Attribute-Based Encryption with Key Cloning Protection

M. J. Hinek[1], S. Jiang[1,2], R. Safavi-Naini[1], and S. F. Shahandashti[3]

[1] *i*CORE Information Security Lab, Dept of CS, University of Calgary, Calgary, Canada.
[2] School of Computer Science and Technology, University of Electronic Science and Technology of China, Chengdu 610054, China.
[3] School of Computer Science and Software Engineering, University of Wollongong, Australia.

mjhinek@alumni.uwaterloo.ca, jiangshq@calliope.uwaterloo.ca, rei@ucalgary.ca , sfs166@uow.edu.au

November 12, 2008

Abstract. In this work, we consider the problem of key cloning in attribute-based encryption schemes. We introduce a new type of attribute-based encryption scheme, called token-based attribute-based encryption, that provides strong deterrence for key cloning, in the sense that delegation of keys reveals some personal information about the user. We formalize the security requirements for such a scheme in terms of indistinguishability of the ciphertexts and two new security requirements which we call uncloneability and privacy-preserving. We construct a privacy-preserving uncloneable token-based attribute-based encryption scheme based on Cheung and Newport's ciphertext-policy attribute-based encryption scheme and prove the scheme satisfies the above three security requirements. We discuss our results and show directions for future research.

Keywords: Attribute-Based Encryption (ABE), Access Control, Key Delegation, User Revocation.

1 Introduction

Attribute-Based Encryption (ABE) schemes provide an elegant way of encrypting messages such that users satisfying an access control policy can decrypt the ciphertext. In an ABE scheme, users' private keys are issued by a trusted authority which assigns privileges to the users to access encrypted content and hence allows the system to be a powerful and efficient approach to controlling users' access. There are two flavors for ABE schemes. In a *ciphertext-policy (CP-ABE)* scheme, a message is encrypted with a specific access policy determined by the encrypter. For example, when CP-ABE is used to control access to messages in a bulletin board of a company, the encrypter can specify that the message may only be decrypted by "technical staff in Department A". In a CP-ABE scheme, the users' keys, issued by the trusted authority, reflects their attributes and defines their access rights. A user whose attributes matches the decryption policy associated with a ciphertext can decrypt that ciphertext. A second flavor of ABE systems are *key-policy (KP-ABE)* schemes. Here, users' keys (issued by the systems trusted authority) captures an access structure (policy) while ciphertexts are associated with attribute sets; again decryption requires that the users' policy 'match' the attribute set of the ciphertext. In both types of ABE systems the encryption system can be seen as a secure method of enforcing access control policy.

Linking access rights to users' private keys means that if a user gives away their private key, then the system's access control policy is breached. This is because private keys issued to users reflect the users' privileges. This is very different from users leaking their private keys in a traditional public key encryption (PKE) system. In these latter types of systems, a user's private key is chosen by the user and does not reflect any access control policy of the system. Here users are wary of giving away their private keys because this will give away their personal information. The reluctance of users to expose such information makes the problem of 'giving away' private keys not very critical in a PKE scheme, while in ABE scheme this presents a serious threat and security breach. A similar problem exists in broadcast encryption systems where a ciphertext can be decrypted by a specified subgroup of users. The issue of key cloning in such system is addressed by introducing mechanism such as *decoder fingerprinting* and *traitor tracing* [7, 6].

In all known ABE schemes [18, 14, 10, 3, 16, 2, 19], users can give away their private keys (referred to as *key cloning*). There are many examples in which key cloning undermines the intended purpose of the encryption scheme and thus providing protection against it is an important issue. For example, Staddon et al. [19] present an ABE scheme for document redaction. Here, parts of a document are 'blacked-out' by encrypting them with the ABE scheme. Only users with attributes satisfying the decryption policy of the redacted document are then able to decrypt these blacked-out regions and fully read the document. For instance, legal documents might black-out the names of minors in a report. The decryption policy might specify that only law enforcement personnel, judges, or high ranking politicians are able to decrypt this information. If a decryption key is leaked to the press in such a scenario, the privacy of the minors in the document are violated and perhaps local laws too. Another suggested use for ABE schemes, by Bethencourt et al. [3], is for encryption of documents by the FBI. In such a scheme, FBI agents encrypt memos so that only people that have certain credentials are able to access it. Since there is no recourse for delegating keys in the scheme, a low paid agent might be tempted to sell his decryption key to the press or a private investigator. In both of these examples, ABE provides an elegant solution to a specific access control problem. However, these examples are 'secure' if we assume that no user clone their private keys.

We note that in existing ABE systems there are many other ways that malicious users can misuse their keys. For example they can construct *pseudo-keys*: a private key constructed by a user A such that, although it looks different from A's own private key, it allows the holder of the pseudo-key to decrypt messages that A can decrypt (or possibly only a subset of the ciphertexts that A can decrypt). It is also possible for users A and B to collude with each other to construct new keys that can decrypt all (or some) of the ciphertexts that both A and B can decrypt. Even in ABE systems in which key delegation is a feature, for example [3], there is no mechanism to revoke the delegated keys or any constraints on who can issue delegated keys. In all of these scenarios, one or more users of the system can, effectively, issue access rights to another entity by giving them a key that can decrypt some ciphertexts: a role that in a secure implementation of the system is designated only to the system trusted authority.

The aim of this work is to address uncloneability of keys and can be seen as a first step towards the more general problem of protection against key misuse in ABE schemes. We only consider CP-ABE schemes although our ideas and approaches can be extended to KP-ABE schemes.

1.1 Our Contributions

The discussion above motivates us to consider a new security property for ABE schemes, called *uncloneability*, to capture the protection of an ABE system against users who are misusing their access rights (decryption privileges) by *cloning* their keys for other entities.

To achieve this security property, we introduce a new type of ABE system called *token-based ABE (tk-ABE)* that provides strong disincentive (or *deterrence*) for users to misuse their privileges. The intuition is that ABE private keys are *impersonal* keys and related to users' attributes that are group properties. So users would be willing to share them with their 'friends' or because of other incentives. However, if the keys are personalized and can be directly linked to one's 'personal information' such as a credit card number, there will be a strong incentive for users *not to share them.* Thus, we use the concept of *self-policing* (with penalty of personal information exposure) to prevent key delegation. We note that without assuming trusted hardware it is always possible for users to make clones of their private keys and so a protection mechanism against key delegation will always be in the form of a deterrent rather than a strictly preventative method.

Model: A tk-ABE scheme consist of three types of entities: a *Trusted Key Generator (TKG)*, a *Token Server (TS)*, and *Users*. The TKG is a trusted entity that stores the system master key MK and generates users' private keys based on the access control policy of the organization. The TKG also used some user specific personal information b (a credit card for example) to create token information for each user which is then securely sent to TS who maintains a database of users in the system (along with each users' token information). Users will only be enrolled in the system by presenting their personal information b at the time of key issuance. The system guarantees that this personal information will never be revealed, learnt or misused even if TKG is corrupted (at a later time) and colludes with TS. This personal information, however, will become accessible to anyone in possession of the users' private key (i.e., the user and any other entity that the user shares its private key with).

This privacy guarantee is very strong and ensures that each user's privacy (with respect to their personal information b) remains intact as long as the user does not misuse their privileges and provided they can trust TKG at the time of entering the system. In a real-life implementation the key issuing process can be a trusted process including trusted hardware implementing the private key derivation algorithm.

Any entity with the system public key PK can encrypt a plaintext message M with some decryption policy W that is allowable by the scheme. The resulting ciphertext C includes a *ciphertext coupon* $\widehat{C}$. A user who wants to decrypt C presents the ciphertext coupon $\widehat{C}$ to TS and receives a decryption token T that is computed (by TS) using $\widehat{C}$ and $\widehat{D}$ (for that user). This token can only be used for decryption by a user whose attributes satisfy the decryption policy of C, and does not pose a security threat if captured by an adversary. The plaintext can be recovered provided the user has a private key corresponding to attributes that satisfy the decryption policy for C and the decryption token T.

Security Properties: The desired security properties for tk-ABE are *Ciphertext Indistinguishability*, *Uncloneability* and *Privacy-Preserving.*

Ciphertext Indistinguishability is defined as a game between a challenger and an adversary similar to the security games in typical ABE schemes, except that oracle access to TS must also be included. It should be noted that the ciphertext indistinguishability also implies collusion resistance. That is, two (or more) users in the system cannot work together using their private keys to decrypt any ciphertext that any of the individual users could not have decrypted alone.

A tk-ABE scheme is said to be uncloneable if any user giving a clone of their private key will enable the receiver of the cloned information to compute that user's personal information.

The privacy-preserving property of a tk-ABE scheme requires that the personal information b remains private even if TKG is corrupted (after a trusted enrollment) and colludes with TS. By trusted enrollment, we mean that TKG deletes the personal information b as soon as they complete the registration of the user. We say that a scheme that satisfies the property is *privacy-preserving.*

Constructions: We give a construction of an uncloneable privacy-preserving tk-ABE scheme that uses the ciphertext-policy ABE scheme by Cheung and Newport [5] (referred to as the CN scheme in the sequel) as the base scheme. That is, we convert the CN scheme into a tk-ABE scheme, by showing how to modify the encryption and decryption algorithms to generate ciphertext coupons and decrypt using the ciphertext and a decryption token associated with that ciphertext and a specific user with a set of attributes satisfying the decryption policy. We prove the security properties of the scheme in standard model. The ideas and methods of this construction can be used to convert other known CP-ABE schemes ([16]) and key-policy ABE schemes ([10, 19]). Indeed, we provide a second tk-ABE scheme based on Bethencourt et al.'s CP-ABE scheme [3] in the Appendix B.

