New Publicly Verifiable Databases with Efficient Updates

Xiaofeng Chen, Jin Li, Xinyi Huang, 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 it by assigning a new value. Also, any attempt by the server to tamper with the data will be detected by the client. Very recently, Catalano and Fiore [17] proposed an elegant framework to build efficient VDB that supports public verifiability from a new primitive named vector commitment. In this paper, we point out Catalano-Fiore's VDB framework from vector commitment is vulnerable to the so-called forward automatic update (FAU) attack. Besides, we propose a new VDB framework from vector commitment based on the idea of commitment binding. The construction is not only public verifiable but also secure under the FAU attack. Furthermore, we prove that our construction can achieve the desired security properties.

Index Terms—Verifiable database, cloud computing, secure outsourcing, vector commitment

1 Introduction

With the rapid development of cloud computing, the techniques for securely outsourcing prohibitively expensive computations are getting widespread attentions in the scientific community. In the outsourcing computation paradigm, the client with resource-constraint devices can outsource the heavy computation workloads into the cloud server and enjoy unlimited computing resources in a pay-per-use manner.

Despite the tremendous benefits, outsourcing computation inevitably suffers from some new security challenges. Firstly, the computation tasks often contain some sensitive information that should not be exposed to (semi-trusted) cloud servers. Therefore, one security challenge is the *secrecy* of inputs/outputs of the outsourcing computation. We argue that the traditional encryption techniques can only provide a partial solution to this problem since it is very difficult to perform meaningful computations over the encrypted data. Though the fully homomorphic encryption could be a potential solution, the existing schemes are not practical yet. Secondly, a semi-trusted cloud server may return a computationally indistinguishable (invalid) result due to financial incentives. Therefore, another security challenge is the *verifiability* of the outsourcing computation. That is, the client can verify the validity of computation result efficiently. Trivially, the verification should never be involved in some other complicated computations. At the very least, it must be far more efficient than accomplishing the computation task itself.

The primitive of verifiable computation has been well studied by plenty of researchers in the past decades [8], [9], [10], [30], [32], [39], [40], [41], [43], [53], [56]. Most of the prior work focused on generic solutions for arbitrary functions (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 et al. [15] first proposed the notion of verifiable database (VDB, for short), which is extremely useful to solve the problem 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 it 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 achieve the goal 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 update on the database. As noted in [15], the main technical difficulty is that the client must have a mechanism to revoke the signatures given to the server for the previous

- *X. Chen is with the State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, Shannxi, China, and the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061. E-mail: xfchen@xidian.edu.cn.*
- *J. Li is with the School of Computer Science, Guangzhou University, Guangzhou, Guangdong, China, and the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061. E-mail: lijin@gzhu.edu.cn.*
- *X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, Fujian, China. E-mail: xyhuang81@gmail.com.*
- *J. Ma is with the School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China. E-mail: jfma@mail.xidian.edu.cn.*
- *W. Lou is with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061. E-mail: wjlou@vt.edu.*

Manuscript received 4 July 2014; revised 23 Sept. 2014; accepted 27 Sept. 2014. Date of publication 30 Oct. 2014; date of current version 16 Sept. 2015. 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/TDSC.2014.2366471

values. Trivially, the client could keep track of every change locally in order to solve this issue. 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 [19], [20], [48] and authentication data structures [44], [46], [50], [52]. However, these solutions 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 et al. [15] 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, but does not support the public verifiability (i.e., only the owner of the database can verify the correctness of the proofs). Recently, Catalano and Fiore [17] proposed an elegant solution to build VDB from a primitive named vector commitment. The concrete construction relies on standard constant-size assumption and supports the public verifiability.

1.1 Our Contribution

In this paper, we further study the problem of constructing verifiable database with efficient updates. Our contribution is two fold:

- We point out a security weakness of Catalano-Fiore's elegant VDB framework from vector commitment. That is, the framework is vulnerable to the so-called forward automatic update (FAU) attack defined in this paper.
- We propose a new VDB framework from vector commitment based on the binding commitment. The construction is not only public verifiable but also secure under the FAU attack. Additionally, we present a concrete scheme based on the standard constant-size computational Diffie-Hellman (CDH) assumption. The proposed scheme uses the bilinear pairing groups of prime order instead of composite one and thus is more efficient than Benabbas-Gennaro-Vahlis's scheme [15].

1.2 Related Work

Plenty of researchers have devoted considerable attention to the problem of how to securely outsource different kinds of expensive computations. Abadi et al. [1] first proved the impossibility of secure outsourcing an exponential computation while locally doing only polynomial time work. Therefore, it is meaningful only to consider outsourcing expensive polynomial time computations.

In the theoretical computer science community, Atallah et al. [3] presented a framework for secure outsourcing of scientific computations such as matrix multiplications and quadrature. However, the solution used the disguise technique and thus led to the leakage of private information. Later, there are plenty of research work that also investigated this problem [2], [6], [54], [55]. Atallah and Li [4] investigated the problem of computing the edit distance between two sequences and presented an efficient protocol to securely outsource sequence comparisons to two servers. Recently, Blanton et al. proposed a more efficient scheme for secure outsourcing sequence comparisons [11].

In the cryptographic community, Chaum and Pedersen [28] firstly introduced the notion of wallets with observers, a piece of secure hardware installed on the client's computer to perform some expensive computations. Hohenberger and Lysyanskaya [38] proposed the first outsource-secure algorithm for modular exponentiations based on the two previous approaches of precomputation [13], [47], [51] and server-aided computation [12], [14], [31]. Chen et al. [26] proposed more efficient outsource-secure algorithms for (simultaneously) modular exponentiation in the two untrusted program model. Chevallier-Mames et al. [16] presented the first algorithm for secure delegation of elliptic-curve pairings based on an untrusted server model. However, an obvious disadvantage of the algorithm is that the outsourcer should carry out some other expensive operations such as scalar multiplications and exponentiations. Green et al. [37] proposed new methods for efficiently and securely outsourcing decryption of attribute-based encryption (ABE) ciphertexts.

