Identity-Based Proxy Re-Encryption

Matthew Green Giuseppe Ateniese

Information Security Institute
Johns Hopkins University
3400 N. Charles St.
Baltimore, MD 21218
{ateniese,mgreen}@cs.jhu.edu

Abstract

In a proxy re-encryption scheme a semi-trusted proxy converts a ciphertext for Alice into a ciphertext for Bob *without* seeing the underlying plaintext. A number of solutions have been proposed in the public-key setting. In this paper, we address the problem of Identity-Based proxy re-encryption, where ciphertexts are transformed from one *identity* to another. Our schemes are compatible with current IBE deployments and do not require any extra work from the IBE trusted-party key generator. In addition, they are non-interactive and one of them permits multiple re-encryptions. Their security is based on a standard assumption (DBDH) in the random oracle model.

Keywords: proxy re-encryption, identity-based encryption, bilinear maps.

1 Introduction

In a proxy re-encryption scheme, a proxy can convert an encryption computed under Alice's public-key into an encryption intended for Bob. Such a scheme can be used by Alice to temporarily forward encrypted messages to Bob without giving him her secret key. The fundamental property of proxy re-encryption schemes is that the proxy is not fully trusted, i.e., it does not know the secret keys of Alice or Bob and does not learn the plaintext during the conversion. The proxy and Bob, however, are not allowed to collude, thus it is usually assumed that at least one of the two is honest or that their collusion is preventable or detectable via other means.

A number of proxy re-encryption protocols have been proposed in the context of public-key encryption [16, 3, 1, 15, 12]. In this work we extend the notion of proxy re-encryption to the area of Identity-Based Encryption (IBE), in which senders encrypt messages using the recipient's identity (a string) as the public key. For instance, Charles could encrypt a message for Alice by just using her email address. First introduced by Shamir in 1984 and then realized by Boneh-Franklin [7] and by Cocks [11] several years later, identity-based encryption has proven useful in solving several key-distribution issues, and has permitted the development of a variety of novel cryptographic protocols, e.g., secret handshakes [2], public-key searchable encryption [5, 19], CCA2-secure public-key encryption [10], and digital signatures [6]. The Boneh-Franklin scheme is particularly efficient, and has been practically deployed [17].

Our identity-based proxy re-encryption (IB-PRE) schemes allow a proxy to translate an encryption under Alice's identity into one computed under Bob's identity. The proxy uses *proxy keys*, or *re-encryption keys*, to perform the translation without being able to learn the plaintext. Moreover, no information on the secret

keys of Alice and Bob can be deduced from the proxy keys. Our constructions are compatible with existing Boneh-Franklin IBE deployments, and can be implemented using existing secrets and parameters.

Remember that users in an Identity-Based Encryption scheme request keys from a trusted party known as a Private Key Generator (PKG). Thus, in principle, it is possible that proxy keys could be generated by the PKG directly. However, we categorically exclude this possibility and we focus only on schemes where individual users delegate their own decryption rights, without the involvement of the Private Key Generator. This is for theoretical and practical reasons: (1) From a theoretical point of view, having the PKG, or any other trusted party, generating the proxy keys makes the problem of finding IB-PRE schemes quite unchallenging given prior art, (2) from a practical point of view, it is clearly undesirable to have the PKG involved in the generation of proxy keys. It would constitute a considerable bottleneck in many applications, it would force the PKG to be *on-line* and available even during the generation of proxy keys (other than IBE keys), and, in certain applications, it would make the PKG liable for creating (potentially unwanted) decryption rights.

Previous Work. Mambo and Okamoto proposed a technique for delegating decryption rights in [16]. Blaze, Bleumer and Strauss [3] later presented the first secure "atomic" primitive: an Elgamal-based approach in which the proxy could not learn the message being processed. Unfortunately, the approach in [3] is inherent bidirectional: a corrupted proxy can re-encrypt ciphertexts not only from Alice to Bob, but also from Bob to Alice. Even worse, collusion between the Proxy and "delegator" Alice could reveal the secret key of "delegatee" Bob. Jakobsson [15], and Zhou, Mars, Schneider and Redz [21] partially addressed these concerns by proposing a quorum-based protocol which divided the proxy into many components.

More recent works have focused on the development of unidirectional proxy re-encryption schemes, where collusion between a delegator and the proxy does not compromise the delegatee. Dodis and Ivan [12] realized a form of unidirectional proxy *encryption* by using double-encryption (or by splitting a single decryption key into two parts). Their approach permits a form of single-delegation proxy re-encryption when parties hold pre-shared keys. Ateniese, Fu, Green and Hohenberger [1] proposed an improved, *non-interactive* unidirectional scheme which removed the need for pre-shared keys and permitted arbitrary delegations.

Dodis and Ivan [12] also proposed a very different identity-based proxy encryption scheme in which the PKG delegates decryption rights for *all* identities in the system (*e.g.,* to provide key escrow for law enforcement). Such delegation is non-divisible, *i.e.,* the PKG cannot delegate decryption rights for only a subset of identities in the system. This approach differs conceptually from our non-interactive approach, where individual users delegate their decryption rights. Finally, the Dodis/Ivan system has significant security implications: collusion between the proxy and delegatee results in a system-wide compromise, allowing the colluders to reconstruct the IBE master secret.

Recently, Boneh, Goh and Matsuo [8] presented a *hybrid* form of proxy re-encryption based on IBE. In such schemes, the PKG performs all delegations; thus users are unable to perform offline ("non-interactive") delegations and each delegation requires a costly online request to the PKG. Furthermore, the Boneh-Goh-Matsuo approach specifies a new private-key generation algorithm and it seems therefore incompatible with existing IBE deployments.

Paper Outline. The outline of the rest of this paper is as follows. In section 3 we present definitions for Identity-Based Proxy Re-encryption and for the hardness assumptions used in our proofs. In section 4 we introduce our constructions. In section 5 we discuss several applications for the new primitives. Finally, section 6 lists open research problems and provides our conclusions.

2 Properties of Our Schemes

Ateniese *et. al.* [1] propose a series of properties by which to compare proxy re-encryption schemes. We briefly reiterate some of these properties, in particular those that our scheme provides and that, we believe, are relevant for practical instantiations of Identity-Based Proxy Re-encryption.

- *Unidirectionality.* A unidirectional scheme permits user A to delegate to user B, without permitting A to decrypt user B's ciphertexts.
- *Non-Interactivity.* Non-interactive schemes permit user A to construct a re-encryption key $rk_{id_A \to id_B}$ while offline, (*i.e.,*without the participation of B or the Private Key Generator).
- *Multi-use.* A multi-use scheme permits the proxy (or proxies) to perform multiple re-encryptions on a single ciphertext, e.g., re-encrypt from A to B, then re-encrypt the result from B to C, etc..
- *Non-transitivity.* In a non-transitive scheme, the proxy is not authorized to re-delegate decryption rights. In particular, the proxy cannot combine re-encryption keys to create new delegations. For example, the proxy cannot construct a re-encryption key $rk_{id_A \to id_C}$ from the two keys $rk_{id_A \to id_B}$ and $rk_{id_B \to id_C}$.
- *Space-optimal.* Many existing schemes (*e.g.,* [12, 8, 1]) incur additional communication costs in order to support re-encryption. This inefficiency takes several common forms, including: (a) ciphertext expansion upon re-encryption (see the practical implementations of [1]), (b) a required pre-distribution stage in which secrets are shared with delegatees (as in [12]), or (c) the inclusion of "extra" ciphertext material used solely to support proxy re-encryption (see [8]).

In this paper we focus on unidirectional schemes only. A unidirectional IB-PRE allows Alice to delegate decryption rights to Bob without requiring Bob to do the same. Unidirectional IB-PRE is clearly a more powerful primitive than a bidirectional one but also harder to devise. Notice, also, that a bidirectional scheme can always be achieved by running a unidirectional one in both directions, i.e., from Alice to Bob and viceversa.

In addition, we believe that non-interactivity is a fundamental property and our schemes provide it. In a non-interactive scheme, Alice can generate the re-encryption key from Bob's identity, without ever involving Bob. In the identity-based setting, this property provides an interesting twist: Alice can delegate decryption rights to delegatees that do not exist yet or will join the system later. Moreover, as noted by Boneh and Franklin [7], identities can be seen as credentials and express conditions. For instance, an encryption under *"Alice* $||$ *security-clearance* $||$ *time period"* can be opened by Alice only if she has security clearance and within the time period specified in the string. Analogously, in our schemes, Alice can specify the conditions under which the delegation of decryption rights has to happen. We will explore applications of this feature in section 5.