Non-interactive systems and future work: Decryption in tk-ABE system requires interaction between users and the token server. This has the drawback of requiring the token server to be on-line. In Section 6 we discuss non-interactive uncloneable ABE schemes and present a construction of a non-interactive scheme that provides protection against uncloneability in Appendix C. Unlike tk-ABE, the non-interactive scheme in its current form is specifically based on Cheung and Newport's scheme and extending it to other CP-ABE schemes remains an open problem.

Key misuse can pose a severe security risk to real-life implementations of ABE schemes. Our approach to introduce a token server breaks the decryption process into a two step process and requires interaction with a token server. The trust requirement on this server is minimal in the sense that the server neither has decryption power nor can extract users' personal information. In addition to introducing a deterrence for key delegation, the use of a token server provides a simple mechanism to *revoke* users: the TKG can simply inform the TS to remove a user's entry from the database.

It is interesting to consider alternative approaches to enforce non-delegatability, such us traceability of users who clone their keys.

1.2 Outline for rest of paper

In Section 2, we discuss related work. Section 3 reviews the main properties of attribute-based encryption schemes.In Section 4, we formalize our new type of ABE scheme, token-based ABE schemes. Section 5 contains our tk-ABE construction. In Section 6, we consider non-interactive uncloneable ABE schemes. Finally, in Section7, we conclude with a discussion of this and future work.

2 Related Work

Attribute-Based Encryption: Attribute-Based Encryption (ABE) was introduced by Sahai and Waters [18]. The first concrete ciphertext policy ABE was proposed by Nali, Adams and Miri [14]. The decryption policies in that work consist of the conjunction of threshold gates, where the number of gates and the threshold of each is determined at encryption time. Bethencourt et al. [3], extend Sahai and Waters' scheme [18] to construct a CP-ABE scheme in which decryption policies are described by threshold trees. Here, each node in the tree corresponds to a threshold gate and each leaf corresponds to an attribute. If a user's attributes satisfies the tree, then they can decrypt the ciphertext. Another CP-ABE scheme was proposed by Cheung and Newport [5], in which decryption policies are restricted to a single AND gate, but attributes are allowed to be either positive or negative. This scheme is easily extended to allow for decryption policies consisting of a disjunction of AND gates by simply encrypting the plaintext once for each AND.

Security Issues of Key Delegation: Recently, Wang et al. [20] consider the security problems of key delegation in access control systems. Key cloning is a special case of key delegation, in which the entire key is delegated.

Non-Delegatability: Lipmaa et al. [13] first proposed and formalized the property of non-delegatability as a requirement for designated-verifier signatures. Our definition of uncloneability has a similar essence to theirs, in the sense that in both, the ability to generate certain output implies knowledge of a secret. In our case, the certain output is the decrypted message and the secret is the user's sensitive personal info embedded into the system. Closer to our work is digital signets from Dwork et al. [8]. They add personal information to bind users to decryption as well. However, their scenario is different, as they need this binding to be small in size.

Broadcast Encryption: Fiat and Naor [9] laid the foundations of broadcast encryption. These schemes are used to encrypt broadcast content in a way that only a privileged subset of users can decrypt the content. Key misuse problem is addressed by assuming tamper-resistant hardware or provisions to trace 'traitors' who share their key information. In particular key cloning is protected by tamper resistant hardware. Assuming tamper resistant hardware for storing keys provides security against key cloning. However it will result in very restricted applications. In this paper we proposed a solution that is in fact a deterrence measure, namely leakage of sensitive information.

Key Revocation: The ability to revoke malicious users in the system has been seen a desirable property for many cryptographic systems including credential systems and access control systems. An ABE scheme with key revocation was presented by Staddon et al. [19] in 2008 (their scheme was inspired by Naor and Pinkas [15]). The number of users that can be revoked in their scheme is fixed. Once this number is exceeded, the system needs to be re-created.

Tokens: Baek et al. [1] formalize the use of token-controlled public key encryption. Our situation is different, since we require a token to be specific to an individual user, whereas they allow multiple user use. Our tokens are also more embedded into the system as they depend on the user's decryption key.

3 Attribute-Based Encryption (ABE)

Attribute-Based Encryption (ABE) schemes, introduced by Sahai and Waters [18], are an extension of Identity-Based Encryption (IBE) schemes. We review some basic properties ABE schemes below.

Ciphertext-Policy Attribute-Based Encryption (CP-ABE): In a CP-ABE scheme, a user's private key is determined by the set of attributes S that they possess. A Ciphertext is created with a specific decryption policy W over the set of possible attributes. If a user's attributes satisfy the decryption policy, which we denote by $S \in W$, then the user's private key can decrypt the ciphertext.

Definition 1 (CP-ABE). *A ciphertext-policy attribute-based encryption (CP-ABE) scheme $\mathcal{S}$ is four-tuple of algorithms $\mathcal{S} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}\}$. The algorithms are specified as follows:*

$\mathsf{Setup}(\ell,\ [n])$: Takes as input a security parameter ℓ and possibly the number of possible attributes n in the scheme. Outputs the system public key PK and the system private key, called the master key, MK.

$\mathsf{KeyGen}(MK,\ PK,\ S)$: The trusted key generator takes as input the master key MK, the system public key PK, and a user's attribute set S. Outputs a private key D to the user.

$\mathsf{Encrypt}(PK,\ M,\ W)$: Takes as input the system public key PK, a plaintext message M, and a decryption policy W. Outputs the ciphertext C (which includes specifying W).

$\mathsf{Decrypt}(PK,\ C,\ D)$: Takes as input the system public key PK, a ciphertext C and a user's decryption key D. If the user's attribute set satisfies the decryption policy W, compute and output a plaintext M.

Key-Policy Attribute-Based Encryption (KP-ABE): In a KP-ABE scheme, the attributes are associated with the plaintext message instead of the user. The decryption policy in a KP-ABE scheme for each user is specific (perhaps not unique though) to each user and is embedded within each user's private key. The policy is set by the organization running the ABE scheme and might, for example, assign decryption policies based on a user's role in the organization. Each ciphertext is created with a set of attributes that describes the plaintext. If the attributes of the ciphertext satisfy the decryption policy of a given user, then that user is able to decrypt it. The entity encrypting a plaintext does not, in general, know who will be able to decrypt it.

A KP-ABE scheme $\mathcal{S}$, like a CP-ABE scheme, is also defined as a four-tuple of algorithms $\mathcal{S} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}\}$. The algorithms are very similar to that of a CP-ABE scheme (see [10] for example).

4 Token-Based Attribute-Based Encryption (tk-ABE)

In a token-based ABE scheme, we bind some personal user information into the decryption process. Since private keys and ciphertexts in a typical ABE scheme are not bound to a user in the same way as in a typical public key encryption scheme, we introduce tokens and the token server to bind a user to decryption. In particular, in a tk-ABE scheme, in order for a user to decrypt any ciphertext, that user must first obtain a decryption token from the token server. This decryption token is specific to both the ciphertext that the user wishes to decrypt and to the user. In order for the issued token to be useful to the user (i.e., help the user decrypt the ciphertext), the user must provide some personal information. The idea is that if a user wishes to delegate their (cloned) private key, then their personal information is easily exposed, thus providing a deterrent for key delegation.

More formally, we define a ciphertext-policy tk-ABE scheme as follows:

Definition 2 (CP tk-ABE). *A ciphertext-policy token-based ABE (CP tk-ABE) scheme $\mathcal{S}$ is a five-tuple of algorithms $\mathcal{S} = \{\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{GetToken}, \mathsf{Encrypt}\}$. These algorithms are specified as follows:*

$\mathsf{Setup}(\ell,\ [n])$: Takes as input a security parameter ℓ and possibly the number of possible attributes n in the scheme. Outputs the system public key PK and the system private key, called the master key, MK.

$\mathsf{KeyGen}(MK,\ PK,\ ID,\ S,\ b)$: The trusted key generator takes as input the master key MK, the system public key PK, and a user's information: identification ID, attributes S and personal information b. Outputs a private key D to the user and sends token information about the user $\widehat{D}$ to the token server. Finally erase b and randomness in generating $\hat{D}, D$.

$\mathsf{Encrypt}(PK, M, W)$: Take as input the system public key PK, a plaintext message M, and a decryption policy W. Outputs the ciphertext C which includes the decryption policy W and information needed to compute a ciphertext coupon $\widehat{C}$.

$\mathsf{GetToken}(ID, \widehat{C})$: The token server takes as input the user's identity ID, and a ciphertext coupon $\widehat{C}$ (computed by the user from a ciphertext C possibly using D, b) and computes a token T using $\widehat{D}$, where $\widehat{D}$ corresponds to the user with identity ID. Return T to the user.

$\mathsf{Decrypt}(PK, C, T, D, b)$: Take as input the system public key PK, a ciphertext C, a decryption token T (corresponding to C and the user) and a user's private key D and personal information b. If the user's private key corresponds to attributes that satisfy the decryption policy for C, output the plaintext M.

This definition for a ciphertext-policy tk-ABE scheme can easily be modified to define a key-policy tk-ABE scheme as well.

4.1 Security Model

For a token-ABE scheme to be secure, we require that scheme be ciphertext indistinguishable, uncloneable and privacy-preserving. We formally define these security properties here.

Ciphertext Indistinguishability: We define security with respect to ciphertext indistinguishability with the following game, which we call the *tk-ABE security game*. The game is played between an *adversary* and a *challenger* and is given in Security Game 1.