Since the servers are not fully trusted by the outsourcers, there should exist an efficient way for the client to check the validity of the computation result. Gennaro et al. [36] 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 techniques [34], [35]. Besides, another disadvantage of the schemes based on fully homomorphic encryption 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. Benabbas et al. [15] presented the first practical verifiable computation scheme for high degree polynomial functions [36] and used it to construct efficient VDB schemes. Very recently, Chen et al. [27] introduced the notion of verifiable database with incremental updates. Parno et al. [49] showed a construction of a multi-function verifiable computation scheme based on the outsourced ABE.

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], [22], [23], [25], [33], 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 [21], [25], [38]. 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 [30], [39], [40], [43]. Obviously, an essential requirement is that the client must verify the proofs efficiently.

1.3 Organization

This paper is organized as follows. In Section 2, we present the formal definition and security requirements of VDB. Some preliminaries are presented in Section 3. In Section 4, we overview Catalano-Fiore's VDB Framework from vector commitment and present some security flaws of the construction. We propose a new efficient VDB framework and a concrete VDB scheme in Section 5. The security analysis of the proposed VDB scheme and comparison with existing schemes are given in Section 6. Finally, concluding remarks will be made in Section 7.

2 Verifiable Database with Updates

2.1 Formal Definition

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. Informally, a VDB scheme allows a resource-constrained 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 database records from the server. Inherently, any attempts to tamper with the data by the dishonest server will be detected with an overwhelming probability when the client queries the database. In order to achieve the confidentiality of the data record m_x, the client can use a master secret key to encrypt each m_x using a symmetric encryption scheme such as AES. Trivially, given the ciphertext v_x, only the client can compute the record m_x. Therefore, we only need to consider the case of encrypted database (x, v_x). This is implicitly assumed in the existing academic research.

The formal definition for verifiable databases with updates is given as follows [15], [17]:

Definition 1. *A verifiable database scheme with updates* $\mathsf{VDB} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ *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, a database encoding* S *that is given to the server, and a public key* PK *that is distributed to all users (including the client itself) for verifying the proofs.*
- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$*: 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 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 [17], 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 [15]. A verifiable database scheme should support both verifiability for various applications.

2.2 Security Requirements

In the following, we introduce some security requirements for VDB. The first requirement is the **security** of VDB scheme. Intuitively, a 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. Benabbas et al. [15] presented the following definition:

Definition 2 (Security). *A* 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,*

$$\mathrm{Adv}_A(\mathsf{VDB}, DB, k) \leq \mathrm{negl}(k),$$

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

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

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

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

Definition 3 (Correctness). *A* 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 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]

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 [36].

Definition 4 (Efficiency). *A* 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 q.*

Finally, we introduce a new requirement named *accountability* for VDB scheme [27]. 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 5 (Accountability). *A* VDB *scheme is accountable if for any database $DB \in [q] \times \{0,1\}^*$, where $q = poly(k)$, the client can provide a proof if the dishonest server has tampered with the database.*

3 PRELIMINARIES

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

3.1 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.* $e(u^a, v^b) = e(u,v)^{ab}$ 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 problem in $\mathbb{G}_1$.

Definition 6. *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].

3.2 Vector Commitments

Commitment is a fundamental primitive in cryptography and plays an important role in almost all security protocols such as voting, identification, zero-knowledge proof, etc. Intuitively, a commitment scheme can be viewed as the digital equivalent of a sealed envelope. The sender places a message in the sealed envelope and gives it to the receiver. On one hand, no one except the sender could open the envelope to learn the message from the commitment (this is called *hiding*). On the other hand, the sender could not change the message any more (this is called *binding*).

Very recently, Catalano and Fiore [17] proposed a new primitive called Vector Commitment, which is closely related to zero-knowledge sets [18], [24], [42], [45]. Informally speaking, a vector commitment scheme 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. Therefore, the property of hiding is not considered in Catalano and Fiore's constructions.[2] 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 [17].

Definition 7. *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 ith 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 ith 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 any user who holds a proof π_i for some message at the position j w.r.t. C. It allows the user to compute an updated proof π'_i (and the updated commitment C') such that π'_i is valid w.r.t 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. Trivially, a vector commitment scheme with hiding property can be constructed by composing a standard commitment scheme with any vector commitment scheme that does not satisfy hiding.

4 Catalano-Fiore VDB Framework

Catalano and Fiore presented an elegant construction for building a general VDB framework from vector commitment [17]. In this section, we first overview their VDB general framework and then present a security weakness of the construction.

4.1 The General Framework

Catalano-Fiore's VDB general construction from vector commitment is given as follows.

- $\mathsf{Setup}(1^k, DB)$. Let the database be $DB = (i, v_i)$ for $1 \le i \le q$. 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 $\mathsf{PK} = (\mathsf{PP}, C)$ as the public key of VDB scheme, $\mathsf{S} = (\mathsf{PP}, \mathsf{aux}, DB)$ as the database encoding, and $\mathsf{SK} = \perp$ as the secret key of the client.
- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$. 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 $\perp$.
- $\mathsf{Update}(\mathsf{SK}, x, v')$. To update the record of index x, the client firstly retrieves the current record v_x from the server. That is, the client obtains $\tau \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x \neq \perp$. Also, the client computes $(C', U) \leftarrow \mathsf{VC.Update}_{\mathsf{PP}}(C, v_x, x, v'_x)$ and outputs $\mathsf{PK}' = (\mathsf{PP}, C')$ and $t'_x = (\mathsf{PK}', v'_x, U)$. Then, the server uses v'_x to update the database record of index x, PK' to update the public key, and U to update the auxiliary information.

