J Supercomput
DOI 10.1007/s11227-013-1039-z

Verifiable computation with access control in cloud computing

Lingling Xu · Shaohua Tang

Abstract With the tremendous growth of cloud computing, verifiable computation has been firstly formalized by Gennaro et al. and then studied widely to provide integrity guarantees in the outsourced computation. However, existing verifiable computation protocols either work in the secret key setting or in the public key setting, namely, work either for single client or for all clients, which rules out some practical applications with access control policies. In this paper, we introduce and formalize the notion of *verifiable computation with access control* (AC-VC), in which only the computationally weak clients with necessary access control permissions can be allowed by a trusted source to apply the outsourced computation of a function to a server. We present a formal security definition and a proved secure black-box construction for AC-VC. This construction is built based on any verifiable computation in the secret key model and ciphertext-policy attribute-based encryption (CP-ABE). The access control policies that our AC-VC can realize depend on that realized in the based CP-ABE.

Keywords Cloud computing · Verifiable computation · Access control · Attribute-based encryption

1 Introduction

The ubiquitous availability of high-capacity networks, low-cost computers, and storage devices as well as utility computing have led to a tremendous growth in cloud

L. Xu · S. Tang (✉)
School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
e-mail: csshtang@gmail.com

L. Xu
e-mail: csllxu@scut.edu.cn

Published online: 29 October 2013

computing. Cloud computing enables computationally weak clients to outsource expensive computational tasks to the cloud, where massive computational power can be easily utilized in a pay-per-use manner. In cloud computing, since the cloud may have the financial incentive to run an extremely fast but incorrect computation, the clients cannot trust the cloud. Thus, the fundamental security issue in this setting is how the clients verify the results computed by the cloud, which makes verifiable computation (VC) a must for this setting.

In the last few years, a considerable amount of research was devoted to verifiable computation which considers a scenario where a computationally weak client wishes to outsource the computation of a function F on an input x to a computationally strong but untrusted server. Verifiable computation has been extensively studied in various settings mainly including the secret-key setting and public-key setting. In a verifiable computation in the secret-key setting (SK-VC) [5, 10, 12, 15], a client who wishes to outsource computation of a function F is required to first run an expensive pre-processing phase to generate a secret key SK_F and a public key PK_F with respect to F. This large initial cost is then amortized over multiple executions of the protocol on different inputs x_i, and the client needs the secret key SK_F in order to initiate each such execution. Namely, in the online phase, the client must use the secret key SK_F to interact with the server to obtain $F(x_i)$ for an input x_i. So a secret-key verifiable computation is for single client. In contrast, in a verifiable computation in the public-key setting [4, 11, 14, 27, 28], a trusted source runs the expensive pre-processing phase once to generate a public key PK_F related to F, such that any client can use PK_F to outsource computation of F to the cloud and verify the results. That is, a verifiable computation in the public-key setting is for multiple clients.

While the recent solutions consider and solve the verifiable computation problem in various settings, there are a number of desirable features that they fail to achieve. Take outsourced operations on medical records as an example, a hospital stores a database of medical records on a cloud, and due to the sensitivity of the records, the hospital requests that only some doctors, who have designated, can outsource the computation of some statistical function F on some records to the cloud. More generally, we will consider the following scenario where a trusted source has developed a cloud application such as the computation of a function F and outsourced it to an untrusted server. The source defines some access control permissions and wants to make the application only available to those clients satisfying the access permissions. In a nutshell, a protocol is required such that:

- Only those clients satisfying the access control permissions can access the application, and those clients who does not satisfy the access conditions cannot access the application even if they collude with each other;
- The untrusted server cannot convince a client, who satisfies the access permissions that an incorrectly computed output for F is correct;
- The time costs for an honest client during the interaction with the server must be less than that to compute the function.

To realize a protocol as described above, the traditional server-based access control methods are no longer applicable since the cloud server is untrusted. Thus, it is desirable to enforce the access permissions on the cloud application in the hands of

the source. A simply method is that the source can give each client, who satisfies the access permissions, a secret key used to access the cloud application for this function F. However, this method is not efficient enough especially when the number of clients is large since the source needs to interact with each client satisfying the access permissions for F. And for a different outsourced function F', the source needs to repeat the similar works. According to this problem, we aim to present a protocol in which the source can achieve the access control on the cloud applications more efficiently and securely.

1.1 Our contributions

We present a formal notion of *verifiable computation with access control*: this is a protocol among these polynomial-time parties, a source, some clients and a server as well as an issuer, to collaborate on the access control of outsourced computation of a function F. Concretely, the source, who is assumed to be a trusted party, can enforce some access structure ω on the verifiable computation of F by initialization such that only the clients, who possess the attributes satisfying ω, can outsource the computation of F on any dynamically-chosen inputs to the server.

The assigning of attributes to clients is done by a separate entity called the issuer, external to the function F and ω. Specifically, a verifiable computation with access control is as follows:

- The issuer first generates the system-wide parameters. Then for each client, the issuer issues a certificate for its attributes.
- Taking the system-wide parameters, a function F and an access structure ω as inputs, the source computes some public information $PK_{F,\omega}$. This stage can take time comparable to computing the function from scratch.
- When a client wants the server to compute $F(x)$, it takes its certificate, prepares some public information σ_x and private information τ_x about x. Then the client sends σ_x to the server.
- Once the server obtains the public information σ_x associated with x, it computes an output σ_y which encodes the value $F(x)$ by using $PK_{F,\omega}$ and returns it to the client. If and only if the client's attributes satisfy the access structure ω, it can recover the value $F(x)$ from σ_y and verify its correctness.

For the verifiable computation with access control, we also present a formal security definition. After that, we present a provably secure black-box construction for AC-VC by combining any secret-key verifiable computation (SK-VC) (formally defined in [12]) and ciphertext-policy attribute-based encryption (CP-ABE) [7, 31]. The computational assumptions underlying the security of our construction are the security of based SK-VC and the full security of CP-ABE.

Theorem 1 (Informal) *For a function F and an access control policy ω, assuming that there exist a secure secret-key verifiable computation for the function F and a fully secure ciphertext-policy attribute-based encryption with the access control policy ω, then there is a verifiable computation with access control for F and ω.*

The main idea of our construction is simple. In a secret-key verifiable computation [12], the client must use the secret key SK_F, which is preprocessed to outsource the computation of F for any input x to the server. Inspired by this point, in our construction, we let the source first generate (SK_F, PK_F) by performing the preprocessing of SK-VC, and then encrypt SK_F with an access control condition ω by using attribute-based encryption. Thus, when a client wants to outsource the computation of $F(x)$ for any input x, it has to first decrypt SK_F using its certificate issued by the issuer for its entitled attributes. Informally, on the one hand, due to the security of based attribute-based encryption, any client whose entitled attributes do not satisfy the access condition ω cannot decrypt SK_F, let alone outsource any input to the server. Moreover, colluding users cannot pool their certificates, meaning that they cannot collude with each other by using their certificates to access the outsourced computation if none of them has the entitled attributes satisfying ω. On the other hand, in terms of the security of the based SK-VC, the server cannot cheat a client, who has attributes satisfying ω, with incorrect results.