Initialization: The adversary chooses a decryption policy W that it wishes to be challenged upon.

System Setup: The challenger runs $\mathsf{Setup}(\ell,n)$, for a given security parameter ℓ and number of attributes n, and gives the public parameters PK to the adversary.

Phase 1: The adversary is allowed to make polynomially many adaptive queries consisting of

(a) private decryption keys and personal information requests for any user whose attribute set S does not satisfy W. The adversary specifies ID and S of the user.

(b) private decryption keys for any user whose attribute set do not satisfy W. The adversary specifies ID, S and b for the user.

(c) token requests for any ciphertext and any user. The adversary specifies $\widehat{C}$ and a user (ID if already known user, ID, S, b for new user).

For any quantity that the adversary must specify, it may choose to allow the challenger to choose a random input instead.

Challenge: The adversary submits two plaintexts M_0 and M_1 ($M_0 \neq M_1$) randomly taken from plaintext domain. The challenger randomly picks $\alpha \in \{0,1\}$, and encrypts M_α with decryption policy W. The challenger gives the ciphertext C_α to the adversary.

Phase 2: The adversary may repeat Query Phase 1.

Guess: The adversary outputs $\alpha' \in \{0,1\}$, and wins if $\alpha = \alpha'$.

Security Game 1: tk-ABE Security Game.

Since the adversary can simply guess α' randomly and win the game with probability 1/2, the advantage of the adversary in this game is defined as $\mathcal{ADV} := \Pr[\alpha = \alpha'] - \frac{1}{2}$.

Notice that the adversary must declare the decryption policy at the start of the game. This model is called the *selective ID model* (see [5] for more details). A stronger game would involve the adversary declaring W in the challenge stage. The formal definition of ciphertext indistinguishability that we use for tk-ABE is as follows.

Definition 3 (Ciphertext Indistinguishability). *A tk-ABE scheme Δ is secure against chosen plaintext attacks (CPA) in the selective ID model if all polynomial time adversaries have at most a negligible advantage in the token-ABE security game.*

Notice that ciphertext indistinguishability also implies collusion-resistance. That is, if a tk-ABE scheme Δ is ciphertext indistinguishable, then no collection of users in Δ can combine their keys to decrypt any ciphertext that any of the individual users could not have decrypted alone. This follows since the adversary in the tk-ABE security game is allowed to query for multiple private keys (and hence simulate the colluding users) before and after selecting the plaintexts for the challenge stage (as pointed out in [5]).

Uncloneability: The intuition of an uncloneable tk-ABE scheme is the following. Suppose Alice has private key D and personal information b. If Alice gives Bob a copy of her key D (a clone of D) then Bob can (efficiently) compute Alice's personal information b. Such a scheme is said to be uncloneable. We formalize this with the following definition. Here, let B be the domain of valid users' personal information and let $|B|$, the size of this space, be polynomial in ℓ (the size/security parameter of the scheme).

Definition 4 (Uncloneability). *Let Δ be a ciphertext-policy tk-ABE scheme. We say that Δ is* uncloneable *if the following holds for all users in Δ: Let ID be any user in Δ with private key D and personal information b. If D is known then b can be computed in time $O(|B|)$ given access to the token server TS.*

In fact, we can strengthen this definition of uncloneability by relaxing the meaning of a cloned key. To this end we define a partial cloned key D' of the key D (denoted by $D' \subseteq D$) to be any unchanged portion of D. A cloned key is then also a partial cloned key. With this definition, we have the following strengthened notion of uncloneability.

Definition 5 (Strong Uncloneability). *Let Δ be a ciphertext-policy tk-ABE scheme with public key PK. We say that Δ is* strongly uncloneable *if the following holds for all users in Δ: Let ID be any user in Δ with private key D and personal information b. Let $D' \subseteq D$ be any partial cloned key of D, C be any valid ciphertext, and W be any policy. If there exists a polynomial time algorithm $\mathcal{F}$ that, given oracle access to* $\mathsf{GetToken}$ *and given input PK, C , ID and D', outputs the decryption of C, that is*

$$\mathcal{F}^{\mathsf{GetToken}(\cdot,\cdot)}(PK, C, ID, D') = \mathsf{Decrypt}(PK, C, \mathsf{GetToken}(ID, \hat{C}), D', b),$$

then there exists an extractor algorithm $\mathcal{X}$ that, given oracle access to $\mathsf{GetToken}$ *and given input PK, C , ID and D', outputs user ID's personal information b in time $O(|B|)$.*

Notice that in this definition, we also require that the partial cloned key D' is sufficient for user ID to decrypt a ciphertext C using $\mathsf{Decrypt}$. Basically, this notion of uncloneability means that knowledge of enough components (or bits, or parts) of a user's private key that enables one to decrypt a ciphertext is sufficient to compute that user's personal information.

Privacy-Preserving: We formalize our notion of privacy as follows.

Definition 6 (Privacy-Preserving). *Let Δ be a tk-ABE scheme with public key PK and master key MK. Δ is said to be* privacy-preserving *if for every user with identification ID, private key D, token information $\widehat{D}$, attribute set S and personal information b, it holds that*

$$\Pr[b \mid PK, MK, ID, S, \widehat{D}] = \Pr[b \mid ID, S].$$

The intuition behind this definition is that a user's personal information b is no more at risk due to their membership in the tk-ABE scheme than it would be otherwise. Even the TKG and TS working together (thus MK and $\widehat{D}$ are known) cannot compute b. This of course assumes that at key issuance time, the TKG deletes b once it has used it to compute $\widehat{D}$. If a corrupt TKG keeps this information, then all privacy is lost. Here we assume that a user's identification ID and attributes S are known or can be easily obtained.

5 A Privacy-Preserving Uncloneable tk-ABE Scheme

We now present a privacy-preserving uncloneable ciphertext-policy tk-ABE scheme. The scheme allows for decryption policies that consist of a single conjunction attributes and negated attributes. The ciphertext contains a component for each attribute in the set of all possible attributes $\mathcal{N}$, whether they appear in the conjunction or not. The components correspond to positive, negated or *don't care* attributes. Any attribute not appearing in the decryption policy is considered a don't care attribute, as the decryption policy does not depend on the user having or not having that attribute. For more details,

The algorithms in our tk-ABE scheme are defined as follows:

$\mathsf{Setup}(\ell, n)$: First the bilinear groups in which the scheme will operate are chosen: let $\mathbb{G}$ and $\mathbb{G}_1$ be bilinear groups of prime order p (where p is an ℓ-bit prime), let g be a randomly chosen generator of $\mathbb{G}$, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be an efficient bilinear map. Next, let $\mathcal{N} = \{1, \ldots, n\}$ be the set of attributes and randomly choose y, $t_1, \ldots, t_{3n}$ from $\mathbb{Z}_p$. The system public key PK is

$$PK = \langle Y = e(g, g)^y, T_1 = g^{t_1}, \ldots, T_{3n} = g^{t_{3n}}, g, e, \mathbb{G}, \mathbb{G}_1, \mathcal{N} \rangle,$$

and the system private key, called the master key, is $MK = \langle y, t_1, \ldots, t_{3n} \rangle$. The system public key is made public and the master key is given to the trusted key generator TKG.

Note: We assume that elements in $\mathbb{G}$ and $\mathbb{G}_1$ have binary representation with bitlength ℓ.

$\mathsf{KeyGen}(MK, PK, ID, S, b)$: To generate a private decryption key for a user with identification ID, attributes S and personal information b, the trusted key generator TKG does the following: For each attribute $i \in \mathcal{N}$, choose a random $r_i \in \mathbb{Z}_p$ and compute

$$D_i = \begin{cases} g^{\frac{r_i}{t_i}} & \text{if } i \in S \text{ (user has attribute } i) \\ g^{\frac{r_i}{t_{n+i}}} & \text{if } i \notin S \text{ (user does not have attribute } i), \end{cases}$$

$$F_i = g^{\frac{r_i}{t_{2n+i}}} \qquad \text{(for don't care attributes in } W).$$

Letting $r = \sum_{i \in \mathcal{N}} r_i$, TKG then computes the user's token information $\widehat{D} = g^{(y-r)/b}$ and sends $\langle ID, \widehat{D} \rangle$ to the token server TS. The user's private key is $D = \langle \{D_i, F_i\}_{i \in \mathcal{N}} \rangle$.

Encrypt(PK, M, W): To encrypt a message $M \in \mathbb{G}_1$ with decryption policy W over I, pick a random $s \in \mathbb{Z}_p$ (the nonce for the ciphertext) and compute $\widetilde{C} = MY^s = Me(g,g)^{ys}$, $\widehat{C} = g^s$, and for each attribute $i \in \mathcal{N}$ compute

$$C_i = \begin{cases} T_i^s = g^{st_i} & \text{if } i \in I,\ \underline{i} = i \quad \text{(must have attribute } i\text{)} \\ T_{n+i}^s = g^{st_{n+i}} & \text{if } i \in I,\ \underline{i} = \neg i \quad \text{(must not have attribute } i\text{)} \\ T_{2n+i}^s = g^{st_{2n+i}} & \text{if } i \notin I \quad \text{(don't care about attribute } i\text{).} \end{cases}$$

The ciphertext is $C = \langle W, \widetilde{C}, \widehat{C}, \{C_i\}_{i \in \mathcal{N}} \rangle$, where $\widehat{C} = g^s$ is the ciphertext coupon.