4.2 Forward Automatic Update Attack

We argue that Catalano-Fiore's VDB construction suffers from the following attack.

The attack is based on the fact that an adversary A (i.e., the malicious server) can perform Update in a same way as the client. That is, the adversary A firstly retrieves the current record v_x. Then, A computes $(C^*, U) \leftarrow \mathsf{VC.Update}_{\mathsf{PP}}(C, v_x, x, v^*_x)$ and outputs $\mathsf{PK}^* = (\mathsf{PP}, C^*)$ and $t^*_x = (\mathsf{PK}^*, v^*_x, U)$ (note that all of the computations do not need any secret knowledge of the client). Finally, the server updates the corresponding database record with v^*_x, and the public key with PK^*. Trivially, the server can generate a valid proof for any query based on PK^*. Besides, this *forward* updated public key PK^* and the real one PK' are totally computationally indistinguishable from a viewpoint of any third party. Therefore, when a dispute occurs, a judge cannot deduce that the server is dishonest. We define this kind of adversary as *forward automatic update* attacker.

We analyze why Catalano-Fiore's VDB construction suffers from the FAU attack. The main reason is that the secret key in Catalano-Fiore's VDB framework is assumed to be empty, i.e., $\mathsf{SK} = \perp$. Trivially, if $\mathsf{SK} = \perp$, then anyone can verify the validity of output τ and thus the construction supports the public verifiability. However, this also allows the adversary A to update the database in an indistinguishable manner as the client since no secret information is required in the update algorithm.[3] Besides, it is more difficult for the third party to detect the FAU attack than the client. Therefore, VDB schemes that support the public verifiability are more vulnerable to the FAU attack in the real-world applications.

In the following, we present the formal proof that Catalano-Fiore's framework violates the *security* definition of VDB (i.e., Definition 2).

Proposition 4.1. *The Catalano-Fiore's VDB framework does not satisfy the property of security.*

Proof. A VDB scheme does not satisfy the property of security means that the adversary A (i.e., the dishonest server) can successfully simulate the experiment $\mathbf{Exp}_A^{\mathsf{VDB}}[DB, k]$ and win the game with a non-negligible probability. In the Catalano-Fiore's VDB framework, the secret key is assumed to be empty, i.e., $\mathsf{SK} = \perp$. Therefore, the adversary A can perform the algorithm Update freely. Our main trick is that we require A to perform an additional round of Update after finishing l rounds of Update queries of the client. However, in the last round of Update, A also plays the role of client. More precisely, the simulated experiment $\mathbf{Exp'}_A^{\mathsf{VDB}}[DB, k]$ is defined as follows:

$$\begin{aligned}
&\text{Experiment } \mathbf{Exp'}_A^{\mathsf{VDB}}[DB, k] \\
&\quad (\mathsf{PK}, \perp) \leftarrow \mathsf{Setup}(DB, k); \\
&\quad \text{For } i = 1, \ldots, l+1; \text{where } l = poly(k); \\
&\qquad \mathbf{Verify} \text{ query}: \\
&\qquad\quad (x_i, \tau_i) \leftarrow A\big(\mathsf{PK}, t'_1, \ldots, t'_{i-1}\big); \\
&\qquad\quad v_i \leftarrow \mathsf{Verify}(\mathsf{PK}/\perp, x_i, \tau_i); \\
&\qquad \mathbf{Update} \text{ query}: \\
&\qquad\quad \big(x_i, v_{x_i}^{(i)}\big) \leftarrow A\big(\mathsf{PK}, t'_1, \ldots, t'_{i-1}\big); \\
&\qquad\quad t'_i \leftarrow \mathsf{Update}\big(\perp, x_i, v_{x_i}^{(i)}\big); \\
&\quad (\hat{x}, \hat{\tau}) \leftarrow A\big(\mathsf{PK}, t'_1, \ldots, t'_{l+1}\big); \\
&\quad \hat{v} \leftarrow \mathsf{Verify}(\mathsf{PK}/\perp, \hat{x}, \hat{\tau}).
\end{aligned}$$

Since we implicitly assign $\mathsf{PK} \leftarrow \mathsf{PK}_i$ after every update query in the experiment, the final public key $\mathsf{PK} = \mathsf{PK}_{l+1}$. Then, let $\hat{v} = v_{\hat{x}}^{(l+1)}$. Trivially, $\hat{v} \neq \perp$ and $\hat{v} \neq v_{\hat{x}}^{(l)}$. This violates the *security* definition of VDB scheme. □

Remark 2. It seems that there are two naive approaches against FAU attack for Catalano-Fiore's VDB framework. The first solution is that we can require the server to compute a signature on the (updated) public key. However, we argue that this solution does not

3. Note that the construction [15] only support the private verifiability since the (non-empty) secret key SK is involved in the verification. Besides, SK is also involved in the update algorithm and hence only the client can update the database.

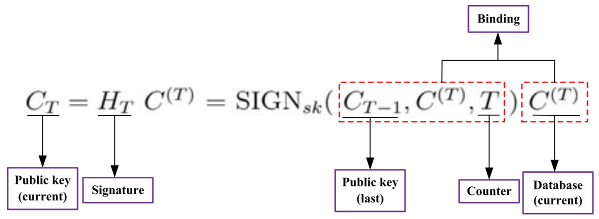


Fig. 1. Commitment binding.

work since the dishonest server is an inherent FAU attacker in Catalano-Fiore's VDB framework (the server has the ability to compute the signature on any public key). The second one is that the client computes a signature on the (updated) public key. Obviously, the server cannot forge the client's signature. However, it requires that the client must have a mechanism to revoke the previous signatures (surprisingly, it reverted to the same problem of VDB). Therefore, neither of the two approaches can solve the security issue of Catalano-Fiore's constructions.

