2012

Dynamic Secure Cloud Storage with Provenance

Sherman S. M. CHOW
University of Waterloo

Cheng-Kang CHU
Institute for Infocomm Research

Xinyi HUANG
Institute for Infocomm Research

Jianying ZHOU
Institute for Infocomm Research

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-642-28368-0_28

Published in Cryptography and Security: From Theory to Applications. Lecture Notes in Computer Science, vol 6805. Springer, pp. 442-464.
https://doi.org/10.1007/978-3-642-28368-0_28

Dynamic Secure Cloud Storage with Provenance*

Sherman S.M. Chow[1], Cheng-Kang Chu[2], Xinyi Huang[2], Jianying Zhou[2], and Robert H. Deng[3]

[1] University of Waterloo
smchow@math.uwaterloo.ca
[2] Institute for Infocomm Research
{ckchu,xhuang,jyzhou}@i2r.a-star.edu.sg
[3] Singapore Management University
robertdeng@smu.edu.sg

Abstract. One concern in using cloud storage is that the sensitive data should be confidential to the servers which are outside the trust domain of data owners. Another issue is that the user may want to preserve his/her anonymity in the sharing or accessing of the data (such as in Web 2.0 applications). To fully enjoy the benefits of cloud storage, we need a confidential data sharing mechanism which is fine-grained (one can specify *who* can access *which classes* of his/her encrypted files), dynamic (the total number of users is not fixed in the setup, and any new user can decrypt previously encrypted messages), scalable (space requirement does not depend on the number of decryptors), accountable (anonymity can be revoked if necessary) and secure (trust level is minimized).

This paper addresses the problem of building a secure cloud storage system which supports dynamic users and data provenance. Previous system is based on specific constructions and does not offer all of the aforementioned desirable properties. Most importantly, dynamic user is not supported. We study the various features offered by cryptographic anonymous authentication and encryption mechanisms; and instantiate our design with verifier-local revocable group signature and identity-based broadcast encryption with constant size ciphertexts and private keys. To realize our concept, we equip the broadcast encryption with the dynamic ciphertext update feature, and give formal security guarantee against adaptive chosen-ciphertext decryption and update attacks.

Keywords: Anonymity, broadcast encryption, cloud storage, dynamic encryption, group signatures, pairings, secure provenance.

1 Introduction

New computing paradigms keep emerging. One notable example is the cloud computing paradigm, a new economic computing model made possible by the advances in networking technology, where a client can leverage a service provider's

* Funded by A*STAR project SecDC-112172014.

computing, storage or networking infrastructure. With the unprecedented exponential growth rate of information, there is an increasing demand for outsourcing data storage to cloud services such as Microsoft's Azure and Amazon's S3.

The use of public cloud infrastructure introduces significant security and privacy risks. For the sensitive data, one can always use data encryption before outsourcing to mitigate the confidentiality concern. However, the hurdle often lies in its management. Consider that a certain organization is a cloud service client; different individual users within an organization should have different access privileges of the organization's data. The cloud client may not want to trust the cloud server in performing the access control faithfully, or put the *whole* system under the control of a reference monitor inside a trusted hypervisor. Apart from the management of access control, *dynamic user management* is also an important feature for any *practical* system. New users may join the system after the data is encrypted, and it is desirable that a new user can decrypt previously created ciphertexts if necessary. This also implies that the same copy of ciphertext can be decryptable by more than one user, and one may not want to maintain multiple ciphertexts corresponding to the same plaintext data, either for easier data management or minimizing storage overhead.

Another issue is privacy. A cloud client hoping to enforce access control does not necessary need to let the cloud server to know the identity of the users. Actually, anonymity is a desirable feature for many web or collaborative applications. Active involvement of discussion or collaboration over web can be partially attributed to the (pseudo) anonymity perceived by the users. On the other hand, perfect anonymity might be abused by misbehaving users. It is thus equally important to support *data provenance*, especially, to record who created, modified, deleted data stored in a cloud.

Can the current advances in cryptography solve our problem? Recall that using group signatures, each group member can sign a message on behalf of a group such that anyone can verify that the group signature is produced by someone enrolled to the group, but not exactly whom. Can we just employ any group signature scheme for the data provenance, and any public key encryption for the data confidentiality requirement of cloud storage? Concretely, for a user to upload (encrypted) data, he or she has to sign the ciphertext using the member signing key. The cloud service provider accepts this ciphertext if the signature is valid. All the users' action regarding insertion, modification and deletion will be accountable due to the use of group signature as an anonymous authentication mechanism. A group manager can then open the signature to reveal the uploader's identity in case the data is in dispute. Indeed, it is actually the approach taken by a recent secure provenance system for cloud computing [12]. We revisit this problem, identify and realize the missing features which are desirable in the cloud setting, investigate the subtle issues involved in the interaction of these two cryptographic primitives, and contribute to the study of secure cloud storage system in the following four aspects.

1.1 Survey of Cryptographic Toolkits and a Generic System Design

The system proposed by Lu *et al.* [12] (hereinafter referred to as LLLS scheme) is based on two existing constructions: the group signature scheme proposed by Boyen and Waters [5] and a simplified version of an attribute-based encryption scheme proposed by Waters [14], which are not explicitly mentioned in [12] though. It may be difficult for a general computer scientist or a cloud computing researcher/practitioner to make sense of what is going on behind the equations. Indeed, we believe that the encryption part of their system can be replaced by a single-user public key encryption scheme according to what we re-engineered from their number-theoretic description. We also believe that it is good for both cryptography community and security (in cloud computing) community to establish the connection between the set of properties that cryptographic schemes can possibly offer, and the desirable features one may expect in the cloud computing setting. Otherwise, we may need to design a "new" cryptographic scheme whenever there is a slight change in the application scenario. Finally, the decoupling of specific instantiations from the generic system design is good for replacing the building block with better constructions in the future.

1.2 Revocation in Group Signatures

Regarding the concrete contribution of our proposed system, we first notice that a "vanilla" group signature scheme, which is used in the LLLS scheme [12], may not be suitable for the cloud storage scenario. Recall that the ciphertexts in the cloud are contributed by different users. When we notice something wrong with a particular message, we want to reveal the authorship of that message. After we learn the real author, we may worry that this user is compromised without being noticed, or this user has been turned malicious for a while and has been uploading "wrong" data to the cloud. It is unclear which other ciphertexts on the system are possibly uploaded by the same user without opening all signatures. This is actually a well known shortcoming of normal group signatures, and it becomes more apparent in our scenario.

On the other hand, the power of anonymity revocation, if tuned correctly, can be a mean to revoke the signing power of a user. For example, if it is possible for the group manager to generate a user-specific trapdoor which can be used to check whether the hidden identity of a signature is a particular one, without opening the hidden identity in clear, verifier-local revocation is possible when the verifier is being issued with the trapdoor for the revoked users. In our concrete instantiation, we will realize this verifier-local revocation concept using the scheme proposed by Boneh and Shacham [4]. Indeed, one may take this one step further using a recently proposed notion of real traceable signature [7] such that the tracing of data uploaded by "bad" users can be efficiently performed.

1.3 Dynamic Broadcast Encryption

In the LLLS scheme [12], every ciphertext in the system can be decrypted by any users. For access control, one should employ a multi-recipient encryption scheme

where the encryptor can decide who is allowed to decrypt. This is exactly the reason why a traditional (broadcast) encryption scheme is not suitable for the dynamic user setting. Once a ciphertext is generated, the decryption set is also fixed. No one can add/delete users for that ciphertext without decrypting it.

An efficient realization of multi-recipient encryption is broadcast encryption, in which the ciphertext size is independent of the size of the recipients. In this paper, we propose a method that allows the master secret key holder to re-encrypt or update the ciphertext based on the identity-based broadcast encryption proposed by Delerablée [9]. In this case, the group manager can adjust the decryption set of the existing ciphertexts on the cloud storage. This update procedure is more efficient than the trivial "decrypt-then-encrypt" method. The possibility of updating a ciphertext has been briefly mentioned in [9]. However, no exact algorithm was provided in [9], not to say a formal security model capturing any possible vulnerability which may be introduced by this update procedure. We will show that a simple and straightforward way to update a ciphertext will make it easily decryptable by anyone outside of the parties who have been designated as legitimate decryptors. We provide a better way of updating, and a formal security definition of adaptive chosen-ciphertext decryption and update attacks. The proof turns out to be non-trivial as we need to use a slightly stronger number theoretic assumption than what has been assumed in [9] to obtain our formal security guarantee.

1.4 Linkage between Group Signatures and Broadcast Encryption

For group signatures on plaintext messages, identification of bad messages can be achieved in a straightforward manner: a user reports against a certain message, everybody can read this and the revocation manager can open the identity. This mechanism cannot be directly applied when messages are encrypted. In particular, due to confidentiality concern, we do not want the revocation manager to have the power to decrypt every ciphertext. To the best of our knowledge, this issue is never formally studied.

