, T_x + 1, \cdots, T_q)^{\beta}$ and sends it to the client. If σ is valid, the client updates it together with $T_x + 1$ locally. Finally, set $T_x \leftarrow T_x + 1$.

Remark 2. As pointed out in [6], incremental encryption leaks some information that is kept secret when using a traditional encryption scheme. In the resulting incremental encryption scheme (ENC, P) in our construction, an adversary can determine where a modification takes place, but still cannot determine the symbol being modified (i.e., hide details about the data record and its modifications). This is similar to previous incremental encryptions [6, 7, 26]. Actually, we can prove that the incremental encryption (ENC, P) is semantically-secure if and only if the original one ENC is semantically-secure (the formal proof will be given in the full version of this paper). On the other hand, though we only focus on the bit flipping operation in the our construction, it can be extended to other operations such as insert, delete, etc.

Remark 3. The storage overhead of client in our construction is all counters T_x and the latest BLS signature σ. Note that the number of T_x is dependent of q, it is highly undesirable when q becomes very large. Trivially, we can still use the vector commitment to solve this issue. The server computes the signature $\sigma = \mathcal{H}(C_R, C_T)^{\beta}$, where C_T is the vector commitment on all counters $(T_1, T_2, \cdots, T_q)$. Therefore, the client only requires to store σ and C_T and the storage overhead is independent of q. Trivially, the server should provide a valid opening of C_T as a proof during the verification phase. Due to the property of vector commitment, the update of C_T is still incremental.

5 Analysis of Our Proposed Inc-VDB Scheme

5.1 Security Analysis

Theorem 1. *The proposed Inc-VDB scheme is secure.*

Proof. Similar to [13], we prove the theorem by contradiction. Assume there exists a polynomial-time adversary A that has a non-negligible advantage ϵ in the experiment $\mathbf{Exp}_A^{\mathsf{Inc\text{-}VDB}}[DB, k]$ for some initial database DB, then we can use A to build an efficient algorithm B to break the Squ-CDH assumption. That is, B takes as input a tuple (g, g^a) and outputs g^{a^2}.

Without loss of generality, we assume that the secret/public key pairs of B and A are $(\alpha, Y = g^\alpha)$ and $(\beta, S = g^\beta)$, respectively. First, B randomly chooses an element $x^* \in_R \mathbb{Z}_q$ as a guess for the index x^* on which A succeeds in the experiment $\mathbf{Exp}_A^{\mathsf{Inc\text{-}VDB}}[DB, k]$. Then, B randomly chooses $z_i \in_R \mathbb{Z}_p$ and computes $h_i = g^{z_i}$ all $1 \le i \ne x^* \le q$. Let $h_{x^*} = g^a$. Besides, B computes:

$h_{i,j} = g^{z_i z_j}$ for all $1 \le i \ne j \le q$ and $i, j \ne x^*$;
$h_{i,x^*} = h_{x^*,i} = (g^a)^{z_i}$ for all $1 \le i \le q$ and $i \ne x^*$.

Set $\mathsf{PP} = (p, q, \mathbb{G}_1, \mathbb{G}_2, \mathcal{H}, e, g, \{h_i\}, \{h_{i,j}\})$, where $1 \le i \ne j \le q$. Given a database DB, B computes the commitment $C_R = \prod_{i=1}^q h_i^{v_i}$. Also, B computes $H_x^{(0)} = \mathcal{H}(C_R, x, 0)^\alpha$ for $1 \le x \le q$. Set $\mathsf{aux} = \{\mathsf{aux}_1, \cdots, \mathsf{aux}_q\}$, where $\mathsf{aux}_x = (H_x^{(0)}, 0)$ for $1 \le x \le q$.

Define $\mathsf{PK} = (\mathsf{PP}, C_R, \mathsf{aux}, DB)$ and $\mathsf{SK} = \alpha$. Note that PK is perfectly distributed as the real ones. B sends PK to A and A responds with $\sigma = \mathcal{H}(C_R, 0, 0, \cdots, 0)^\beta$.

