Verifiable Computation over Large Database with Incremental Updates

Xiaofeng Chen, Jin Li, Jian Weng, Jianfeng Ma, and Wenjing Lou

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.

Index Terms—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 attention in the scientific community. 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, 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 [9], [13], [14], [34], [35], [42], [43], [45]. Most of the prior work focused on generic solutions for an 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.

- *X. Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China, and the State Key Laboratory of Cryptology, PO Box 5159, Beijing 100878, China. E-mail: xfchen@xidian.edu.cn.*
- *J. Li is with the School of Computer Science, Guangzhou University, Guangzhou, China. E-mail: jinli71@gmail.com.*
- *J. Weng is with the Department of Computer Science, Jinan University, Tianhe 510632, China. E-mail: cryptjweng@gmail.com.*
- *J. Ma is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, China. E-mail: jfma@mail.xidian.edu.cn.*
- *W. Lou is with the Department of Computer Science, Virginia Polytechnic Institute and State University, Falls Church, VA 22043. E-mail: wjlou@vt.edu.*

Manuscript received 19 July 2015; revised 4 Dec. 2015; accepted 18 Dec. 2015. Date of publication 24 Dec. 2015; date of current version 14 Sept. 2016. Recommended for acceptance by F. Rodríguez-Henríquez. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2015.2512870

Benabbas et al. [19] first proposed the notion of the verifiable database (VDB) in order to solve the problem of verifiable outsourcing storage. That is, 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).

Trivially, we can construct efficient VDB schemes based on message authentication codes or digital signatures for a static database. However, it is another thing if the client (frequently) performs updates on the database. As noted in [19], the main technical difficulty in this case 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 respond 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.

Benabbas et al. [19] 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). Basically, it is sufficient to just achieve private verifiability for VDB schemes in most applications. While in some special scenarios (especially in the case of the database owner is not the database user), it is essential to achieve public verifiability. For example, the client (or data

owner) outsources a database which supports public traffic information services to every driver (data user). In this case, each user should be able to publicly verify the validity of the proof by the cloud server in VDB schemes. Very recently, Catalano and Fiore [22] 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. For example, one anti-virus company outsources its virus database to a cloud server. Also, the company must add the new-discovered viruses to the database everyday. Generally, the daily new-discovered viruses are a very tiny part of whole database and almost all parts of database remain unchanged. In this case, the client must *re-compute* and *update* the encrypted version (ciphertext) on the server at all times [10], [11]. 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 propose 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 et al. [10], [11] 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 ciphertext of other blocks remains identical [12], [46]. Nevertheless, we argue that the incremental encryption does not provide a full solution for constructing efficient Inc-VDB schemes. The reasons are two-fold: First, 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 1 bit for each single block is changed. If this case happens, the client must re-compute the whole ciphertext from scratch. Second, 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-fold:

- 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 [22] and the encrypt-then-incremental MAC mode of encryption [12]. 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 do not satisfy the property of accountability. We prove that the proposed Inc-VDB scheme satisfies the property of accountability.

This is the full version of the paper that has been presented in ESORICS 2014 [31]. The main differences from the conference version are as follows: First, we present the related work of secure outsourcing computations to illustrate the research progress. We also introduce the primitive of vector commitment and Catalano-Fiore's VDB framework in Sections 2.3 and 2.4, respectively. Second, we add a new Section 4 to present an incremental encryption mechanism based on bit flipping. We also present the formal security proof. Finally, we add a new Section 6 to provide a thorough experimental evaluation of the proposed incremental VDB scheme.

1.2 Related Work

Gennaro et al. [39] first formalized the notion of verifiable computation. Though the solution allows a client to outsource the computation of an arbitrary function, it is inefficient for practical applications due to the complicated fully homomorphic encryption (FHE) techniques [37], [38]. Besides, another disadvantage of the schemes based on FHE is that the client must repeat the expensive pre-processing stage if the malicious server tries to cheat and learns a bit of information, i.e., the client has accepted or rejected the computation result. Therefore, plenty of researchers investigated verifiable computation for specific functions in order to obtain much more efficient protocols such as matrix multiplications and quadrature [1], [3], [6], [53], [54], sequence comparisons [4], [16], and cryptographic algorithms [21], [30], [32], [40], [41], [50].

Previous research works for VDB are mainly based on accumulators [24], [25], [49] and authentication data structures [47], [48], [51], [52]. However, it seems that these solutions either rely on non-constant size assumptions or require expensive operation. Benabbas et al. [19] presented the first practical VDB scheme based on the hardness of the subgroup membership problem in bilinear groups. However, the scheme does not satisfy the property of public verifiability. Catalano and Fiore [22] proposed an elegant solution to construct the public verifiable VDB schemes from vector commitment. Both of the schemes assumed that

the size of the outsourced database should be fixed and the client can know the outsourcing function in advance. Recently, Backes et al. [8] presented a flexible VDB with two additional properties that eliminates the assumption.

Generally, there are three kinds of approaches to achieve the verifiability of outsourcing computations. The first one is mostly suitable for the case that the verification itself is never involved in any expensive computations. For example, for the "inversion of one-way function" class of outsourcing computations [5], [27], [28], [29], [36], the client can directly verify the result since the verification is just equivalent to compute the one-way functions. The second approach is that the client uses multiple servers to achieve verifiability [26], [29], [41]. That is, the client sends the random test query to multiple servers and it accepts only if all the servers output the same result. Trivially, the approach can only ensure the client to detect the error with probability absolutely less than 1. The last approach is based on one malicious server and might leverage some proof systems [34], [42], [43], [45]. Obviously, an essential requirement is that the client must verify the proofs efficiently.

1.3 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. In Section 4, we propose an incremental encryption mechanism based on bit flipping besides the formal security proof. In Section 5, we firstly propose a general and efficient Inc-VDB framework and then present a concrete Inc-VDB scheme. We also present the security and efficiency analysis of the proposed Inc-VDB scheme. The experimental evaluation of the proposed scheme is given in Section 6. Finally, concluding remarks will be made in Section 7.