With our dynamic broadcast encryption, a ciphertext can be updated such that it will become decryptable by the revocation manager, but now another problem arises as the update in the ciphertext means it is no longer the "content" signed by the group signature. To reclaim the public verifiability offered by group signatures so as to mitigate any concern about false accusations, we propose the use of a linkable ciphertext transformation to achieve the dynamic decryption and public verifiability simultaneously.

1.5 Organization

The rest of the paper is organized as follows. Section 2 introduces the system and threat models. In Section 3, we discuss our design of cloud storage system which supports both data provenance and dynamic users. Section 4 will detail our concrete scheme, followed by some concluding remarks.

2 Model

2.1 System Model

We consider a secure cloud storage system involving three different entities: the Cloud Service Provider (denoted by CSP), who provides data storage service and has significant storage space and computation resources, the Group Manager (denoted by GM) who represents the IT department of an organization which has a large amount of data files to be stored in the cloud, and is responsible for the maintenance of the system deployed on the cloud and the management of the credential of users. Users (denoted by U), the staffs of an organization, who upload and download data from the organization's storage on the cloud. A CSP may provide the network storage service for different clients. For simplicity, we assume there is only one GM, and all users belong to this client.

Running of a cloud storage system consists of the following phases:

Setup. This phase is performed by GM. It takes as input a security parameter, and outputs a public parameter PK and a master secret key MK. PK is published in the system, and MK is kept secret by GM.

User Registration. This is an initial user registration phase performed by GM and users. A user U_{ID} can register to the GM and get a private authentication and decryption key pair $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$. Besides, ID is added to a user list S which is published and used for broadcast encryption.

Data Access. This phase is performed by a user U_{ID} who owns the key $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ and the service provider CSP. There are two kinds of accesses, read and write. U_{ID} can read data by downloading the ciphertext and decrypting it using dk_{ID}. To write, U_{ID} uploads ciphertexts with anonymous signatures computed from ak_{ID}. CSP accepts the ciphertext if the corresponding signature is valid and generated by an unrevoked user.

User Joining. This phase is performed by GM and a new user U_{ID}. GM first gives the key pair $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ to U_{ID} and adds ID to the user list S. GM also re-encrypts the existing ciphertexts on the storage so that the new user can decrypt them.

User Revocation. This phase is performed by GM and CSP. If a user U_{ID} is revoked, GM removes ID from the user list S and gives a token to CSP so that CSP can add it to the revocation list. For the existing ciphertexts, GM re-encrypts them to exclude U_{ID}.

Tracing. This phase is performed by GM. Given a ciphertext and its corresponding signature, GM outputs the identity of the signer (perhaps only when the (GM) targeted to that specific signer) or an error symbol "$\perp$" (indicating tracing failure).

2.2 Security Model

A secure cloud storage system should ensure confidentiality, anonymity and traceability. We discuss the security models for these three properties.

Confidentiality. We define the confidentiality of the data on the storage by extending the notion of selective-ID chosen ciphertext security of broadcast encryption in [9]. In our construction, the adversary can query an update oracle. The security can be modeled as the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, which are both given as input n, the maximal size of a set of receivers.

Init. The adversary outputs a set $S^* = \{\mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_s\}$ of identities it wants to attack (with $s \leq n$).

Setup. The challenger performs Setup phase and sends the public parameter PK to the adversary.

Query Phase 1. The adversary adaptively issues queries to the following oracles:

- Extract(ID): if $\mathsf{ID} \notin S^*$, $\mathcal{C}$ performs User Registration phase and returns $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ to $\mathcal{A}$.
- Decryption(C): $\mathcal{C}$ performs the read part of Data Access phase and returns the decrypted message.
- Update(C, *act*, ID): $\mathcal{C}$ considers the following two cases:
  - *act* = "add": $\mathcal{C}$ performs the update procedure on C like in User Joining phase to add ID to the decryption set of C.
  - *act* = "delete": $\mathcal{C}$ performs the update procedure on C like in User Revocation phase to delete ID from the decryption set of C.

Challenge. $\mathcal{A}$ outputs two equal-length messages m_0, m_1. $\mathcal{C}$ randomly chooses a bit $b \in \{0, 1\}$ and outputs a ciphertext C^*, an encryption of m_b.

Query Phase 2. This phase is the same as Query Phase 1 except that $\mathcal{A}$ cannot issue

- C^* to the decryption oracle, and
- (C^*, add, ID) to the update oracle and then ID to the extract oracle for any ID.

Guess. $\mathcal{A}$ outputs a guess b'. If $b' = b$, $\mathcal{A}$ wins.

We define $\mathcal{A}$'s advantage in winning the game as $\mathtt{Succ}_A$ as $|\Pr[b = b'] - 1/2|$. The probability is taken over the coin tosses of $\mathcal{A}$ and all the randomized algorithms. We say the storage system provides IND-sID-CCA confidentiality if no polynomial time adversary can win the above game with a non-negligible advantage. If the decryption and update oracles are not allowed to query, then we say the system provides IND-sID-CPA confidentiality. Furthermore, we define the IND-sID-CPA* confidentiality as the IND-sID-CPA security game except that the adversary is allowed to choose any set $\mathcal{T}^* \subset S^*$ and ask for an "update" of the challenge ciphertext encrypted for $S^* \backslash \mathcal{T}^*$.

Anonymity. In the anonymity game, the adversary's goal is to determine which of two keys generated a signature, without given either of the keys. The game is defined as follows.

Init. The challenger $\mathcal{C}$ runs algorithm Setup, obtaining PK and MK. He then runs algorithm User Registration, obtaining n key pairs $(ak_{\mathsf{ID}_1}, dk_{\mathsf{ID}_1}), (ak_{\mathsf{ID}_2}, dk_{\mathsf{ID}_2}), \ldots, (ak_{\mathsf{ID}_n}, dk_{\mathsf{ID}_n})$. $\mathcal{C}$ provides the adversary $\mathcal{A}$ with PK.

Query Phase 1. The adversary can make queries as follows:
- Extract(ID_i): $\mathcal{A}$ requests the private key of the user at index $i, 1 \leq i \leq n$. The challenger responds with $(ak_{\mathsf{ID}_i}, dk_{\mathsf{ID}_i})$.
- Sign(ID_i, C): $\mathcal{C}$ performs the write part of Data Access phase and computes a signature σ_i on C using ak_{ID_i}. $\mathcal{A}$ is given σ_i.
- Revocation(ID_i). $\mathcal{A}$ can request the revocation token of the user at index $i, 1 \leq i \leq n$. The challenger responds with the revocation token of ID_i.

Challenge. $\mathcal{A}$ outputs a ciphertext C^* and two indices i_0 and i_1. It must have made neither an extraction nor a revocation query on either index. The challenger chooses a bit $b \in \{0, 1\}$ uniformly at random, computes a signature σ^* on C^* using $ak_{\mathsf{ID}_{i_b}}$ and provides σ^* to $\mathcal{A}$.

Query Phase 2. After obtaining the challenge, $\mathcal{A}$ can make additional queries of the challenger, restricted as follows.
- Extract(ID_i): As before, but $\mathcal{A}$ cannot make extraction queries at ID_{i_0} or ID_{i_1}.
- Sign(ID_i, C): $\mathcal{A}$ can make signing queries as before.
- Revocation(ID_i). As before, but $\mathcal{A}$ cannot make revocation queries at ID_{i_0} or ID_{i_1}.

Guess. Finally, $\mathcal{A}$ outputs a bit b', its guess of b. The adversary wins if $b' = b$.

We define $\mathcal{A}$'s advantage in winning the game as $\mathtt{Succ}_A$ as $|\Pr[b = b'] - 1/2|$. The probability is taken over the coin tosses of $\mathcal{A}$ and the randomized key generation and signing algorithms. We say the storage system provides anonymity if no polynomial time adversary can win the above game with a non-negligible advantage.

Traceability. We say that a storage system is traceable if no adversary can win the traceability game. In the traceability game, the adversary's goal is to create a signature that cannot be traced to one of the users in the compromised coalition using the tracing algorithm above. Let n be the maximal size of a set of users. The traceability game, between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is defined as follows.

Init. The challenger $\mathcal{C}$ runs algorithm Setup, obtaining PK and MK. He then runs algorithm User Registration, obtaining n key pairs $(ak_{\mathsf{ID}_1}, dk_{\mathsf{ID}_1}), (ak_{\mathsf{ID}_2}, dk_{\mathsf{ID}_2}), \cdots, (ak_{\mathsf{ID}_n}, dk_{\mathsf{ID}_n})$ and their revocation tokens. $\mathcal{C}$ provides the adversary $\mathcal{A}$ with PK and revocation tokens of all users, and sets $U = \varnothing$.