4.3 Backward Substitution Update (BSU) Attack

In this section, we consider another attack in VDB schemes that is not explicitly stated in previous literatures. We call it *Backward Substitution Update* attack. That is, the dishonest server can utilize the previous (while valid) public key and the corresponding database to substitute the current ones (trivially, this can also be viewed as an update). We argue that the server in VDB has the ability to update the public key freely. If this case happens, the effort of the later update by the client is no longer meaningful. Furthermore, if the client did not store the public key locally, it is difficult for him to distinguish the past public key from the latest one. On the other hand, even if the client has stored the latest public key, it seems still to be difficult for him to prove the fact that the stored public key is the latest one.

We argue that Benabbas-Gennaro-Vahlis's scheme [15] does not suffers from BSU attack since the secret key of the client is updated each time. Without loss of generality, assume that the latest secret key of client is SK_l. When the dishonest server presents a previous public key PK_p including a counter T_p and the corresponding database DB_p, it is trivial that the output of query algorithm by the server cannot pass the verification with the secret key SK_l. As a result, the tampering will be detected by the client. However, this cannot be viewed as a proof even if the client presents his secret key to a judge. The reason is that a malicious client also has the ability to frame a honest server. That is, the malicious client can present a random value as the secret key to invalidate the verification on the output of an honest server. Thus, the judge cannot deduce who is dishonest when a dispute occurred. In this sense, we argue that Benabbas-Gennaro-Vahlis's scheme does not satisfy the property of accountability.

We provide a straightforward effective solution to this problem: Similarly, we also introduce a counter T in the public key to denote the update times. The difference is that the server computes a signature σ on the latest counter T_l and the identity ID_c of the client. Given a past public key with the counter T_p, the client provides the pair (σ, T_l) to the judge as a proof. If σ is valid and $T_p < T_l$, the judge claims that the server is dishonest. Trivially, the storage workload of the client is only the latest pair (σ, T_l). Therefore, we do not focus on the BSU attack any more in our proposed construction.

5 New Efficient VDB Framework

In this section, we present a new efficient VDB framework from vector commitment which is secure against the FAU attack. Additionally, we present a concrete VDB construction from Catalano-Fiore's vector commitment scheme based on the CDH assumption [17].

5.1 Design Philosophy

As stated above, the main reason that Catalano-Fiore's framework suffers from the FUA attack is that the secret key SK of the client (actually, SK is assumed to be $\perp$) is not involved in the computation and update of the public key PK. This will enable the adversary (especially the dishonest server) to update PK freely. Note that $\mathsf{PK} = (\mathsf{PP}, C')$ and the public parameter PP of vector commitment is never updated.[4] That is, the server can update the vector commitment C' at its own will and this equals to update the databases. We argue that it is meaningless if we add a signature of the server on C' to PK since the server can compute such a signature on any message. On the other hand, if we use SK to compute the updated public key PK (more precisely, the commitment C') just as in the scheme [15], the proposed VDB scheme might no longer support the public verifiability. The main reason is that the secret key SK might be also involved in the verification of C and the corresponding proofs (i.e., the openings of C). That is, only the client with SK can verify the validity of the proofs. Therefore, it seems to be contradictory to construct a VDB scheme that is public verifiable and secure under the FUA attack simultaneously.

We utilize the idea of *commitment binding* to solve this problem. The main trick is that the client uses the secret key SK to compute a signature on some binding information which will be explained later. Also, the signature is used to compute the updated commitment C'. Since the signature is different for each updating, the server cannot compute a new C' without the cooperation of client.

The binding information consists of the last public key C_{T-1} (a commitment value), the commitment $C^{(T)}$ on the the current database vector, and the current counter T. Assume that the signature of client on binding information is $H_T = \textsc{Sign}_{sk}(C_{T-1}, C^{(T)}, T)$, then the current public key $C_T = H_T C^{(T)}$. So, the solution binds the commitment C_T to the 3-tuple $(C_{T-1}, C^{(T)}, T)$ in a recursion manner as shown in Fig. 1. As a result, the adversary includes the server cannot update the database and public key freely.

4. The definition and constructions of vector commitment are actually given in the public parameter model (also as know as auxiliary string model). That is, the public parameters are generated and published by a trusted party [29].

5.2 New Proposed VDB Framework

The new proposed VDB framework is given as follows.

- $\mathsf{Setup}(1^k, DB)$. Let the database be $DB = (i, v_i)$ for $1 \le i \le q$. Let VC be any secure vector commitment scheme. 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 $(C_R, \mathsf{aux}) \leftarrow \mathsf{VC.Com}_{\mathsf{PP}}(v_1, \dots, v_q)$. Let (sk, pk) be the secret/public key pair of the client. Let $\textsc{Sign}$ be a provably secure digital signature scheme. Let T be a counter with the initial value 0. Let $C^{(T)} = \mathsf{VC.Com}_{\mathsf{PP}}(v_1^{(T)}, \dots, v_q^{(T)})$ be the commitment on the latest database vector after the original database DB has been updated T times. Trivially, $C^{(0)} = C_R$. Specially, let $C_{-1} = C_R$. The client computes and sends the signature $H_0 = \textsc{Sign}_{sk}(C_{-1}, C^{(0)}, 0)$ to the server. If H_0 is valid, then the server computes $C_0 = H_0 C^{(0)}$. Also, the server adds the information of $\Sigma_0 = (H_0, C_{-1}, C^{(0)}, 0)$ to aux.