2 Preliminaries

In this section, we first introduce the basic definition and properties of bilinear pairings. We then present the formal definition of VDB. Besides, we also introduce the primitive of vector commitment and Catalano-Fiore's elegant VDB framework.

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 $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties:

1) Bilinear: $e(u^\alpha, v^\beta) = e(u, v)^{\alpha\beta}$ for all $u, v \in \mathbb{G}_1$, and $\alpha, \beta \in \mathbb{Z}_p^*$.
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 problem in $\mathbb{G}_1$.

Definition 1. *The Computational Diffie-Hellman 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 $\Pr[\mathcal{A}(1^k, g, g^x, g^y) = g^{xy}] \leq \mathsf{negl}(k)$ 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 [7].

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 [19], [22]:

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 triples (x, 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 [22], 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 [19]. Trivially, a verifiable database scheme should support both verifiability for various applications.

2.3 Vector Commitment

Informally speaking, a vector commitment scheme [22] allows to commit to an ordered sequence of values $(m_1, \ldots, m_q)$ in such a way that the committer can later open the commitment at specific positions. Furthermore, anyone should not be able to open a commitment to two different values at the same position (this is called *position binding*). Besides, vector can be required to be *hiding*. That is, any adversary cannot distinguish whether

a commitment was created to a sequence $(m_1, \ldots, m_q)$ or to $(m'_1, \ldots, m'_q)$, even after seeing some openings at some positions. However, hiding is not a critical property in the realization of vector commitment for some applications, e.g., constructing verifiable database with efficient updates. Besides the properties of position binding and hiding, vector commitment needs to be *concise*, i.e., the size of the commitment string and the opening are both independent of q. In the following, we present a formal definition of vector commitment [22].

Definition 3. *A vector commitment scheme* $\mathsf{VC} = (\mathsf{VC.KeyGen}, \mathsf{VC.Com}, \mathsf{VC.Open}, \mathsf{VC.Veri}, \mathsf{VC.Update}, \mathsf{VC.ProofUpdate})$ *consists of the following algorithms:*

- $\mathsf{VC.KeyGen}(1^k, q)$*: On input the security parameter* k *and the size* $q = poly(k)$ *of the committed vector, the key generation algorithm outputs some public parameters* PP *which also implicitly define the message space* $\mathcal{M}$*.*
- $\mathsf{VC.Com}_{\mathsf{PP}}(m_1, \ldots, m_q)$*: On input a sequence of* q *messages* $(m_1, \ldots, m_q) \in \mathcal{M}^q$*, and the public parameters* PP*, the committing algorithm outputs a commitment string* C *and an auxiliary information* aux*.*
- $\mathsf{VC.Open}_{\mathsf{PP}}(m, i, \mathsf{aux})$*: This algorithm is run by the committer to produce a proof* π_i *that* m *is the* i*th committed message.*
- $\mathsf{VC.Ver}_{\mathsf{PP}}(C, m, i, \pi_i)$*: The verification algorithm outputs 1 only if* π_i *is a valid proof that* C *is a commitment to a sequence* $(m_1, \ldots, m_q)$ *such that* $m = m_i$*.*
- $\mathsf{VC.Update}_{\mathsf{PP}}(C, m, i, m')$*: This algorithm is run by the original committer who wants to update* C *by changing the* i*th message to* m'*. It takes as input the old message* m *at the position* i*, the new message* m'*, outputs a new commitment* C' *together with an update information* U*.*
- $\mathsf{VC.ProofUpdate}_{\mathsf{PP}}(C, U, m', i, \pi_i)$*: The algorithm can be run by any user who holds a proof* π_i *for some message at the position* j *with respect to.* C*. It allows the user to compute an updated proof* π'_i *(and the updated commitment* C'*) such that* π'_i *is valid with respect to* C' *which contains* m' *as the new message at the position* i*. Basically, the value* U *contains the update information which is needed to compute such values.*

2.4 VDB Framework from Vector Commitment

Catalano-Fiore's VDB general framework from vector commitment is given as follows [22].

- $\mathsf{Setup}(1^k, DB)$: Let the database be $DB = (x, v_x)$ for $1 \le x \le q$. Run the key generation algorithm of vector commitment to obtain the public parameters $\mathsf{PP} \leftarrow \mathsf{VC.KeyGen}(1^k, q)$. Run the committing algorithm to compute the commitment and auxiliary information $(C, \mathsf{aux}) \leftarrow \mathsf{VC.Com}_{\mathsf{PP}}(v_1, \ldots, v_q)$. Define $\mathsf{PK} = (\mathsf{PP}, C, \mathsf{aux}, DB)$ as the public key of VDB scheme, and $\mathsf{SK} = /$ as the secret key of the client.
- $\mathsf{Query}(\mathsf{PK}, x)$: On input an index x, the server firstly runs the opening algorithm to compute $\pi_x \leftarrow \mathsf{VC.Open}_{\mathsf{PP}}(v_x, x, \mathsf{aux})$ and then returns $\tau = (v_x, \pi_x)$.
- $\mathsf{Verify}(\mathsf{PK}, x, \tau)$: Parse the proofs $\tau = (v_x, \pi_x)$. If $\mathsf{VC.Ver}_{\mathsf{PP}}(C, x, v_x, \pi_x) = 1$, then return v_x. Otherwise, return an error $\perp$.
- $\mathsf{Update}(\mathsf{SK}, x, v'_x)$: To update the record of index x, the client firstly retrieves the current record v_x from the server. That is, the client obtains $\tau \leftarrow \mathsf{Query}(\mathsf{PK}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x \neq \perp$. Also, the client computes $t'_x = (C', U) \leftarrow \mathsf{VC.Update}_{\mathsf{PP}}(C, v_x, x, v'_x)$ and sends the server (x, t'_x, v'_x). Then, the server uses v'_x to update the database record of index x, and t'_x to update the public key.