In our construction, for given F and ω, the source needs only to perform an initialization, and afterwards neither interacts with any client nor the server. The runtime of a client in the scheme are basically equal to that of a client in the based SK-VC plus an amortized cost of attribute-based encryption over multiple executions of this client on different inputs. So if the based SK-VC and CP-ABE are both efficient, our construction is also efficient.

From the construction, we can see that the access control policies that can be realized in our AC-VC are exactly the same as that realized in the based CP-ABE.

1.2 Related work

The notion of verifiable computation was first formalized by Gennaro et al. [12]. They presented a secret-key verifiable computation scheme for any function by combining Yao's Garbled Circuit [32, 33] with a fully homomorphic encryption system [13]. Their definition uses an amortized notion of complexity for the client, that is, the client can perform some expensive pre-processing in the off-line stage, but it is required to run very efficiently in the online stage. This scheme also provides input and output privacy. Chung et al. [10] showed how to remove the need for the public-key (though still making use of FHE) and suggested an interactive approach to delegate the off-line phase using universal arguments. Goldwasser et al. [15] presented a verifiable computation scheme by using designated verifier CS proofs. As for some concrete functions such as high degree polynomial functions, Benabbas et al. [5] gave a practical verifiable computation scheme. Kamara's scheme [20] gives a general-purpose server-aided multiparty protocols that are more efficient (in terms of computation and communication) for the parties than the alternative of running a standard multiparty computation.

The above verifiable computation schemes are constructed in the secret-key setting, namely, a preprocessing stage is first performed to generate a pair of keys (sk_F, pk_F) for a function F, and then in the online stage the client uses sk_F to outsource the computation of F on any input to the server. Therefore, these constructions only work for a single client.

In contrast, verifiable computation scheme in the public-key setting were presented [4, 11, 14, 27, 28], which work for multiple clients. In some of these schemes, a trusted source first performs a pre-processing stage to generate some public information PK_F for a function F such that any client can uses the information to outsource computation of F on an input to the server. Whereas in the other schemes such as [3, 15, 19, 25] which are constructed based on Probabilistic Checkable Proofs (PCP) [2], scheme [1] which converts the secrecy property of multiparty computation protocols into soundness for a VC scheme via the use of message authentication codes and scheme [30], which is for some concrete functions, a preprocessing phase is not required and any client can directly outsource the computation of F to the server.

Besides the outsource of computation in the cloud, data outsourcing has also attracted a lot of attention. To assure the increased privacy of outsoured data, many data outsourcing systems require flexible access control approaches. In this case, CP-ABE is regarded as one of the most suitable technologies for data access control, because it gives the data owner more direct control on access policies. Similarly to data outsourcing, we will use CP-ABE to realize the access control in verifiable computation. However, we will also present a formal security model for AC-VC which cannot be directly borrowed from the existing work on data outsourcing, and prove that our construction for AC-VC is secure under the security model. As to CP-ABE, Bethencourt et al. [7] proposed the first CP-ABE which supports monotonic access structures. Lewko et al. in [21] proposed the first fully secure CP-ABE in the standard model, which supports access policies with linear secret sharing scheme (LSSS) [6]. As to the attribute-based access control in data outsourcing systems, many solutions have also been presented in [9, 16–18, 23, 26, 34, 35].

1.3 Organization

The rest of this paper is organized as follows. In Sect. 2, we will present the functionality and security definition of AC-VC. The building blocks including SK-VC and CP-ABE will be introduced in Sect. 3. Subsequently, we will present our construction for AC-VC based on SK-VC and CP-ABE in Sect. 4. In Sect. 5, the security analysis of our construction for AC-VC will be given. We give some discussions about the multiauthority and attribute revocation problem in our AC-VC construction in Sect. 6. Finally, Sect. 7 concludes the paper.

2 Verifiable computation with access control: definition

Throughout this paper, we use "$A \models \omega$" to denote that "A satisfies ω" where A is an attribute set and ω is an access structure. If A is a randomized algorithm, then $y \leftarrow \mathsf{A}(x)$ denotes the assignment to y of the output of A on input x. Unless noted, all algorithms are probabilistic polynomial-time (PPT) and we implicitly assume they take an extra parameter κ in their input, where κ is a security parameter. A function $\nu : \mathbb{N} \rightarrow [0, 1]$ is negligible if for all $c \in \mathbb{N}$ there exists a $\kappa_c \in \mathbb{N}$ such that $\nu(k) < k^{-c}$ for all $k > \kappa_c$.

A verifiable computation with access control (AC-VC) is run among the following parties: an issuer providing access certificates for some attributes; a trusted source

developing a cloud application to compute a function F; some computationally weak clients; and a cloud server who provides the computation of F for clients. A verifiable computation with access control consists of six algorithms ISetup, CertIssue, Initialization, Query, Compute, Verify described as follows.

- ISetup: The issuer runs the randomized ISetup algorithm to generate a public key pk_I and the corresponding secret key sk_I for the security parameter κ. It publishes the public key as the system-wide parameter.

$$(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$$

– CertIssue: A client obtains an access certificate for an attribute set A by engaging in the protocol with the issuer.

$$cert_A \leftarrow \mathsf{CertIssue}(pk_I, sk_I, A)$$

– Initialization: Taking κ, pk_I, a function F and an access control structure ω as inputs, the source generates some public information $PK_{F,\omega}$ related to F and ω.

$$PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$$

– Query: For an input x, $PK_{F,\omega}$ and pk_I, the client computes some public information σ_x and private information τ_x by using a certificate $cert_A$ corresponding to the attribute set A, and sends σ_x to the server.

$$(\sigma_x, \tau_x) \leftarrow \mathsf{Query}(PK_{F,\omega}, pk_I, cert_A, x)$$

– Compute: The server computes an encoded version σ_y of the function's output $y = F(x)$ and returns σ_y to the client.

$$\sigma_y \leftarrow \mathsf{Compute}(PK_{F,\omega}, \sigma_x)$$

- Verify: Taking τ_x and σ_y as inputs, the client recovers $y = F(x)$.

$$y \cup \perp \leftarrow \mathsf{Verify}(\tau_x, \sigma_y)$$

According to the security requirements of an AC-VC scheme, in the following, we will define *Correctness*, *Verifiability*, *Access Security*, and *Efficiency* for it based on the definition in [12].