To answer the verify and update queries of A in the experiment, B just simply runs the real $\mathsf{Query}(\mathsf{PK}, x)$ and $\mathsf{Inc\text{-}Update}(\mathsf{SK}, x, P_x^{(T_x+1)})$ algorithms and responds with the same value. Note that the $\mathsf{Inc\text{-}Update}(\mathsf{SK}, x, P_x^{(T_x+1)})$ algorithm requires the secret key α of B, and A cannot perform this algorithm without the help of B. After every update query, A responds with $\sigma = \mathcal{H}(C_R, T_1, T_2, \cdots, T_x + 1, \cdots, T_q)^\beta$.

Suppose that $(\hat{x}, \hat{\tau})$ be the tuple returned by A at the end of the experiment, where $\hat{\tau} = (\hat{v}, \hat{\pi}_{\hat{x}}, H_{\hat{x}}^{(T_{\hat{x}})})$ and $\hat{v} = (\hat{v}_{\hat{x}}, \hat{P}_{\hat{x}}^{(1)}, \cdots, \hat{P}_{\hat{x}}^{(T_{\hat{x}})}, T_{\hat{x}})$. Besides, note that if A wins with a non-negligible advantage ϵ in the experiment, then we have $\hat{v} \ne \perp$, $\hat{v} \ne v_{\hat{x}}^{(T_{\hat{x}})}$. Since $H_{\hat{x}}^{(T_{\hat{x}})}$ is a valid BLS signature generated with the secret key α of B, we have $\hat{P}_{\hat{x}}^{(i)} = P_{\hat{x}}^{(i)}$ for all $1 \le i \le T_{\hat{x}}$. Otherwise, A successfully forged a new BLS signature. Therefore, we have $\hat{v}_{\hat{x}} \ne v_{\hat{x}}$.

If $\hat{x} \ne x^*$, B aborts the simulation and fails. Otherwise, note that $h_{\hat{x}} = g^a$ and $e(C_R, h_{\hat{x}}) = e(h_{\hat{x}}^{v_{\hat{x}}}, h_{\hat{x}}) e(\pi_{\hat{x}}, g) = e(h_{\hat{x}}^{\hat{v}_{\hat{x}}}, h_{\hat{x}}) e(\hat{\pi}_{\hat{x}}, g)$, B can compute

$$g^{a^2} = (\hat{\pi}_{\hat{x}} / \pi_{\hat{x}})^{(v_{\hat{x}} - \hat{v}_{\hat{x}})^{-1}}.$$

The success probability of B is ϵ / q.

Theorem 2. *The proposed Inc-VDB scheme is correct.*

Proof. If the server is assumed to be honest, then the proofs

$$\tau = (v_x, \pi_x, H_x^{(T_x)}, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x).$$

Firstly, note that $C_R / h_x^{v_x} = \prod_{1 \le j \le q, j \ne x} h_j^{v_j}$ and $\pi_x = \prod_{1 \le j \le q, j \ne x} h_{x,j}^{v_j}$, we have $e(C_R / h_x^{v_x}, h_x) = e(\pi_x, g)$. Secondly, since $H_x^{(T_x)} = \mathcal{H}(C_R, x, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x)^\alpha$, we have $e(H_x^{(T_x)}, g) = e(\mathcal{H}(C_R, x, P_x^{(1)}, \cdots, P_x^{(T_x)}, T_x), Y)$. Hence, the output of the verification algorithm is always the value $v_x^{(T_x)}$.

Theorem 3. *The proposed Inc-VDB scheme is efficient.*

Proof. It is trivial that the computational and storage resources invested by the client in our scheme is independent of the size of the database (except for a one-time Setup phase). More precisely, in the Verify algorithm, the client requires the workload of four pairings and an exponentiation in $\mathbb{G}_1$ (note that it can be reduced to two pairings and two exponentiations in $\mathbb{G}_1$). Besides, in the Inc-Update algorithm, the client only requires the workload of computing an incremental BLS signature. On the other hand, the storage of client is only two elements in $\mathbb{G}_1$ (please refer to Remark 3 for more discussions).