3 VDB with Incremental Updates

In this section, we introduce the formal definition and security requirements of VDB 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 notation) 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 as 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 4. *A verifiable database scheme with incremental updates* $\mathsf{Inc\text{-}VDB} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Inc\text{-}Update})$ *consists 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.*
- $\mathsf{Inc\text{-}Update}(\mathsf{SK}, x, v')$*: 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 triples* (x, 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 [19], 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 5 (Security). *An* Inc-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*

$$\mathrm{Adv}_A(\textsf{Inc-VDB}, DB, k) \leq \mathrm{negl}(k),$$

where

$$\mathrm{Adv}_A(\textsf{Inc-VDB}, DB, k) = \Pr[\mathbf{Exp}_A^{\textsf{Inc-VDB}}(DB, k) = 1]$$

is defined as the advantage of A in the experiment as follows:

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

In the above experiment, we implicitly assign $\textsf{PK} \leftarrow \textsf{PK}_i$ 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 6 (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 7 (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 8 (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.*

4 Incremental Encryption Based on Bit Flipping

In this section, we propose a new incremental encryption based on the bit flipping operation. More precisely, we give a general mechanism for converting any provable secure encryption scheme into an incremental one. The construction $\Pi = (\textsf{KG}, \textsf{ENC}, \textsf{DEC}, \textsf{Inc-ENC}, \textsf{Inc-DEC})$ is defined as follows:

- KG: On input the security parameter k, the key generation algorithm outputs the secret/public key pair (SK, PK). Without loss of generality, let $\Pi_0 = (\textsf{KG}, \textsf{ENC}, \textsf{DEC})$ be any IND-CCA secure (symmetric or asymmetric) encryption scheme and the key is implicit in the notation for simplicity. Trivially, the public key PK is an empty string if Π_0 is a symmetric scheme.
- ENC: On input a message m, the encryption algorithm outputs a ciphertext $c = \textsf{ENC}(m)$.
- DEC: On input the ciphertext c, the decryption algorithm outputs the message $m = \textsf{DEC}(c)$.
- Inc-ENC: On input a slightly modified message m', the original message m, and the ciphertext c on m, the incremental encryption algorithm outputs the (incremental) ciphertext $c' = \textsf{Inc-ENC}(m') = (c, P)$, where $P = (p_1, p_2, \ldots, p_\omega)$ denotes the bit positions where m' and m have different values, i.e., $m[p_i] \neq m[p_i]$ for $1 \leq i \leq \omega$.
- Inc-DEC: On input the ciphertext $c' = (c, P)$, the incremental decryption algorithm outputs the message m'. Trivially, it firstly decrypts c to obtain m and then performs the bit flipping operation on the location p_i $(1 \leq i \leq \omega)$ of m.

In the following, we present the formal security proof of our construction.

Theorem 4.1. *If Π_0 is an IND-CCA secure (symmetric or asymmetric) encryption scheme, then Π is also an IND-CCA secure encryption scheme.*

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 [39].

Proof. We prove by contradiction. Assume that there exists a polynomial time IND-CCA adversary $\mathcal{A}$ can successfully attack the scheme Π with a non-negligible probability ϵ in time T, then we can construct another polynomial time IND-CCA adversary $\mathcal{A}_0$ that uses $\mathcal{A}$ as a subroutine to attack the scheme Π_0. Without loss of generality, we assume that $\mathcal{A}$ can make at most $q_1 + q_2$ decryption queries.

Let $c_i' = (c_i, P_i)$ be a decryption query issued by $\mathcal{A}$. Trivially, $\mathcal{A}_0$ can relay the partial decryption query c_i to the challenger $\mathcal{C}$ in Π_0. Suppose the response of $\mathcal{C}$ is m_i, $\mathcal{A}_0$ performs the bit flipping operation on the location P_i of m_i to obtain m_i'. Then, $\mathcal{A}_0$ responds m_i' to the query c_i' of $\mathcal{A}$.

After issuing q_1 decryption queries, $\mathcal{A}$ chooses two distinct (challenge) messages $(\mathbf{m}_0', \mathbf{m}_1')$ and sends them to $\mathcal{A}_0$. Similarly, $\mathcal{A}_0$ can compute two corresponding (challenge) messages $(\mathbf{m}_0, \mathbf{m}_1)$ for scheme Π_0, where $\mathbf{m}_b$ $(b \in \{0,1\})$ is obtained by performing the bit flipping operation on the location $\mathbf{P}$ of $\mathbf{m}_b'$ and $\mathbf{P}$ is some bit positions randomly chosen by $\mathcal{A}_0$.

Let the challenge ciphertext by $\mathcal{C}$ be $\mathbf{c} = \mathsf{ENC}(\mathbf{m}_b)$. Trivially, $\mathcal{A}_0$ can compute the corresponding challenge ciphertext $\mathbf{c}' = (\mathbf{c}, \mathbf{P})$ for $\mathcal{A}$. Then, $\mathcal{A}$ can issue further q_2 decryption queries except $\mathbf{c}'$ and $\mathcal{A}_0$ responds in the same way as above.

Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. Then, $\mathcal{A}_0$ can replay b' as its guess in the scheme Π_0. Trivially, the success probability of $\mathcal{A}_0$ is also ϵ. □

Remark 2. As pointed out in [11], incremental encryption leaks some information that is kept secret when using a traditional encryption scheme. In the proposed incremental encryption scheme Inc-ENC, 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 m and m'). This is similar to previous incremental encryptions [11], [12], [46]. In order to achieve stronger privacy, it should also encrypt the modified location information P. That is, $\mathsf{Inc\text{-}ENC}(m') = (\mathsf{ENC}(m), \mathsf{ENC}(P))$. On the other hand, though we only focus on the bit flipping operation in our construction, it can be extended to other operations such as insert, delete, etc.