Definition 1 (Correctness) A verifiable computation with access control is correct if for any choice of F and access control structure ω, the issuer setup algorithm generates keys $(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$, the certificate issue algorithm generates the certificate $cert_A \leftarrow \mathsf{CertIssue}(pk_I, sk_I, A)$ corresponding to any attribute set A and the source initialization algorithm generates keys $PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$ such that, $\forall x \in Domain(F)$ and attribute set A, if $A \models \omega$, $\sigma_x \leftarrow \mathsf{Query}(PK_{F,\omega}, pk_I, cert_A, x)$, $\sigma_y \leftarrow \mathsf{Compute}(PK_{F,\omega}, \sigma_x)$, then $y = F(x) \leftarrow \mathsf{Verify}(\tau_x, \sigma_y)$.

As desired in a verifiable computation with access control, any client who has the attributes satisfying ω can outsource the computation of F to the server on any input and verify the correctness of the results. In other words, for a given function F and access structure ω, after a client, whose attributes satisfy ω, makes query for an input x to the server, the server cannot return an incorrectly computed results so as to convince the client to output y in the verification algorithm such that $F(x) = y$. We first formally define an experiment as follows where $poly()$ denotes a polynomial function of its input.

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Verif}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$:
$(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$
$PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$
for $i = 1, 2, \ldots, l = poly(\kappa)$
$(A_i, x_i) \leftarrow \mathcal{A}(PK_{F,\omega}, x_1, A_1, \sigma_{x_1}, \ldots, x_{i-1}, A_{i-1}, \sigma_{i-1})$
if $A_i \models \omega$, $(\sigma_{x_i}, \tau_i) \leftarrow \mathsf{Query}(PK_{F,\omega}, pk_I, cert_{A_i}, x_i)$
else $\perp \leftarrow \mathsf{Query}(PK_{F,\omega}, pk_I, cert_{A_i}, x_i)$
$(i, \hat{\sigma_y}) \leftarrow \mathcal{A}(PK_{F,\omega}, x_1, A_1, \sigma_{x_1}, \ldots, x_l, A_l, \sigma_{x_l})$
$\hat{y} \leftarrow \mathsf{Verify}(\tau_x, \hat{\sigma_y})$
if $A_i \models \omega$, $\hat{y} \neq \perp$ and $\hat{y} \neq F(x)$, output '1', else '0'.

In above experiment, (pk_I, sk_I) and $PK_{F,\omega}$ are generated by running ISetup and $\mathsf{Initialization}$ algorithms respectively, where $PK_{F,\omega}$ is related to a function F and an access structure ω. Then, for $i = 1, \ldots, poly(\kappa)$, the adversary makes a polynomial number of oracle queries to the Query algorithm, each time specifying (x_i, A_i). If $A_i \models \omega$, then the adversary is returned σ_{x_i}, and else '$\perp$'. Finally the adversary outputs a tuple $(i, \hat{\sigma_y})$. The adversary succeeds if it produces an output that convinces the verification algorithm to accept on the wrong output value for a given input value.

Definition 2 (Verifiability) We say that a verifiable computation with access control satisfies verifiability for a function F and an access control structure ω, if for any adversary $\mathcal{A}$ running in probabilistic polynomial time, the advantage of $\mathcal{A}$ in the experiment above, which is defined as

$$Adv_{\mathcal{A}}^{\mathrm{Verif}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega) = Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Verif}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega] = 1\right],$$

satisfies

$$Adv_{\mathcal{A}}^{\mathrm{Verif}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega) \leq \mathsf{negli}(\kappa),$$

where $\mathsf{negli}()$ is a negligible function of its input.

Note that we present the definition of *Verifiability* for AC-VC based on the definition of verifiability in [12] without considering the rejection problem, that is, a malicious server that is able to observe the result of the verification procedure (namely, the accept/reject decision) on polynomially many inputs can eventually break the soundness of the protocol. However, we can obtain a definition of stronger verifiability based on that in [5], which does not suffer from the rejection problem. In a nutshell, the stronger verifiability is the same the definition of verifiability above except that

let the server learn the output of the verification procedure on all inputs. We will not describe it in detail here.

Moreover, in a AC-VC scheme, it is desired that only the clients, whose attributes satisfy ω, can outsource the computation of F to the server, and other clients cannot access the application even if they collude with each other. This security that we call *Access Security* is defined formally as follows:

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Access}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$:
$(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$
$PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$
$A_1, A_2, \ldots, A_q \leftarrow \mathcal{A}$
if $A_i \models \omega$,
  $\perp \leftarrow \mathsf{CertIssue}(pk_I, sk_I, A_i)$
else $cert_{A_i} \leftarrow \mathsf{CertIssue}(pk_I, sk_I, A_i)$
$\hat{\sigma_x} \leftarrow \mathcal{A}$
$\hat{\sigma_y} \leftarrow \mathsf{Compute}(PK_{F,\omega}, \sigma_x)$
$\hat{y} \leftarrow \mathcal{A}(\sigma_y)$
if $\hat{y} = F(x)$ for some x, output '1', else '0'.

Definition 3 (Access security) We say that a verifiable computation with access control satisfies access security for a function F and an access control structure ω, if for any adversary $\mathcal{A}$ running in probabilistic polynomial time, the advantage of $\mathcal{A}$ in the experiment above, which is defined as

$$Adv_{\mathcal{A}}^{\mathrm{Access}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega) = Pr\big[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Access}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega] = 1\big],$$

satisfies

$$Adv_{\mathcal{A}}^{\mathrm{Access}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega) \leq \mathsf{negli}(\kappa),$$

where $\mathsf{negli}()$ is a negligible function of its input.

Definition 4 (Efficiency) A verifiable computation with access control is efficient if for a public key pk_I, $PK_{F,\omega}$ related to a function F and ω, a certificate $cert_A$ for an attribute set A, any x and σ_y, $(\sigma_x, \tau_x) \leftarrow \mathsf{Qurey}(x, pk_I, PK_{F,\omega}, cert_A)$ plus the time required for $\mathsf{Verify}(\tau_x, \sigma_y)$ is $o(T)$, where T is the time required to compute $F(x)$.

3 Building blocks

In this section, two building blocks for AC-VC including verifiable computation in the secret-key setting (SK-VC) and ciphertext-policy attribute-based encryption (CP-ABE) will be introduced.

3.1 Verifiable computation in the secret-key setting (SK-VC)

A SK-VC scheme consists of four algorithms: VC.KeyGen, VC.ProbGen, VC.Compute and VC.Verify.

- $(pk, sk) \leftarrow \mathsf{VC.KeyGen}(\kappa)$: Taking a security parameter κ and a function F as inputs, outputs a pair of keys (pk, sk), publishes pk and keeps sk secret.
- $(\sigma_x, \tau_x) \leftarrow \mathsf{VC.ProbGen}(x, sk)$: For an input x, outputs (σ_x, τ_x) by using the secret key sk, publishes σ_x and keeps τ_x secret.
- $\sigma_y \leftarrow \mathsf{VC.Compute}(pk, \sigma_x)$: Take pk and σ_x as inputs, outputs a result σ_y which is an encoded value of $F(x)$.
- $y \cup \perp \leftarrow \mathsf{VC.Verify}(\tau_x, \sigma_y)$: Outputs $y = F(x)$ or '$\perp$' with inputs τ_x and σ_y.