Finally, one of our schemes is *multi-use* in the sense that once a re-encryption from Alice to Bob is computed, the resulting ciphertext can be re-encrypted again from Bob to Charles, etc., multiple times. Finding an unidirectional and multi-use scheme was left as an open problem in prior art for the public-key case. We show how to achieve this property for our IB-PRE but at the cost of allowing the ciphertext to expand linearly with respect to the number of re-encryptions (but this seems to be inevitable for a non-interactive scheme).

3 Definitions

We begin by describing the setting and computational problems used within this work. We then formally define an Identity-Based Proxy Re-encryption scheme and propose a new, generalized security definition.

Definition 3.1 (Bilinear Map) We say a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is a *bilinear map* if:

1. $\mathbb{G}_1$, $\mathbb{G}_T$ are groups of the same prime order q.
2. For all $a, b \in \mathbb{Z}_q^*$, $g \in \mathbb{G}_1$, $e(g^a, g^b) = e(g,g)^{ab}$.
3. The map is non-degenerate (i.e., if $\mathbb{G}_1 = \langle g \rangle$, then $\mathbb{G}_T = \langle e(g,g) \rangle$).
4. e is efficiently computable.

For simplicity our constructions are defined in the symmetric setting, however they also work in the *asymmetric* one with a bilinear map of the form: $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

Definition 3.2 (Decisional Bilinear Diffie Hellman Assumption (DBDH)) Our schemes are based on the assumed intractability of the Decisional Bilinear Diffie-Hellman problem (DBDH) in $\mathbb{G}_1, \mathbb{G}_T$. This assumption is believed to hold in certain groups, and used as the basis of several Identity-Based Encryption schemes, *e.g.,* [4, 18].

We define the DBDH problem as follows: Let $(\mathbb{G}_1, \mathbb{G}_T)$ be a pair of bilinear groups with an efficiently computable pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, and let g be a random generator of $\mathbb{G}_1$. The DBDH problem is to decide, given a tuple of values $(g, g^a, g^b, g^c, T) \in \mathbb{G}_1^4 \times \mathbb{G}_T$ (where $a, b, c \in_R \mathbb{Z}_q^*$), whether $T = e(g,g)^{abc}$ or if T is a random element of $\mathbb{G}_T$.

Let k be a security parameter of sufficient size. Formally, we say that the DBDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$ if for all probabilistic polynomial time algorithms $\mathcal{A}$, the following condition is true:

$$\left| \begin{array}{l} \Pr\left[a,b,c \xleftarrow{\$} \mathbb{Z}_q^*;\ 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, e(g,g)^{abc}). \right] - \\ \Pr\left[a,b,c \xleftarrow{\$} \mathbb{Z}_q^*;\ T \xleftarrow{\$} \mathbb{G}_T;\ 1 \leftarrow \mathcal{A}(g, g^a, g^b, g^c, T). \right] \end{array} \right| \leq \nu(k)$$

Where $\nu(\cdot)$ is defined as a *negligible* function, *i.e.,* for all polynomial functions $p(\cdot)$, $\nu(k) < 1/p(k)$.

3.1 Identity-Based Proxy Re-Encryption

An Identity-Based Proxy Re-encryption (IB-PRE) scheme is an extended Identity Based Encryption scheme. The first extension is an algorithm that generates *re-encryption keys* that can be given to the proxy. The proxy uses the second algorithm to apply these re-encryption keys to ciphertexts and "atomically" re-encrypt them from one identity to another. In a *non-interactive* scheme, re-encryption keys may be generated by the delegator using only her IBE secret key— the IBE master secret is not required.

Encryption Levels. Our definitions refer to the notion of an "encryption level" as an implicit property of a ciphertext. A ciphertext generated directly using the Encrypt algorithm is termed a "level-1" ciphertext. Applying the re-encryption algorithm to a level-ℓ ciphertext results in a level-$(\ell+1)$ ciphertext. Specific constructions may optionally place bounds on the number of consecutive re-encryptions; for instance, non-"multi-use" schemes such as [12, 8, 1] are limited to a single re-encryption. In our definitions below, we define MaxLevels as the highest-possible encryption level (for a single-use scheme, this value is 2).

Definition 3.3 (Non-interactive Identity-Based Proxy Re-encryption (IB-PRE)) A *non-interactive identity-based proxy re-encryption scheme* is tuple of algorithms (Setup, KeyGen, Encrypt, Decrypt, RKGen, Reencrypt):

- $\mathsf{Setup}(1^k, \mathsf{MaxLevels})$ accepts a security parameter and a value indicating the maximum number of consecutive re-encryptions permitted by the scheme. The algorithm outputs both the master public parameters (params) which are distributed to users, and the master secret key (msk) which is kept private.
- $\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$ on input an identity $id \in \{0,1\}^*$ and the master secret key, outputs a decryption key sk_{id} corresponding to that identity.
- $\mathsf{Encrypt}(\mathsf{params}, id, m)$ on input a set of public parameters, an identity $id \in \{0,1\}^*$, and a plaintext $m \in \mathcal{M}$, output c_{id}, the encryption of m under the specified identity.
- $\mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$ on input a secret key sk_{id_1} (derived via the KeyGen algorithm) and identities $(id_1, id_2) \in \{0,1\}^*$, outputs a *re-encryption key* $rk_{id_1 \to id_2}$.
- $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})$ on input a ciphertext c_{id_1} under identity id_1, and a re-encryption key $rk_{id_1 \to id_2}$ (generated by the RKGen routine), outputs a *re-encrypted ciphertext* c_{id_2}.
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c_{id})$ decrypts the ciphertext c_{id} using the secret key sk_{id}, and outputs m or $\perp$.

Correctness. Intuitively, an IB-PRE scheme is *correct* if the Decrypt algorithm always outputs the expected decryption of a properly-generated ciphertext (when supplied with the appropriate decryption key). We define "proper generation" as the process of (1) encrypting a plaintext using Encrypt, and subsequently (2) iteratively applying the Reencrypt algorithm up to $\mathsf{MaxLevels} - 1$ times using valid re-encryption keys.

Slightly more formally, let $c_{id_1} \leftarrow \mathsf{Reencrypt}^n(\cdots, \mathsf{Encrypt}(\mathsf{params}, \cdot, m))$ be a properly-generated ciphertext. Then $\forall m \in \mathcal{M}, \forall id_1, id_2 \in \{0,1\}^*, \forall n < \mathsf{MaxLevels} - 1$, where $sk_{id_1} = \mathsf{KeyGen}(\mathsf{msk}, id_1)$, $sk_{id_2} = \mathsf{KeyGen}(\mathsf{msk}, id_2)$, $rk_{id_1 \to id_2} \leftarrow \mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_1, id_2)$, the following propositions hold:

- $\mathsf{Decrypt}(\mathsf{params}, sk_{id_1}, c_{id_1}) = m$
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id_2}, \mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})) = m$

Security. Security definitions for Identity-Based Encryption (see [7]) ensure that no reasonable set of colluding keyholders will obtain an advantage against non-colluding users. Identity-Based Proxy re-encryption requires a further extension of this collusion guarantee, to model the presence of a semi-trusted *proxy* that possesses re-encryption keys, and may behave maliciously or fall under an adversary's control. Previous security definitions for proxy re-encryption (*e.g.,* [3, 12, 1]) have treated this issue by specifying a separate definitional game to model security against a malicious proxy. We eliminate the need for separate games by granting the adversary access to a "re-encryption key" extraction oracle, which returns re-encryption (proxy) keys for arbitrary pairs $(id_i \to id_j)$.

Definition 3.4 (Security of Non-Interactive Identity Based Proxy Re-Encryption (IND-PrID-ATK))
Let $\mathcal{S}$ be an IB-PRE scheme defined as a tuple of algorithms (Setup, KeyGen, Encrypt, Decrypt, RKGen, Reencrypt). Security is defined according to the following game $Exp^{\mathcal{A}, \mathsf{IND\text{-}PrID\text{-}ATK}, i}$, where $\mathsf{ATK} \in (\mathsf{CPA}, \mathsf{CCA})$.