Remark 3. After performing l rounds of update, the ciphertext is $(\mathsf{ENC}(m), P_1, \ldots, P_l)$. We present a more efficient method to represent the ciphertext. We can give the following recursive definition if we view P_j as a set:

$$\bar{P}_1 = P_1,$$
$$\bar{P}_{j+1} = \bar{P}_j \oplus P_{j+1} = (\bar{P}_j - P_{j+1}) \cup (P_{j+1} - \bar{P}_j).$$

As a result, the ciphertext is now $(\mathsf{ENC}(m), \bar{P}_l)$ (or $(\mathsf{ENC}(m), \mathsf{ENC}(\bar{P}_l))$ to enhance the privacy). This ensures to delete the identical positions information of P_j (thus no bit flipping operation is requires in these positions) and the ciphertext is also shortened.

5 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.

5.1 High Description

Catalano and Fiore presented an elegant construction for building a general VDB framework from vector commitment [22]. 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 provides 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 the above incremental encryption to generate the ciphertext v_x'. More precisely, we define $v_x' = (v_x, P_x)$, where $P_x = (p_1, p_2, \ldots, p_\omega)$ 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 performs 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 $|P_x| \ll |v_x'|$ (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 [12], i.e., an incremental encryption together with an incremental MAC of the ciphertext (the encrypt-then-MAC approach [18]). Trivially, we could use an incremental signature scheme to substitute the incremental MAC. In our concrete construction, we adopt the (incremental) BLS signature scheme [17]. For every update, the client first verify the current BLS signature on the commitment C_R and all the current modifications $(P_x^{(1)}, \ldots, P_x^{(T)})$ of the data record v_x, where $P_x^{(i)}$ denotes the modification in the ith update for $1 \leq i \leq T$. This ensures that the current database is not tampered with by the server.[2] 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

2. Bellare, Goldreich, and Goldwasser pointed out that some incremental signature schemes may suffer from the so-called substitution attack in some scenarios. However, it assumed that the adversary can successfully tamper with the data and the signer does not check the corresponding signatures. Obviously, the attack does not work in our scheme.

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.

If we use different vector commitment schemes and incremental encryption/signature schemes, we can obtain various constructions for Inc-VDB schemes. That is, the paradigm by incorporating the primitive of vector commitment and the encrypt-then-incremental MAC mode of encryption actually provides a general framework for constructing Inc-VDB schemes.

5.2 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 $\mathcal{M} = \mathbb{Z}_p$.[3]

  Let $(\alpha, Y = g^\alpha)$ and $(\beta, S = g^\beta)$ be the secret/public key pair of the client and server, respectively, where $\alpha, \beta \in_R \mathbb{Z}_p^*$. 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 $(v_1, v_2, \ldots, v_q)$. 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 [23] 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 $\sigma = \mathcal{H}(C_R, 0, 0, \ldots, 0)^\beta$ on C_R and all initial counters $(0, 0, \ldots, 0)$ (note that all T_x has an initial value 0). Also, set $\mathsf{aux} = \{\mathsf{aux}_1, \ldots, \mathsf{aux}_q\}$, 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

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

- $\mathsf{Verify}(\mathsf{PK}, x, \tau)$: Parse the proofs $\tau = (v_x, \pi_x, H_x^{(T_x)}, P_x^{(1)}, \ldots, P_x^{(T_x)}, T_x)$. 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 $e(H_x^{(T_x)}, g) = e(\mathcal{H}(C_R, x, P_x^{(1)}, \ldots, P_x^{(T_x)}, T_x), Y)$ hold.[4] If the proofs τ are valid, the verifier accepts them and outputs $v_x^{(T_x)} = (v_x, P_x^{(1)}, \ldots, P_x^{(T_x)})$. 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 $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x^{(T_x)} \ne \perp$. Then, the client computes the incremental signature

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

  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 in PK, i.e., $\mathsf{aux}_x \leftarrow (t'_x, P_x^{(1)}, \ldots, P_x^{(T_x+1)}, T_x + 1)$. Also, the server computes an updated incremental signature $\sigma = \mathcal{H}(C_R, T_1, T_2, \ldots, T_x + 1, \ldots, T_q)^\beta$ 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 4. Note that the proof $\pi_x = \prod_{1 \le j \le q, j \ne x} h_{x,j}^{v_j}$ is always identical for all queries to the same index x. Therefore, the server only needs to compute π_x once for the first query on index x (this is different from the scheme [22]). Trivially, the server requires much less computational overhead for the query algorithms. Besides, we can use the techniques in Remark 3 to shorten the proofs τ.

Remark 5. It is trivial that the above framework supports the property of public verifiability (note that any verifier should also check the validity of the counter T_x). Similarly, we can also adopt the idea of using a verifiable random function to achieve private verifiability. For more details, please refer to [22].

Remark 6. 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, we estimate the storage overload of client for very large q.

Assume that $q = 10^8$ and the counter T_x for each index x is 10^6 (that is, the database has 10^8 records and for each index x it has been updated 1 million times). This means that total updated times for the database is 10^{6*10^8} (this is a giant update times in the real applications). However, the storage of client is only about $7 * 10^8$ bits (less that 700 M). It is still tolerable even for a resource-limited user.

In the following we present a new solution to further reduce the storage overload of client. The trick is to still use

3. Though the message space $\mathcal{M}$ in this construction is $\mathbb{Z}_p$, it can be easily extended to support possibly large payload v_i by using a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$.

4. If the verifier is client, then he needs only check whether the equation $H_x^{(T_x)} = \mathcal{H}(C_R, x, P_x^{(1)}, \ldots, P_x^{(T_x)}, T_x)^y$ holds in order to decrease the computation overload.

vector commitment. The server computes the signature $\sigma = \mathcal{H}(C_R, C_T)^{\beta}$, where C_T is the vector commitment on all counters $(T_1, T_2, \ldots, T_q)$. 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.