  Set $\mathsf{PK} = (\mathsf{PP}, pk, C_R, C_0)$ as the public key of VDB scheme, $\mathsf{S} = (\mathsf{PP}, \mathsf{aux}, DB)$ as the database encoding, and $\mathsf{SK} = sk$ as the secret key of the client.
- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$. Assume that the current public key $\mathsf{PK} = (\mathsf{PP}, Y, C_R, C_T)$. On input an index x, the server runs the opening algorithm to compute $\pi_x \leftarrow \mathsf{VC.Open}_{\mathsf{PP}}(v_x, x, \mathsf{aux})$ and returns $\tau = (v_x, \pi_x, \Sigma_T)$, where $\Sigma_T = (H_T, C_{T-1}, C^{(T)}, T)$.
- $\mathsf{Verify}(\mathsf{PK}, x, \tau)$. Parse $\tau = (v_x, \pi_x, \Sigma_T)$. If $\mathsf{Ver}_{pk}(\Sigma_T) = 1$ (this means that H_T is a valid signature of the client on message $(C_{T-1}, C^{(T)}, T)$) and $\mathsf{VC.Ver}_{\mathsf{PP}}(C_T, H_T, x, v_x, \pi_x) = 1$, 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 as in the above Verify algorithm. That is, the client obtains $\tau \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x \ne \perp$. Set $T \leftarrow T + 1$, the client firstly computes $C^{(T)} = \mathsf{VC.Com}_{\mathsf{PP}}(v_1^{(T)}, \dots, v_q^{(T)})$ and $t'_x = H_T = \textsc{Sign}_{sk}(C_{T-1}, C^{(T)}, T)$, and then sends (t'_x, v'_x) to the server. If t'_x is valid, then the server computes $C_T = H_T C^{(T)}$ and updates the public key $PK = (\mathsf{PP}, pk, C_R, C_T)$ (note that only the value of C_T needs to be updated). Also, the server uses the value of v'_x to update the database record of index x, i.e., $DB(x) \leftarrow v'_x$. Finally, the server adds the information of $\Sigma_T = (H_T, C_{T-1}, C^{(T)}, T)$ to aux in S.

Remark 3. It is trivial that the above framework supports the property of public verifiability. Similarly, we can also adopt the idea of using a verifiable random function to achieve private verifiability. For more details, please refer to [17].

5.3 A Concrete VDB Scheme

In this section, we follow the proposed new VDB framework to propose a concrete VDB scheme from the vector commitments based on the CDH assumption [17].

- $\mathsf{Setup}(1^k, DB)$. 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 $\mathcal{H} : \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^* \to \mathbb{G}_1$ 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 $\mathcal{M} = \mathbb{Z}_p$. The client randomly selects an element $y \in_R \mathbb{Z}_p$ and then computes $Y = g^y$.

  Let $C_R = \prod_{i=1}^q h_i^{v_i}$ be the root commitment on the database vector $(v_1, v_2, \dots, v_q)$. Let T be a counter with the initial value 0. Let $C^{(T)}$ be the commitment on the the latest database vector after the original database DB has been updated T times. Trivially, $C^{(0)} = C_R$. Specially, let $C_{-1} = C_R$. The client computes and sends $H_0 = \mathcal{H}(C_{-1}, C^{(0)}, 0)^y$ to the server. If H_0 is valid, then the server computes $C_0 = H_0 C^{(0)}$. Also, the server adds the information of $(H_0, C_{-1}, C^{(0)}, 0)$ to aux.
  Set $\mathsf{PK} = (\mathsf{PP}, Y, C_R, C_0)$, $\mathsf{S} = (\mathsf{PP}, \mathsf{aux}, DB)$ and $\mathsf{SK} = y$.
- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$. Assume that the current public key $\mathsf{PK} = (\mathsf{PP}, Y, C_R, C_T)$. Given a query index x, the server computes $\pi_x^{(T)} = \prod_{1\le j\le q, j\ne x} h_{x,j}^{v_j^{(T)}}$ and returns the proofs

$$\tau = \left(v_x^{(T)}, \pi_x^{(T)}, H_T, C_{T-1}, C^{(T)}, T\right).$$

- $\mathsf{Verify}(\mathsf{PK}, x, \tau)$. Parse the proofs $\tau = (v_x^{(T)}, \pi_x^{(T)}, H_T, C_{T-1}, C^{(T)}, T)$. Then, anyone (including the client) can verify the validity of the proofs τ by checking whether the following two equations $e(H_T, g) = e(\mathcal{H}(C_{T-1}, C^{(T)}, T), Y)$ and $e(C_T / H_T h_x^{v_x^{(T)}}, h_x) = e(\pi_x^{(T)}, g)$ hold.[5] If the proofs τ is valid, the verifier accepts it and outputs $v_x^{(T)}$. Otherwise, outputs 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 $\tau \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \tau) = v_x \ne \perp$.

  Set $T \leftarrow T + 1$, the client firstly computes $C^{(T)} = \frac{C_{T-1}}{H_{T-1}} h_x^{v'_x - v_x}$ and $t'_x = H_T = \mathcal{H}(C_{T-1}, C^{(T)}, T)^y$, and then sends (t'_x, v'_x) to the server. If t'_x is valid, then the server computes $C_T = H_T C^{(T)}$ and updates the public key with $PK = (\mathsf{PP}, Y, C_R, C_T)$. Also, the server uses the value of v'_x to update the database record of index x, i.e., $DB(x) \leftarrow v'_x$. Finally, the server adds the information of $(t'_x = H_T, C_{T-1}, C^{(T)}, T)$ to aux in S.