Query Phase. The adversary can make queries as follows:
- Extract(ID_i): $\mathcal{A}$ requests the private key of the user at index $i, 1 \leq i \leq n$. The challenger appends i to U, the adversary's coalition, and responds with $(ak_{\mathsf{ID}_i}, dk_{\mathsf{ID}_i})$.
- Sign(ID_i, C): $\mathcal{C}$ performs the write part of Data Access phase and generates a signature σ_i of C using ak_{ID_i}. $\mathcal{A}$ is given σ_i.

Output. Finally, $\mathcal{A}$ outputs a ciphertext C^*, a revocation list RL^*, and a signature σ^*. $\mathcal{A}$ wins if: (1) σ^* is a valid signature on C^* with revocation list RL^*; (2) σ^* traces (using the tracing algorithm) to some user outside of the coalition $U \backslash RL^*$, or the tracing algorithm fails; and (3) σ is non-trivial, i.e., $\mathcal{A}$ did not obtain σ by making a signing query on C^*.

We denote by $\mathtt{Succ}_T$ the probability that $\mathcal{A}$ wins the game. The probability is taken over the coin tosses of $\mathcal{A}$ and the randomized key generation and signing algorithms. We say the storage system provides traceability if no polynomial time adversary can win the above game with a non-negligible probability.

3 Cryptographic Provenance-Aware Cloud Storage

The first part of this section is about data provenance. We start by describing different features provided by various group signature schemes, followed by the design of provenance-aware cloud storage system enabled by group signatures. Next, we will focus on user dynamics. We will describe the update feature of broadcast encryption scheme and how group signatures can be used to authenticate updatable ciphertexts. To make our discussion more concrete, our presentation in this part will be based on the identity-based broadcast encryption proposed by Delerablée [9]. Hence we will also review the properties of pairing, which is the primitive operation used by this scheme.

3.1 Group Signatures with Different Features

We first describe how a traditional group signature scheme works. When a user joins a group, the GM and the user executes a joining protocol. As a result, the GM gives this new member a signer key. A group member wants to preserve anonymity in signing. The signing process is actually a proof of knowledge (PoK) of a signer key, with respect to the message to be signed. Another feature of group signature is that it can be opened to reveal the true signer. Thus, it should contain an *encryption* of some information that uniquely identifies a user, such that only a designated party (e.g., the GM, or another party being assigned to do that) can decrypt it. This process is often called as opening.

Verifier-Local Revocation. In a group signature scheme which supports verifier-local revocation (VLR), the encryption of unique identification information in the signature is not necessary. However, there should be enough "structure" in the group signature that links it to a "unique revocation token". Such a token is a by-product generated from the joining protocol. Unless instructed, this revocation token is kept confidential by the GM. When there is a need to revoke a user, the corresponding revocation token is released such that everyone can identify if a signature is produced by the revoked user and discard the signature if this is the case, which effectively enables the signing key revocation. In our instantiation to be presented, we will use a VLR group signature scheme proposed in [4].

Traceability. Opening and VLR are not the only possible anonymity management mechanisms. Here we describe a new way supported by a recently proposed notion which is called real traceable signature [7]. It is similar to the VLR group signature. A difference is that, instead of *checking* whether a signature was issued

by a given user, the GM can generate a tracing trapdoor which allows the reconstruction of user-specific tags. These tags can uniquely identify a signature of a misbehaving user. Identifying N signatures just requires N tag-reconstructions instead of requiring the checking of all N' ($>> N$) signatures ever produced in the system. This feature is desirable when the cloud storage is storing time-sensitive information, such as real-time financial data.

Exculpability. Along with the aforementioned properties, another strengthening of the basic group signature notion is exculpability. Note that the member signing key generated by the GM can be regarded as a signature produced by the GM. Instead of generating this signature and giving it as a member signing key to the user directly, the GM can sign on something that is related to some valuable secrets of users, such that they would not share it with others easily. To sign on behalf of the group, a member can only do so using both the member's own valuable secret and the member signing key. In this way, even the GM cannot sign "on behalf" of a group member.

3.2 A Basic Design

Setup. The GM first selects a group signature scheme ($\mathcal{GS}$) suitable for the application scenario and a multi-recipient public-key encryption scheme, may it be an identity-based encryption ($\mathcal{IBE}$), an attribute-based encryption ($\mathcal{ABE}$) (e.g., [14]) or a broadcast encryption ($\mathcal{BE}$). The public parameter PK and the master secret key will include the corresponding public/secret parameter of $\mathcal{GS}$ and $\mathcal{BE}$.

User Registration. The user's key is $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$, where ak_{ID} is the member signing key given by $\mathcal{GS}$ and dk_{ID} is the private decryption key of the encryption scheme. The generation of ak_{ID} may involve a tracing trapdoor for the user ID, which will be held confidentially by the GM or the revocation manager RM.

Data Access. For read operation, U_{ID} downloads the ciphertext from the cloud and decrypts it using dk_{ID}. For write operation, U_{ID} first prepares a ciphertext using the encryption scheme according to the expected intended recipient lists; then signs the resulting ciphertext using $\mathcal{GS}$. CSP only accepts and stores both the ciphertext and the signature if the signature is valid.

In essence, this simplifies the framework implicitly used by the LLLS system [12]. One difference is that the user in their system will also generate a one-time signing key and use it to sign on the ciphertext to be uploaded. The group signature is signed on the corresponding verification key of the one-time signature instead. So the whole uploaded data also consists of a verification key, a one-time signature in additional to the ciphertext and the group signature. Similar technique has been used in the group signature paradigm to achieve "security against adaptive opening attack", e.g., as used in [7]. However, this requires the underlying group signature scheme to offer a similar level of security while the underlying scheme [5] employed in [12] can only provide CPA security. The benefit of using this "2-layer" signing approach is not clear. Another difference is that

[12] requires the existence of a system manager which is "a trustable and powerful entity, located at the top of the cloud computing system". In particular, this entity can decrypt all ciphertexts of the system and issue signatures on behalf of any user. While in our design, we can employ a $\mathcal{GS}$ scheme with exculpability to avoid the later problem, and the power of decryption is only confined to the group manager GM but not the cloud service provider CSP. (And one may take a step further to reduce the trust assumption by employ constructions such as "escrow-free" IBE [8] and multi-authority ABE [6]). We thus believe our design is more preferable.

3.3 Bilinear Group

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three (multiplicative) cyclic groups of prime order p, $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be an isomorphism and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a map with the following properties:

- Bilinear: for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Non-degenerate: $e(g_1, g_2) \neq 1$ for some $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.
- Efficiently computable: e can be computed efficiently.

We say that $BGS = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot, \cdot))$ is a bilinear group system if all the above operations are efficiently computable.

In this paper, we say a bilinear group system is *symmetric* if $\mathbb{G}_1 = \mathbb{G}_2$, and it is *asymmetric* otherwise.

Throughout this paper, we only work with bilinear group pairs of prime order. We remark that the implicit group signature scheme [5] used in the LLLS system requires bilinear group pairs of composite order. Realizing cryptographic scheme in a composite order group tends to be less efficient in general since a larger modulus should be chosen to withstand the best known factorization attack.

3.4 Update in Dynamic Broadcast Encryption

Now we move forward to show how to equip the encryption scheme with dynamic update features so as to support dynamic users. We will adopt Delerablée's [9] broadcast encryption scheme to encrypt data. Here we explain why a trivial update mechanism would not work for this scheme.

Before delving into the details of the scheme, we focus on a particular component of the ciphertext, namely, $c_2 = h_2^{k \cdot \prod_{id \in S}(\alpha + H_0(id))}$, where α is the master secret key (that is used to generate the user decryption key), S is the set of designated users who can decrypt the ciphertext, and k is the randomness of the ciphertext which uniquely determines the padding used to encrypt the message, specifically, $e(h_1, h_2)^k$.

Let us consider a simple case that the ciphertext is originally intended for user id_1, i.e., $c_2 = h_2^{k(\alpha + H_0(id_1))}$. Suppose at a later stage we want to also allow user id_2 to decrypt this ciphertext and we want to revoke the decryption power of id_1. A trivial way to do is exponentiating c_2 to the power of $\frac{\alpha + H_0(id_2)}{\alpha + H_0(id_1)}$, which gives us $c_2' = (h_2^{k(\alpha + H_0(id_1))})^{\frac{\alpha + H_0(id_2)}{\alpha + H_0(id_1)}} = h_2^{k(\alpha + H_0(id_2))}$.

Now, note that we have two equations c_2 and c_2' regarding two unknown k and α in the exponent with a common base element h_2. Hence, it is possible to derive the value of h_2^k from c_2 and c_2'. With this value, one can easily recover the padding $e(h_1, h_2^k) = e(h_1, h_2)^k$.