1. SELECT. Choose $i \xleftarrow{\$} \{0,1\}$.
2. SETUP. Run $\mathsf{Setup}(1^k)$ to get $(\mathsf{params}, \mathsf{msk})$, and give params to $\mathcal{A}$.

3. FIND PHASE. $\mathcal{A}$ makes the following queries. At the conclusion of this phase $\mathcal{A}$ will select $id^* \in \{0,1\}^*$ and $(m_0, m_1) \in \mathcal{M}^2$.

   (a) For $\mathcal{A}$'s queries of the form $(\texttt{extract}, id)$, return $sk_{id} = \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$ to $\mathcal{A}$.
   (b) For $\mathcal{A}$'s queries of the form $(\texttt{rkextract}, id_1, id_2)$, where $id_1 \neq id_2$, return $rk_{id_1 \to id_2} = \mathsf{RKGen}(\mathsf{params}, \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id_1), id_1, id_2)$ to $\mathcal{A}$.
   (c) For $\mathcal{A}$'s queries of the form $(\texttt{decrypt}, id, c)$, if $\mathsf{ATK} = \mathsf{CCA}$ then return $m = \mathsf{Decrypt}(\mathsf{params}, \mathsf{RKGen}(\mathsf{params}, \mathsf{msk}, id), c)$ to $\mathcal{A}$. Otherwise if $\mathsf{ATK} = \mathsf{CPA}$, return $\perp$ to $\mathcal{A}$.
   (d) For $\mathcal{A}$'s queries of the form $(\texttt{reencrypt}, id_1, id_2, c)$, if $\mathsf{ATK} = \mathsf{CCA}$ then derive a re-encryption key $rk_{id_1 \to id_2} = \mathsf{RKGen}(\mathsf{params}, \mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id_1), id_1, id_2)$, and return $c' = \mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, id_1, id_2, c)$ to $\mathcal{A}$. If $\mathsf{ATK} = \mathsf{CPA}$, return $\perp$ to $\mathcal{A}$.

   Note that $\mathcal{A}$ is not permitted to choose id^* such that trivial decryption is possible using keys extracted during this phase (*e.g.,* by using extracted re-encryption keys to translate from id^* to some identity for which $\mathcal{A}$ holds a decryption key).
4. CHOICE AND CHALLENGE. When $\mathcal{A}$ presents $(\texttt{choice}, id^*, m_0, m_1)$, compute $c^* = \mathsf{Encrypt}(\mathsf{params}, id^*, m_i)$ and give c^* to $\mathcal{A}$.
5. GUESS STAGE. $\mathcal{A}$ continues to make queries as in the FIND stage, with the following restrictions. Let $\mathcal{C}$ be a set of ciphertext/identity pairs, initially containing the single pair $\langle c^*, id^* \rangle$. For all $c \in \mathcal{C}$ and for all rk given to $\mathcal{A}$, let $\mathcal{C}'$ be the set of all possible values derived via (one or more) consecutive calls to Reencrypt:

   (a) $\mathcal{A}$ is not permitted to issue any query of the form $(\texttt{decrypt}, id, c)$ where $\langle c, id \rangle \in (\mathcal{C} \cap \mathcal{C}')$.
   (b) $\mathcal{A}$ is not permitted to issue any queries $(\texttt{extract}, id)$ or $(\texttt{rkextract}, id_1, id_2)$ that would permit trivial decryption of any ciphertext in $(\mathcal{C} \cap \mathcal{C}')$.
   (c) $\mathcal{A}$ is not permitted to issue any query of the form $(\texttt{reencrypt}, id_1, id_2, c)$ where $\mathcal{A}$ possesses the keys to trivially decrypt ciphertexts under id_2 and $\langle c, id_1 \rangle \in (\mathcal{C} \cap \mathcal{C}')$. On successful execution of any re-encrypt query, let c' be the result and add the pair $\langle c', id_2 \rangle$ to the set $\mathcal{C}$.

   At the conclusion of this stage, $\mathcal{A}$ outputs i', where $i' \in \{0,1\}$.

The outcome of the game is determined as follows: If $i' = i$ then $\mathcal{A}$ wins the game. Let $(\texttt{ext}, \texttt{rk}, \texttt{dec}, \texttt{renc})$ be the oracles for the FIND phase of the game, and $(\texttt{ext}', \texttt{rk}', \texttt{dec}'\texttt{renc}')$ be the same oracles modified for the GUESS stage. $\mathcal{A}$'s advantage in the above game, $Adv_{\mathcal{A}}^{\mathsf{IND\text{-}PrID\text{-}ATK}}$ is defined as:

$$\left| Pr\left[i' = i \;\middle|\; \begin{array}{c} i \xleftarrow{\$} \{0,1\};\ (\mathsf{params}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^k); \\ (id^*, m_0, m_1, t) \leftarrow \mathcal{A}^{\texttt{ext}(\cdot), \texttt{rk}(\cdot), \texttt{dec}(\cdot), \texttt{renc}(\cdot)}(\mathsf{params}); \\ c^* \leftarrow \mathsf{Encrypt}(\mathsf{params}, id^*, m_i); \\ i' \leftarrow \mathcal{A}^{\texttt{ext}'(\cdot), \texttt{rk}'(\cdot), \texttt{dec}'(\cdot), \texttt{renc}'(\cdot)}(\mathsf{params}, c^*, t); \end{array} \right] - 1/2 \right|$$

We say that the Identity-Based Proxy Re-encryption scheme $\mathcal{S}$ is IND-Pr-ID-CPA-secure if for all probabilistic polynomial time algorithms $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathsf{IND\text{-}Pr\text{-}ID\text{-}CPA}} \leq \nu(k)$. We say that the Identity-Based Proxy Re-encryption scheme $\mathcal{S}$ is IND-PrID-CCA-secure if for all probabilistic polynomial time algorithms $\mathcal{A}$, $Adv_{\mathcal{A}}^{\mathsf{IND\text{-}PrID\text{-}CCA}} \leq \nu(k)$.

Comparison to IBE Security. Observe that the security experiments we present are equivalent to an execution of the experiments defined by Boneh/Franklin security model (IND-ID-CPA/IND-ID-CCA) [7], provided

that $\mathcal{A}$ makes no queries of the form $(\mathsf{rkextract}, \ldots)$ or $(\mathsf{reencrypt}, \ldots)$. Therefore, when re-encryption keys are not deployed, it is easy to see that any IB-PRE secure in the IND-PrID-ATK remains secure in the IND-ID-ATK sense.

4 Non-interactive Unidirectional Proxy Re-encryption Schemes

The first schemes we present are based on Boneh and Franklin's IBE scheme [7], and are secure under the Decisional Bilinear Diffie-Hellman Assumption (DBDH) in the random oracle model. While ciphertexts in the proposed schemes have a different form from those in the standard Boneh-Franklin scheme, the master parameters and secret keys remain unchanged. As a result, it is possible to implement proxy re-encryption within an existing Boneh-Franklin deployment (*i.e.,* using pre-existing parameters and keys).

4.1 A First Attempt (IBP1)

Consider a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1 = \langle g \rangle$. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two independent hash functions[1] such that: $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathcal{H}_2 : \mathbb{G}_T \rightarrow \mathbb{G}_1$. Finally, let s and g^s be the master secret and public key of the PKG, respectively. For some $r \in_R \mathbb{Z}_q^*$, an encryption of $m \in \mathbb{G}_T$ under Alice's identity can be computed as:

$$IBE_{Alice}(m) = (g^r, m \cdot e(g^s, \mathcal{H}_1(Alice))^r)$$

Suppose Alice wants to delegate her decryption rights to Bob. She must generate a re-encryption key to give to the proxy. Let $IBE_{Bob}(\cdot)$ be a standard identity-based encryption under Bob's identity. Alice selects a random $X \stackrel{\$}{\leftarrow} \mathbb{G}_T$ and generates the re-encryption key as:

$$rk_{Alice \rightarrow Bob} = \mathcal{H}_1(Alice)^{-s} \cdot \mathcal{H}_2(X),\ IBE_{Bob}(X),$$

Given an encryption for Alice, $IBE_{Alice}(m) = (c_1, c_2)$ the proxy can transform it into an encryption for Bob by releasing: $(c'_1 = c_1, c'_2 = c_2 \cdot e(g^r, rk_{Alice \rightarrow Bob}), c'_3 = IBE_{Bob}(X))$. Indeed, notice that:

$$\begin{aligned} c'_1 &= g^r \\ c'_2 &= m \cdot e(g^r, \mathcal{H}_2(X)), \\ c'_3 &= IBE_{Bob}(X). \end{aligned}$$

Bob can recover X from c'_3 and then m by computing $c'_2 / e(c'_1, \mathcal{H}_2(X))$.