GetToken(ID, $\widehat{C}$): The token server retrieves $\langle ID, \widehat{D} \rangle$ from its database of users and computes (and outputs) the ciphertext token $T = e(\widehat{C}, \widehat{D}) = e(g^s, g^{(y-r)/b}) = e(g,g)^{s(y-r)/b}$. Here $\widehat{C}$ is provided by the user, which is extracted from a ciphertext $C = \langle W, \widetilde{C}, \widehat{C}, \{C_i\}_{i=1..n} \rangle$.

Decrypt(PK, C, T, D, S, b): To decrypt a ciphertext C (with decryption policy W over the literals I) with decryption token T corresponding to C and the user, a user with private decryption key D, attributes S satisfying the decryption policy W, and private information b does the following: For each attribute $i \in \mathcal{N}$ compute $E_i = e(g,g)^{sr_i}$ where

$$E_i = e(D_i, C_i) = \begin{cases} e(g^{st_i}, g^{r_i/t_i}) = e(g,g)^{sr_i} & \text{if } i \in I, \underline{i} = i, i \in S \\ e(g^{st_{i+n}}, g^{r_i/t_{n+i}}) = e(g,g)^{sr_i} & \text{if } i \in I, \underline{i} = \neg i, i \notin S \end{cases}$$
$$E_i = e(C_i, F_i) = e(g^{st_{i+2n}}, g^{r_i/t_{2n+i}}) = e(g,g)^{sr_i} \quad \text{if } i \notin I \text{ (don't care).}$$

Computing the product $\widetilde{E} = \prod_{i \in \mathcal{N}} E_i = \prod_{i \in \mathcal{N}} e(g,g)^{sr_i} = e(g,g)^{s \sum_{i \in \mathcal{N}} r_i} = e(g,g)^{sr}$, and recalling that $T = e(g,g)^{s(y-r)/b}$, the plaintext is recovered since

$$\frac{\widetilde{C}}{(T)^b\, \widetilde{E}} = \frac{M\, e(g,g)^{sy}}{(e(g,g)^{s(y-r)/b})^b\, e(g,g)^{sr}} = \frac{M\, e(g,g)^{sy}}{e(g,g)^{sy-sr}\, e(g,g)^{sr}} = M.$$

Our construction of a tk-ABE scheme satisfies each of the three desired security properties: ciphertext indistinguishability, strong uncloneability and privacy-preserving. Proofs for the security results are given in Appendix A.

5.1 Efficiency

The overall computational complexity (of all entities combined) is essentially the same as in the CN scheme, except that the TKG has an additional exponentiation at key generation (for $\widehat{D}$) for each user, and there is also an additional exponentiation in Decrypt (unmasking T) for each decryption. The TKG needs to compute $O(n)$ exponentiations for each user (in the KeyGen algorithm) added to the scheme, each user needs to compute $O(n)$ pairing operations and $O(n)$ multiplications to decrypt a single ciphertext, and the TS needs to compute one paring operation for each $\mathsf{GetToken}$ request. The $\mathsf{Encrypt}$ algorithm is identical to the CN scheme and requires $O(n)$ exponentiations.

Even though the TS only needs to compute one pairing operation per $\mathsf{GetToken}$ request, the number of such requests may be very large. In order to lessen the load of the TS, multiple token servers can be used, with user's assigned to a particular server.

There is also a significant communication complexity in our tk-ABE scheme, since each user must request a token from the TS each time they wish to decrypt a ciphertext. Again, adding multiple token servers will distribute this cost.

When the number of attributes n is large, the CN scheme (and hence this scheme) is very inefficient as decryption needs $O(n)$ pairing operations and exponentiations, and both the ciphertext and private keys have $O(n)$ size. In this situation, the CP-ABE scheme by Bethencourt, Sahai and Waters [3] is much more efficient. We show how the BSW scheme can be *tokenized* (i.e., turned into a tk-ABE scheme) in Appendix B.

5.2 User Revocation

The main intention of introducing a token server to ABE schemes was to create uncloneable schemes. A side affect of this (introduction of TS) is that revoking a user's decryption privileges in a tk-ABE scheme is trivial. Since a user needs a decryption token in order to decrypt any ciphertext (using their private key), we can simply remove the user's entry in the token server's database. Clearly, a policy for user revocation must be enforced (i.e., who can and cannot revoke users). By introducing more data into the TS's database of users, more complex policies can also be incorporated into tk-ABE schemes also. For example, user's might be put on 'probation' for a certain period of time in which they are not allowed to be issued decryption tokens. Or perhaps 'premium users' in a system are allowed to decrypt content before non-premium users.

6 Non-Interactive Uncloneable CP-ABE

We now propose a CP-ABE scheme that is (strongly) uncloneable without the use of decryption tokens. This is achieved by adding a single dummy attribute that each entity in the system possesses and is needed for decryption of any ciphertext. Essentially, *all* valid decryption policies for any ciphertext implicitly requires the user to have this attribute. In the scheme, this attribute (we'll denote it as the zeroth attribute i_0) for a given user will consist of some personal information about the user (that they will not wish to disclose) appended by some random bits. If a user distributes a clone of this key (or a partial clone that is able to decrypt) then the user must reveal this zeroth attribute and hence reveal their personal information. Thus, the uncloneability property comes from a user not wishing to divulge their personal information.

In Appendix C, we formalize the security model of the non-interactive scheme and present our construction. A short discussion and comparison of tk-ABE and our non-interactive scheme is also given.

7 Discussion/Conclusion

ABE schemes provide an elegant method of controlling access to information. We have described a security weakness in the current definition of ABE systems that makes all known ABE systems what we called 'delegatable'. That is, the ABE schemes allow users in the system to clone their keys, or construct keys that bypass the role of the system trusted key generator. We have introduced a new type of ABE scheme, token-based ABE, which provides a disincentive for users to mis-use their key information (key cloning in particular) by linking any leakage of their ABE private key to the complete loss of some personal information that the user wants to protect. We have proposed a security model for tk-ABE by motivating and formalizing three security properties, ciphertext indistinguishability, uncloneability and privacy-preserving properties, and presented a tk-ABE scheme. We have proved the security of this scheme in the proposed model. We have also given a second construction (Appendix B) that provides higher efficiency and is based on the scheme of Bethencout, Sahai and Waters.

Our scheme is the first ABE scheme to consider key delegation (cloning) as a security threat and proposes a solution that is practical and in fact adds extra functionality (key revocation) to the system. It also opens many new interesting avenues for future work, some of which are outlined below.

Firstly, the methods that we use can easily be extended/modified to any of the other known ABE schemes (both ciphertext-policy and key-policy) to create a tk-ABE scheme. An open problem is to construct a tk-ABE scheme that is not based on an existing ABE scheme using new techniques.

Secondly, our solution introduces a token server that issues tokens that are specific to a user requesting them and is necessary for decryption of a ciphertext. While we have presented a non-interactive scheme, it remains an open question if a non-interactive ABE scheme can be constructed using other known ABE schemes as a base (see Appendix C.4).

Thirdly, in a tk-ABE scheme, the deterrence for key cloning is obtained by adding the token server (and decryption tokens). The computational load on the token server can be quite high if many users request decryption tokens. One possible research direction to address this is to find a method in which the communication and computational overhead for issuing a token by the TS is very small, like in the token-controlled PKE of Baek et al. [1]. In this way, a tk-ABE scheme would have minimal TS costs.

Fourthly, the schemes we present only offer security against key delegation of cloned keys. An open (and important) problem is to create uncloneable schemes for any kind of key delegation (including pseudo-keys and keys created by colluding users).

Finally, and at a more fundamental level, is the issue of providing secure support for delegation in ABE systems. The ability to delegate keys may be required in some systems (hierarchical schemes for example). However, in a secure system with key delegation, a delegation policy must be well defined in advance and enforced by the system. All current ABE systems implicitly support on open delegation policy in the sense that they allow a user to clone their key, or give away part of their keys and their associated decryption power. Our approach may be seen as a closed approach in the sense that only the TKG can allocate keys and register new users and any mis-use of ABE keys will endanger an individuals personal information. A challenging research direction is to allow defining delegation policies for users and ensuring that it is enforced.

References

1. J. Baek, R. Safavi-Naini and W. Susilo, Token-Controlled Public Key Encryption. R. H. Deng, F. Bao, H. Pang and J. Zhou (Eds): ISPEC 2005, LNCS 3439, pp. 386–397, 2005.
2. J. Baek, W. Susilo and J. Zhou, New Constructions of Fuzzy Identity-Based Encryption. ASIACCS 2007.