3.5 Signatures on Updated Ciphertext

In the case of revoking a user based on a bad message he or she has posted, we do not care much about the message confidentiality for this particular ciphertext anymore. So the above simple update suggests a simple linkable ciphertext transformation such that any user can still verify the group signature on the original ciphertext. The GM first updates the ciphertext by allowing any revocation manager (the entity in the organization who is responsible for asserting misbehavior of a user) to decrypt the ciphertext in question. Specifically c_2 will be updated to $c_2^{(\alpha + H_0(\mathsf{RM}))}$ where RM is the identity of the revocation manager. Now, it is easy to verify the linkage between the original ciphertext component c_2 and the newly updated component c_2' by the bilinearity of the pairing, i.e., checking if $e(w_1 \cdot h_1^{H_0(\mathsf{RM})}, c_2) = e(h_1, c_2')$ where $w_1 = h_1^\alpha$ is a component in the public parameter.

3.6 Updating Ciphertexts in Practice

The outcome of the update algorithm is that the ciphertext becomes decryptable by any user specified in the input, *independent* of the original specified decryptor set. By the strong functionality provided by the update algorithm, it seems unavoidable to require the knowledge of the master secret key α to update a ciphertext.

While we advocate that supporting dynamic user is an essential feature, adding or removing users from the system should be a relatively less frequent operation. One choice to realize the update functionality in practice is that whenever a new staff is added to an organization, his/her supervisor, who knows what classes of data he/she should be entitled to access, submits the corresponding update requests to the GM. Another way is to rely on a trusted hypervisor. Note that it is different from the approach of putting the *entire* storage system within a trusted virtual machine maintained on the cloud. In our case, only the update module which works on a relatively smaller set of ciphertexts is put in the trusted execution environment.

4 Concrete Construction

Now we are ready to present our concrete construction, which is based on the VLR group signature scheme proposed in [4] and our variant of the identity-based broadcast encryption proposed by Delerablée [9].

4.1 Instantiation of Our Design

Let n be the maximum number of receivers the file can be encrypted to.

- Setup. On input a security parameter κ, the GM performs the following steps:
  - It generates a bilinear group system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot,\cdot))$ as described above where $p \geq 2^\kappa$ and cryptographic hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_p^*, H_1 : \{0,1\}^* \to \mathbb{G}_2^2, H_2 : \{0,1\}^* \to \mathbb{Z}_p$.
  - It randomly chooses $h_1 \in_R \mathbb{G}_1, h_2, g_2 \in_R \mathbb{G}_2$ and computes $g_1 = \psi(g_2)$.
  - It randomly chooses two secrets $\alpha, \gamma \in \mathbb{Z}_p^*$ and sets $w_1 = h_1^\alpha, w_2 = g_2^\gamma$.

  The public parameter is defined as

$$PK = (\ BGS = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot,\cdot)), H_0(\cdot), H_1(\cdot), H_2(\cdot), \\ g_1, g_2, h_2, h_2^\alpha \ldots, h_2^{\alpha^n}, w_1, w_2, e(h_1, h_2)\ ).$$

  The master secret key is $MK = (h_1, \alpha, \gamma)$.
- User Registration. For each user $\mathsf{U_{ID}}$, the GM issues a private key $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$, where

$$ak_{\mathsf{ID}} = (x_{\mathsf{ID}} \in_R \mathbb{Z}_p, g_1^{1/(\gamma + x_{\mathsf{ID}})}) \quad \text{and} \quad dk_{\mathsf{ID}} = h_1^{1/(\alpha + H_0(\mathsf{ID}))}.$$

  The identity ID is added to the access user list S and $(\mathsf{ID}, g_1^{1/(\gamma+x_{\mathsf{ID}})})$ is added to a token list T.
- Data Access. This phase is performed by a user $\mathsf{U_{ID}}$ and the service provider CSP. The user $\mathsf{U_{ID}}$ with the private key $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ can read or write data as follows.
  - *Write*: Given a message m and a user list S, $\mathsf{U_{ID}}$ first chooses a random $k \in \mathbb{Z}_p^*$ and computes the ciphertext $C = (c_1, c_2, c_3)$ as follows.

$$c_1 = w_1^{-k}, \qquad c_2 = h_2^{k \cdot \prod_{id \in S}(\alpha + H_0(id))} \qquad \text{and} \qquad c_3 = m \cdot e(h_1, h_2)^k.$$

    Let $ak_{\mathsf{ID}} = (x_{\mathsf{ID}}, y_{\mathsf{ID}})$. $\mathsf{U_{ID}}$ generates a signature on C:
    1. Pick a random nonce $r \in \mathbb{Z}_p$ and let $(\hat{u}, \hat{v}) = H_1(g_1||g_2||w_2||C||r) \in \mathbb{G}_2^2$. Compute their images in $\mathbb{G}_1 : u = \psi(\hat{u}); v = \psi(\hat{v})$.
    2. Select a random $a \in \mathbb{Z}_p$ and compute $T_1 = u^a$ and $T_2 = y_{\mathsf{ID}} v^a$.
    3. Set $\delta = a x_{\mathsf{ID}}$. Pick three random blinding values r_a, r_x and $r_\delta \in \mathbb{Z}_p$.
    4. Compute helper values

$$R_1 = u^{r_a}, \quad R_2 = e(T_2, g_2)^{r_x} e(v, w_2)^{-r_a} e(v, g_2)^{-r_\delta}, \quad R_3 = T_1^{r_x} u^{-r_\delta}.$$

    5. Compute a challenge $c = H_2(g_1||g_2||w_2||C||r||T_1||T_2||R_1||R_2||R_3) \in \mathbb{Z}_p$.
    6. Compute $s_a = r_a + ca, s_x = r_x + c x_{\mathsf{ID}}$, and $s_\delta = r_\delta + c\delta \in \mathbb{Z}_p$.

  The signature on C is defined as $\sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta)$. Then (C, σ) is sent to CSP. Upon receiving (C, σ), CSP first verifies the signature $\sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta)$:
  1. Compute $(\hat{u}, \hat{v}) = H_1(g_1||g_2||w_2||C||r) \in \mathbb{G}_2^2$ and their images in $\mathbb{G}_1 : u = \psi(\hat{u}); v = \psi(\hat{v})$.

2. Compute $\tilde{R}_1 = u^{s_a}/T_1^c, \tilde{R}_2 = e(T_2, g_2)^{s_x} \cdot e(v, w_2)^{-s_a} \cdot e(v, g_2)^{-s_\delta} \cdot (e(T_2, w_2)/e(g_1, g_2))^c$ and $\tilde{R}_3 = T_1^{s_x} u^{-s_\delta}$.
3. (C, σ) is said to be a correct cipher-signature pair if we have $c = H_2(g_1||g_2||w_2||C||r||T_1||T_2||\tilde{R}_1||\tilde{R}_2||\tilde{R}_3)$.

CSP accepts (C, σ) if it is a correct cipher-signature pair and there is no element y_{ID} in the revocation list R such that $e(T_2/y_{\mathsf{ID}}, \hat{u}) = e(T_1, \hat{v})$. The description of the revocation list will be given shortly.

- *Read*: Given a ciphertext $C = (c_1, c_2, c_3)$ and a user list S, $\mathsf{U_{ID}}$ computes

$$m = c_3/(e(c_1, h_2^{\omega_{\mathsf{ID},S}}) e(dk_{\mathsf{ID}}, c_2))^{\frac{1}{\prod_{id \in S, id \neq \mathsf{ID}} H_0(id)}},$$

where

$$\omega_{\mathsf{ID},S} = \frac{1}{\alpha} \cdot \left(\prod_{id \in S, id \neq \mathsf{ID}} (\alpha + H_0(id)) - \prod_{id \in S, id \neq \mathsf{ID}} H_0(id) \right).$$

– User Joining. When a user $\mathsf{U_{ID}}$ joins, the GM gives the user a private key pair $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ as in the registration phase. Let $ak_{\mathsf{ID}} = (x_{\mathsf{ID}}, y_{\mathsf{ID}})$. The identity ID is added to the user list S and $(\mathsf{ID}, y_{\mathsf{ID}})$ is added to a token list T. For an existing ciphertext $C = (c_1, c_2, c_3)$, the GM chooses a random $k' \in \mathbb{Z}_p^*$ and replaces C with $C' = (c_1', c_2', c_3')$, where

$$c_1' = c_1 \cdot (h_1^\alpha)^{-k'}, \quad c_2' = c_2^{\alpha + H_0(\mathsf{ID})} \cdot h_2^{k' \cdot \prod_{id \in S} (\alpha + H_0(id))}, \quad c_3' = c_3 \cdot e(h_1, h_2)^{k'}.$$

– User Revocation. If a user $\mathsf{U_{ID}}$ is revoked, the GM first removes ID from the user list S. Then the GM searches for the pair $(\mathsf{ID}, y_{\mathsf{ID}})$ in the token list T and sends y_{ID} to CSP so that CSP can add it to the revocation list R. For an existing ciphertext $C = (c_1, c_2, c_3)$, the GM chooses a random $k' \in \mathbb{Z}_p^*$ and replaces C with $C' = (c_1', c_2', c_3')$, where

$$c_1' = c_1 \cdot (h_1^\alpha)^{-k'}, \quad c_2' = c_2^{(\alpha + H_0(\mathsf{ID}))^{-1}} \cdot h_2^{k' \cdot \prod_{id \in S} (\alpha + H_0(id))}, \quad c_3' = c_3 \cdot e(h_1, h_2)^{k'}.$$

– Tracing. Given a valid cipher-signature pair $(C, \sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta))$, the GM finds the first pair $(\mathsf{ID}, y_{\mathsf{ID}}) \in T$ satisfying $e(T_2/y_{\mathsf{ID}}, \hat{u}) = e(T_1, \hat{v})$ and outputs ID.