Remark 7. The existing VDB schemes [19], [22] and our construction only support the query based on index. The reason is that we just view the database as a set of tuples (x, m_x). In the real applications, these schemes could be extended to allow queries based on some values. Besides, for the simplicity of description, all three schemes only considered the queries over single index in each update. Nevertheless, they could also be extended to queries over multiple indexes in one update. We appreciate an anonymous referee for pointing out this issue.

5.3 Security Analysis

Theorem 5.1. *The proposed Inc-VDB scheme is secure under the CDH assumption holds.*

Proof. Similar to [22], 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 (which is equivalent to 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 $\mathsf{aux} = \{\mathsf{aux}_1, \ldots, \mathsf{aux}_q\}$, 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 $\sigma = \mathcal{H}(C_R, 0, 0, \ldots, 0)^{\beta}$.

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 $\sigma = \mathcal{H}(C_R, T_1, T_2, \ldots, T_x + 1, \ldots, T_q)^{\beta}$.

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 $\hat{v} = (\hat{v}_{\hat{x}}, \hat{P}_{\hat{x}}^{(1)}, \ldots, \hat{P}_{\hat{x}}^{(T_{\hat{x}})}, T_{\hat{x}})$. 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 5.2. *The proposed Inc-VDB scheme is correct.*

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

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

First, 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 $e(C_R / h_x^{v_x}, h_x) = e(\pi_x, g)$. Second, due to $H_x^{(T_x)} = \mathcal{H}(C_R, x, P_x^{(1)}, \ldots, P_x^{(T_x)}, T_x)^{\alpha}$, we have $e(H_x^{(T_x)}, g) = e(\mathcal{H}(C_R, x, P_x^{(1)}, \ldots, P_x^{(T_x)}, T_x), Y)$. Hence, the output of the verification algorithm is always the value $v_x^{(T_x)}$. □

Theorem 5.3. *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 $\mathsf{Inc\text{-}Update}$ algorithm, the client only requires the workload of computing an incremental BLS signature. That is, the computational overload of client is to perform an exponentiation in $\mathbb{G}_1$ and an incremental hashing operation. On the other hand, the storage of client is only two elements in $\mathbb{G}_1$ (please refer to Remark 6 for more discussions). □

Theorem 5.4. *The proposed Inc-VDB scheme is accountable.*

Proof. Given the proofs τ with the counter T_x for index x, the client firstly compares 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 $1 \le i \le T_x$). □

5.4 Efficiency Analysis

In this section, we present the efficiency analysis of the proposed scheme and give a comparison with schemes [19], [22].

First, all of the three schemes require a one-time expensive computational effort in the Setup phase. Second, our

TABLE 1
Efficiency Comparison

Scheme	Benabbas-Gennaro-Vahlis Scheme	Catalano-Fiore Scheme	Our Proposed Scheme
Computational Model	Amortized Model	Amortized Model	Amortized Model
Computational Assumption	Subgroup Member Assumption	CDH Assumption	CDH Assumption
Public Verifiability	No	Yes	Yes
Accountability	No	No	Yes
Server Computation (Query)	$(q-1)M_1+2P$	$(q-1)(M_1+E)$	/
Verifier Computation (Verify)	$3M_1+1M_2+3E+2F+1P+1H$	$1M_1+1E+2P+1H$	$1M_1+1E+4P+1H+1\mathcal{H}$
Client Computation (Update)	$1M_1+1M_2+3E+2F+1P+1\mathcal{E}+1\mathcal{D}+1H$	$1M_1+1E+1\mathcal{E}+1\mathcal{D}+2H$	$1E+1\mathcal{D}+1\mathcal{H}$
Server Computation (Update)	$1M_1$	/	$1E+1\mathcal{H}$

proposed scheme simultaneously satisfies the properties of public verifiability and accountability. Besides, our scheme is efficient since the computational resources invested by the client are 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 6, 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_1 (resp. M_2) a multiplication in $\mathbb{G}_1$ (resp. $\mathbb{G}_2$), by E an exponentiation in $\mathbb{G}_1$, by I an inverse in $\mathbb{G}_1$, by P a computation of the pairing,[5] by F an operation on a pseudo-random function, by H a regular hashing operation,[6] by $\mathcal{E}$ (resp. $\mathcal{D}$) a regular encryption (resp. decryption) operation, and by $\mathcal{H}$ an incremental hashing operation [15]. We omit other operations such as addition in $\mathbb{G}_1$ for all three schemes.

In the query algorithm of our scheme, the server does not need to compute the proof each time as discussed in Remark 2. Betises, in 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 [19], [22] 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.

6 Performance Evaluation

In this section, we provide a thorough experimental evaluation of the proposed Inc-VDB scheme. Our experiments are simulated with the pairing-based cryptography (PBC) library and OpenSSL open-source library on a LINUX machine with Intel Core i7-4600U processors running at 2.70 GHz and 8 G memory. Throughout this experiment, in order to precisely evaluate the computation complexity at both client and server sides, we simulate both entities on this LINUX machine.

Since the Type 1 pairing functions are actually shown to be insecure (or very inefficient after some fixing) [2], we do the experiment using the asymmetric pairings (i.e., either Type 2 or Type 3 pairings). The elliptic curve we used is a MNT d224-curve, where the base field size is 224-bit and the embedding degree of the curve is 6 [44]. Also, we adopt the famous CryptDB [20] database system in the experiments and the dataset is the encrypted file with the length of 8 MB.

We provide the time costs simulation for schemes [19], [22] and our scheme in Figs. 1, 2 and 3. Also, we provide the performance analysis of our incremental encryption in Fig. 4. The time cost of query, verify and update algorithms for all three schemes are shown in Figs. 1, 2, 3a and 3b, respectively. Fig. 1 shows that the query time cost of our scheme is always 0, and the query time cost of scheme [22] is relatively small compared with scheme [19]. As shown in Fig. 2, the verification time cost of the three schemes are all linear with the commutating count, and our verification algorithm is the most efficient one. The main reason is that we use the incremental hash algorithm in our scheme (the input for the incremental hash algorithm in our simulation is 8 MB).