3. J. Bethencourt, A. Sahai and B. Waters, Ciphertext-Policy Attribute-Based Encryption. IEEE Symposium on Security and Privacy, pp. 321–334, 2007 (Oakland 2007).
4. M. Chase, Multi-authority Attribute Based Encryption, TCC 2007, LNCS 4392, pp. 515–534, 2007.
5. L. Cheung and C. Newport, Porvably Secure Ciphertext Policy ABE. ACM Conference on Computer and Communications Security, pp. 456–465, 2007 (CCS'07).
6. Benny Chor, Amos Fiat, Moni Naor. "Tracing traitors". Advances in Cryptology - CRYPTO 94. LNCS 839: 257-270. Springer. 1994.
7. Fingerprinting Long Forgiving Messages, G. R. Blakley, C. Meadows and G. B. Purdy. Advances in Cryptology - CRYPTO '85. LNCS 218: 180-189, 198.
8. C. Dwork, J. B. Lotspiech and M. Naor, Digital Signets: Self-Enforcing Protection of Digital Information (Preliminary Version), STOC 1996, pg. 489–498, 1996.
9. Amos Fiat and Moni Naor. "Broadcast Encryption". Advances in Cryptology: CRYPTO 1993. LNCS 773: 480–491. 1993. (Extended version available at: http://www.wisdom.weizmann.ac.il/∼naor)
10. V. Goyal, O. Pandey, A. Sahai and B. Waters, Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Dada. ACM Conference on Computer and Communications Security, pp. 89–98, 2006 (CCS'06).
11. J. Katz, A. Sahai and B. Waters, Predicate Encryption Supporting Disjunction, Polynomial Equations, and Inner Products. N. Smart (Ed.): EUROCRYPT 2008, LNCS 4965, pp. 146–162, 2008.
12. D. Khader, Attribute Based Group Signature with Revocation. Cryptology ePrint Archive, Report 2007/241, 2007. Available online at http://eprint.iacr.org/
13. H. Lipmaa, G. Wang and F. Bao, Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction. L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi and M. Yung (Eds): ICALP 2005, LNCS 3580, pp. 459–471, 2005.
14. D. Nali, C. Adams and A. Miri, Using Threshold Attribute-Based Encryption for Practical Biometric-Based Access Control. International Journal of Network Security, vol. 1, no. 3, November, 2005, pp. 173-182.
15. M. Naor and B. Pinkas, Efficient Trace and Revoke Schemes. Y. Frankel (Ed.): Financial Cryptography 2000, LNCS 1962, pp. 1–20, 2000.
16. R. Ostrovsky, A. Sahai and B. Waters, Attribute-Based Encryption with Non-Monotonic Access Structures. Cryptology ePrint Archive, Report 2007/323, 2007. Full version of CCS 2007 paper (to appear). Available online at http://eprint.iacr.org/
17. M. Pirretti, P. Traynor, P. McDaniel and B. Waters, Secure Attribute-Based Systems. ACM Conference on Computer and Communications Security, pp. 99–112, 2006 (CCS'06).
18. A. Sahai and B. Waters, Fuzzy Identity-Based Encryption. R. Cramer (Ed.): EUROCRYPT 2005, LNCS 3494, pp. 457–473, 2006.
19. J. Staddon, P. Golle, M. Gagné and P. Rasmussen, A Content-Driven Access Control System. K. E. Seamons, N. McBurnett and T. Polk (Eds): IDtrust 2008, ACM International Conference Proceeding Series, vol. 283, pp. 26–34, 2008.
20. Q. Wang, N. Li and H. Chen, On the Security of Delegation in Access Control Systems, S. Jajodia and J. López (Eds): ESORICS 2008, LNCS 5283, pp. 317–332.
21. D. Yao, N. Fazio, Y. Dodis and A. Lysyanskaya, ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and broadcast Encryption. ACM Conference on Computer and Communications Security, pp. 354–363, 2004 (CCS'04).

A Security Proofs

We now show that our scheme satisfies the desired security properties. To prove ciphertext indistinguishability, we reduce the problem to another (believed) hard problem: the Decisional Bilinear Diffie-Hellman (DBDH) problem. In particular, we rely on the following assumption.

Assumption 1 (DBDH) *Let $\mathbb{G}, g \in \mathbb{G}, \mathbb{G}_1$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ be defined as above. Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ uniformly at random. The Decisional Bilinear Diffie-Hellman (DBDH) assumption is that no polynomial-time adversary is able to distinguish the tuple $\langle g^a, g^b, g^c, e(g,g)^{abc} \rangle$ from the tuple $\langle g^a, g^b, g^c, e(g,g)^z \rangle$ with more than a negligible advantage.*

We say that $\langle A = g^a, B = g^b, C = g^c, Z\rangle$ is a DBDH instance (or challenge) and the DBDH problem is to decide if $Z = e(g,g)^{abc}$ or $Z = e(g,g)^z$. With this, we can state our first security result. The advantage in solving the DBDH problem is the probability of success minus 1/2 (since one can always guess).

Theorem 2. *Our scheme is secure against chosen plaintext attacks (CPA) in the selective ID model, provided that the Decisional Bilinear Diffie-Hellman (DBDH) assumption holds.*

Proof. Our proof is modified from [5]. Let $\mathcal{A}$ denote an adversary who can win the tk-ABE security game with non-negligible advantage ϵ. We construct a simulator $\mathcal{S}$ that uses $\mathcal{A}$ to help it solve the DBDH problem with non-negligible advantage $\epsilon/2$. This is a contradiction if the DBDH assumption holds, so we conclude that such an $\mathcal{A}$ cannot exist.

Suppose $\mathcal{S}$ is given a DBDH challenge $\langle A, B, C, Z\rangle = \langle g^a, g^b, g^c, Z\rangle$, where $Z = e(g,g)^{abc}$ or $Z = e(g,g)^z$ for (uniformly chosen) random $a, b, c, z \in \mathbb{Z}_p$. $\mathcal{S}$ will simulate our tk-ABE scheme in the tk-ABE security with $\mathcal{A}$ playing the game as follows:

Init: $\mathcal{A}$ gives $\mathcal{S}$ a challenge access policy W over the literals $I \subseteq \mathcal{N}$.

Setup: $\mathcal{S}$ implicitly sets $y = ab$ (unknown), by letting $Y = e(A, B) = e(g,g)^{ab}$. For each $i \in \mathcal{N}$, $\mathcal{S}$ chooses random $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p$ computes the T_i according to

	$i \in I$		$i \notin I$
	$\underline{i} = i$	$\underline{i} = \neg i$	
T_i	g^{α_i}	B^{α_i}	B^{α_i}
T_{n+i}	B^{β_i}	g^{β_i}	B^{β_i}
T_{2n+i}	B^{γ_i}	B^{γ_i}	g^{γ_i}

The public parameters

$$PK = \langle e, g, Y = e(g,g)^{ab}, T_1, \ldots, T_{3n}\rangle$$

is given to $\mathcal{A}$. The master key is (implicitly)

$$MK = \langle ab, \{t_i\}_{i=1..3n}\rangle,$$

where $t_i = \log_g T_i$(i.e., t_i is the exponent of T_i with base g). Some are known to $\mathcal{S}$ and some are not (i.e., those created based on B).

Phase 1: To help $\mathcal{S}$ reply to $\mathcal{A}$'s queries, we show how $\mathcal{S}$ can generate private keys and token information for users with attribute sets S^* that do *not* satisfy the challenge decryption policy W. The algorithm $\mathsf{NewUser}$ takes as input a user's ID^*, attribute set S^* and personal information b^* and outputs a valid private key D and token information $\widehat{D}$ for that user.

$\mathsf{NewUser}(ID^*, S^*, b^*)$: There must exist a $j \in I$ such that either $j \in S^*$ and $\underline{j} = \neg j$ or $j \notin S^*$ and $\underline{j} = j$. $\mathcal{S}$ chooses such a j. Without loss of generality, assume $j \notin S^*$ and $\underline{j} = j$ (i.e., the user does not have attribute j).

For every $i \in \mathcal{N}$, $\mathcal{S}$ randomly chooses $r'_i \in \mathbb{Z}_p$. Then it sets $r_j = ab + r'_j b$ and for each $i \neq j$ it sets $r_i = r'_i b$. Finally, it sets $r = \sum_{i=1}^{n} r_i = ab + \sum_{i=1}^{n} r'_i b$. Note r_i and r are all unknown to $\mathcal{S}$. However, D_i are computed as follows: For $i = j$,

$$D_j = A^{1/\beta_i} g^{r'_j/\beta_j} = g^{(ab+r'_j b)/b\beta_j} = g^{r_j/b\beta_j}, \text{ since } j \notin S^* \text{ and } \underline{j} = j,\ t_{j+n} = b\beta_j.$$

For $i \neq j$, when $i \in S^*$

$$D_i = \begin{cases} B^{r'_i/\alpha_i} = g^{r_i/t_i} & i \in I \wedge \underline{i} = i, \text{ since } t_i = \alpha_i \\ g^{r'_i/\alpha_i} = g^{r_i/b\alpha_i} & (i \in I \wedge \underline{i} = \neg i) \vee i \notin I, \text{ since } t_i = b\alpha_i \end{cases}$$

and when $i \notin S^*$

$$D_i = \begin{cases} g^{r'_i/\beta_i} = g^{r_i/b\beta_i} & (i \in I \wedge \underline{i} = i) \vee i \notin I, \text{ since } t_{n+i} = b\beta_i \\ B^{r'_i/\beta_i} = g^{r_i/\beta_i} & (i \in I \wedge \underline{i} = \neg i), \text{ since } t_{n+i} = \beta_i. \end{cases}$$

For the F_i, when $i = j$

$$F_j = A^{1/\gamma_j} g^{r'_j/\gamma_j} = g^{(ab+br'_j)/b\gamma_j} = g^{r_j/b\gamma_j}, \text{ since } j \notin S^*,\ \underline{j} = j \text{ and } t_{2n+j} = b\gamma_j.$$

and for $i \neq j$, we have

$$F_i = \begin{cases} g^{r'_i/\gamma_i} = g^{r_i/b\gamma_i} & i \in I, \text{ since } t_{2n+i} = b\gamma_i \\ B^{r'_i/\gamma_i} = g^{r_i/\gamma_i} & i \notin I, \text{ since } t_{2n+i} = \gamma_i. \end{cases}$$

The private exponent is then $D = \langle \{D_i, F_i\}_{i \in \mathcal{N}} \rangle$. For the token information, $\mathcal{S}$ computes

$$\widehat{D} = (\prod_{i=1}^{n} B^{-r'_i})^{1/b^*} = g^{-\sum_{i=1}^{n} r'_i b/b^*} = g^{(ab-r)/b^*} = g^{(y-r)/b^*}.$$

The algorithm outputs $D, \widehat{D}$

For user's with attribute sets that do satisfy the challenge decryption policy W, the adversary is only allowed to query for decryption tokens. To generate token information for such users, $\mathcal{S}$ uses the following helper function $\mathsf{NewToken}$, defined as

$\mathsf{NewToken}(ID^*, S^*, b^*)$: Since $\mathcal{A}$ is not allowed to query for this private key, $\mathcal{S}$ does not need to explicitly compute the D_i and F_i for user ID, $\mathcal{S}$ only needs to generate a valid $\widehat{D}$. To do this, $\mathcal{S}$ randomly chooses $r'_1, r_2, r_3, \ldots, r_n \in \mathbb{Z}_p$ and sets $r_1 = ab + r'_1$ and $r = r_1 + \cdots r_n$. Thus, the distribution of the r_i are the same here as they are in $\mathsf{Decrypt}$. Therefore, the D_i and F_i are correctly (implicitly) defined. The token information for user ID is then computed as

$$\widehat{D} = g^{-(r'_1+r_2+\cdots+r_n)/b^*} = g^{-(-ab+r_1+r_2+\cdots+r_n)/b^*} = g^{(y-r)/b^*}.$$

The algorithm outputs $\widehat{D}$.