4.2 Efficiency Analysis

The operational efficiency of our system inherits the merit of the underlying schemes. Regarding the system setup, the length of the system parameter (which is shared by all users of the system) is independent of the total number of users (which can be exponential in theory due to the use of identity-based system) and is only dependent on the maximum number of decryptors for a ciphertext. The user private key is of constant size, so as the ciphertext.

For computation requirement, pairing is the dominating operation in pairing-based cryptosystems like ours. Thanks to the design of the underlying encryption scheme, encryption itself does not require any pairing operation (the value

$e(h_1, h_2)$ can be pre-computed and put into the system parameter), and the generation of group signature only requires the computation of one pairing ($e(v, w_2)$ and $e(v, g_2)$ can be pre-computed). Verification of group signature is a little bit more costly (two pairing operations which cannot be pre-computed). However, this part is done by the CSP which is assumed to be computationally powerful. Decryption of a ciphertext requires two pairing operations, which is again independent of the total number of users of the system.

For the ciphertext update operation, we must compare its performance with the naïve "decrypt-then-encrypt" approach. It is clear that our update algorithm outperforms this approach since no pairing is required by ours but two evaluations of pairing are needed in the decryption. Indeed, it is not difficult to see that our update operation's computational complexity is similar to that of the encryption algorithm. We consider this as a natural requirement.

4.3 Security Analysis

We start by listing out the required number-theoretical assumptions.

The General (Decisional) Diffie-Hellman Exponent Assumption. Boneh *et al.* [2] introduced the General Diffie-Hellman Exponent (GDHE) assumption. Consider a symmetric bilinear group system $(p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$. Let s, n be positive integers, $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$. We require that $p_1 = q_1 = 1$. For a set Ω, a function $h : \mathbb{F}_p \to \Omega$ and a vector $x_1, x_2, \ldots, x_n \in \mathbb{F}_p$, we write

$$h(P(x_1, x_2, \ldots, x_n)) = (h(p_1(x_1, \ldots, x_n)), \ldots, h(p_s(x_1, \ldots, x_n))) \in \Omega^s.$$

We use similar notation for the s-tuple Q. Let $g \in \mathbb{G}$ be a generator of $\mathbb{G}$ and set $g_T = e(g, g) \in \mathbb{G}_T$. We define the (P, Q, f)-General Diffie-Hellman Problem in $\mathbb{G}$ as follows.

Definition 1 ((P, Q, f)-GDHE). *Given the vector*

$$H(x_1, \ldots, x_n) = (g^{P(x_1,\ldots,x_n)}, g_T^{Q(x_1,\ldots,x_n)}) \in \mathbb{G}^s \times \mathbb{G}_T^s,$$

compute $g_T^{f(x_1,\ldots,x_n)} \in \mathbb{G}_T$.

Definition 2 ((P, Q, f)-GDDHE). *Given $H(x_1, \ldots, x_n)$ as above and an element $T \in \mathbb{G}_T$, decide whether $T = g^{f(x_1,\ldots,x_n)}$.*

The data confidentiality of our instantiation is based on the decisional version of the GDHE assumption: the General Decisional Diffie-Hellman Exponent (GDDHE) assumption. The specific choice of P, Q and f will be given in Appendix. For the unforgeability of the group signature, we require the following assumption that is outside of the GDHE framework.

The Strong Diffie-Hellman Assumption. The traceability of our instantiation is based on the q-Strong Diffie-Hellman assumption (q-SDH), which was also used by Boneh and Boyen in the security proof of their short signature scheme [1].

Definition 3 (q-SDH). *The q-SDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows: given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \cdots, g_2^{\gamma^q})$ as input, output a pair $(g_1^{1/(\gamma+x)}, x)$, where $x \in \mathbb{Z}_p^*$.*

The Decisional Linear Assumption. The anonymity of our instantiation is based on the hardness of Decisional Linear (D-Lin) problem, introduced by Boneh, Boyen, and Shacham [3].

Definition 4 (D-Lin). *Given $u, v, h, u^a, v^b, h^c \in \mathbb{G}_1$ as input, decide whether $a + b = c$.*

With our generic design, it is easy to argue the security of our cloud storage system. Basically, the security guarantees follow from those of the respective underlying schemes. For confidentiality, as we have pointed out in Section 3.4, extra care must be taken to have a secure update. The formal security guarantee against adaptive chosen-ciphertext decryption and update attacks for our extension of Delerablée's scheme can be found in Appendix.

5 Conclusion

We studied the problem of building a secure cloud storage system. We believe that supporting dynamic users and data provenance are essential in cloud storage. In this paper, we proposed a cryptographic design of such systems with a number of desirable features. During the course of our design we also devised new cryptographic techniques to build a provably secure dynamic system. We also discussed different features that existing anonymous authentication mechanisms can provide. We hope that our study can help cloud storage service providers and cloud clients to pick the right system to use for their application scenarios.

References

1. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
2. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
3. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
4. Boneh, D., Shacham, H.: Group Signatures with Verifier-Local Revocation. In: Proceedings of ACM Conference on Computer and Communications Security (CCS 2004), pp. 168–177. ACM, New York (2004)

5. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
6. Chase, M., Chow, S.S.M.: Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In: Proceedings of ACM Conference on Computer and Communications Security (CCS 2010), pp. 121–130 (2009)
7. Chow, S.S.M.: Real Traceable Signatures. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009)
8. Chow, S.S.M.: Removing Escrow from Identity-Based Encryption. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 256–276. Springer, Heidelberg (2009)
9. Delerablée, C.: Identity-Based Broadcast Encryption with Constant Size Ciphertexts and Private Keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)
10. Even, S., Goldreich, O., Micali, S.: On-line/Off-line Digital Signatures. J. Cryptology 9(1), 35–67 (1996)
11. Lamport, L.: Constructing Digital Signatures from a One Way Function. Technical report (1979)
12. Lu, R., Lin, X., Liang, X., Shen, X.S.: Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing. In: Proceedings of ACM Symposium on Information, Computer & Communication Security (ASIACCS 2010), pp. 282–292. ACM, New York (2010)
13. Rabin, M.O.: Digitalized Signatures. In: Foundations of Secure Computations, pp. 155–166. Academic Press, London (1978)
14. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011); Also available at Cryptology ePrint Archive, Report 2008/290

A Security with Update

We prove the IND-sID-CPA* confidentiality of our construction under the general decision Diffie-Hellman exponent (GDDHE) framework introduced by [2]. The assumption is described as follows. The intractability of this assumption can be proved using the similar techniques in [9].

Definition 5 $((f, g, F) - \mathsf{GDDHE})$. *Let* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot,\cdot))$ *be a bilinear group system defined above. Let* f *and* g *be two* coprime *polynomials of orders* t *and* n*, respectively. Let* g_0, h_0 *be two generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively. Given*

$$g_0, \quad g_0^{\alpha}, \quad \cdots, g_0^{\alpha^{t-1}}, g_0^{\alpha \cdot f(\alpha)}$$
$$g_0^{k \cdot \alpha \cdot f(\alpha)}, g_0^{k \cdot \alpha^2 \cdot f(\alpha)}, \cdots, g_0^{k \cdot \alpha^s \cdot f(\alpha)}$$
$$h_0, \quad h_0^{\alpha}, \quad \cdots, h_0^{\alpha^{2n}}, h_0^{k \cdot g(\alpha)}$$

and $T \in \mathbb{G}_T$*, decide whether* T *is equal to* $e(g_0, h_0)^{k \cdot f(\alpha)}$.

We first give the following lemma to show that the adversary can get more information about the challenge ciphertext without influencing the security of our dynamic identity-based broadcast encryption (D-IBBE).

Lemma 1. *Consider the IND-sID-CPA* security game between an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, D-IBBE remains secure even if $\mathcal{A}$ can ask for any challenge ciphertext encrypted for $S^* \backslash \mathcal{T}^*$, where $\mathcal{T}^* \subset S^*$.*

Proof. Given the bilinear group system $BGS = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot,\cdot))$ and the input of $(f, g, F) - \mathsf{GDDHE}$ problem, $\mathcal{S}$ plays the IND-sID-CPA* game with $\mathcal{A}$ as follows.