A SK-VC scheme is secure if it satisfies correctness, verifiability and efficiency, which were defined formally in [12]. In the remainder of this paper, in order to distinguish between the definition of verifiability for SK-VC and AC-VC, we denote vc-verifiability as the verifiability for SK-VC defined in [12].

3.2 Ciphertext-policy attribute-based encryption (CP-ABE)

A CP-ABE scheme [7, 21] consists of four algorithms: ABE.Setup, ABE.KeyGen, ABE.Encrypt, and ABE.Decrypt.

- $(pk, sk) \leftarrow \mathsf{ABE.Setup}(\kappa)$: Takes a security parameter κ, outputs a master public/secret key pair (pk, sk).
- $dk_A \leftarrow \mathsf{ABE.KeyGen}(sk, A)$: Takes the master secret key sk and an attribute set $A \subseteq \Omega$ where Ω is the attribute universe, outputs a decryption key dk_A.
- $c \leftarrow \mathsf{ABE.Encrypt}(m, \omega, pk)$: Takes the master public key pk, a message m and an access policy ω, produces a ciphertext c.
- $m \leftarrow \mathsf{ABE.Decrypt}(c, dk_\omega)$: Takes the master public key pk, a decryption key dk_A and a ciphertext c, outputs a message m if A satisfies ω which is associated with c.

The definition of full security of ciphertext-policy attribute-based encryption [7, 21] is given as follows.

Definition 5 (Full security of CP-ABE) A CP-ABE is fully secure if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the game below is negligible in κ where κ is a security parameter.

Setup: The challenger runs $\mathsf{ABE.Setup}(1^\kappa)$ and gives pk to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ may query for the decryption keys of attribute sets $A_1, \ldots, A_{q_1} \subseteq \Omega$.

Challenge: $\mathcal{A}$ submits two equal-length messages $M_0, M_1 \in \{0,1\}^*$ and a challenge access policy ω such that none of the sets $A_1, \ldots, A_{q_1}$ satisfies ω. The challenger flips a random coin $b \in \{0,1\}$ and encrypts M_b with respect to $\mathcal{A}$. The ciphertext C^* is given to $\mathcal{A}$.

Phase 2: Same as Phase 1 with the restriction that none of the additional attribute sets $A_{q_1+1}, \ldots, A_q$ satisfies $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs $b' \in \{0,1\}$. The advantage of $\mathcal{A}$ is defined as $Adv_{\mathcal{A}}^{\text{CP-ABE}} = |Pr[b' = b] - \frac{1}{2}|$.

4 Verifiable computation with access control: construction

In a AC-VC, a trusted source develops a cloud application, here the computation of a function F on any input, and outsources it to a cloud server. It's required that the

source can decide which of its clients can use this application. One possible method may be that the source grants the access control rights to the server and lets the server perform control. However, the server is untrusted and may collude with some clients due to some benefits. Another method is that the source can give each client, who satisfies the access permissions, a secret key used to access the cloud application for a function F. However, the source needs to interact with each of these clients so that this method is not efficient enough especially when the number of these clients is large. So how the source enforce access control on the application efficiently is a key problem in the construction of AC-VC.

The main idea of our construction is described as follows. The participants include an issuer who is responsible for the generation of certificates for each client's attributes, a trusted source, a server and a client. When the source initializes the application, it generates some public information $PK_{F,\omega}$ with respect to the function F and the access control policy ω so that only the clients, whose attributes satisfy the policy, can use the public information to prepare the inputs for the function F and verify the corresponding outputs provided by the server. When a client wants to outsource the computation of $F(x)$, it first uses its certificate, which is issued by the issuer, to generate some public information σ_x and private information τ_x, and sends σ_x to the server. Then the server computes an output σ_y by taking $PK_{F,\omega}$ and σ_x as inputs. Finally, the client outputs y or '$\perp$' by using σ_y, its certificate and τ_x as well.

Note that in our construction, the source only needs to initialize the application, and neither interact with the server nor the client afterward.

Based on any SK-VC which consists of ($\mathsf{VC.KeyGen}$, $\mathsf{VC.ProbGen}$, $\mathsf{VC.Compute}$, $\mathsf{VC.Verify}$) and CP-ABE := ($\mathsf{ABE.Setup}$, $\mathsf{ABE.KeyGen}$, $\mathsf{ABE.Encrypt}$, $\mathsf{ABE.Decrypt}$), we can construct a verifiable computation with access control as follows. We also illustrate it in Fig. 1.

- $(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$: Taking a security parameter κ as input, the issuer runs $\mathsf{ABE.Setup}$ algorithm to generate a pair of keys $(pk_I, sk_I) \leftarrow \mathsf{ABE.Setup}(\kappa)$, publishes pk_I as the system-wide parameter and keeps sk_I secret.
- $cert_A \leftarrow \mathsf{CertIssue}(sk_I, A)$: For any attribute set A, the issuer runs $\mathsf{ABE.KeyGen}$ algorithm to generate a decryption key $dk_A \leftarrow \mathsf{ABE.KeyGen}(sk_I, A)$. Then let $cert_A := dk_A$ be a certificate corresponding to A.
- $PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$: By inputting a function F and an attribute structure ω, the source first generates a pair of keys $(pk_S, sk_S) \leftarrow \mathsf{VC.KeyGen}(\kappa)$, and encrypts the secret key sk_S by using the access structure ω to generate $pk_\omega \leftarrow \mathsf{ABE.Encrypt}(sk_S, \omega, pk_I)$. Thus, only the clients, who have attributes satisfying ω, can decrypt sk_S by using their certificates and then outsource any computation to the server by performing SK-VC afterward. Finally, the source publishes $PK_{F,\omega} := (pk_S, pk_\omega)$.
- $(\sigma_x, \tau_x) \leftarrow \mathsf{Query}(cert_A, PK_{F,\omega}, x)$: Taking $PK_{F,\omega}$ and a certificate $cert_A$ for an attribute set A as inputs where $PK_{F,\omega}$ is parsed as (pk_S, pk_ω), the client first decrypts pk_ω by using its certificate $cert_A$ to obtain $sk^* \leftarrow \mathsf{ABE.Decrypt}(cert_A, pk_\omega)$. Then it outsources the computation on input x to the server by running $(\sigma_x, \tau_x) \leftarrow \mathsf{VC.ProbGen}(x, sk^*)$. Obviously, if $A \models \omega$, then $sk^* = sk_S$. Otherwise, due to the full security of the based CP-ABE, the clients who do not satisfy the access permissions cannot obtain sk_S even if they collude with each other.