Now, to answer $\mathcal{A}$'s queries, $\mathcal{S}$does the following: (note, if $\mathcal{A}$ makes the same query twice, or makes an inconsistent query, $\mathcal{A}$ rejects the query)

(a) private decryption keys and personal information requests for any user whose attribute set S does not satisfy W. $\mathcal{A}$ specifies ID^* and S^*. If $S \notin W$, $\mathcal{S}$ randomly generates b^*, runs $\mathsf{NewUser}(ID^*, S^*, b^*)$ and gives D to $\mathcal{A}$. $\mathcal{S}$ also records $\langle ID^*, S^*, b^*, D, \widehat{D}\rangle$ for future queries.

(b) private decryption keys for any user whose attribute set do not satisfy W. $\mathcal{A}$ specifies ID^*, S^* and b^*. If $S^* \notin W$, $\mathcal{S}$ runs $\mathsf{NewUser}(ID^*, S^*, b^*)$ and gives $\mathcal{A}$ D. $\mathcal{S}$ also records $\langle ID^*, S^*, b^*, D, \widehat{D}\rangle$ for future queries.

(c) token requests for any ciphertext and any user. $\mathcal{A}$ specifies $\widehat{C}$ and user information. There are three cases
   - If user ID^* has not already been created and $S^* \in W$ then $\mathcal{A}$ submits ID^*, S^* and b^* for the user. $\mathcal{S}$ runs $\mathsf{NewToken}(ID^*, S^*, b^*)$ to obtain $\widehat{D}$ and records $\langle ID^*, S^*, b^*, \widehat{D}\rangle$ for future queries.
   - If user ID^* has not already been created and $S^* \notin W$ then $\mathcal{A}$ submits ID^*, S^* and b^* for the user. $\mathcal{S}$ runs $\mathsf{NewUser}(ID^*, S^*, b^*)$ to obtain $\widehat{D}$ and records $\langle ID^*, S^*, b^*, D, \widehat{D}\rangle$ for future queries.
   - If user ID^* has already been created, $\mathcal{A}$ submits ID^* and $\mathcal{S}$ looks up the $\widehat{D}$ already computed.

   In all cases, once $\mathcal{S}$ has a $\widehat{D}$ for the user, it computes $T = e(\widehat{C}, \widehat{D})$ and gives this to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives two different plaintext messages M_0 and M_1 of the same size to $\mathcal{S}$. $\mathcal{S}$ randomly chooses $\mu \in \{0, 1\}$ and encrypts M_μ as follows: (note: the value C used below is from the DBDH challenge, $C = g^c$, not to be confused with the ciphertext)

$$\widetilde{C} = Zm_\mu$$

$$\widehat{C} = C = g^c$$

$$C_i = \begin{cases} C^{\alpha_i} = g^{c\alpha_i} & i \in I \wedge \underline{i} = i \\ C^{\beta_i} = g^{c\beta_i} & i \in I \wedge \underline{i} = \neg i \\ C^{\gamma_i} = g^{c\gamma_i} & i \notin I \end{cases}$$

The ciphertext, $\langle W, \widetilde{C}, \widehat{C}, \{C_i\}_{i\in\mathcal{N}}\rangle$ is given to $\mathcal{A}$.

Notice that if $Z = e(g, g)^{abc}$, then the ciphertext is a valid ciphertext with nonce $s = c$. If $Z = e(g, g)^z$, however, then $\widetilde{C}$ is a random element in $\mathbb{G}_1$. (It is a valid encryption of the plaintext $M_\mu e(g, g)^z e(g, g)^{-abc}$, which will not be M_0 or M_1 with very high probability, though).

Phase 2: Same as phase 1.

Guess: At the end of game, $\mathcal{A}$ guesses that $M_{\mu'}$ was encrypted. If $\mathcal{A}$ guesses correctly ($\mu' = \mu$), then $\mathcal{S}$ answers the DBDH challenge with $Z = e(g, g)^{abc}$. If $\mathcal{A}$ does not guess correctly, $\mathcal{S}$ answers $Z = e(g, g)^z$.

When $Z = e(g, g)^{abc}$, the ciphertext is a valid encryption of M_μ, with all system parameters having the same distribution as tk-ABE. Since $\mathcal{A}$ has advantage ϵ in the tk-ABE security game, $\mathcal{A}$ answers correctly with probability $\epsilon + 1/2$. Thus, it follows that

$$\Pr[\mathcal{S} \text{ guesses } e(g, g)^{abc} | Z = e(g, g)^{abc}] = \Pr[\mu' = \mu | Z = e(g, g)^{abc}] = 1/2 + \epsilon.$$

When $Z = e(g, g)^z$, the ciphertext does not correspond to an encryption of M_0 or M_1 and hence is an invalid session of the tk-ABE security game. Thus, the advantage that $\mathcal{A}$ in the game does not apply and $\mathcal{A}$'s guess will be random. It then follows that

$$\Pr[\mathcal{S} \text{ guesses } e(g,g)^z | Z = e(g,g)^z] = \Pr[\mu' \neq \mu | Z = e(g,g)^z] = 1/2.$$

Thus, $\mathcal{S}$'s advantage in solving the DBDH problem is at least $\epsilon/2$. ❑

Theorem 3. *Our scheme is (strongly) uncloneable.*

Proof. Let Δ be the ciphertext-policy tk-ABE scheme with public key PK. Let ID be the user in Δ with private key D and personal information b, for which we have the partial cloned key $D' \subseteq D$. Let C and W be the valid ciphertext and policy for which we have

$$\mathcal{F}^{\mathsf{GetToken}(\cdot,\cdot)}(PK, C, ID, D') = \mathsf{Decrypt}(PK, C, \mathsf{GetToken}(ID, \hat{C}), D', b).$$

Now, the extractor algorithm $\mathcal{X}$ does the following: First, using algorithm $\mathcal{F}$, it computes the plaintext M. Since the $\mathsf{Decrypt}$ algorithm works with input D', $\mathcal{X}$ can compute $\widetilde{E} = e(g,g)^{sr}$ with input PK, C and D'.Recall that the plaintext (in the $\mathsf{Decrypt}$ algorithm) is computed by

$$\frac{\widetilde{C}}{(T)^b\, \widetilde{E}} = \frac{M\, e(g,g)^{ys}}{(T)^b\, e(g,g)^{sr}} = M. \tag{1}$$

Since $\mathcal{X}$ has access to $\mathsf{GetToken}$, it can request $T = \mathsf{GetToken}(ID, \widehat{C})$. With $\widetilde{C}$, $\widetilde{E}$, T and M known, $\mathcal{X}$ simply tries all values of $b' \in B$ (the personal information space) until $\widetilde{C} = MT^{b'}\,\widetilde{E}$, which happens when $b' = b$ in time $O(|B|)$ in the worst case. ❑

Theorem 4. *Our scheme is privacy-preserving.*

Proof. Consider any user with identification ID, attribute set S, private key D, token information $\widehat{D}$ and personal information b. Recall that PK is the tk-ABE public key and MK is the master key. It follows that

$$\Pr[b \mid ID, PK, MK, \widehat{D}, S] = \Pr[b \mid ID, \widehat{D}, S],$$

since PK and MK are chosen independently of b. Recall that $\widehat{D} = g^{(y-r)/b}$ and that the order of group $\mathbb{G}_1$ is q. This, we have

$$\begin{aligned}\Pr[b \mid ID, \widehat{D}, S] &= \Pr[b \mid ID, (y-r)/b \bmod q, S]\\ &= \Pr[b \mid ID, S],\end{aligned}$$

since y, r and b are independent. ❑

B A Second tk-ABE Scheme

The privacy-preserving uncloneable tk-ABE scheme in Section 4 is only suitable when the total number of attributes, n, is relatively small. This follows since the ciphertext size, private key size, and complexity of decryption are all linear in the total number of attributes.

In this section, we present a second privacy-preserving uncloneable token-CP-ABE scheme that is more suitable when the total number of attributes is large. The scheme is based on the CP-ABE scheme of Bethencourt et al. [3]. Here, the decryption policy can be any formula that can be realized by combinations of AND, OR and threshold gates with attributes as inputs. The decryption policy W is realized by a tree access structure $\mathcal{T}$, where each internal node represents a single AND, OR or threshold gate, and leaf nodes correspond to the input attributes. For more details, we refer the reader to [3]. The algorithms that define this scheme are given below.