Init: The adversary $\mathcal{A}$ outputs a target set $S^* = \{\mathsf{ID}_1^*, \cdots, \mathsf{ID}_s^*\}$ of identities that it wants to attack (with $s \leq n-1$).

Setup: $\mathcal{S}$ first defines $h_1 = g_0^{f(\alpha)}$ and sets

$$\begin{aligned} h_2 &= h_0^{\prod_{i=t+s+1}^{t+n}(\alpha+x_i)}, \\ w_1 &= g_0^{\alpha \cdot f(\alpha)} = h_1^{\alpha}, \\ e(h_1, h_2) &= e(g_0, h_0)^{f(\alpha)\cdot\prod_{i=t+s+1}^{t+n}(\alpha+x_i)}. \end{aligned}$$

Note that the degree of $\prod_{i=t+s+1}^{t+n}(\alpha + x_i)$ is $(n-s)$. All other values and functions $(g_1, g_2, H_0(\cdot), H_1(\cdot), H_2(\cdot), w_2)$ are chosen as in the scheme. The public parameter is defined as

$$PK = (BGS, H_0(\cdot), H_1(\cdot), H_2(\cdot), g_1, g_2, h_2, h_2^{\alpha}, \cdots, h_2^{\alpha^n}, w_1, w_2, e(h_1, h_2)),$$

where $h_2^{\alpha^n}$ can be computed from $\{h_0^{\alpha^i}\}_{i \leq (2n-s)}$.

Hash Queries: Same as the original scheme, and omitted.

Query Phase 1: Same as the original scheme, and omitted.

Challenge: $\mathcal{A}$ asks for any challenge ciphertext encrypted for $S^* \backslash \mathcal{T}^*$, where $\mathcal{T}^* \subset S^*$. Let $S = \{x_i\}_{\mathsf{ID}_i \in \mathcal{S}^*, H_0(0||\mathsf{ID}_i) = x_i}$. Let $\mathcal{T} = \{x_i\}_{\mathsf{ID}_i \in \mathcal{T}^*, H_0(0||\mathsf{ID}_i) = x_i}$. When $\mathcal{T} = \emptyset$, we can encrypt the message m_b:

$$c_1 = g_0^{-k\cdot\alpha\cdot f(\alpha)}, \qquad c_2 = h_0^{k\cdot g(\alpha)}, \qquad c_3 = T^{\prod_{i=t+s+1}^{t+n} x_i} \cdot e(g_0^{k\cdot\alpha\cdot f(\alpha)}, h_0^{q(\alpha)})$$

with $q(\alpha) = \frac{1}{\alpha}(\prod_{i=t+s+1}^{t+n}(\alpha + x_i) - \prod_{i=t+s+1}^{t+n} x_i)$ which is of degree $(n-s-1)$.

One can verify that

$$c_1 = w_1^{-k}, \qquad c_2 = h_0^{\prod_{i=t+s+1}^{t+n}(\alpha+x_i)\prod_{i=t+1}^{t+s}(\alpha+x_i)} = h_2^{k\prod_{i=t+1}^{t+s}(\alpha+x_i)}$$

and $c_3 = e(h_1, h_2)^k$ if $T = e(g_0, h_0)^{k\cdot f(\alpha)}$.

Now, to encrypt when $\mathcal{T}^* \neq \emptyset$, the idea is to define c_2 as if it uses the random factor $k(\prod_{\mathcal{T}}(\alpha + x_i))$ and "adjusts" c_1 and c_3 accordingly. After that, we re-randomize all components. Note that the size of $\mathcal{T}$ is at most $s-1$.

Computing $c_1 = g_0^{-k\cdot(\prod_{\mathcal{T}}(\alpha+x_i))\cdot\alpha\cdot f(\alpha)}$ requires $g_0^{k\cdot\alpha\cdot f(\alpha)}, g_0^{k\cdot\alpha^2\cdot f(\alpha)}, \cdots, g_0^{k\cdot\alpha^s\cdot f(\alpha)}$.

We define

$$q'(\alpha) = \frac{1}{\alpha}\Big(\prod_{i=t+s+1}^{t+n}(\alpha + x_i)\cdot\prod_{\mathcal{T}}(\alpha + x_i) - \prod_{i=t+s+1}^{t+n}(x_i)\cdot\prod_{\mathcal{T}}(x_i)\Big)$$

which is of degree $(n-2)$, and compute

$$K = T^{\prod_{i=t+s+1}^{t+n}(x_i)\cdot\prod_{\mathcal{T}}(x_i)}\cdot e(g_0^{k\cdot\alpha\cdot f(\alpha)}, h_0^{q'(\alpha)}).$$

If $T = e(g_0, h_0)^{k\cdot f(\alpha)}$, the discrete logarithm of K to the base $v = e(g_0, h_0)^{f(\alpha)}$ is

$$\begin{aligned}
&k\prod_{i=t+s+1}^{t+n}(x_i)\cdot\prod_{\mathcal{T}}(x_i) + k\cdot\alpha\cdot q'(\alpha)\\
&= k\Big(\prod_{i=t+s+1}^{t+n}(x_i)\cdot\prod_{\mathcal{T}}(x_i) + \prod_{i=t+s+1}^{t+n}(\alpha + x_i)\cdot\prod_{\mathcal{T}}(\alpha + x_i) - \prod_{i=t+s+1}^{t+n}(x_i)\cdot\prod_{\mathcal{T}}(x_i)\Big)\\
&= k\Big(\prod_{i=t+s+1}^{t+n}(\alpha + x_i)\cdot\prod_{\mathcal{T}}(\alpha + x_i)\Big)
\end{aligned}$$

Recall that $e(h_1, h_2) = e(g_0, h_0)^{f(\alpha)\cdot\prod_{i=t+s+1}^{t+n}(\alpha+x_i)}$, $c_3 = e(h_1, h_2)^{k\cdot\prod_{\mathcal{T}}(\alpha+x_i))}$ as desired.

Re-randomizations of all components are straightforward.

B Intractability of Our $(f, g, F) - \mathsf{GDDHE}$

In this section, we prove the intractability of distinguishing the two distributions involved in our $(f, g, F) - \mathsf{GDDHE}$ problem. We first review some results on the General Diffie-Hellman Exponent Problem, from [2]. In order to be the most general, we assume the easiest case for the adversary: when $\mathbb{G}_1 = \mathbb{G}_2$, or at least that an isomorphism that can be easily computed in either one or both ways is available.

Theorem 1 ([2]). *Let $P, Q \in \mathbb{F}_p[X_1, \cdots, X_m]$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$ and let $F \in \mathbb{F}_p[X_1, \cdots, X_m]$. Let d_P (respectively d_Q, d_F) denote the maximal degree of elements of P (respectively of Q, F) and pose $d = \max(2d_P, d_Q, d_F)$). If $F \notin \langle P, Q\rangle$ then for any generic-model adversary $\mathcal{A}$ that makes a total of at most q queries to the oracles (group operations in $\mathbb{G}, \mathbb{G}_T$ and evaluations of e) which is given $H(x_1, \cdots, x_m)$ as input and tries to distinguish $g^{F(x_1,\cdots,x_m)}$ from a random value in $\mathbb{G}_T$, one has $\mathsf{Adv}(\mathcal{A}) \leq \frac{(q+2s+2)^2\cdot d}{2p}$.*

Proof (of Generic Security of Our $(f, g, F) - \mathsf{GDDHE}$). We need to prove that our $(f, g, F) - \mathsf{GDDHE}$ lies in the scope of Theorem 1. Our proof will be very similar to that in [9] since the argument does not really depend on the maximum

degree of the polynomials involved. (Of course, this certainly affects the final bound regarding the advantage of a generic adversary.) Similar to [9], we consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and thus pose $h_0 = g_0^{\beta}$. Our problem can be reformulated as $(P, Q, F) - \mathsf{GDDHE}$ where

$$\begin{aligned} P = (\ & 1, \alpha, \cdots, \alpha^{t-1}, \alpha \cdot f(\alpha) \\ & k \cdot \alpha \cdot f(\alpha), k \cdot \alpha^2 \cdot f(\alpha), \cdots, k \cdot \alpha^s \cdot f(\alpha) \\ & \beta, \beta \cdot \alpha, \cdots, \beta \cdot \alpha^{2n}, k \cdot \beta \cdot g(\alpha)\ ) \\ Q &= 1 \\ F &= k \cdot \beta \cdot f(\alpha), \end{aligned}$$

and thus $m = 3$ and $s = t + s + 2n + 3$. We have to show that F is independent of (P, Q). By making all possible products of two polynomials from P which are multiples of $k \cdot \beta$, we want to prove that no linear combination among the polynomials from the list R below leads to F:

$$\begin{aligned} R = (\ & k \cdot \beta \cdot \alpha \cdot f(\alpha), k \cdot \beta \cdot \alpha^2 \cdot f(\alpha), \cdots, k \cdot \beta \cdot \alpha^{2n+s} \cdot f(\alpha), \\ & k \cdot \beta \cdot g(\alpha), k \cdot \beta \cdot \alpha \cdot g(\alpha), k \cdot \beta \cdot \alpha^{t-1} \cdot g(\alpha), \\ & k \cdot \beta \cdot \alpha \cdot f(\alpha) \cdot g(\alpha)\ ) \end{aligned}$$

where the first line is "generated" from $\{k \cdot \alpha^i \cdot f(\alpha)\}$ and the second line is "generated" from $\{k \cdot \beta g(\alpha)\}$.