In practice, the scheme presented above can be seen as a variant of the efficient Dodis/Ivan [12] key-splitting approach applied to settings where the decryption process makes use of a bilinear map. Note that (1) The scheme is *unidirectional* since the key $rk_{Alice \rightarrow Bob}$ can be used to convert ciphertexts from Alice to Bob but not vice versa. (2) It is *non-interactive* since Bob is not involved during the generation of the re-encryption key. (3) It provides *non-transitivity* since the proxy is not allowed to create new re-encryption keys from the existing ones. (4) Finally, we observe that the scheme is *multi-use* since the proxy can re-encrypt the result of a re-encryption and do it multiple times. To see this, consider the re-encryption ciphertext above: (c'_1, c'_2, c'_3). Notice that c'_3 is just a standard IBE encryption for Bob! A proxy equipped with a re-encryption

[1] Both $\mathcal{H}_1(\cdot)$ and $\mathcal{H}_2(\cdot)$ are more properly "hash-and-encode" functions (see Boneh-Franklin [7] for a detailed definition). Each function consist of a standard hash function which maps inputs to elements of the finite field of order q and then uses an admissible encoding function, MapToPoint, to map those elements into points in $\mathbb{G}_1$.

key $rk_{Bob \to Charles}$ could just apply the re-encryption algorithm *recursively* to c'_3 and allow Charles to recover X which in turn allows him to recover the original message m.

Scheme Description. We now provide a formal description of the scheme (IBP1).

- Setup. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}_1 = \langle g \rangle$ and $\mathbb{G}_T$ have order q. Let $\mathcal{H}_1, \mathcal{H}_2$ be independent full-domain hash functions $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ and $\mathcal{H}_2 : \mathbb{G}_T \to \mathbb{G}_1$. To generate the scheme parameters, select $s \xleftarrow{\$} \mathbb{Z}_q^*$, and output $\mathsf{params} = (\mathbb{G}_1, \mathcal{H}_1, \mathcal{H}_2, g, g^s)$, $\mathsf{msk} = s$.
- $\mathsf{KeyGen}(\mathsf{params}, \mathsf{msk}, id)$. To extract a decryption key for identity $id \in \{0,1\}^*$, return $sk_{id} = \mathcal{H}_1(id)^s$.
- $\mathsf{Encrypt}(\mathsf{params}, id, m)$. To encrypt m under identity id, select $r \xleftarrow{\$} \mathbb{Z}_q^*$ and output $c_{id} = (g^r, m \cdot e(g^s, \mathcal{H}_1(id))^r)$.
- $\mathsf{RKGen}(\mathsf{params}, sk_{id_1}, id_2)$. Select $X \xleftarrow{\$} \mathbb{G}_T$ and compute $\langle R_1, R_2 \rangle = \mathsf{Encrypt}(\mathsf{params}, id_2, X)$. Return $rk_{id_1 \to id_2} = \langle R_1, R_2, sk_{id_1}^{-1} \cdot \mathcal{H}_2(X) \rangle$.
- $\mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c_{id_1})$. To re-encrypt a level-$\ell$ ciphertext from id_1 to id_2, first parse c_{id_1} as $(C_1, \ldots, C_{2\ell})$ and $rk_{id_1 \to id_2}$ as (R_1, R_2, R_3). Next:
  1. If $\ell = 1$, output $c_{id_2} = \langle C_1, C_2 \cdot e(C_1, R_3)), R_1, R_2 \rangle$.
  2. If $\ell > 1$, treat the elements $\langle C_{2\ell-1}, C_{2\ell} \rangle$ as a first-level ciphertext δ. Compute $\langle C'_1, C'_2, C'_3, C'_4 \rangle = \mathsf{Reencrypt}(rk_{id_1 \to id_2}, \delta)$. Output the ciphertext $c_{id_2} = \langle C_1, \cdots, C_{2\ell-2}, C'_1, C'_2, C'_3, C'_4 \rangle$.
- $\mathsf{Decrypt}(\mathsf{params}, sk_{id}, c_{id})$. Parse the level-$\ell$ ciphertext c_{id} as $(C_1, \ldots, C_{2\ell})$. Next:
  1. If $\ell = 1$ output $m = C_2 / e(C_1, sk_{id})$.
  2. If $\ell > 1$, treat the pair $\langle C_{2\ell-1}, C_{2\ell} \rangle$ as a first-level ciphertext c'_{id}, and compute $X_\ell = \mathsf{Decrypt}(sk_{id}, c'_{id})$. For $i = (\ell - 1)$ descending to 1, compute $X_i = C_{2i} / e(C_{2i-1}, \mathcal{H}_2(X_{i+1}))$. Finally, output X_1 as the plaintext.

Each level-ℓ ciphertext in the above scheme contain 2ℓ elements. In principle, the scheme permits an arbitrary number of re-encryptions on a ciphertext, with a two-element ciphertext expansion on each re-encryption.

Correctness. We first show correctness for first-level ciphertexts (*i.e.,* those produced by $\mathsf{Encrypt}$). Let $c_{id_1} = (g^r, m \cdot e(g^s, \mathcal{H}_1(id_1))^r)$ be the first-level encryption of m under id_1, and $sk_1 = \mathcal{H}_1(id_1)^s$ be the corresponding decryption key. The decryption process produces the following result:

$$(m \cdot e(g^s, \mathcal{H}_1(id_1))^r) / e(g^r, \mathcal{H}_1(id_1)^s) = m$$

The correctness under re-encryption is shown as follows. Given a first-level ciphertext $c_{id_1} = (g^r, C_2)$ and a correctly-formed re-encryption key $rk_{id_1 \to id_2} = (\langle R_1, R_2 \rangle = \mathsf{Encrypt}(\mathsf{params}, id_2, X), R_3)$, we obtain the "second-level" ciphertext $c_{id_2} = (g^r, C'_2 = C_2 \cdot e(g^r, R_3), R_1, R_2)$ where C'_2 is:

$$\begin{aligned} C'_2 &= C_2 \cdot e(g^r, R_3) \\ &= m \cdot e(g^s, \mathcal{H}_1(id_1))^r) \cdot e(g^r, \mathcal{H}_1(id_1)^{-s} \cdot \mathcal{H}_2(X)) \\ &= m \cdot e(g, \mathcal{H}_2(X))^r \end{aligned}$$

Given $sk_{id_2} = \mathcal{H}_1(id_2)^s$ we decrypt $c_{id_2} = (g^r, C'_2, R_1, R_2)$ as follows. Begin by decrypting the first-level ciphertext $\hat{c}_{id_2} = \langle R_1, R_2 \rangle$ under sk_{id_2}: $X = \mathsf{Decrypt}(\mathsf{params}, sk_{id_2}, \hat{c}_{id_2})$. Then compute $C'_2 / e(g^r, \mathcal{H}_2(X))$ to

obtain m. Having shown correctness for a single re-encryption, the correctness for multiple re-encryptions follows. Given level-ℓ ciphertext c_{id_i} and sk_{id_i}, strip the the final two elements and treat them as a first-level ciphertext under id_i, decrypting to reveal X_ℓ. Use the value X_ℓ as a decryption secret for the previous two elements, and repeat until the final two elements remain. The final value in this chain contains the original message m.

Security. We next show that IBP1 scheme defined above meets the IND-Pr-ID-CPA definition if the Decisional Bilinear Diffie-Hellman assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$. Our proof is in the random oracle model, and is an extension of the original proof of Boneh/Franklin [7].

Theorem 4.1 *If there exists a p.p.t. adversary $\mathcal{A}$ that wins the* IND-Pr-ID-CPA *game on IBP*1 *with non-negligible advantage, then there exists an adversary $\mathcal{B}$ that solves the DBDH problem over $\mathbb{G}_1, \mathbb{G}_T$ with non-negligible advantage.*

A proof of Theorem 4.1 is presented in Appendix A.