5. If the verifier is client, then he needs only to check whether $H_T = \mathcal{H}(C_{T-1}, C^{(T)}, T)^y$ holds in order to decrease the computation overload.

TABLE 1
Comparison among Three Schemes

Schemes	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
Secure against FAU Attack	Yes	No	Yes
Public Verifiability	No	Yes	Yes
Accountability	No	No	Yes
Server Computation (Query)	$(q-1)M+2P$	$(q-1)(M+E)$	$(q-1)(M+E)$
Verifier Computation (Verify)	$4M+3E+2F+1P$	$1M+1E+1I+2P$	$2M+1E+1I+4P$
Client Computation (Update)	$2M+3E+2F+1P$	$1M+1E$	$2M+2E+1I$

6 Analysis of Our Proposed VDB

6.1 Security Analysis

Theorem 6.1. *The proposed VDB scheme is secure.*

Proof. Similar to [17], we prove the theorem by contradiction. Assume there exists a polynomial-time adversary A that has a non-negligible advantage ϵ in the experiment $\mathbf{Exp}_A^{\mathsf{VDB}}[DB,k]$ 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}.

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 $\mathbf{Exp}_A^{\mathsf{VDB}}[DB,k]$. 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 $1 \le i \ne j \le q$. Then, B randomly selects an element $y \in_R \mathbb{Z}_p$ and computes $Y = g^y$. Given a database DB, B computes the commitment $C_R = \prod_{i=1}^q h_i^{v_i}$. Also, B computes $H_0 = \mathcal{H}(C_R, C_R, 0)^y$ and $C_0 = H_0 C_R$. Define $\mathsf{PK} = (\mathsf{PP}, Y, C_R, C_0)$, $\mathsf{S} = (\mathsf{PP}, \mathsf{aux}, DB)$ and $\mathsf{SK} = y$. B sends PK to A. Note that PK and S are perfectly distributed as the real ones. To answer the verify and update queries of A in the experiment, B just simply runs the real $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ and $\mathsf{Update}(\mathsf{SK}, x, v'_x)$ algorithms and responds with the same value. Note that the $\mathsf{Update}(\mathsf{SK}, x, v'_x)$ algorithm requires the secret key y of B, and A cannot perform this algorithm without the help of B. Therefore, the FAU attack is no longer successful in the experiment $\mathbf{Exp'}_A^{\mathsf{VDB}}[DB,k]$.

Suppose that $(\hat{x}, \hat{\tau})$ be the tuple returned by A at the end of the experiment, where $\hat{\tau} = (\hat{v}, \hat{\pi}, \Sigma_l)$. Besides, note that if A wins with a non-negligible advantage ϵ in the experiment, then we have $\hat{v} \ne \perp$, $\hat{v} \ne v_{\hat{x}}^{(l)}$ and $e(C^{(l)}, h_{\hat{x}}) = e(h_{\hat{x}}^{v_{\hat{x}}^{(l)}}, h_{\hat{x}}) e(\pi_{\hat{x}}^{(l)}, g) = e(h_{\hat{x}}^{\hat{v}}, h_{\hat{x}}) e(\hat{\pi}, g)$.

If $\hat{x} \ne x^*$, B aborts the simulation and fails. Otherwise, we have $h_{\hat{x}} = g^a$. Trivially, B can compute

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

The success probability of B is ϵ/q. □

Theorem 6.2. *The proposed VDB scheme is correct.*

Proof. If the server is assumed to be honest, then the proofs $\tau = (v_x^{(T)}, \pi_x^{(T)}, H_T, C_{T-1}, C^{(T)}, T)$, where $\pi_x^{(T)} = \prod_{1 \le j \le q, j \ne x} h_{x,j}^{v_j^{(T)}}$. Firstly, note that $H_T = \mathcal{H}(C_{T-1}, C^{(T)}, T)^y$, therefore we have $e(H_T, g) = e(\mathcal{H}(C_{T-1}, C^{(T)}, T), Y)$. Secondly, due to $C_T/H_T h_x^{v_x^{(T)}} = C^{(T)}/h_x^{v_x^{(T)}} = \prod_{1 \le j \le q, j \ne x} h_j^{v_j^{(T)}}$, we have $e(C_T/H_T h_x^{v_x^{(T)}}, h_x) = e(\pi_x^{(T)}, g)$. Hence, the output of the verification algorithm is always the value $v_x^{(T)}$. □

Theorem 6.3. *The proposed VDB scheme is efficient.*

Proof. It is trivial that the computational resources invested by the client in our scheme is independent of q (except for a one-time Setup phase). □

Theorem 6.4. *The proposed VDB scheme is accountable.*

Proof. Given the proofs τ with the counter T, the client firstly compares it with the latest counter T_c he stored. If $T < T_c$, then the client sends the corresponding signature σ on T_c to the judge as the proof. Otherwise, he sends τ to the judge as the proof since the verification of τ will fail if the server has tampered with the database. □

6.2 Comparison

In this section, we compare the proposed scheme with Benabbas-Gennaro-Vahlis scheme and Catalano-Fiore scheme.

Firstly, all of the three schemes require a one-time expensive computational effort in the Setup phase. Secondly, our proposed scheme is secure against FAU attack and can support the public verifiability simultaneously (this is different from the other two schemes). Besides, our scheme is efficient since the computational resources invested by the client is independent on the size of the database. Trivially, most of the expensive computational overhead are outsourced to the server (this is same in all three schemes). Finally, the server invests all of the storage resources in order to store and update the database. That is, the client does not require to store any data locally.