$\mathsf{Setup}(\ell)$: First the bilinear groups in which the scheme will operate are chosen: let $\mathbb{G}$ and $\mathbb{G}_1$ be bilinear groups of prime order p (where p is an ℓ-bit prime), let g be a randomly chosen generator of $\mathbb{G}$, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be an efficient bilinear map. Let $H : \{0,1\}^* \to \mathbb{G}$ be a hash function. Choosing random $y \in \mathbb{Z}_p$, the system public key PK and private key, called the master key, MK are given by

$$PK = \langle Y = e(g,g)^y, g, e, \mathbb{G}, \mathbb{G}_1 \rangle,$$

$$MK = \langle g^y \rangle.$$

PK is made public and MK is given to the trusted key generator TKG.

$\mathsf{KeyGen}(MK, PK, ID, S, b)$: To generate a private decryption key for a user with identification ID, attributes S and personal information b, the TKG does the following: Choose a random $r \in \mathbb{Z}_p$, and for each attribute $j \in S$, choose a random $r_j \in \mathbb{Z}_p$ The private decryption key for the user ID is

$$D = \langle \{D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}\}_{j \in S} \rangle.$$

The TKG computes the user's token information $\widehat{D} = g^{\frac{y+r}{b}}$ and sends $\langle ID, \widehat{D} \rangle$ to the token server.

$\mathsf{Encrypt}(MK, M, W)$: To encrypt a plaintext message $M \in \mathbb{G}_1$ with decryption policy W, described by the tree access structure $\mathcal{T}$, the user does the following. A polynomial q_x is chosen for each node x (including the leaves) in the tree $\mathcal{T}$. These polynomials are chosen in the following way in a top-down manner, starting from the root node r. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$. Starting with the root node r the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_r(0) = s$. Then, it chooses d_r other points of the polynomial q_r randomly to define it completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points randomly to completely define q_x. Letting $\mathcal{L}$ be the set of leaf nodes in $\mathcal{T}$, the ciphertext is constructed as

$$C = \langle W, \widetilde{C} = Me(g,g)^{sy}, \widehat{C} = g^s, \{C_x = g^{q_x(0)}, C'_x = H(att(x))^{q_x(0)}\}_{x \in \mathcal{L}} \rangle.$$

The ciphertext coupon is $\widehat{C} = g^s$.

$\mathsf{GetToken}(ID, \widehat{C})$: The token server retrieves $\langle ID, \widehat{D} \rangle$ from its database of users and computes (and outputs) the ciphertext token $T = e(\widehat{C}, \widehat{D}) = e(g^s, g^{\frac{y+r}{b}}) = e(g,g)^{s(y+r)/b}$.

Decrypt(PK, C, ID, D, S, b): To decrypt a ciphertext C (with decryption policy W realized by the tree access structure $\mathcal{T}$), a user with identification ID, private decryption key D, attributes S satisfying W, and personal information b does the following: (The decryption algorithm is a recursive procedure and only the simplest form of the decryption algorithm is described) First define the recursive algorithm $\textsc{DecryptNode}(C, D, x)$ that takes as input a ciphertext $C = \langle W, \widetilde{C}, \{C_j, C'_j\}_{j\in\mathcal{L}}\rangle$, a private key D, which is associated with a set S of attributes, and a node x from $\mathcal{T}$. If the node x is a leaf node then we let $i = att(x)$ and have the base case as follows: if $i \in S$, then

$$\textsc{DecryptNode}(C, D, x) = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} = \frac{e(g^r H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = e(g,g)^{rq_x(0)}.$$

If $i \notin S$, then $\textsc{DecryptNode}(C, sk_u, x) = \perp$.
For the recursive case, when x is a non-leaf node, the algorithm $\textsc{DecryptNode}(C, D, x)$ then proceeds as follows: for all nodes z that are children of x, it calls $\textsc{DecryptNode}(C, D, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$. Otherwise, compute and return

$$\begin{aligned}
F_x &= \prod_{z\in S_x} F_z^{\Delta_{i,S'_x}(0)} \quad \text{where } i = index(z),\ S'_x = \{index(z) : z \in S_x\} \\
&= \prod_{z\in S_x} \left(e(g,g)^{rq_x(0)}\right)^{\Delta_{i,S'_x}(0)} = \prod_{z\in S_x} \left(e(g,g)^{rq_{parent(z)}(index(z))}\right)^{\Delta_{i,S'_x}(0)} \quad \text{(by construction)} \\
&= \prod_{z\in S_x} e(g,g)^{rq_x(i)\Delta_{i,S'_x}(0)} = e(g,g)^{rq_x(0)} \quad \text{(using polynomial interpolation).}
\end{aligned}$$

Using $\textsc{DecryptNode}$ called on the root node r of the tree $\mathcal{T}$, if the tree is satisfied by the attribute set S, then compute $A = \textsc{DecryptNode}(C, D, r) = e(g,g)^{rq_r(0)} = e(g,g)^{rs}$.
The user then sends $(ID, \widehat{C})$ to the token server to obtain the decryption token $T = e(g,g)^{s(y+r)/b}$.
Using $\widetilde{C}$, A, T and b, the plaintext is recovered by computing

$$\frac{\widetilde{C}A}{(T)^b} = \frac{\widetilde{C}e(g,g)^{rs}}{(e(g,g)^{s(y+r)/b})^b} = \frac{Me(g,g)^{ys}e(g,g)^{rs}}{e(g,g)^{s(y+r)}} = M.$$

B.1 Complexity

The space and computation complexity of this scheme, just as with the previous, is the same as the scheme that is based on (i.e., Bethencourt et al.'s scheme). In comparison to the previous scheme, notice that the size of the ciphertext is now linear in the number of inputs to the formula describing the decryption formula instead of linear in the total number of attributes. The ciphertext must also, however, contain the access tree $\mathcal{T}$. A user's private key is now linear in the number of attributes that they possess instead of linear in the total number of attributes, and the computational costs are linear in the size of the access tree $\mathcal{T}$ instead of linear in the total number of attributes.

B.2 Security

The scheme in this section satisfies all of the three security requirements that we desire for a token-ABE scheme: ciphertext indistinguishability, uncloneability and privacy-preserving. Due to space requirements, we only state the security and omit the proofs.

The ciphertext indistinguishability of the scheme follows from Bethencourt et al.'s scheme. The security model is different from the previous section though. Here, the security game allows the adversary to choose the decryption policy in which it will be challenged after the Phase 1 queries, so we are no longer in the selective-ID model. We will call this the non-selective-ID model. While this allows for a more powerful adversary, the underlying assumption that the security is reduced to is weaker. Here, security is based on the generic bilinear group model. In this model, an adversary can be shown to have advantage $O(q^2/p)$ in the CP-ABE game (choosing W after Phase 1), where q is the number of group elements received from all queries. For more details, see [3, Appendix A]. The uncloneability and privacy-preserving properties of the scheme can be shown in a similar way as the tk-ABE scheme in Section 5.

C Non-Interactive Uncloneable Scheme

In a non-interactive uncloneable CP-ABE scheme, we bind a user's personal information directly into the user's private key. In addition, we require that this personal information that embedded into the private key must be needed to encrypt any ciphertext (that the user is allowed to decrypt).

C.1 Security Model

We define the three security properties needed for a uncloneable CP-ABE scheme in the non-interactive model.

As with other CP-ABE schemes, we define security with respect to ciphertext indistinguishability using a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The game is given below in Security Game 2.

Init: The adversary chooses a valid decryption policy W to be challenged upon.
Setup: The Challenger runs $\mathsf{Setup}(\ell, n)$ and gives PK to the adversary.
Phase 1: The adversary can adaptively make polynomially many private key requests for any user with any attribute set $S \notin W$. In these requests, either the adversary provides the personal information b for the private key request, or the challenger randomly chooses $b \in B$ for the new key.
Challenge: The adversary submits two different plaintext messages M_0 and M_1 of equal length to the challenger. The challenger randomly chooses $c \in [0, 1]$ and gives $\mathsf{Encrypt}(PK, M_c, W)$ to the adversary.
Phase 2: The adversary repeats phase 1.
Guess: The adversary guesses $c' \in [0, 1]$ and wins if $c = c'$.

Security Game 2: Non-Interactive Uncloneable CP-ABE

Since the adversary can simply guess correctly with probability 1/2, the advantage of the adversary in the game is defined as $\mathcal{ADV} = \Pr[c' = c] - 1/2$. A CP-ABE scheme is said to be secure against CPA attacks in the selective ID model if no polynomial time adversary has non-negligible advantage in the above game.

The intuition of an uncloneable non-interactive ABE scheme is the same as for tk-ABE except that we do not require oracle access to the token server here. We define it as follows.

Definition 7 (Uncloneability). *Let Δ be a ciphertext-policy ABE scheme. We say that Δ is* uncloneable *if the following holds for all users in Δ: Let ID be any user in Δ with private key D and personal information b. If D is known then b can be computed in time $O(|B|)$.*

Again, we can strengthen this definition of uncloneability by relaxing the meaning of a cloned key. To this end we define a partial cloned key D' of the key D (denoted by $D' \subseteq D$) to be any unchanged portion of D. A cloned key is then also a partial cloned key. With this definition, we have the following strengthened notion of uncloneability.