4.2 An Optimization

Ciphertexts in scheme section 4.1 expand upon re-encryption. This is caused by the inclusion (within the re-encrypted ciphertext) of a portion of the re-encryption key. There are scenarios where Bob knows that the original ciphertext was intended for Alice (this information can even be appended to the ciphertext) and there is no need for multiple re-encryptions. In such cases we could simplify our scheme by noticing that, in the Boneh-Franklin IBE symmetric setting, Alice and Bob inherently share a secret key $K_{AB} = e(\mathcal{H}_1(Alice), \mathcal{H}_1(Bob))^s$. Alice can use this value to compute the re-encryption key as follows:

$$rk_{Alice \to Bob} = \mathcal{H}_1(Alice)^{-s} \cdot \mathcal{H}_3(\{K_{AB}\} || Alice \to Bob).$$

Where $\{K_{AB}\}$ denotes binary representation, and $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{G}_1$ is an independent full domain hash function. The string "$Alice \to Bob$" is added to ensure that a re-encryption key from Bob to Alice is computed under a distinct secret (bidirectional re-encryption). Note that the resulting scheme permits only a single re-encryption for each ciphertext. A primary advantage of this construction is the absence of ciphertext expansion during re-encryption.

4.3 A Chosen Ciphertext Secure Scheme (IBP2)

The scheme presented above is secure under chosen plaintext attack. While this is the level of security provided by many IBE and proxy re-encryption schemes (*e.g.,* [1, 18] and the practical proxy encryption constructions of Dodis/Ivan [12]), it is important to consider stronger definitions such as security under adaptive chosen ciphertext attack.

Background. A common approach to building CCA-secure Identity-Based *Encryption* schemes is to begin with a CPA-secure construction, and then apply the generic Fujisaki-Okamoto conversion [13] (see *e.g.,* [7, 20]). It is tempting to believe that this approach is by itself sufficient to construct CCA-secure IB-PRE schemes. Unfortunately, this does not appear to be the case. Notice that a re-encryption proxy grants adversaries an alternative means by which adversaries may decrypt ciphertexts: a malicious delegatee B may decrypt A's ciphertexts by first using the proxy to re-encrypt from $id_A \to id_B$, and then decrypting the result under his own secret key. When a malicious delegatee uses the proxy to "alternatively decrypt" in this

manner, he need not follow the specified F-O decryption algorithm, and can ignore the critical ciphertext validity checks. Unfortunately, the validity checks of the F-O approach cannot be moved into the re-encryption process, as they fundamentally require access to the decryption secret.

Intuition. In order to surmount the issues raised above, we propose an approach that provides the proxy with the means to verify ciphertext validity and reject improperly-formed ciphertexts. As a result of this check, a malicious delegatee no longer gains any advantage by using the re-encryption proxy as an oracle. The building block of our construction is a Hierarchical Identity-Based Proxy Re-encryption scheme, which we implement using a modified form of the Gentry-Silverberg HIBE [14] (this scheme is in turn based on the Boneh/Franklin scheme). To achieve IND-PrID-CCA-secure IB-PRE, we make use of the Canetti, Halevi and Katz (CHK) [10] technique, which allows us to transform a CPA-secure HIBE into a CCA-secure scheme with a type of *publicly-verifiable* ciphertext validity check. In order to present a more efficient construction, we re-use randomness and implement the CHK transform using a Boneh/Lynn/Shacham short signature [9].

The Construction. We now present a *single-use*, non-interactive CCA-secure IB-PRE construction (IBP2).

- Setup. Let n be polynomial in the security parameter k. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}_1, \mathbb{G}_T$ have order q and $\mathbb{G}_1 = \langle g \rangle$. To generate the scheme parameters, select $s \xleftarrow{\$} \mathbb{Z}_q^*$ and output $\mathsf{params} = (\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, \mathcal{H}_6, g, g^s)$, $\mathsf{msk} = s$, with independent hash functions $\mathcal{H}_{1-6}$ defined as below:

$$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1, \mathcal{H}_2 : \{0,1\}^* \rightarrow \mathbb{G}_1$$
$$\mathcal{H}_3 : \{0,1\}^* \rightarrow \mathbb{G}_1, \mathcal{H}_4 : \mathbb{G}_T \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$$
$$\mathcal{H}_5 : \mathbb{G}_T \rightarrow \{0,1\}^n$$

- KeyGen$(\mathsf{params}, \mathsf{msk}, id)$. To extract a decryption key for identity $id \in \{0,1\}^*$, return $sk_{id} = \mathcal{H}_1(id)^s$.

- Encrypt$(\mathsf{params}, id, m \in \{0,1\}^n)$. To encrypt m under identity $id \in \{0,1\}^*$, first:

  1. Select $\sigma \xleftarrow{\$} \mathbb{G}_T$, and set $r = \mathcal{H}_4(\sigma, m)$.
  2. Compute $c' = (g^r, \sigma \cdot e(g^s, \mathcal{H}_1(id)^r), m \oplus \mathcal{H}_5(\sigma))$.
  3. Compute $S = \mathcal{H}_3(id||c')^r$.
  4. Output the ciphertext $c = \langle S, c' \rangle$.

- RKGen$(\mathsf{params}, sk_{id_1}, id_1, id_2)$. To compute a re-encryption key from $id_1 \rightarrow id_2$:

  1. Select $N \xleftarrow{\$} \{0,1\}^n$, and compute $K = e(sk_{id_1}, \mathcal{H}_1(id_2))$.
  2. Output $rk_{id_1 \rightarrow id_2} = \langle N, \mathcal{H}_2(K||id_1||id_2||N) \cdot sk_{id_1} \rangle$.

- Reencrypt$(\mathsf{params}, rk_{id_1 \rightarrow id_2}, c_{id_1})$. To re-encrypt a first-level ciphertext, first parse c_{id_1} as (S, A, B, C), and parse $rk_{id_1 \rightarrow id_2}$ as $\langle N, R \rangle$. Next:

  1. Let $h = \mathcal{H}_3(id_1||\langle A, B, C \rangle)$.
  2. Check if $e(g, S) = e(h, A)$. If not, return $\perp$.
  3. Otherwise, select $t \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $B' = B / \frac{e(A, R \cdot h^t)}{e(g^t, S)}$.
  4. Output the re-encrypted ciphertext $c_{id_2} = (A, B', C, id_1, N)$.

- Decrypt(params, sk_{id}, c_{id}). To decrypt a first-level (non re-encrypted) ciphertext, first parse c_{id} as (S,A,B,C). Next:

  1. Let $h = \mathcal{H}_3(id, \langle A,B,C\rangle)$.
  2. Select $t \xleftarrow{\$} \mathbb{Z}_q^*$, and compute $\sigma' = B/\frac{e(A, sk_{id}\cdot h^t)}{e(g^t,S)}$.
  3. Compute $m' = C \oplus \mathcal{H}_5(\sigma')$, and $r' = \mathcal{H}_4(\sigma', m')$.
  4. Verify that $S = h^{r'}$ and $A = g^{r'}$. If either check fails, return $\perp$, otherwise output m'.

  To decrypt a second-level (re-encrypted) ciphertext, first parse c_{id} as (A,B,C,id_{src},N). Next:

  1. Compute $K = e(\mathcal{H}_1(id_{src}), sk_{id})$.
  2. Compute $\sigma' = B \cdot e(A, \mathcal{H}_2(K||id_{src}||id||N))$.
  3. Compute $m' = C \oplus \mathcal{H}_5(\sigma')$, and $r' = \mathcal{H}_4(\sigma', m')$.
  4. Verify that $A = g^{r'}$. If this check fails, return $\perp$, otherwise output m'.