In Fig. 3a, we provide the efficiency comparison for data update of the client side. The simulation results show that the growth rate of our scheme is much smaller than that of schemes [19], [22]. Fig. 3b shows the efficiency comparison for data update of server side. The scheme [22] does not need any computation cost in this

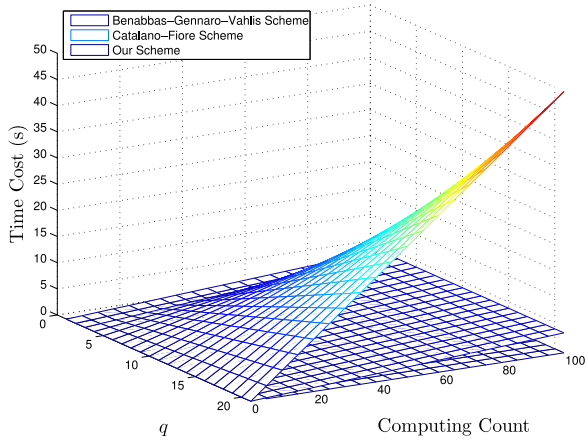


Fig. 1. Query comparison.

5. 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. Actually, the operations in the composite order groups require much more expensive computational overload though we use the same notions for both schemes [33].

6. Note that *regular* means the output of operation should be computed from scratch.

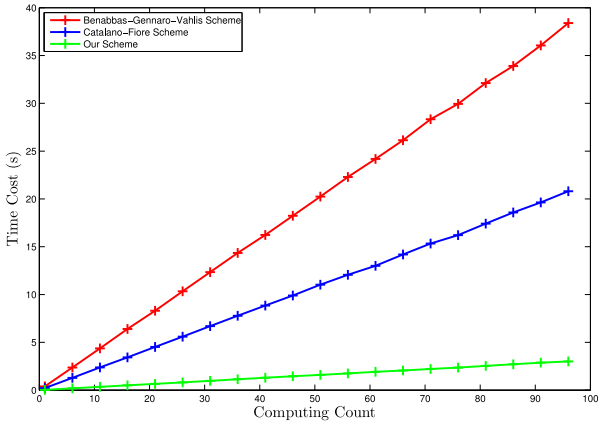


Fig. 2. Verify comparison.

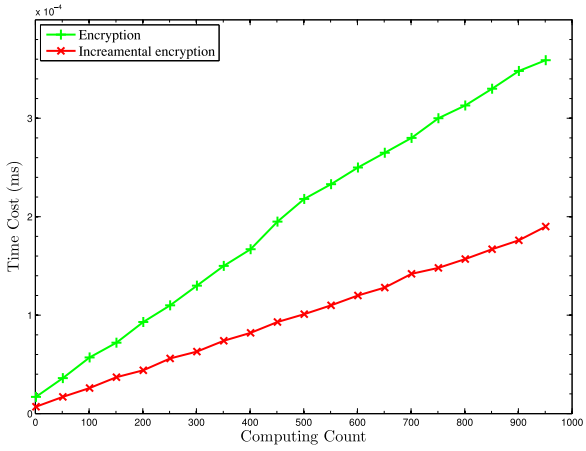


Fig. 4. Efficiency of IncENC.

phase, and thus the computation time is always 0. Since we introduce the pairing computation in the server data update phase, the computation cost of our server side update is relatively higher than that of the scheme [19]. However, we argue that the computational overhead of query algorithm is only performed by the cloud server rather than the resource-constrained client. Therefore, it is reasonable for cloud outsourcing environment. On the other hand, the simulation results in Fig. 4 indicate that our incremental encryption scheme is much more efficient than the normal encryption scheme when the number of updated blocks are sufficiently large. In both verification and update algorithms which are performed by client, the simulation results indicate that our scheme is more efficient than scheme [19]. Besides, since the scheme [22] suffers from the FAU attack and the scheme [19] only provides private verifiability, our scheme is most suitable for real-world applications.

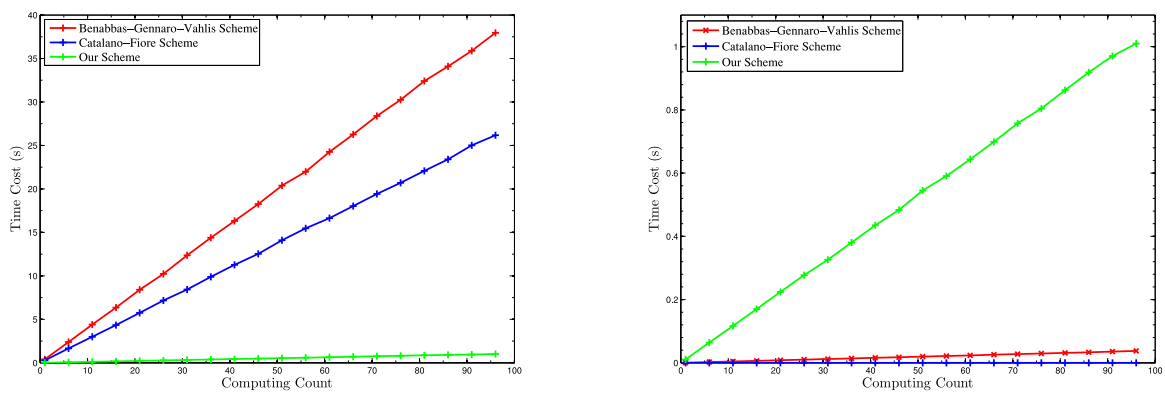
7 Conclusion

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 frequently 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 CDH assumption.