Theorem 4. *The proposed Inc-VDB scheme is account.*

Proof. Given the proofs τ with the counter T_x for index x, the client firstly compare it with the latest counter T_c for same index x that he stored locally. If $T_x < T_c$, then the client sends the corresponding signature σ on T_c to the judge as a proof. Otherwise, he sends τ to the judge as a proof since the verification of τ will fail if the server has tampered with the database (i.e., either v_x or $P_x^{(i)}$ for $1 \leq i \leq T_x$).

5.2 Efficiency Analysis

In this section, we present the efficacy analysis of the proposed scheme and give a comparison with schemes [12, 13]. We compare our scheme with Benabbas-Gennaro-Vahlis's scheme and Catalano-Fiore's scheme.

Firstly, all of the three schemes require a one-time expensive computational effort in the Setup phase. Secondly, our proposed scheme simultaneously satisfies the properties of public verifiability and accountability. Besides, our scheme is efficient since the computational resources invested by the client is independent on the size of the database. Finally, the server invests almost all of the storage resources in order to store and update the database. Trivially, as shown in Remark 3, the storage overhead of client is only two elements in $\mathbb{G}_1$.

Table 1 presents the comparison among the three schemes. We denote by M a multiplication in $\mathbb{G}_1$ (or $\mathbb{G}_2$), by E an exponentiation in $\mathbb{G}_1$, by I an inverse in $\mathbb{G}_1$, by P a computation of the pairing[2], by F an operation on a pseudo-random function, by H a regular hashing operation[3], by En a regular encryption operation, and by $\mathcal{H}$ an incremental hashing operation. We omit other operations such as addition in $\mathbb{G}_1$ for all three schemes.

[2] We argue that the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in Benabbas-Gennaro-Vahlis's scheme are different from those in our scheme since their scheme uses bilinear groups of composite order. Thus, the operations in the groups require different computational overload though we use the same notions for both schemes.

[3] Note that *regular* means the output of operation should be computed from scratch.

In the query algorithm of our scheme, the server does not need to compute the proof each time. Betises, in the the verify and update algorithms, the client in our scheme requires less computational overhead since it does not require to perform the operations of encryption and hashing from scratch. Therefore, our scheme is much more efficient than schemes [12, 13] in these three algorithms. On the other hand, the server in update algorithm of our scheme requires a little more computational overhead, i.e., an incremental BLS signature, in order to achieve accountability. If we use the incremental hash-then-sign paradigm, the server only performs the operations of an an exponentiation in $\mathbb{G}_1$ and an incremental hashing.

Table 1. Efficiency Comparison

Scheme	Scheme [12]	Scheme [13]	Our Scheme
Computational Model	Amortized	Amortized	Amortized
ComputationalAssumption	Subgroup Member	CDH	CDH
Public Verifiability	No	Yes	Yes
Accountability	No	No	Yes
Server Computation (Query)	$(q-1)M+2P$	$(q-1)(M+E)$	/
Verifier Computation (Verify)	$4M+3E+2F$ $+1P+1H$	$1M+1E+1I$ $+2P+1H$	$1M+1E+1I$ $+4P+1\mathcal{H}$
Client Computation (Update)	$2M+3E+2F$ $+1P+1En+1H$	$1M+1E$ $+1En+2H$	$1E+1\mathcal{H}$
Server Computation (Update)	$1M$	/	$1E+1\mathcal{H}$