THE CONSTRUCTION FOR AC-VC

1. The issuer does as follows:

$\underline{\mathsf{ISetup}(\kappa):}$
$(pk_I, sk_I) \leftarrow \mathsf{ABE.Setup}(\kappa)$,
publishes pk_I, keeps sk_I secret;

$\underline{\mathsf{CertIssue}(sk_I, A):}$
$dk_A \leftarrow \mathsf{ABE.KeyGen}(sk_I, A)$,
$cert_A := dk_A$;

2. The source does as follows:

$\underline{\mathsf{Initialization}(\kappa, pk_I, F, \omega):}$
$(pk_S, sk_S) \leftarrow \mathsf{VC.KeyGen}(\kappa)$,
$pk_\omega \leftarrow \mathsf{ABE.Encrypt}(sk_S, \omega, pk_I)$,
publishes $PK_{F,\omega} := (pk_S, pk_\omega)$;

3. The server and the client interact as follows:

Client		Server
$\underline{\mathsf{Query}(cert_A, PK_{F,\omega}, x):}$		
$sk^* \leftarrow \mathsf{ABE.Decrypt}(cert_A, pk_\omega)$,		
$(\sigma_x, \tau_x) \leftarrow \mathsf{VC.ProbGen}(x, sk^*)$		
	$\xrightarrow{\sigma_x}$	
		$\underline{\mathsf{Compute}(cert_A, PK_{F,\omega}, x):}$
		$\sigma_y \leftarrow \mathsf{VC.Compute}(pk_S, \sigma_x)$
	$\xleftarrow{\sigma_y}$	
$\underline{\mathsf{Verify}(\tau_x, \sigma_y):}$		
$y \cup \perp \leftarrow \mathsf{VC.Verify}(\tau_x, \sigma_y)$		

Fig. 1 The construction for AC-VC based on SK-VC and CP-ABE

- $\sigma_y \leftarrow \mathsf{Compute}(PK_{F,\omega}, \sigma_x)$: Parse $PK_{F,\omega}$ as (pk_S, pk_ω). The server computes an output σ_y for the client by running $\mathsf{VC.Compute}(pk_S, \sigma_x)$ and returns σ_y to the client.
- $y \cup \perp \leftarrow \mathsf{Verify}(\tau_x, \sigma_y)$: Taking τ_x and σ_y as inputs, the client computes an output 'y' or '$\perp$' by running $\mathsf{VC.Verify}(\tau_x, \sigma_y)$.

Remark 1 The client can decrypt the secret key sk^* before its first query on an input to the server. Namely, the algorithm ABE.Decrypt is only performed once by the client. Thus, the Query algorithm above can also be split into two new algorithms, one can be off-line that the client decrypt the secret key sk^* from pk_ω by using its certificate, and another algorithm is an online query for the client which is the same as $\mathsf{VC.ProbGen}(x, sk^*)$ in SK-VC.

Remark 2 In the scheme, after the client decrypts sk^*, it takes sk^* and x as inputs, the server takes pk_S as input where $PK_{F,\omega} = (pk_S, pk_\omega)$, and they interact with each other exactly in SK-VC.

Remark 3 From the concrete construction above, we can see that our proposed AC-VC can realize the access control policies exactly the same as the based CP-ABE. For example, when instantiated with the CP-ABE scheme by Bethencourt et al. [7] and Lewko et al. [21], our AC-VC can achieve monotonic access policies and access policies with linear secret sharing scheme (LSSS) [6], respectively.

5 Security analysis of our construction for AC-VC

In this section, we will analyze the security of our AC-VC construction in detail, including *Correctness*, *Verifiability*, *Access Security*, and *Efficiency*.

Correctness When both the client and the server are honest, the output of the scheme will be $y = F(x)$. Concretely, if a client whose attribute set A satisfies the access structure ω, then it can decrypt sk_S by performing $sk_S \leftarrow \mathsf{ABE.Decrypt}(cert_A, pk_\omega)$ in Query. Then, by inputting x, sk_S and pk_S, the client and the server interact with each other by running Query, Compute, and Verify just as in SK-VC. So the correctness of this scheme depends on that of the based SK-VC.

In the following, we will prove that our construction for AC-VC satisfies *Verifiability* and *Access Security*. Informally, in our construction, only the clients whose attributes satisfy the access permissions can access the application. Moreover, the untrusted server cannot cheat a client, whose attributes satisfy the access permissions, with incorrect answers.

Theorem 2 *If the based SK-VC scheme is vc-verifiable and CP-ABE scheme is fully secure, then our construction for AC-VC satisfies verifiability.*

Proof In the following, we will show that the success probability of any adversary $\mathcal{A}$ to break the verifiability of our construction for AC-VC is negligible in κ assuming the based SK-VC scheme is vc-verifiable and the based CP-ABE is fully secure.

To complete the proof, we consider two experiments Experiment-0 and Experiment-1.

Experiment-0: the same as the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Verif}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$ defined in Sect. 2. Clearly,

$$Pr\left[Adv_{\mathcal{A}}^{\mathrm{Verif}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega)\right] = Pr[\mathsf{Experiment\text{-}0} = 1]. \tag{1}$$

Experiment-1: the same as Experiment-0 except that let $pk_\omega = \mathsf{Encrypt}_{\mathrm{ABE}}(r, \omega, pk_I)$ where r is a randomly chosen value.

Due to the full security of the based CP-ABE scheme, we can obtain by Lemma 1 below that the experiments Experiment-0 and Experiment-1 are indistinguishable, that is,

$$\left|Pr[\mathsf{Experiment\text{-}0} = 1] - Pr[\mathsf{Experiment\text{-}1} = 1]\right| \leq \mathsf{negli}(\kappa). \tag{2}$$

In addition, we can prove in Lemma 2 below that any adversary cannot succeed in Experiment-1 as we assumed that the based SK-VC is vc-verifiable, that is,

$$\left|Pr[\mathsf{Experiment\text{-}1} = 1]\right| \leq \mathsf{negli}(\kappa). \tag{3}$$

Summing up the differences between the above experiments, we can conclude by (1), (2) and (3), that for any adversary $\mathcal{A}$, the success probability to break our construction for AC-VC is negligible in κ,

$$\begin{aligned}
&\left|Pr[Adv_{\mathcal{A}}^{\text{Verif}}(\mathcal{AC}\text{-}\mathcal{VC}, F, \omega)]\right| \\
&\quad = \left|Pr[\mathsf{Experiment\text{-}0} = 1]\right| \\
&\quad \leq \left|Pr[\mathsf{Experiment\text{-}0} = 1] - Pr[\mathsf{Experiment\text{-}1} = 1]\right| + \left|Pr[\mathsf{Experiment\text{-}1} = 1]\right| \\
&\quad \leq \mathsf{negli}(\kappa)
\end{aligned}$$

□

Lemma 1 $|Pr[\mathsf{Experiment\text{-}0} = 1] - Pr[\mathsf{Experiment\text{-}1} = 1]| \leq \mathsf{negli}(\kappa)$ *if the based CP-ABE is fully secure.*

Proof If there is an adversary $\mathcal{A}$ who can distinguish the two experiments Experiment-0 and Experiment-1, then we can construct an algorithm $\mathcal{A}'$ who can break the full security of the based CP-ABE scheme.