Note that the last polynomial can be written as a linear combination of the polynomials from the first line. Also, taking out the "common factors" of $k \cdot \beta$ from R and F, we therefore simplify the task to refuting a linear combinations of elements of the list R' below which leads to $f(\alpha)$:

$$\begin{aligned} R' = (\ & \alpha \cdot f(\alpha), \alpha^2 \cdot f(\alpha), \cdots, \alpha^{2n+s} \cdot f(\alpha), \\ & g(\alpha), \alpha \cdot g(\alpha), \alpha^{t-1} \cdot g(\alpha)\ ) \end{aligned}$$

Any such linear combination can be written as

$$f(\alpha) = A(\alpha) \cdot f(\alpha) + B(\alpha) \cdot g(\alpha)$$

where A and B are polynomials such that $A(0) = 0$ (since $f(\alpha)$ does not exist in R'), $\deg A \leq 2n + s$ and $\deg B \leq t - 1$.

Since f and g are coprime by assumption, we must have $f|B$. Since $\deg f = t$ and $\deg B \leq t - 1$, this implies $B = 0$. Hence $A = 1$ which contradicts $A(0) = 0$.

C A Variant with Chosen-Ciphertext Security

In this part we present a variant of our system with chosen-ciphertext security. Before we describe our construction, we first introduce the notion of strong one-time signature, which is used in our construction.

C.1 Strong One-Time Signatures

A strong one-time signature scheme is a signature scheme with the difference that each private key is used only once for signature generation. Like a normal signature scheme, a one-time signature scheme consists of three algorithms, namely, SKeyGen (generates a signing key sk and a verification key vk), Sign (generates a signature ρ on a message m using sk) and Vrfy (verifies a message/signature pair (m, ρ) using vk). The security requirement is that the adversary is unable to generate a valid signature of a new message after making at most one signing query. The strong unforgeability means that it is even impossible for the adversary to generate a new signature on a message whose signature is already known. Since the introduction in [13,11], there have been many schemes proposed (e.g., using one-way function paradigm [10]) and many of them are efficient.

C.2 Our Construction

Let $(n-1)$ be the maximum number of receivers the file can be encrypted to and {SKeyGen, Sign, Vrfy} be a one-time signature scheme.

- Setup. On input a security parameter κ, the GM performs the following steps:
  - It generates a bilinear group system $BGS = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \psi(\cdot), e(\cdot,\cdot))$ as described above where $p \geq 2^{\kappa}$ and cryptographic hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_1 : \{0,1\}^* \to \mathbb{G}_2^2$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_p$.
  - It randomly chooses $h_1 \in_R \mathbb{G}_1, h_2, g_2 \in_R \mathbb{G}_2$ and computes $g_1 = \psi(g_2)$.
  - It randomly chooses two secrets $\alpha, \gamma \in \mathbb{Z}_p^*$ and sets $w_1 = h_1^{\alpha}, w_2 = g_2^{\gamma}$.

  The public parameter is defined as

  $$PK = (BGS, H_0(\cdot), H_1(\cdot), H_2(\cdot), g_1, g_2, h_2, h_2^{\alpha} \ldots, h_2^{\alpha^n}, w_1, w_2, e(h_1, h_2)).$$

  The master secret key is $MK = (h_1, \alpha, \gamma)$.
- User Registration. For each user $\mathsf{U_{ID}}$, the GM issues a private key $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$, where

  $$ak_{\mathsf{ID}} = (x_{\mathsf{ID}} \in_R \mathbb{Z}_p, g_1^{1/(\gamma + x_{\mathsf{ID}})}) \quad \text{and} \quad dk_{\mathsf{ID}} = h_1^{1/(\alpha + H_0(0||\mathsf{ID}))}.$$

  The identity ID is added to the access user list S and $(\mathsf{ID}, g_1^{1/(\gamma + x_{\mathsf{ID}})})$ is added to a token list T.
- Data Access. This phase is performed by a user $\mathsf{U_{ID}}$ and the service provider CSP. The user $\mathsf{U_{ID}}$ with the private key $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ can read or write data as follows.
  - *Write*: Given a message m and a user list S, $\mathsf{U_{ID}}$ performs the following steps:
    1. Run $\mathsf{SKeyGen}(1^{\kappa})$ to generate $\{vk, sk\}$;
    2. Choose a random $k \in \mathbb{Z}_p^*$;
    3. Compute $\mathfrak{C} = (c_1, c_2, c_3)$ as follows.

    $$c_1 = w_1^{-k}, c_2 = h_2^{k \cdot (\alpha + H_0(1||vk)) \prod_{id \in S} (\alpha + H_0(0||id))}, c_3 = m \cdot e(h_1, h_2)^k.$$

4. Run $\mathsf{Sign}_{sk}(\mathfrak{C})$ to generate ρ, the ciphertext is $C = \langle \mathfrak{C}, vk, \rho \rangle$.
5. Let $ak_{\mathsf{ID}} = (x_{\mathsf{ID}}, y_{\mathsf{ID}})$. U_{ID} generates a signature on C:
   (a) Pick a random nonce $r \in \mathbb{Z}_p$ and let $(\hat{u}, \hat{v}) = H_1(g_1||g_2||w_2||C||r) \in \mathbb{G}_2^2$. Compute their images in $\mathbb{G}_1 : u = \psi(\hat{u}); v = \psi(\hat{v})$.
   (b) Select a random $a \in \mathbb{Z}_p$ and compute $T_1 = u^a$ and $T_2 = y_{\mathsf{ID}} v^a$.
   (c) Set $\delta = a x_{\mathsf{ID}}$. Pick three random blinding values r_a, r_x and $r_\delta \in \mathbb{Z}_p$.
   (d) Compute helper values

$$R_1 = u^{r_a}, R_2 = e(T_2, g_2)^{r_x} \cdot e(v, w_2)^{-r_a} \cdot e(v, g_2)^{-r_\delta}, R_3 = T_1^{r_x} \cdot u^{-r_\delta}.$$

   (e) Compute a challenge $c = H_2(g_1||g_2||w_2||C||r||T_1||T_2||R_1||R_2||R_3) \in \mathbb{Z}_p$.
   (f) Compute $s_a = r_a + ca, s_x = r_x + c x_{\mathsf{ID}}$, and $s_\delta = r_\delta + c\delta \in \mathbb{Z}_p$.

   The signature on C is defined as $\sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta)$. Then (C, σ) is sent to CSP. Upon receiving (C, σ), CSP first verifies the signature $\sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta)$:
   (a) Compute $(\hat{u}, \hat{v}) = H_1(g_1||g_2||w_2||C||r) \in \mathbb{G}_2^2$ and their images in $\mathbb{G}_1 : u = \psi(\hat{u}); v = \psi(\hat{v})$.
   (b) Compute $\tilde{R}_1 = u^{s_a}/T_1^c, \tilde{R}_2 = e(T_2, g_2)^{s_x} \cdot e(v, w_2)^{-s_a} \cdot e(v, g_2)^{-s_\delta} \cdot (e(T_2, w_2)/e(g_1, g_2))^c$ and $\tilde{R}_3 = T_1^{s_x} u^{-s_\delta}$.
   (c) (C, σ) is said to be a correct cipher-signature pair if

$$c = H_2(g_1||g_2||w_2||C||r||T_1||T_2||\tilde{R}_1||\tilde{R}_2||\tilde{R}_3).$$

   CSP accepts (C, σ) if it is a correct cipher-signature pair and there is no element y_{ID} in the revocation list R such that $e(T_2/y_{\mathsf{ID}}, \hat{u}) = e(T_1, \hat{v})$. The description of the revocation list will be given shortly.
6. *Read*: Given a ciphertext $C = \langle \mathfrak{C} = (c_1, c_2, c_3), vk, \rho \rangle$ and a user list S, U_{ID} performs the following steps:
   (a) Return $\perp$ if $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 0$.
   (b) Compute

$$m = c_3 / (e(c_1, h_2^{\omega_{\mathsf{ID},S}}) e(dk_{\mathsf{ID}}, c_2))^{\frac{1}{H_0(1||vk) \cdot \prod_{id \in S, id \neq \mathsf{ID}} H_0(0||id)}},$$

   where

$$\omega_{\mathsf{ID},S} = \frac{1}{\alpha} \cdot ((\alpha + H_0(1||vk)) \textstyle\prod_{id \in S, id \neq \mathsf{ID}} (\alpha + H_0(0||id)) - H_0(1||vk) \textstyle\prod_{id \in S, id \neq \mathsf{ID}} H_0(0||id)).$$