Correctness. We first show correctness for first-level ciphertexts (*i.e.,* those produced by Encrypt). Let $c_{id_1} = \langle S,A,B,C\rangle = \langle h^r, g^r, \sigma \cdot e(g^s, \mathcal{H}_1(id)^r), m \oplus \mathcal{H}_5(\sigma)\rangle$ be the first-level encryption of m under id_1, with $h = \mathcal{H}_3(id_1||\langle A,B,C\rangle)$. Let $sk_1 = \mathcal{H}_1(id_1)^s$ be the corresponding decryption key. For a random $t \in \mathbb{Z}_q^*$, the decryption process proceeds as follows:

$$\begin{aligned}(\sigma \cdot e(g^s, \mathcal{H}_1(id_1))^r)/\frac{e(g^r, \mathcal{H}_1(id_1)^s \cdot h^t)}{e(g^t, h^r)} &= \sigma \\ \mathcal{H}_5(\sigma) \oplus (m \oplus \mathcal{H}_5(\sigma)) &= m \\ g^{\mathcal{H}_4(\sigma,m)} &\stackrel{?}{=} g^r \\ h^{\mathcal{H}_4(\sigma,m)} &\stackrel{?}{=} h^r\end{aligned}$$

The correctness under re-encryption is shown as follows. Given a first-level ciphertext $c_{id_1} = \langle S,A,B,C\rangle = \langle h^r, g^r, \sigma \cdot e(g^s, \mathcal{H}_1(id)^r), m \oplus \mathcal{H}_5(\sigma)\rangle$ (where $h = \mathcal{H}_3(id||\langle A,B,C\rangle)$) and a correctly-formed re-encryption key $rk_{id_1 \to id_2} = \langle N, R\rangle$, we compute the ciphertext validity check as:

$$e(g, h^r) \stackrel{?}{=} e(h, g^r)$$

Recall that $R = sk_{id_1} \cdot W$ where $W = \mathcal{H}_2(e(\mathcal{H}_1(id_1)^s, \mathcal{H}_1(id_2))||id_1||id_2||N)$. To generate the "second-level" ciphertext $c_{id_2} = (A, B', C, id_1, N)$, for some $t \in_R \mathbb{Z}_q^*$ we obtain:

$$B' = (\sigma \cdot e(g^s, \mathcal{H}_1(id_1))^r))/\frac{e(g^r, R \cdot h^t)}{e(g^t, h^r)} = \sigma / e(g^r, W)$$

Given $sk_{id_2} = \mathcal{H}_1(id_2)^s$ we decrypt $c_{id_2} = (A, B', C, id_1, N) = (g^r, \sigma/e(g^r, W), m \oplus \mathcal{H}_5(\sigma), N)$ as follows:

$$\begin{aligned}\mathcal{H}_2(e(\mathcal{H}_1(id_1), \mathcal{H}_1(id_2)^s)||id_1||id_2||N) &= W \\ (\sigma/e(g^r, W)) \cdot e(g^r, W) &= \sigma \\ \mathcal{H}_5(\sigma) \oplus (m \oplus \mathcal{H}_5(\sigma)) &= m \\ g^{\mathcal{H}_4(\sigma,m)} &\stackrel{?}{=} g^r\end{aligned}$$

Security. We next show that IBP2 meets the IND-PrID-CCA definition if the Decisional Bilinear Diffie-Hellman assumption holds in $(\mathbb{G}_1, \mathbb{G}_T)$. Our proof is in the random oracle model.

Theorem 4.2 *If there exists a p.p.t. adversary $\mathcal{A}$ that wins the* IND-PrID-CCA *game against IBP*2 *with non-negligible advantage, then there exists an adversary $\mathcal{B}$ that solves the DBDH problem over $\mathbb{G}_1, \mathbb{G}_T$ with non-negligible advantage.*

A proof of Theorem 4.2 is presented in Appendix B.

5 Applications of Identity-Based Proxy Re-encryption

Proxy Re-encryption has a number of practical applications, which have been detailed in previous works. All of these applications translate directly to the Identity-Based setting but with some additional features.

Secure Email with IBE. The most natural application of proxy re-encryption is to allow Bob to read Alice's encrypted emails while she is on vacation. Messages are encrypted under the email address *"alice@company.com"* and are translated by the proxy into encryptions under *"bob@company.com"*. The proxy does not learn the content of the messages being translated.

Attribute-based Delegations. As noted by Boneh and Franklin [7], identities can be created to include attributes or to express conditions. For instance, a message encrypted under *"alice* $\|$ *lawyer* $\|$ *from 01/01/2008"* can be read by Alice only if she is a lawyer and not before the beginning of year 2008. This idea applies directly to our IBE-PRE scheme and it allows Alice to specify under which conditions the proxy is allowed to translate her ciphertexts into Bob's. For instance, consider the case of *temporary* delegations [1] where the time is divided in time intervals t_1, t_2, ..., t_k and Alice can specify that the proxy can translate her ciphertexts for Bob only during t_i. With our scheme, Alice could just create the proxy key:

$$rk_{Alice\|t_i \rightarrow Bob},$$

so that any encryption under $Alice\|t_i$ can be converted into an encryption for Bob but not during other time periods. This eliminates the need for designing a separate and specialized scheme as it was done in [1].

Even more interestingly, Alice could specify the conditions under which Bob can read her messages. For instance, a re-encryption key of this form:

$$rk_{Alice \rightarrow Bob\|after\ Nov\ 2007\|security-clearance},$$

would specify that encryptions under Alice's identity can be converted into encryptions for Bob but that Bob can read the messages only in the future, after Nov 2007, and under the condition that he is able to obtain a security clearance.

Bridging IBE and PKE. *Hybrid* proxy re-encryption is a concept put forward by Boneh, Goh and Matsuo [8] to create a bridge between IBE and public-key based encryption (PKE). Our scheme can also be used to translate from IBE to PKE. Indeed, consider the ciphertext after the re-encryption, which has the form:

$$c'_1 = g^r,\ \ c'_2 = m \cdot e(g^r, \mathcal{H}_2(K)),\ \ c'_3 = IBE_{Bob}(K).$$

Notice that c'_3 is a standard (semantically-secure) id-based encryption of a key K. This encryption can be substituted with a public-key based one (or even a semantically-secure symmetric one). In this way, an encryption under Alice's identity is converted into an encryption under Bob's public-key. Our approach

provides some advantages over the one in [8]. Indeed, no TTP is involved in creating re-encryption keys and parameters in our scheme are compatible with those of the standard Boneh-Franklin IBE.

Travel Key. Boneh and Franklin [7] suggested to use an IBE system to store temporary keys into the laptop during travel so that, if the laptop is lost or stolen, only those keys get exposed. The idea is to let Bob act as a PKG that generates his own master secret and public keys. Alice could use Bob's master public-key to encrypt messages for Bob under identities *day1*, *day2*, ... etc., for all days in which Bob is traveling. Bob can store into his laptop just the keys corresponding to those days while leaving his master secret key safely stored elsewhere.

This solution, however, requires Bob to inform of his travels any of his potential correspondents and have them act according to the encryption scheme (that is, they have to encrypt under *day1*, *day2*, etc.). An alternative solution is to set up a proxy (Bob's mail server, for instance) with a re-encryption key of the form:

$$rk_{Bob \rightarrow Bob's-Travel-Key}.$$

Every encryption intended for Bob will be encrypted under Bob's travel key, which is the only secret key stored into his laptop. Notice that the proxy does not have to be trusted and can be set-up by a system administrator who won't be able to read Bob's messages.

Access Control in Networked File Storage. In [1], the authors describe an application of proxy re-encryption to the distribution of key material within a cryptographic filesystem. Each file stored on an untrusted file server is encrypted using a symmetric key; these keys are encrypted under a public *master key* which is stored alongside the encrypted material. When a user wishes to decrypt a file, the semi-trusted *keyserver* re-encrypts these encapsulated symmetric keys from the master key to the keys of individual users who can then decrypt. The key server provides access control for the encrypted material, but does not itself possess the ability to decrypt files.

This application translates naturally to the Identity Based setting with the additional benefit of allowing the holder of the master key to specify access control policies directly within the identity strings of the users. A re-encryption key can even be generated before an individual has joined the system.

6 Conclusions and Future Work

In this work we introduced new constructions enabling non-interactive, unidirectional proxy re-encryption in the IBE setting. Our schemes are very efficient and can be deployed within standard IBE frameworks. New compelling applications can be realized thanks to our schemes, most notably attribute-based delegation and access control.

An interesting open problem is to find efficient constructions for *multi-use* CCA-secure IBE-PRE schemes. Another important open problem is to find efficient IBE-PRE secure in the standard model (rather than in the RO model).

References

[1] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *the 12th Annual Network and Distributed System Security Symposium*, pages 29–43, 2005. Full version available at http://eprint.iacr.org/2005/028.