$\mathcal{A}'$ is given the public parameters pk_{ABE} of CP-ABE by a challenger. It is allowed to make polynomial times of queries for the decryption key generation to the challenger. Then it runs VC.KeyGen to generate a pair of keys $(pk, sk) \leftarrow \mathsf{VC.KeyGen}(\kappa)$, chooses randomly a value r and sends (sk, r) along with an attribute set ω to the challenger as its challenge. After the challenger returns a ciphertext C, $\mathcal{A}'$ publishes (pk, C) as the public parameters $PK_{F,\omega}$ for a function F and an attribute set ω, namely, let $PK_{F,\omega} := (pk, C)$.

In the case that $C = \mathsf{ABE.Encrypt}(sk, \omega, pk_{\text{ABE}})$, one can observe that the environment that $\mathcal{A}'$ created for $\mathcal{A}$ is exactly that of Experiment-0. In the case that $C = \mathsf{ABE.Encrypt}(r, \omega, pk_{\text{ABE}})$, then one can easily learn that this environment is exactly that of Experiment-1.

Finally if the experiment outputs '1', then $\mathcal{A}'$ outputs '0' meaning that $\mathcal{A}'$ guesses that C is the ciphertext of sk, and otherwise outputs '1' to guess that C is the ciphertext of r.

Since we assume that the based CP-ABE is fully secure, namely, $Adv_{\mathcal{A}'}^{\text{CP-ABE}} \leq \mathsf{negli}(\kappa)$, according to the security definition of full security, we have

$$\begin{aligned}
&\left|Pr[\mathsf{Experiment\text{-}0} = 1] - Pr[\mathsf{Experiment\text{-}1} = 1]\right| \\
&\quad = \left|Pr\left[\mathcal{A}' \textit{ outputs } 0\right] - Pr\left[\mathcal{A}' \textit{ outputs } 1\right]\right| \\
&\quad = 2 \cdot Adv_{\mathcal{A}'}^{\text{CP-ABE}} \\
&\quad \leq \mathsf{negli}(\kappa)
\end{aligned}$$

□

Lemma 2 $|Pr[\mathsf{Experiment\text{-}1} = 1]| \leq \mathsf{negli}(\kappa)$ *if the based SK-VC is vc-verifiable.*

Proof If there is an adversary $\mathcal{A}$ who can succeed in Experiment-1, then an algorithm $\mathcal{A}'$ can be constructed to break the vc-verifiability of the based SK-VC.

$\mathcal{A}'$ is given the public parameter pk_S for a function F by the challenger of SK-VC. It obtains the public key pk_I for the based CP-ABE from the issuer. Then for a

function F and an attribute set ω, it chooses a random value r and sends (pk_S, pk_ω) as the public parameter $PK_{F,\omega}$ to $\mathcal{A}$ where $pk_\omega = \mathsf{ABE.Encrypt}(r, \omega, pk_I)$. Later, when $\mathcal{A}$ makes queries to the oracle Query for (A_i, x_i), if $A_i \models \omega$, then $\mathcal{A}'$ uses its oracle, i.e., sends x_i to the challenger and obtains σ_{x_i}. Then $\mathcal{A}'$ returns σ_{x_i} to $\mathcal{A}$ as the answer. Finally, when $\mathcal{A}$ makes query for (x, A^*), if $A^* \models \omega$, $\mathcal{A}'$ sends x to the challenger as its challenge. After receiving σ_x from the challenger, $\mathcal{A}'$ sends σ_x to $\mathcal{A}$. When $\mathcal{A}$ returns σ_y to $\mathcal{A}'$, $\mathcal{A}'$ sends σ_y to the challenger. If $\mathcal{A}$ succeeds, then $y = F(x)$. Since the Compute and Verify algorithms are the same as VC.Compute and VC.Verify in the based SK-VC, respectively, if $\mathcal{A}$ can succeed in Experiment-1 with nonnegligible probability, then $\mathcal{A}'$ can break the vc-verifiability of based SK-VC. That is,

$$Pr[\mathsf{Experiment\text{-}1} = 1] = Pr\big[\mathcal{A}' \textit{ succeeds}\big] \leq \mathsf{negli}(\kappa). \qquad \square$$

Theorem 3 *If the based SK-VC scheme is vc-verifiable and CP-ABE scheme is fully secure, then our construction for AC-VC satisfies access security.*

Proof In a SK-VC scheme, only the clients with secret key sk_S generated by the source can outsource computation of F to the server. In our construction, to outsource an input, the client has to decrypt sk_S and then conduct the interaction with the server just as in SK-VC. We assume that the based SK-VC is secure such that if and only if a client learns the secret key sk_S, it can outsource the computation of F for any input to the server. Thus, in fact the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{Access}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$ is equivalent to the following experiment $\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$:

$\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$:
$(pk_I, sk_I) \leftarrow \mathsf{ISetup}(\kappa)$
$PK_{F,\omega} \leftarrow \mathsf{Initialization}(\kappa, pk_I, F, \omega)$
where $PK_{F,\omega} = (pk_S, pk_\omega)$
$\omega_1, \omega_2, \ldots, \omega_q \leftarrow \mathcal{A}$
if $\omega_i \models \omega$,
  $\perp \leftarrow \mathsf{CertIssue}(pk_I, sk_I, \omega_i)$
else $cert_{\omega_i} \leftarrow \mathsf{CertIssue}(pk_I, sk_I, \omega_i)$
$sk \leftarrow \mathcal{A}$
$(\sigma_x, \tau_x) \leftarrow \mathsf{VC.ProbGen}(x, sk)$
$\sigma_y \leftarrow \mathsf{VC.Compute}(\sigma_x, PK_S)$
$y \leftarrow \mathsf{VC.Verify}(\tau_x, \sigma_y)$
if $y = F(x)$, output '1', else '0'.

In the above experiment, if $y = F(x)$, then $sk = \mathsf{ABE.Decrypt}(pk_S, cert_\omega) = sk_S$. Let $\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}$ and $\mathsf{Exp}^{\mathrm{Access}}_{\mathcal{A}}$ denote the outputs of $\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$ and $\mathsf{Exp}^{\mathrm{Access}}_{\mathcal{A}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$ respectively. Due to the vc-verifiability of the based SK-VC, we have

$$Pr\big[\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}} = 1\big] = Pr\big[\mathsf{Exp}^{\mathrm{Access}}_{\mathcal{A}} = 1\big].$$

Obviously, due to the full security of the based CP-ABE scheme, the success probability in the experiment $\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}[\mathcal{AC}\text{-}\mathcal{VC}, F, \omega]$ is negligible in κ. By Lemma 3,

we have

$$Pr\left[\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}} = 1\right] \leq \mathsf{negli}(\kappa). \tag{4}$$

Therefore, we can conclude that from above

$$Pr\left[\mathsf{Exp}^{\mathrm{Access}}_{\mathcal{A}} = 1\right] \leq \mathsf{negli}(\kappa). \qquad \square$$

Lemma 3 *$Pr[\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}} = 1] \leq \mathsf{negli}(\kappa)$ if the based CP-ABE is fully secure.*

Proof The concrete proof is shown as follows. If there is an adversary $\mathcal{A}$ who can win the experiment $\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}}$, then we can construct an algorithm $\mathcal{B}$ who can break the full security of CP-ABE.