6 Conclusion

The primitive of verifiable database with efficient updates is useful to solve the problem of verifiable outsourcing of storage. However, the existing schemes cannot satisfy the property of incremental update, i.e., the client must re-compute the new ciphertext and the updated tokens from scratch each time. In this paper, we first introduce the notion of verifiable database with incremental updates (Inc-VDB) that can lead to huge efficiency gain when the database undergoes frequent while small modifications. Besides, we propose a general Inc-VDB framework by incorporating the primitive of vector commitment and the encrypt-then-incremental MAC mode of encryption. We also present a concrete Inc-VDB scheme based on the computational Diffie-Hellman (CDH) assumption.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (Nos. 61272455 and 61100224), China 111 Project (No. B08038), Doctoral Fund of Ministry of Education of China (No. 20130203110004), Program for New Century Excellent Talents in University

(No. NCET-13-0946), and the Fundamental Research Funds for the Central Universities (No. BDY151402). Besides, Lou's work is supported by US National Science Foundation under grant (CNS-1217889).

References

1. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (AsiaCCS), pp. 48–59 (2010)
2. Atallah, M.J., Pantazopoulos, K.N., Rice, J.R., Spafford, E.H.: Secure outsourcing of scientific computations. Advances in Computers 54, 216–272 (2001)
3. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable Delegation of Computation on Outsourced Data. In: Proceedings of the ACM conference on Computer and Communications Security (CCS), pp. 863–874 (2013)
4. Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: How to remove intractability assumptions. In: Proceedings of the ACM Symposium on Theory of Computing (STOC), pp. 113–131 (1988)
5. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 216–233. Springer, Heidelberg (1994)
6. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental Cryptography and Application to Virus Protection. In: Proceedings of the 27th ACM Symposium on the Theory of Computing (STOC), pp. 45–56 (1995)
7. Buonanno, E., Katz, J., Yung, M.: Incremental Unforgeable Encryption. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 109–124. Springer, Heidelberg (2002)
8. Blum, M., Luby, M., Rubinfeld, R.: Program result checking against adaptive programs and in cryptographic settings. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 107–118 (1991)
9. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. Journal of Computer and System Science, 549–595 (1993)
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairings. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
12. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
13. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Heidelberg (2013)
14. Camenisch, J., Hohenberger, S., Pedersen, M.: Batch Verification of Short Signatures. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 246–263. Springer, Heidelberg (2007)
15. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)

16. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
17. Canetti, R., Riva, B., Rothblum, G.: Practical delegation of computation using multiple servers. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), pp. 445–454 (2011)
18. Chen, X., Li, J., Susilo, W.: Efficient Fair Conditional Payments for Outsourcing Computations. IEEE Transactions on Information Forensics and Security 7(6), 1687–1694 (2012)
19. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 541–556. Springer, Heidelberg (2012)
20. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the ACM Symposium on the Theory of Computing (STOC), pp. 113–122 (2008)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. SIAM Journal on Computing 18(1), 186–208 (1989)
22. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
23. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceedings of the ACM Symposium on Theory of Computing (STOC), pp. 723–732 (1992)
24. Kilian, J.: Improved efficient arguments. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 311–324. Springer, Heidelberg (1995)
25. Micali, S.: CS proofs. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS), pp. 436–453 (1994)
26. Mironov, I., Pandey, O., Reingold, O., Segev, G.: Incremental Deterministic Public-Key Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 628–644. Springer, Heidelberg (2012)
27. Martel, C.U., Nuckolls, G., Devanbu, P.T., Gertz, M., Kwong, A., Stubblebine, S.G.: A general model for authenticated data structures. Algorithmica 39(1), 21–41 (2004)
28. Naor, M., Nissim, K.: Certificate revocation and certificate update. In: Proceedings of the 7th conference on USENIX Security Symposium, vol. 7, p. 17 (1998)
29. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
30. Papamanthou, C., Tamassia, R.: Time and space efficient algorithms for two-party authenticated data structures. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 1–15. Springer, Heidelberg (2007)
31. Tamassia, R., Triandopoulos, N.: Certification and authentication of data structures. In: Alberto Mendelzon Workshop on Foundations of Data Management (2010), `http://www.cs.bu.edu/~nikos/papers/cads.pdf`