[2] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP '03)*, page 180, Washington, DC, USA, 2003. IEEE Computer Society.

[3] Matt Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of Eurocrypt '98*, volume 1403, pages 127–144, 1998.

[4] Dan Boneh and Xavier Boyen. Efficient selective-id secure Identity-Based Encryption without random oracles. In *Proceedings of Eurocrypt '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[5] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, 2004.

[6] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[7] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *Advances in Cryptology (CRYPTO 2001)*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[8] Dan Boneh, Eu-Jin Goh, and Toshihiko Matsuo. Proposal for P1363.3 Proxy Re-encryption. `http://grouper.ieee.org/groups/1363/IBC/submissions/NTTDataProposal-for-P1363.3-2006-09-01.pdf`.

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil Pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, London, UK, 2001. Springer-Verlag.

[10] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from Identity Based Encryption. In *Proceedings of Eurocrypt '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer-Verlag, 2004.

[11] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.

[12] Yevgeniy Dodis and Anca Ivan. Proxy cryptography revisited. In *Proceedings of the Tenth Network and Distributed System Security Symposium*, February 2003.

[13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of Crypto '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[14] Craig Gentry and Alice Silverberg. Hierarchical ID-Based cryptography. In *Proceedings of Asiacrypt '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.

[15] Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *Proceedings of Public Key Cryptography*, pages 112–121, 1999.

[16] Masahiro Mambo and Eiji Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. *IEICE Trans. Fund. Electronics Communications and Computer Science*, E80-A/1:54–63, 1997.

[17] Voltage Security, Inc. http://www.voltage.com.

[18] Brent Waters. Efficient Identity-Based Encryption without random oracles. In *Proceedings of Eurocrypt '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[19] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *Proceedings of Network and Distributed System Security Symposium 2004 (NDSS'04)*, San Diego, CA, February 2004.

[20] Peng Yang, Takashi Kitagawa, Goichiro Hanaoka, Rui Zhang, Kanta Matsuura, and Hideki Imai. Applying Fujisaki-Okamoto to Identity-Based Encryption. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 16th International Symposium (AAECC-16)*, volume 3857 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2006.

[21] Lidong Zhou, Michael A. Marsh, Fred B. Schneider, and Anna Redz. Distributed blinding for ElGamal re-encryption. Technical Report 2004–1924, Cornell Computer Science Department, 2004.

A Security Proof of IBP1

Proof. Let $\mathcal{A}$ be a p.p.t. algorithm that has non-negligible advantage ε in attacking the scheme of §4.1. We use $\mathcal{A}$ in order to construct a second algorithm $\mathcal{B}$ which has non-negligible advantage at solving the DBDH problem in $\mathbb{G}_1, \mathbb{G}_T$. Algorithm $\mathcal{B}$ accepts as input a properly-distributed tuple $\langle \mathbb{G}_1 = \langle g \rangle, g^a, g^b, g^c, T \rangle \in \mathbb{G}_1^4 \times \mathbb{G}_T$ and outputs 1 if $T = e(g,g)^{abc}$. We now describe the algorithm $\mathcal{B}$, which interacts with algorithm $\mathcal{A}$ via the IND-Pr-ID-CPA interface.

Oracle Queries. $\mathcal{B}$ simulates the random oracle $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ as follows: On receipt of a query for id (on which it has not previously been queried), select $z \xleftarrow{\$} \mathbb{Z}_q^*$ and randomly flip a weighted coin to set $\alpha \leftarrow 1$ with probability γ (defined below), and $\alpha \leftarrow 0$ otherwise. If $\alpha = 0$ then compute $h \leftarrow (g^c)^z$, else compute $h \leftarrow g^z$. Record the tuple (id, h, z, α). Finally, return h as the result of the query (if id has previously been queried, simply locate the existing tuple and return the previously-computed h). Note that the distribution of the values h returned by the simulated oracle is random, regardless of the choice of α. $\mathcal{B}$ simulates (initial) queries to the random oracle $\mathcal{H}_2 : \mathbb{G}_T \rightarrow \mathbb{G}_1$ by simply returning elements $\in_R \mathbb{G}_1$.

Our simulation proceeds as follows:

1. SELECT. Choose $i \xleftarrow{\$} \{0,1\}$.
2. SETUP. $\mathcal{B}$ generates the scheme's master parameters $\mathsf{params} = (\mathbb{G}_1, \mathcal{H}_1, \mathcal{H}_2, g, g^a)$ and gives this tuple to $\mathcal{A}$.
3. FIND. When $\mathcal{A}$ submits $(\mathsf{extract}, id)$, $\mathcal{B}$ evaluates $\mathcal{H}(id)$ as described above, to obtain (id, h, z, α). $\mathcal{B}$ outputs $sk_{id} = (g^a)^z$ to $\mathcal{A}$.

   When $\mathcal{A}$ submits $(\mathsf{rkextract}, id_1, id_2)$, $\mathcal{B}$ selects $r \xleftarrow{\$} \mathbb{Z}_q^*$, $x \xleftarrow{\$} \mathbb{G}_1$ and $X \xleftarrow{\$} \mathbb{G}_T$, then evaluates $\mathcal{H}_1(id_1)$ and $\mathcal{H}_1(id_2)$ to obtain the values $(\alpha_1, z_1), (\alpha_2, z_2)$ (for id_1, id_2 respectively). Now:

(a) If $\alpha_1 = 0$ then $\mathcal{B}$ returns $rk_{id_1 \to id_2} = \left((g^b)^r, T^{rz_2} \cdot X, x\right)$ to $\mathcal{A}$ (note that this key is incorrectly formed, see section below).
(b) If $\alpha_1 = 1$ then $\mathcal{B}$ returns the correctly-formed $rk_{id_1 \to id_2} = (\langle R_1, R_2 \rangle = \mathsf{Encrypt}(X), (g^a)^{-z_1} \cdot \mathcal{H}_2(X))$.

4. CHOICE AND CHALLENGE. At the conclusion of the FIND phase, $\mathcal{A}$ outputs (id^*, m_0, m_1) with the condition that $\mathcal{A}$'s choice of id^* is not trivial.[2] $\mathcal{B}$ evaluates $\mathcal{H}_1(id^*)$ and recovers (id^*, h, z, α) from the $\mathcal{H}_1$ table, and returns $c^* = \langle g^b, T^z \cdot m_i \rangle$ to $\mathcal{A}$.

5. GUESS. $\mathcal{A}$ makes queries $(\mathsf{extract}, \ldots)$ and $(\mathsf{rkextract}, \ldots)$ as in the FIND stage, except that $\mathcal{A}$ is restricted from making any query that would result in a trivial situation (a valid decryption path from id^* to an identity for which the adversary possesses a secret key). At the conclusion of this phase, $\mathcal{A}$ outputs its guess $i' \in \{0, 1\}$.

   Conditions for Abort. Let α_i represent the value α generated by $\mathcal{H}_1(id_i)$. Prior to outputting a value, $\mathcal{B}$ verifies several conditions:

   (a) The value α corresponding to id^* is 0.
   (b) For each of $\mathcal{A}$'s queries $(\mathsf{extract}, id_i)$, $\alpha_i = 1$.
   (c) For each of $\mathcal{A}$'s queries $(\mathsf{rkextract}, id_i, id_j)$, where $id_i \to id_j$ lies along a path leading from id^*, $\alpha_j = 0$.
   (d) For each of $\mathcal{A}$'s queries $(\mathsf{rkextract}, id_i, id_j)$, where $id_i \to id_j$ does not lie along a path leading from id^*, $\alpha_i = 1$.

   If any of the above conditions are false, $\mathcal{B}$ aborts the simulation. Otherwise, if $i' = i$, $\mathcal{B}$ outputs 1, or 0 otherwise.

Claim. If $\mathcal{B}$ does not abort during the game, then $\mathcal{A}$'s view is identical to the real attack, with the exception of re-encryption keys of the form $rk_{id^* \to \cdot}$. We address this case below by showing that $\mathcal{A}$ cannot distinguish the simulation from the real attack. Hence, when the input to $\mathcal{B}$ is a DBDH tuple, then the challenge ciphertext c^* is a correct encryption of m_i under id^* and hence (subject to the definition of $\mathcal{A}$ and the argument above) $\left|\Pr[i = i'] - \frac{1}{2}\right| \geq \varepsilon$. When the input to $\mathcal{B}$ is random, c^* represents the encryption of a random element. Since $\mathcal{A}$ is unable to distinguish between the simulation and a real attack, it must hold that $\Pr[i = i'] > \frac{1}{2} + \varepsilon$ for a non-negligible ε. Hence, $\mathcal{B}$ succeeds with non-negligible advantage.