$\mathcal{B}$ is given the public key pk_I of CP-ABE. Then $\mathcal{B}$ generates $PK_{F,\omega}$ and sk_S by running $\mathsf{Initialization}(\kappa, pk_I)$ for a function F and an attribute set ω, where $PK_{F,\omega} = (pk_S, pk_\omega)$, $sk_S = \mathsf{Decrypt}_{\mathrm{ABE}}(pk_\omega, cert_\omega)$, and transfers $(pk_I, PK_{F,\omega})$ to $\mathcal{A}$. When $\mathcal{A}$ makes queries for some attribute A_i where A_i does not satisfy ω, $\mathcal{B}$ uses its own oracle to answer $\mathcal{A}$. $\mathcal{B}$ selects a random value t, and sends (t, sk_s) to the challenger. After receiving the answer C, $\mathcal{B}$ sets C as the answer. Finally, if $\mathcal{A}$'s output is sk_S, then $\mathcal{B}$ outputs '1', if $\mathcal{A}$'s output is not equal to sk, then $\mathcal{B}$ outputs '0'. Therefore,

$$Pr\left[Adv^{\text{CP-ABE}}_{\mathcal{B}} = 1\right] \geq Pr\left[\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}} = 1\right]. \tag{5}$$

So if the based CP-ABE is fully secure, namely, $Pr[Adv^{\text{CP-ABE}}_{\mathcal{B}} = 1] \leq \mathsf{negli}(\kappa)$, then for any adversary $\mathcal{A}$, we have

$$Pr\left[\mathsf{Exp}'^{\mathrm{Access}}_{\mathcal{A}} = 1\right] \leq \mathsf{negli}(\kappa). \qquad \square$$

Efficiency analysis To be a true verifiable computation with access control, AC-VC must satisfy the efficiency requirements from Sect. 2; specifically, the runtime required for Query and Verify must be much less than computing the function F itself. For any input, since Query simply requires the computations in SK-VC and an additional ABE.Decrypt, and all of the inputs to ABE.Decrypt are independent of F, the runtime of Query is independent of the function. Moreover, the runtime of Verify is just the same as that of VC.Verify in the based SK-VC, it is independent of F as well.

More concretely, the runtime of the client, the source and the server in AC-VC are respectively as follows. Since the algorithm ABE.Decrypt is performed by the client only once (in its first query to the server) and its cost is amortized over all the inputs by the client, the total runtime of the client are that of a client in the based SK-VC plus that required for ABE.Decrypt in an amortized sense. The runtime of the trusted source is equal to that of a source in the based SK-VC plus the amortized runtime for the algorithm ABE.Encrypt over all executions of the scheme by all clients. As for the runtime of the server, it is the same as that of a server in the based SK-VC. In Table 1, we illustrate the efficiency analysis of our construction for AC-VC.

Next, we will instantiate with a concrete SK-VC scheme for high degree polynomial functions [5] and a concrete CP-ABE scheme [7] under the framework of our AC-VC to illustrate the efficiency of the AC-VC.

Table 1 Efficiency of the general AC-VC

	Client	Source	Server
Runtime	$T_{\text{client}}^{\text{SK-VC}} + T_{\text{Dec}}$	$T_{\text{source}}^{\text{SK-VC}} + T_{\text{Enc}}$	$T_{\text{server}}^{\text{SK-VC}}$

Notation for Table 1:

T_{Enc}: runtime for ABE.Encrypt

$T_{\text{source}}^{\text{SK-VC}}$: runtime of a source in SK-VC

$T_{\text{client}}^{\text{SK-VC}}$: runtime of a client in SK-VC

T_{Dec}: runtime for ABE.Decrypt

$T_{\text{server}}^{\text{SK-VC}}$: runtime of a server in SK-VC

Table 2 Efficiency of the concrete AC-VC comparing with the based SK-VC

Scheme	Runtime of client	Runtime of server	Runtime of source
SK-VC [5]	$O(n \cdot \log d)$	$O((d+1)^n)$	$O((d+1)^n)$
Concrete AC-VC	$O(n \cdot \log d + l)$	$O((d+1)^n)$	$O((d+1)^n + l)$

The based SK-VC scheme can realize the outsourced computation of any d-degree polynomial functions for n variables. In this scheme, the runtime of the client is $O(n \cdot \log d)$, and the runtime of the server and the source are both asymptotically the same as evaluating the polynomial F, namely, $O((d+1)^n)$. Obviously, $O(n \cdot \log d) \ll O((d+1)^n)$. Assume that the access control policy complexity, namely, the number of attributes included in the access control policy is l. Then the runtime required by ABE.Encrypt and ABE.Decrypt are both $O(l)$ which are linearly dependent on l in [7]. Therefore, in the concrete AC-VC, the upper bound of the runtime of a client (namely, the runtime in the first query for an input) is $O(n \cdot \log d + l)$, and in the later executions of the scheme, the runtime of the client is only $O(n \cdot \log d)$. As for the policy complexity, in practice, large complexity-policies may be rare such as in the role based policies. So, for high degree polynomial functions, when $l < n \cdot \log d$ or they are comparative, we have $O(n \cdot \log d + l) \ll O((d+1)^n)$. The detailed efficiency of the concrete AC-VC is shown in Table 2. In addition, we expect further improvements in the efficiency of new ABE schemes which will benefit the proposed AC-VC.

6 Discussions

Multi-authority in AC-VC In our construction for AC-VC, we assume that all clients' attributes come from the same domain and they are managed by a single authority (or issuer). However, in some real applications, the clients' attributes may be from different domains each of which is managed by a different authority. In order to solve this problem, multiauthority CP-ABE has been studied [8, 9, 22, 24]. By combining these multiauthority CP-ABE schemes with SK-VC under our framework

for AC-VC, the source can enforce access policies based on attributes in different domains. For example, the source can define the access policy as "Corporation A. Engineer AND University B. Lecturer" where the attribute "Engineer" is issued by Corporation A and the attribute "Lecturer" is issued by University B.

Attribute revocation in AC-VC. In our construction for AC-VC, when the function F or the access policy ω changes, naturally, the source needs to re-generate a public key $PK_{F,\omega}$ associated with the new function and access policy.