The proposed construction supports the data update such as replacement and deletion operation. However, it seems not be applicable of the insertion operation due to the vector commitment. Thus, an interesting open problem is to propose a general Inc-VDB construction that supports all different updating operations.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61272455 and 61572382), Doctoral Fund of Ministry of Education of China (No. 20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), the Fundamental Research Funds for the Central Universities (No. BDY151402), National High Technology Research and Development Program (863 Program) of China (No. 2015AA016007), and China 111 Project (No. B16037). Besides, Lou's work was supported by US National Science Foundation under grant (CNS-1217889). An extend abstract of this paper has been presented at the 19th European



(a) Client Update Computation

(b) Server Update Computation

Fig. 3. Efficiency comparison for data update.

Symposium on Research in Computer Security (ESORICS 2014), LNCS 8712, Springer, pp. 148-162, 2014.

REFERENCES

[1] M. J. Atallah and K. B. Frikken, "Securely outsourcing linear algebra computations," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Security*, pp. 48-59, 2010.

[2] G. Adj, A. Menezes, T. Oliveira and F. Rodríguez-Henríquez, "Computing discrete logarithms in $F_{3^{6*137}}$ and $F_{3^{6*163}}$ using magma," in *Proc. 5th Int. Workshop Arithmetic Finite Fields*, 2015, pp. 3–22.

[3] M. J. Atallah, K. N. Pantazopoulos, J. R. Rice, and E. H. Spafford, "Secure outsourcing of scientific computations," *Adv. Comput.*, vol. 54, pp. 216–272, 2001.

[4] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Int. J. Inf. Security*, vol. 4, pp. 277–287, 2005.

[5] M. Blanton, "Improved conditional e-payments," in *Proc. 6th Int. Conf. Appl. Cryptograph. Netw. Security*, 2008, pp. 188–206.

[6] D. Benjamin and M. J. Atallah, "Private and cheating-free outsourcing of algebraic computations," in *Proc. 6th Annu. Conf. Privacy, Security Trust*, 2008, pp. 240–245.

[7] F. Bao, R. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Proc. 5th Int. Conf. Inf. Commun. Security*, 2003, pp. 301–312.

[8] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 863–874.

[9] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, "Multi-prover interactive proofs: How to remove intractability assumptions," in *Proc. ACM Symp. Theory Comput.*, 1988, pp. 113–131.

[10] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography: The case of hashing and signing," in *Proc. 14th Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 1994, pp. 216–233.

[11] M. Bellare, O. Goldreich, and S. Goldwasser, "Incremental cryptography and application to virus protection," in *Proc. 27th ACM Symp. Theory Comput.*, 1995, pp. 45–56.

[12] E. Buonanno, J. Katz, and M. Yung, "Incremental unforgeable encryption," in *Proc. 8th Int. Workshop Fast Softw. Encryption*, 2002, pp. 109–124.

[13] M. Blum, M. Luby, and R. Rubinfeld, "Program result checking against adaptive programs and in cryptographic settings," in *Proc. DIMACS Workshop Distrib. Comput. Crypthography*, 1991, pp. 107–118.

[14] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *J. Comput. Syst. Sci.*, pp. 549–595, 1993.

[15] M. Bellare and D. Micciancioy, "A new paradigm for collision-free hashing: Incrementality at reduced cost," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1997, pp. 163–192.

[16] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Proc. 17th Eur. Symp. Res. Comput. Security*, 2012, pp. 505–522.

[17] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairings," in *Proc. 7th Int. Conf. Theory Appl. Cryptol. Inf. Security: Adv. Cryptol.*, 2001, pp. 514–532.

[18] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Proc. 6th Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2000, pp. 531–545.

[19] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. 31st Annu. Conf. Adv. Cryptol.*, 2011, pp. 111–131.

[20] R. A. Popa, N. Zeldovich, and H. Balakrishnan, "CryptDB: A practical encrypted relational DBMS," MIT Comput. Sci. Artif. Intell. Laboratory, Cambridge, MA, USA, Tech. Rep. MIT-CSAIL-TR-2011-005, Jan. 2011.

[21] B. Chevallier-Mames, J. Coron, N. McCullagh, D. Naccache, and M. Scott, "Secure delegation of elliptic-curve pairing," in *Proc. 9th IFIP WG 8.8/11.2 Int. Conf. Smart Card Re. Adv. Appl.*, 2010, pp. 24–35.

[22] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. 16th Int. Conf. Practice Theory Public-Key Cryptography*, 2013, pp. 55–72.

[23] J. Camenisch, S. Hohenberger, and M. Pedersen, "Batch verification of short signatures," in *Proc. 26th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2007, pp. 246–263.

[24] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials," in *Proc. 12th Int. Conf. Practice Theory Public Key Cryptography*, 2009, pp. 481–500.

[25] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. 22nd Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 2002, pp. 61–76.

[26] R. Canetti, B. Riva, G. Rothblum, "Practical delegation of computation using multiple servers," in *Proc. 18th ACM Conf. Comput. Commun. Security*, 2011, pp. 445–454.

[27] B. Carbunar, M. Tripunitara, "Conditional payments for computing markets," in *Proc. 7th Int. Conf. Cryptol. Netw.. Security*, 2008, pp. 317–331.

[28] B. Carbunar, M. Tripunitara, "Fair payments for outsourced computations," in *Proc. Annu. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, 2010, pp. 529–537.

[29] X. Chen, J. Li, and W. Susilo, "Efficient fair conditional payments for outsourcing computations," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 6, pp. 1687–1694, Dec. 2012.

[30] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," in *Proc. Eur. Conf. Res. Comput. Security*, 2012, pp. 541–556.

[31] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," in *Proc. 19th Eur. Symp. Res. Comput. Security*, 2012, pp. 148–162.

[32] D. Chaum and T. Pedersen, "Wallet databases with observers," in *Proc. 12th Annu. Adv. Int. Cryptol. Conf. Adv. Cryptol.*, 1993, pp. 89–105.