Definition 8 (Strong Uncloneability). *Let Δ be a ciphertext-policy ABE scheme with public key PK. We say that Δ is* strongly uncloneable *if the following holds for all users in Δ: Let ID be any user in Δ with private key D and personal information b. Let $D' \subseteq D$ be any partial cloned key of D, C be any valid ciphertext, and W be any policy. If there exists a polynomial time algorithm $\mathcal{F}$ that, given input PK, C , ID and D', outputs the decryption of C, that is*

$$\mathcal{F}(PK, C, ID, D') = \mathsf{Decrypt}(PK, C, \mathsf{GetToken}(ID, \hat{C}), D', b),$$

then there exists an extractor algorithm $\mathcal{X}$ that, given input PK, C , ID and D', outputs user ID's personal information b in time $O(|B|)$.

Notice that in this definition, we also require that the partial cloned key D' is sufficient for user ID to decrypt a ciphertext C using $\mathsf{Decrypt}$. Basically, this notion of uncloneability means that knowledge of enough components (or bits, or parts) of a user's private key that enables one to decrypt a ciphertext is sufficient to compute that user's personal information.

Definition 9 (Privacy-Preserving). *Let Δ be an uncloneable ABE scheme with public key PK and master key MK. Δ is said to be* privacy-preserving *if for every user with identification ID, private key D, attribute set S and personal information b, it holds that*

$$\Pr[b \mid PK, MK, D, ID, S] = \Pr[b \mid ID, S].$$

This definition of privacy-preserving ensures that a user's personal information b is not more at risk of being exposed as a result of being a member of the ABE system.

C.2 Construction

We propose a CP-ABE scheme that is non-delegatable without the use of decryption tokens. This is achieved by adding a single dummy attribute that each entity in the system possesses and is needed for decryption of any ciphertext. Essentially, *all* valid decryption policies for any ciphertext implicitly requires the user to have this attribute. In the scheme, this attribute (we'll denote it as the zeroth attribute i_0) for a given user will consist of some personal information about the user (that they will not wish to disclose) appended by some random bits. If a user delegates their key (or parts of their key needed for decryption) then the user must reveal this zeroth attribute and hence reveal their personal information. Thus, the non-delegatable property comes from a user not wishing to divulge their personal information. In the following, let B be the space of valid personal information.

The four algorithms in the scheme are given below.

$\mathsf{Setup}(\ell,\, n)$: First the bilinear groups in which the scheme will operate are chosen: let $\mathbb{G}$ and $\mathbb{G}_1$ be bilinear groups of prime order p (where p is an ℓ-bit prime), let g be a randomly chosen generator of $\mathbb{G}$, and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be an efficient bilinear map. Next, let $\mathcal{N} = \{1, \ldots, n\}$ be the set of attributes and randomly choose $y,\, t_0, t_1, \ldots, t_{3n}$ from $\mathbb{Z}_p$. The system public key PK is

$$PK = \langle Y = e(g, g)^y, T_0 = g^{t_0}, T_1 = g^{t_1}, \ldots, T_{3n} = g^{t_{3n}}, g, e, \mathbb{G}, \mathbb{G}_1, \mathcal{N} \rangle,$$

and the system private key, called the master key, is $MK = \langle y, t_0, t_1, \ldots, t_{3n} \rangle$. The system public key is made public and the master key is given to the trusted key generator TKG.

Note: We assume that elements in $\mathbb{G}$ and $\mathbb{G}_1$ have binary representation with bitlength ℓ.
Note: This is the same as in our tk-ABE Setup algorithm except for the addition of t_0 and T_0.

$\mathsf{KeyGen}(MK,\, PK,\, S,\, b)$: To generate a private decryption key for a user with identification ID, attributes S and personal information b, the trusted key generator TKG does the following: If $b \notin B$ then abort. Otherwise, for each attribute $i \in \mathcal{N}$, choose a random $r_i \in \mathbb{Z}_p$ and compute

$$D_i = \begin{cases} g^{\frac{r_i}{t_i}} & \text{if } i \in S \text{ (user has attribute } i) \\ g^{\frac{r_i}{t_{n+i}}} & \text{if } i \notin S \text{ (user does not have attribute } i), \end{cases}$$

$$F_i = g^{\frac{r_i}{t_{2n+i}}} \qquad \text{(for don't care attributes in } W).$$

Let m be the bitlength of b. Repeatedly choose a random $(\ell - m)$-bit positive integer a until $D_0 = a||b \in \mathbb{G}_1$, where $||$ denotes concatenation. Let r_0 and r be (implicitly) defined as $D_0 = g^{r_0/t_0}$ and $r = \sum_{i=0..n} r_i$. Finally, compute

$$\widehat{D} = \frac{g^y}{D_0^{t_0} g^{r_1+\cdots+r_n}} = \frac{g^y}{g^{r_0+r_1+\cdots+r_n}} = \frac{g^y}{g^r} = g^{y-r},$$

and output the private decrypting key $D = \langle \widehat{D}, D_0, \{D_i, F_i\}_{i \in \mathcal{N}} \rangle$. The TKG then deletes a, b and D.

$\mathsf{Encrypt}(PK,\, M,\, W)$: To encrypt a message $M \in \mathbb{G}_1$ with decryption policy W over I, pick a random $s \in \mathbb{Z}_p$ (the nonce for the ciphertext) and compute $\widetilde{C} = MY^s = Me(g, g)^{ys}$, $\widehat{C} = g^s$, and $C_0 = T_0^s = g^{st_0}$. For each attribute $i \in \mathcal{N}$ compute

$$C_i = \begin{cases} T_i^s = g^{st_i} & \text{if } i \in I,\ \underline{i} = i & \text{(must have attribute } i) \\ T_{n+i}^s = g^{st_{n+i}} & \text{if } i \in I,\ \underline{i} = \neg i & \text{(must not have attribute } i) \\ T_{2n+i}^s = g^{st_{2n+i}} & \text{if } i \notin I & \text{(don't care about attribute } i). \end{cases}$$

The ciphertext is $C = \langle W, \widetilde{C}, \widehat{C}, \{C_i\}_{i=0..n} \rangle$.

$\mathsf{Decrypt}(PK,\, C,\, D,\, S)$: To decrypt a ciphertext C (with decryption policy W over attributes I), a user with private decryption key D and attributes S satisfying the decryption policy W does the following: for each attribute $i \in \mathcal{N}$ compute $E_i = e(g, g)^{sr_i}$ where

$$E_i = e(D_i, C_i) = \begin{cases} e(g^{st_i}, g^{r_i/t_i}) = e(g, g)^{sr_i} & \text{if } i \in I, \underline{i} = i, i \in S \\ e(g^{st_{i+n}}, g^{r_i/t_{n+i}}) = e(g, g)^{sr_i} & \text{if } i \in I, \underline{i} = \neg i, i \notin S, \end{cases}$$

or, if i is *do not care* in I (i.e., $i \notin I$),

$$E_i = e(C_i, F_i) = e(g^{st_{2n+i}}, g^{r_i/t_{2n+i}}) = e(g,g)^{sr_i},$$

and also $E_0 = e(C_0, D_0) = e(g^{st_0}, g^{r_0/t_0}) = e(g,g)^{sr_0}$. Multiplying all of the E_i together yields

$$Z = \prod_{i=0}^{n} E_i = \prod_{i=0}^{n} e(g,g)^{sr_i} = e(g,g)^{s\sum_{i=0}^{n} r_i} = e(g,g)^{sr},$$

which when multiplied by $e(\widehat{C}, \widehat{D})$ gives

$$U = e(\widehat{C}, \widehat{D})Z = e(g^s, g^{y-r})e(g,g)^{sr} = e(g,g)^{(s(y-r)+sy)} = e(g,g)^{sy}.$$

The plaintext M can then be revealed by computing

$$\frac{\widetilde{C}}{U} = \frac{Me(g,g)^{sy}}{e(g,g)^{sy}} = M.$$

C.3 Security of Non-Interactive Scheme

Our non-interactive non-delegatable ABE scheme satisfies the three security properties outlined above (ciphertext indistinguishability, uncloneability, privacy-preserving). We state the results below, but leave the proofs to the full version of the paper. The ciphertext indistinguishability follows from the CN-scheme, while the other properties follow closely with the tk-ABE scheme presented in Section 5.

Theorem 5. *Our scheme is secure against chosen plaintext attacks in the selective ID model provided that the decisional bilinear Diffie-Hellman problem (DBDH) assumption holds.*

Theorem 6. *Our scheme is uncloneable.*

Theorem 7. *Our scheme is privacy-preserving.*

While this non-interactive scheme satisfies the three security properties, we observe that user's in the system (with valid private keys) that have the same attributes can combine their keys by simply multiplying their keys together (componentwise multiplication) to create a new valid key corresponding to the same attributes. The component of the new key corresponding to a users personal information is now the product of the colluding users personal information. Thus, the colluding users expose their personal information to each other. Without knowledge of one of the user's personal information though, the new key does not reveal anything about the colluding user's personal information. Thus, this scheme is weaker than the tk-ABE schemes that are presented earlier.

C.4 Additional Comments

We also note that this non-interactive scheme seems to be specific to Cheung and Newport's scheme. The idea behind the transformation (ABE to non-delegatable ABE) does not seem apply to any of the other known ABE scheme. This is because in the other schemes, typically, a user can re-randomize their secret key buy multiplying all of the components in their private key by some random number. In our scheme, we require that a user's personal information explicitly appear in the private key. Re-randomization destroys this.

C.5 Interactive vs Non-Interactive Uncloneable ABE Schemes

The following table shows which type of non-delegatable scheme is better suited for different conditions. A ✓ denotes that this scheme is suitable for the given condition to be efficient, while a ✗ denotes that the scheme is not well suited for that property/condition.

	tk-ABE	Non-Interactive
Key revocation	✓	✗
Communication complexity	✗	✓
Many attributes	✗	✓
Based on different ABE schemes	✓	✗