Table 1 presents the comparison among the three schemes. We denote by M a multiplication in $\mathbb{G}_1$ (or $\mathbb{G}_2$), E an exponentiation in $\mathbb{G}_1$, I an inverse in $\mathbb{G}_1$, P a computation

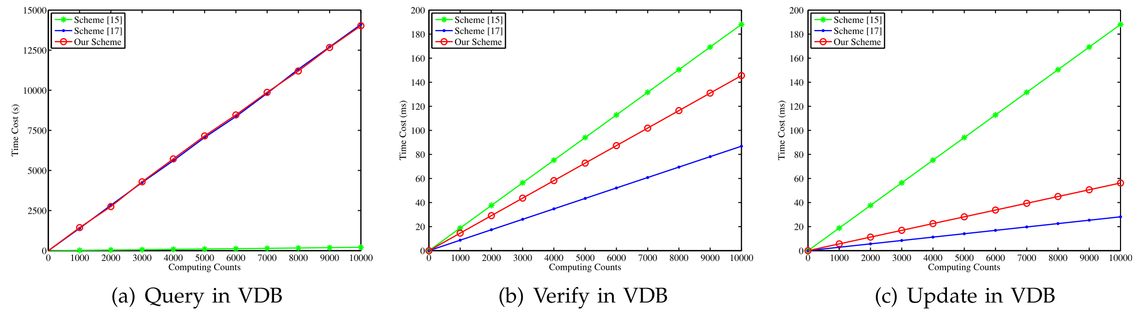


Fig. 2. Efficiency comparison.

of the pairing, and F an operation on a pseudo-random function. We omit other operations such as addition in $\mathbb{G}_1$ for all three schemes.

We argue that the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in Benabbas-Gennaro-Vahlis 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. As a result, our scheme is more efficient than Benabbas-Gennaro-Vahlis's scheme since our scheme uses bilinear groups of prime order.

6.3 Performance Evaluation

In this section, we provide a thorough experimental evaluation of the proposed VDB scheme. Our experiments are simulated with the pairing-based cryptography (PBC) library on a LINUX machine with Intel Core i5-3470 processors running at 3.20 GHz and 4 G memory. Throughout this experiment, to precisely evaluate the computation complexity at both client and server sides, we simulate both entities on this LINUX machine.

We provide the time costs simulation for schemes [15], [17] and our scheme in Fig. 2 when $q = 500$. The time cost of query, verify and update algorithm for all three schemes are shown in Figs. 2a, 2b and 2c, respectively. The simulation results of Fig. 2a reveal that the growth rate of our scheme is the same as that of scheme [17], while relatively higher than that of scheme [15]. Therefore, both the scheme [17] and ours require more overhead than that of scheme [15]. 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 practical in cloud outsourcing environment. On the other hand, the simulation results in Figs. 2b and 2c indicate that our scheme is much more efficient than scheme [15] in both verify and update algorithms which are performed by client. Besides, since the scheme [17] suffers from the FAU attack and the scheme [15] only provides private verifiability, our scheme is most suitable for real-world applications.

In Fig. 3, we provide the efficiency comparison for client side (i.e., the total computational overhead for verify and update algorithms) among three schemes. We present the comparison for three schemes that support public and private verifiability in Figs. 3a and 3b, respectively. Trivially, the public verification requires more overhead than the private one in our scheme. Also, it is obvious that our scheme is superior to scheme [15] for client side overhead.

Fig. 4 shows the efficiency comparison for server side with the increasing of data size q. Trivially, the computational overhead of server side in our scheme is the same as that of scheme [17], while much higher than that of scheme [15]. However, our scheme is still efficient for real-word applications. For example, given a very large database that has 100,000 data records and each record could be arbitrary payload sizes, the time cost of the server in our scheme is only less than 5 minutes.

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 either does not support the public verifiability or suffer from the forward automatic update attack. In this paper, we propose a new framework for verifiable database with efficient updates from vector commitment, which is not only public verifiable but also secure under the FAU attack. Besides, we prove that our construction can achieve the desired security properties.

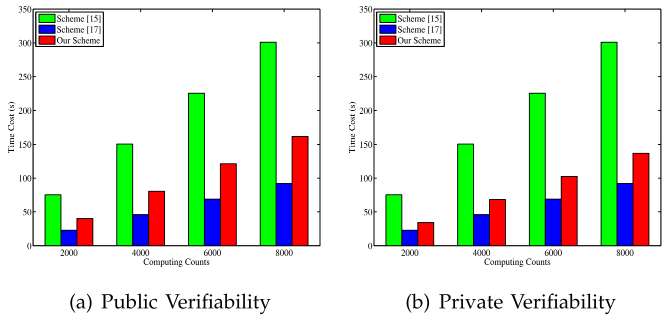


Fig. 3. Client efficiency comparison.

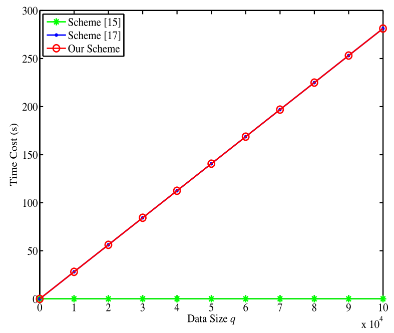


Fig. 4. Server efficiency comparison.

ACKNOWLEDGMENTS

The authors are grateful to the anonymous referees for their invaluable suggestions. This work was supported by the National Natural Science Foundation of China (Nos. 61272455, 61202450 and 61100224), China 111 Project (No. B08038), 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 was supported by US National Science Foundation under grant (CNS-1217889).