[33] A. Guillevic, "Comparing the pairing efficiency over composite-order and prime-order elliptic curves," in *Proc. 11th Int. Conf. Appl. Cryptography Netw. Security*, 2013, pp. 357–372.

[34] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," in *Proc. ACM Symp. Theory Comput.*, 2008, pp. 113–122.

[35] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[36] P. Golle, I. Mironov, "Uncheatable distributed computations," in *Proc. Conf. Topics Cryptol.: Cryptographer's Track RSA*, 2001, pp. 425–440.

[37] C. Gentry, "Fully homomorphic encryption using ideal lattices," *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.

[38] C. Gentry and S. Halevi, "Implementing Gentry's fully-homomorphic encryption scheme," in *Proc. 30th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2011, pp. 129–148.

[39] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. 30th Annu. Conf. Adv. Cryptol.*, 2010, pp. 465–482.

[40] M. Green, S. Hohenberger, B. Waters. (2011). Outsourcing the decryption of ABE ciphertexts. *Proc. 20th USENIX Conf. Security*. The full version can be found at [Online]. Available: http://static.usenix.org/events/sec11/tech/full-papers/Green.pdf

[41] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in *Proc. 2nd Theory Cryptography Conf.*, 2005, pp. 264–282.

[42] J. Kilian, "A note on efficient zero-knowledge proofs and arguments," in *Proc. ACM Symp. Theory Comput.*, 1992, pp. 723–732.

[43] J. Kilian, "Improved efficient arguments (preliminary version)," in *Proc. 15th Annu. Int. Cryptol. Conf.*, 1995, pp. 311–324. 1995.

[44] B. Lynn. (2014). PBC library [Online]. Available: http://crypto.stanford.edu/pbc

[45] S. Micali, "CS proofs," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 436–453.

[46] I. Mironov, O. Pandey, O. Reingold, and G. Segev, "Incremental deterministic public-key encryption," in *Proc. 31st Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2012, pp. 628–644.

[47] C. U. Martel, G. Nuckolls, P. T. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine, "A general model for authenticated data structures," *Algorithmica*, vol. 39, no. 1, pp. 21–41, 2004.

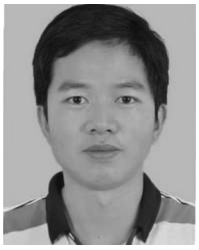
[48] M. Naor and K. Nissim, "Certificate revocation and certificate update," in *Proc. 7th Conf. USENIX Security Symp.*, 1998, vol. 7, pp. 17–17.

[49] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proc. Cryptographers Track RSA Conf.*, 2005, pp. 75–292.

[50] B. Parno, M. Raykova and V. Vaikuntanathan, "How to delegate and verify in public: Verifiable computation from attribute-based encryption," in *Proc. 9th Theory of Cryptography Conf.*, 2012, pp. 422–439.

[51] C. Papamanthou and R. Tamassia, "Time and space efficient algorithms for two-party authenticated data structures," in *Proc. 9th Int. Conf. Inf. Commun. Security*, 2007, pp. 1–15.
[52] R. Tamassia and N. Triandopoulos. (2010). Certification and authentication of data structures. *Proc. Alberto Mendelzon Workshop Found. Data Manage.* [Online]. Available: http://www.cs.bu.edu/nikos/papers/cads.pdf
[53] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *Proc. 30th IEEE Int. Conf. Comput. Commun.*, 2011, pp. 820–828.
[54] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, Jun. 2013.

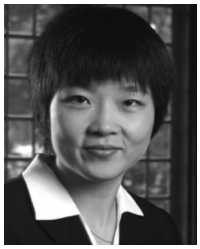
Xiaofeng Chen received the BS and MS degrees in mathematics from Northwest University, China, in 1998 and 2000, respectively. He received the PhD degree in cryptography from Xidian University in 2003. He is currently at Xidian University as a professor. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. His work has been cited more than 3,000 times at Google Scholar. He is in the Editorial Board of Security and Communication Networks (SCN), Computing and Informatics (CAI), and International Journal of Embedded Systems (IJES) etc. He has served as the program/general chair or program committee member in over 30 international conferences.



Jin Li received the BS degree in mathematics in 2002 from Southwest University. He received the PhD degree in information security from Sun Yat-sen University at 2007. He is currently at Guangzhou University as a professor. He has been selected as one of science and technology new star in Guangdong province. His research interests include applied cryptography and security in cloud computing. He has published over 50 research papers in refereed international conferences and journals and has served as the program chair or program committee member in many international conferences.

Jian Weng received the MS and BS degrees in computer science and engineering from South China University of Technology, in 2004 and 2000, respectively, and the PhD degree in computer science and engineering from Shanghai Jiao Tong University, in 2008. From April 2008 to March 2010, he was a postdoctor in the School of Information Systems, Singapore Management University. He is currently a professor and executive dean with the School of Information Technology, Jinan University. He has published more than 60 papers in cryptography conferences and journals such as Eurocryp, Asiacrypt, PKC, and IEEE TIFS. He served as PC co-chair or PC member for more than 10 international conferences.

Jianfeng Ma received the BS degree in mathematics from Shaanxi Normal University, China, in 1985, and the ME and PhD degrees in computer software and communications engineering from Xidian University, China, in 1988 and 1995, respectively. From 1999 to 2001, he was with Nanyang Technological University of Singapore as a research fellow. He is currently a professor in the School of Computer Science, Xidian University, China. His current research interests include distributed systems, computer networks, and information and network security.



Wenjing Lou received the BS and MS degrees in computer science and engineering from Xi'an Jiaotong University, China, the MASc degree in computer communications from the Nanyang Technological University in Singapore, and the PhD degree in electrical and computer engineering from the University of Florida. She is currently a professor in the Computer Science Department, Virginia Polytechnic Institute and State University.