– User Joining. When a user U_{ID} joins, the GM gives the user a private key pair $(ak_{\mathsf{ID}}, dk_{\mathsf{ID}})$ as in the registration phase. Let $ak_{\mathsf{ID}} = (x_{\mathsf{ID}}, y_{\mathsf{ID}})$. The identity ID is added to the user list S and $(\mathsf{ID}, y_{\mathsf{ID}})$ is added to a token list T.
For an existing ciphertext $C = \langle \mathfrak{C} = (c_1, c_2, c_3), vk, \rho \rangle$, the GM performs the following steps:

1. Return $\perp$ if $\mathsf{Vrfy}_{vk}(C, \rho) = 0$.
2. Run $\mathsf{SKeyGen}(1^\kappa)$ to generate $\{vk', sk'\}$;
3. Choose a random $k' \in \mathbb{Z}_p^*$;
4. Compute $c_1' = c_1 \cdot (h_1^\alpha)^{-k'}$;
5. Compute $c_2' = c_2^{\frac{(\alpha+H_0(0||\mathsf{ID}))(\alpha+H_0(1||vk'))}{\alpha+H_0(1||vk)}} \cdot h_2^{k' \cdot (\alpha+H_0(1||vk')) \cdot \prod_{id \in S}(\alpha+H_0(0||id))}$;
6. Compute $c_3' = c_3 \cdot e(h_1, h_2)^{k'}$.
7. Define $\mathfrak{C}' = (c_1', c_2', c_3')$.
8. Run $\mathsf{Sign}_{sk}(\mathfrak{C}')$ to generate ρ', the ciphertext is $C' = \langle \mathfrak{C}', vk', \rho' \rangle$.

– User Revocation. If a user $\mathsf{U_{ID}}$ is revoked, the GM first removes ID from the user list S. Then the GM searches for the pair $(\mathsf{ID}, y_{\mathsf{ID}})$ in the token list T and sends y_{ID} to CSP so that CSP can add it to the revocation list R.
For an existing ciphertext $C = \langle \mathfrak{C} = (c_1, c_2, c_3), vk, \rho \rangle$, the GM performs the following steps:
1. Return $\perp$ if $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 0$.
2. Run $\mathsf{SKeyGen}(1^\kappa)$ to generate $\{vk', sk'\}$;
3. Choose a random $k' \in \mathbb{Z}_p^*$;
4. Compute $c_1' = c_1 \cdot (h_1^\alpha)^{-k'}$;
5. Compute $c_2' = c_2^{\frac{\alpha+H_0(1||vk')}{(\alpha+H_0(0||\mathsf{ID}))(\alpha+H_0(1||vk))}}) \cdot h_2^{k' \cdot (\alpha+H_0(1||vk')) \cdot \prod_{id \in S}(\alpha+H_0(0||id))}$;
6. Compute $c_3' = c_3 \cdot e(h_1, h_2)^{k'}$.
7. Define $\mathfrak{C}' = (c_1', c_2', c_3')$.
8. Run $\mathsf{Sign}_{sk}(\mathfrak{C}')$ to generate ρ', the ciphertext is $\langle \mathfrak{C}', vk', \rho' \rangle$.

– Tracing. Given a valid cipher-signature pair $(C, \sigma = (r, T_1, T_2, c, s_a, s_x, s_\delta))$, the GM finds the first pair $(\mathsf{ID}, y_{\mathsf{ID}}) \in T$ satisfying $e(T_2/y_{\mathsf{ID}}, \hat{u}) = e(T_1, \hat{v})$ and outputs ID. The GM outputs "$\perp$" if that pair does not exist.

C.3 Security Analysis

Now, the IND-sID-CCA confidentiality is given by the following theorem. Let Π' denote the IND-sID-CPA* construction provided in Section 4.1 and Π denote the construction above.

Theorem 2. *If Π' is IND-sID-CPA* secure and* {SKeyGen, Sign, Vrfy} *is a strong one-time signature scheme, then Π is IND-sID-CCA secure.*

Proof. (sketch) Assume we are given a PPT adversary $\mathcal{A}$ attacking Π in an adaptive chosen-ciphertext attack. Say a ciphertext $\langle \mathfrak{C}, vk, \rho \rangle$ is *valid* if $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 1$. Let $\{\langle \mathfrak{C}_i^*, vk_i^*, \rho_i^* \rangle\}$ denote the set of challenge ciphertexts received by $\mathcal{A}$ during a particular run of the experiment, and let Forge denote the event that $\mathcal{A}$ submits a valid ciphertext $\langle \mathfrak{C}, vk, \rho \rangle$ to the decryption oracle where $vk \in \{vk_i^*\}$ (we may assume that $\{vk_i^*\}$ is chosen at the outset of the experiment so this event is well-defined even before $\mathcal{A}$ is given the challenge ciphertext). Recall also that $\mathcal{A}$ is disallowed from submitting the challenge ciphertext to the decryption oracle, and transforming the ciphertext to be decryptable by anyone outside S^*, once the challenge ciphertext is given to $\mathcal{A}$.

It is easy to show that the probability that Forge happens is negligible.

We use $\mathcal{A}$ to construct a PPT adversary $\mathcal{A}'$ which attacks Π'. Define adversary $\mathcal{A}'$ as follows:

1. $\mathcal{A}'(1^\kappa)$ runs $\mathsf{SKeyGen}(1^\kappa)$ q_U times to generate $\{vk_i^*, sk_i^*\}$, and outputs the "target" identity set as $\{0||\mathsf{ID}_i^*\}_{\mathsf{ID}_i^* \in S} \cup \{1||vk_i^*\}$.
2. $\mathcal{A}'$ is given a master public key PK, and runs $\mathcal{A}(1^\kappa, PK)$ in turn.
3. When $\mathcal{A}$ makes extraction oracle query on $\mathsf{ID} \notin S$, $\mathcal{A}'$ issues the query $(0||\mathsf{ID})$ to its own extraction oracle and forwards the result.
4. When $\mathcal{A}$ makes decryption oracle query on the ciphertext $\langle \mathfrak{C}, vk, \rho \rangle$ adversary $\mathcal{A}'$ proceeds as follows:
   (a) If $vk \in \{vk_i^*\}$ then $\mathcal{A}'$ checks whether $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 1$. If so, $\mathcal{A}'$ aborts and outputs a random bit. Otherwise, it simply responds with $\perp$.
   (b) If $vk \notin \{vk_i^*\}$ and $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 0$ then $\mathcal{A}'$ responds with $\perp$.
   (c) If $vk \notin \{vk_i^*\}$ and $\mathsf{Vrfy}_{vk}(\mathfrak{C}, \rho) = 1$, then $\mathcal{A}'$ makes the extraction query to obtain the secret key of $1||vk$, decrypts C and responds with the result.
5. Since decryption oracle queries can be simulated, update oracle queries can be easily simulated too.
6. At some point, $\mathcal{A}$ outputs a set $\mathcal{T}_j^*$, for $\mathcal{T}_j^* \subset S^*$, where $1 \leq j \leq q_U$. $\mathcal{A}'$ outputs the set $\{0||\mathsf{ID}_i^*\}_{\mathsf{ID}_i^* \in \mathcal{T}_j^*} \cup \{1||vk_i^*\}_{i \neq j}$. In return, $\mathcal{A}'$ is given a challenge ciphertext $\mathfrak{C}_i^*$. It then computes $\rho_i^* \leftarrow \mathsf{Sign}_{sk_i^*}(\mathfrak{C}_i^*)$ and returns $\langle \mathfrak{C}_i^*, vk_i^*, \rho^* \rangle$ to $\mathcal{A}$. This effectively simulates the maximum number of q_U update of the challenge ciphertext that can be queried by $\mathcal{A}$.
7. $\mathcal{A}$ may continue to make update and decryption oracle queries, and these are answered by $\mathcal{A}'$ as before, with the natural restrictions regarding challenge ciphertext.
8. Finally, $\mathcal{A}$ outputs a guess b'; this same guess is output by $\mathcal{A}'$.

Note that $\mathcal{A}'$ represents a legal adversarial strategy for attacking Π'; in particular, $\mathcal{A}'$ never requests the secret key corresponding to any of the target identity in the set $\{0||\mathsf{ID}_i^*\}_{\mathsf{ID}_i^* \in S} \cup \{1||vk_i^*\}$. Furthermore, $\mathcal{A}'$ provides a perfect simulation for $\mathcal{A}$ until event Forge occurs.