REFERENCES

[1] M. Abadi, J. Feigenbaum, and J. Kilian, "On hiding information from an oracle," in *Proc. 19th Annu. ACM Symp. Theory Comput.*, 1987, pp. 195–203.

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

[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. Cryptography 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. Int. Conf. Inf. Commun. Syst.*, 2003, pp. 301–312.

[8] 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.

[9] 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.

[10] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *J. Comput. Syst. Sci.*, vol. 47, pp. 549–595, 1993.

[11] 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.

[12] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, "Locally random reductions: Improvements and applications," *J. Cryptol.*, vol. 10, no. 1, pp. 17–36, 1997.

[13] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1998, pp. 221–232.

[14] K. Bicakci and N. Baykal, "Server assisted signatures revisited," in *Proc. Cryptographers' Track RSA Conf.*, 2004, pp. 1991–1992.

[15] 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.

[16] 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 Res. Adv. Appl.*, 2010, pp. 24–35.

[17] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Proc. PKC*, 2013, pp. 55–72.

[18] D. Catalano, D. Fiore, and M. Messina, "Zero-knowledge sets with short proofs," in *Proc. Adv. Cryptol.-EUROCRYPT*, 2008, pp. 433–450.

[19] 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.

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

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

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

[23] B. Carbunar and M. Tripunitara, "Fair payments for outsourced computations," in *Proc. 7th Annu. IEEE Commun. Soc. Conf. Sens., Mesh Ad Hoc Commun. Netw.*, 2010, pp. 529–537.

[24] X. Chen, W. Susilo, F. Zhang, H. Tian, and J. Li, "Identity-based trapdoor mercurial commitment and applications," *Theor. Comput. Sci.*, vol. 412, no. 39, pp. 5498–5512, 2011.

[25] 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.

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

[27] 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*, 2014, pp. 148–162.

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

[29] M. Fischlin and R. Fischlin, "Efficient non-malleable commitment schemes," in *Proc. Adv. Cryptol.-Crypto*, 2000, pp. 413–431.

[30] 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.

[31] M. Girault and D. Lefranc, "Server-aided verification: Theory and practice," in *Proc. 11th Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2005, pp. 605–623.

[32] 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.

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

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

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

[36] 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.

[37] M. Green, S. Hohenberger, and B. Waters. (2011). Outsourcing the decryption of ABE Ciphertexts. *Proc. 20th USENIX Conf. Security*, p. 34 [Online]. Available: http://static.usenix.org/events/sec11/tech/full-papers/Green.pdf

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

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

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

[41] J. Lai, R. H. Deng, H. Pang, and J. Weng, "Verifiable computation on outsourced encrypted data," in *Proc. Eur. Symp. Res. Comput. Security*, 2014, pp. 273–291.

[42] B. Libert and M. Yung, "Concise mercurial vector commitments and independent zero-knowledge sets with short proofs," in *Proc. 7th Int. Conf. Theory Cryptography*, 2010, pp. 499–517.

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

[44] 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.

[45] S. Micali, M. Rabin, and J. Kilian, "Zero-knowledge sets," in *Proc. 44th IEEE Symp. Found. Comput. Sci.*, 2003, pp. 80–91.

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

[47] P. Q. Nguyen, I. E. Shparlinski, and J. Stern, "Distribution of modular sums and the security of server aided exponentiation," in *Proc. Workshop Comp. Number Theory Cryptol.*, 1999, pp. 1–16.

[48] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Proc. Int. Conf. Topics Cryptol.*, 2005, pp. 275–292.

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

[50] 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.

[51] C. P. Schnorr, "Efficient signature generation for smart cards," *J. Cryptol.*, vol. 4, no. 3, pp. 239–252, 1991.

[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

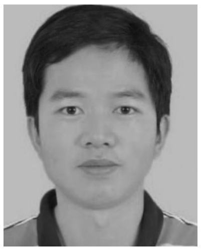
[53] V. Vu, S. Setty, A. J. Blumberg, and M. Walfish, "A hybrid architecture for interactive verifiable computation," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 223–237.

[54] 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.

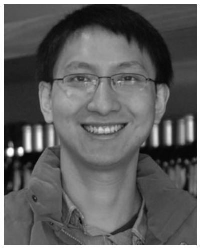
[55] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1172–1181, Jun. 2013. (A preliminary version of this paper is presented at ICDCS, pp. 820-828, 2011.)

[56] L. Zhang and R. Safavi-Naini, "Verifiable delegation of computations with storage-verification trade-off," in *Proc. 19th Eur. Symp. Res. Comput. Security*, 2014, pp. 112–129.

Xiaofeng Chen received the BS and MS degrees in mathematics from Northwest University, China. He received the PhD degree in cryptography from Xidian University in 2003. He is currently a professor at Xidian University. His research interests include applied cryptography and cloud computing security. He has published more than 100 research papers in refereed international conferences and journals. His work has been cited more than 1,600 times at Google Scholar. He has served as the program/general chair or a program committee member in more than 30 international conferences.

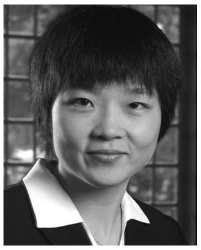


Jin Li received the BS degree in mathematics from Southwest University in 2002. He received the PhD degree in information security from Sun Yat-sen University in 2007. He is currently a professor at Guangzhou University. He has been selected as one of science and technology new star in Guangdong province.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



Xinyi Huang received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 60 research papers in refereed international conferences and journals. His work has been cited more than 1,000 times at Google Scholar. He is in the editorial board of *International Journal of Information Security* (IJIS, Springer) and has served as the program/general chair or a program committee member in more than 40 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 at 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 in 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 at Virginia Polytechnic Institute and State University.