In our construction, if the attribute set of a client satisfies the access policy ω, then the client will obtain the secret key sk_S from $PK_{F,\omega}$ to outsource any input x to the server for $F(x)$. In real applications, the clients' attributes may change dynamically, for example, a client may be entitled some new attributes or revoked some attributes. In the case that a client is entitled some new attributes, the client will obtain new decryption keys issued by the authorities and, therefore, obtain the secret key sk_S for some function F and ω if the client's attributes satisfy ω. In the case that a client is revoked some attributes, this client should not be able to decrypt any new secret key sk_S which requires the revoked attributes to decrypt. Concretely, assume that a new sk_S is associated with a function F' and ω' where ω' requires the revoked attributes, then the client cannot decrypt sk_S. As for the secret key sk_S that he has decrypted before he is revoked, it seems that there is no effective method to prevent him continuing using it during the later outsourcing any more. In respect to the attribute revocation problem in CP-ABE, many methods have been presented [16, 18, 29, 35] to solve it. Since our construction for AC-VC is simply based on CP-ABE and SK-VC, we can apply these schemes [16, 18, 29, 35] to our AC-VC to solve the attribute revocation problem.

7 Conclusion

In this paper, we present the verifiable computation with access control. A construction is given by combining any verifiable computation in the secret-key setting and ciphertext-policy attribute-based encryption. In this construction, the source only needs to perform an initialization to enforce the access control permissions on the cloud application so that only the clients who satisfy the permissions can access the application. This scheme satisfies the correctness, verifiability, and access security that we defined. In addition, our scheme is efficient and the time costs of the participants depend on that required by the based CP-ABE and SK-VC.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 61202466, U1135004, 61170080), Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. 2012LYM_0017), Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011), High-level Talents Project of Guangdong Institutions of Higher Education (2012), and Fundamental Research Funds for the Central Universities (No. 2012zb0015).

References

1. Applebaum B, Ishai Y, Kushilevitz E (2010) From secrecy to soundness: efficient verification via secure computation (extended abstract). In: ICALP 2010. LNCS, vol 6198. Springer, Berlin, pp 152–163

2. Arora S, Safra S (1998) Probabilistic checking of proofs: a new characterization. J ACM 45:70–122
3. Babai L, Fortnow L, Levin LA, Szegedy M (1991) Checking computations in polylogarithmic time. In: STOC 1991. ACM, New York, pp 21–32
4. Barbosa M, Farshim P. Delegatable homomorphic encryption with applications to secure outsourcing of computation. Cryptology ePrint archive: report 2011/215
5. Benabbas S, Gennaro R, Vahlis Y (2011) Verifiable delegation of computation over large datasets. In: CRYPTO 2010. LNCS, vol 6841. Springer, Berlin, pp 111–131
6. Beimel A (1996) Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel
7. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: IEEE symposium on security and privacy. IEEE Computer Society, Los Alamitos, pp 321–334
8. Chase M (2007) Multi-authority attribute based encryption. In: TCC 2007. Springer, Berlin, pp 515–534
9. Chase M, Chow SS (2009) Improving privacy and security in multi-authority attribute-based encryption. In: CCS 2009. ACM, New York, pp 121–130
10. Chung KM, Kalai Y, Vadhan S (2010) Improved delegation of computation using fully homomorphic encryption. In: CRYPTO 2010. LNCS, vol 6223. Springer, Berlin, pp 483–501
11. Fiore D, Gennaro R (2012) Publicly verifiable delegation of large polynomials and matrix computations, with applications. In: CCS 2012. ACM, New York, pp 501–512
12. Gennaro R, Gentry C, Parno B (2010) Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: CRYPTO 2010. LNCS, vol 6223. Springer, Berlin, pp 465–482
13. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: STOC 2009. ACM, New York, pp 169–178
14. Goldwasser S, Kalai YT, Rothblum GN (2008) Delegating computation: interactive proofs for muggles. In: STOC 2008. ACM, New York, pp 113–122
15. Goldwasser S, Lin H, Rubinstein A. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint archive: report 2011/456
16. Hur J, Noh DK (2011) Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans Parallel Distrib Syst 22(7):1214–1221
17. Ibraimi L, Asim M, Petkovic M (2009) Secure management of personal health records by applying attribute-based encryption. Technical report, University of Twente
18. Jahid S, Mittal P, Borisov N (2011) EASiER: encryption-based access control in social networks with efficient revocation. In: ASIACCS 2010. ACM, New York, pp 411–415
19. Kilian J (1992) A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC 1992. ACM, New York, pp 723–732
20. Kamara S, Raykova M (2011) Secure outsourced computation in a multi-tenant cloud. In: Workshop on cryptography and security in clouds
21. Lewko A, Okamoto T, Sahai A, Takashima K, Waters B (2010) Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: EUROCRYPT 2011, LNCS, vol 6110. Springer, Berlin, pp 62–91
22. Lewko AB, Waters B (2011) Decentralizing attribute-based encryption. In: EUROCRYPT 2011. Springer, Berlin, pp 568–588
23. Li M, Yu S, Zheng Y, Ren K, Lou W (2012) Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans Parallel Distrib Syst 24(1):131–143
24. Müller SM, Katzenbeisser S, Eckert C (2008) Distributed attribute-based encryption. In: ICISC 2008. Springer, Berlin, pp 20–36
25. Micali S (2000) Computationally sound proofs. SIAM J Comput 30(4):1253–1298
26. Narayan S, Gagné M, Safavi-Naini R (2010) Privacy preserving EHR system using attribute-based infrastructure. In: CCSW 2010. ACM, New York, pp 47–52
27. Parno B, Raykova M, Vaikuntanathan V (2012) How to delegate and verify in public: verifiable computation from attribute-based encryption. In: TCC 2012, pp 422–439
28. Papamanthou C, Shi E, Tamassia R. Signatures of correct computation. Cryptology ePrint archive: report 2011/587
29. Sahai A, Seyalioglu H, Waters B (2012) Dynamic credentials and ciphertext delegation for attribute-based encryption. In: CRYPTO 2012. Springer, Berlin, pp 199–217
30. Wang C, Ren K, Wang J (2011) Secure and practical outsourcing of linear programming in cloud computing. In: INFOCOM 2011. IEEE Computer Society, Los Alamitos, pp 820–828
31. Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: PKC 2011. LNCS, vol 6571. Springer, Berlin, pp 53–70

32. Yao A (1982) Protocols for secure computations. In: FOCS 1982. IEEE Computer Society, Los Alamitos, pp 160–164
33. Yao A (1986) How to generate and exchange secrets. In: FOCS 1986. IEEE Computer Society, Los Alamitos, pp 162–167
34. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: INFOCOM 2010. IEEE Computer Society, Los Alamitos, pp 534–542
35. Yu S, Wang C, Ren K, Lou W (2010) Attribute based data sharing with attribute revocation. In: ASIACCS 2010. ACM, New York, pp 261–270