Invalid Re-encryption keys. In the simulation above, every re-encryption key that lies along a path from id^* is incorrectly formed. At the same time, it is easy to see that all other re-encryption keys *are* correctly formed. Unfortunately, this condition is unavoidable, as the simulator does not possess the knowledge required in order generate a valid re-encryption key from the challenge identity id^*. To complete our proof, therefore, we make a separate argument that no adversary $\mathcal{A}$ can distinguish our simulation from a "real-world" interaction in which all values have the correct form. The heuristic argument for security is simple: each correctly-formed re-encryption key $rk_{id_1 \to id_2}$ consists of a semantically secure encryption (R_1, R_2) of some element $X \in_R \mathbb{G}_T$, along with the value $R_3 = (\mathcal{H}_1(id_2)^{-s} \cdot \mathcal{H}_2(X)) \in \mathbb{G}_1$. An incorrectly-formed re-encryption key replaces R_3 with some value $x \in_R \mathbb{G}_1$; this x can naturally be expressed as $(\mathcal{H}_1(id_2)^{-s} \cdot y)$ for some unknown $y \in \mathbb{G}_1$. Intuitively, an adversary who can distinguish malformed re-encryption keys

[2] We reject a choice of id^* when $\mathcal{A}$ has previously extracted a series of re-encryption keys, and a decryption key $sk_{id'}$ such that $\mathcal{A}$ can consecutively re-encrypt ciphertexts from id^* to id'.

in our simulation must therefore be able to determine that (R_1, R_2) do *not* encrypt some value $Y \in \mathbb{G}_T$ *s.t.* $\mathcal{H}_2(Y) = y$. We formalize the statement via the following Lemma.

Lemma A.1 (Indistinguishability of simulations) If there exists a *p.p.t.* algorithm $\mathcal{A}'$ with non-negligible advantage ε' at distinguishing the simulation above from a "correct" simulation (in which all values are correctly-formed), then we can construct an algorithm $\mathcal{B}'$ that solves the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_T)$ with non-negligible advantage.

Probability of abort. A variety of conditions in the above simulation can lead the simulator to abort. Boneh and Franklin [7] provide a technique for computing the value γ used in simulating the random oracle $\mathcal{H}_1$, and for placing bounds on the abort probability. We refer the reader to this discussion, and provide a detailed argument in the full version. □

B Security Proof of IBP2

Below we sketch a proof of Theorem 4.2. The full version of this paper includes a more detailed proof with specifics on the reduction.

Proof. Let $\mathcal{A}$ be a p.p.t. algorithm that has non-negligible advantage ε in attacking the scheme of §4.3. We use $\mathcal{A}$ in order to construct a second algorithm $\mathcal{B}$ which has non-negligible advantage at solving the DBDH problem in $\mathbb{G}_1, \mathbb{G}_T$. Algorithm $\mathcal{B}$ accepts as input a properly-distributed tuple $\langle \mathbb{G}_1 = \langle g \rangle, g^a, g^b, g^c, T \rangle \in \mathbb{G}_1^4 \times \mathbb{G}_T$ and outputs 1 if $T = e(g,g)^{abc}$. We now describe the algorithm $\mathcal{B}$, which interacts with algorithm $\mathcal{A}$ via the IND-Pr-ID-CPA interface.

Oracle Queries. $\mathcal{B}$ simulates the random oracles $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5$ as follows.

- $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$. On receipt of a new query for id, select $y, z \xleftarrow{\$} \mathbb{Z}_q^*$ and randomly set $\alpha \in \{0,1\}$ such that $\Pr[\alpha = 0] = \gamma$ (defined below). If $\alpha = 0$ compute $h \leftarrow (g^c)^z$; if $\alpha = 1$ compute $h \leftarrow g^z$. Record the tuple (id, h, y, z, α) and return h. Notice that h is correctly distributed.
- $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{G}_1$. On a new query $X \in \{0,1\}^*$, return $x \xleftarrow{\$} \mathbb{G}_1$ and record (X, x).
- $\mathcal{H}_3 : \{0,1\}^* \to \mathbb{G}_1$. On receipt of a new query $s \in \{0,1\}^*$ of the form $(id || \langle A, B, C \rangle)$, select $z \xleftarrow{\$} \mathbb{Z}_q^*$ and randomly set $\alpha \in \{0,1\}$ such that $\Pr[\alpha = 0] = \gamma$ (defined below). Compute $\mathcal{H}_1(id)$ to obtain the tuple $(id, h_1, y_1, z_1, \alpha_1)$. If $\alpha = \alpha_1$ then set $h = g^z$; if $\alpha = 0$ and $\alpha_1 = 1$ then set $h = (g^c)^z$; if $\alpha = 1$ and $\alpha_1 = 0$ then set $h = g^z / (g^c)^{y_1^{-1} z_1}$. Record the tuple (s, h, z, α) and return h. Notice that h is correctly distributed.
- $\mathcal{H}_4 : \mathbb{G}_T \times \{0,1\}^n \to \mathbb{Z}_q^*$. On receipt of a new query (σ, m), return $r \xleftarrow{\$} \mathbb{Z}_q^*$ and record (σ, m, r, g^r).
- $\mathcal{H}_5 : \mathbb{G}_T \to \{0,1\}^n$. On receipt of a new query σ, return $p \xleftarrow{\$} \{0,1\}^n$ and record (σ, p).

Our simulation proceeds as follows:

1. SELECT. Choose $i \xleftarrow{\$} \{0,1\}$.
2. SETUP. $\mathcal{B}$ generates the scheme's master parameters $\mathsf{params} = (\mathbb{G}_1, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, g, g^a)$ and gives params to $\mathcal{A}$.

3. FIND/GUESS. During the FIND stage, there are no restrictions on which queries $\mathcal{A}$ may issue. The scheme permits only a single consecutive re-encryption, therefore (per definition 3.4), during the GUESS stage, $\mathcal{A}$ is restricted from issuing the following queries:

   - $(\mathsf{extract}, id^*)$ where id^* is the challenge identity.
   - $(\mathsf{decrypt}, id^*, c^*)$ where c^* is the challenge ciphertext.
   - Any pair of queries $\langle(\mathsf{rkextract}, id^*, id_i), (\mathsf{extract}, id_i)\rangle$
   - Any pair of queries $\langle(\mathsf{reencrypt}, id^*, id_i, c^*), (\mathsf{extract}, id_i)\rangle$.
   - Any pair of queries $\langle(\mathsf{rkextract}, id^*, id_i), (\mathsf{decrypt}, id_i, c_i)\rangle$ where $c_i = \mathsf{Reencrypt}(rk_{id^* \to id_i}, c^*)$.

   In the GUESS stage, let id^* be the target identity, and parse the challenge ciphertext c^* as (S^*, A^*, B^*, C^*). In both phases, $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows.

   - On $(\mathsf{extract}, id)$, where (in the GUESS) stage $id \neq id^*$, $\mathcal{B}$ evaluates $\mathcal{H}_1(id)$ to obtain (z, α). $\mathcal{B}$ outputs $sk_{id} = (g^a)^z$ to $\mathcal{A}$.
   - On $(\mathsf{rkextract}, id_1, id_2)$, $\mathcal{B}$ evaluates $\mathcal{H}_1(id_1)$ to obtain the recorded values (α_1, z_1), and evaluates $\mathcal{H}_1(id_2)$ to obtain (α_2, z_2). $\mathcal{B}$ selects $N \xleftarrow{\$} \{0,1\}^n$ and returns the value:

     $$rk_{id_1 \to id_2} = \langle (g^a)^{z_1} \cdot \mathcal{H}_2(e(g^{az_1}, \mathcal{H}_1(id_2)) || id_1 || id_2 || N), N \rangle.$$

     Note that when $\alpha_1 = 1$ this re-encryption key is correctly formed; when $\alpha_1 = 0$ this key is incorrectly formed. See section below.
   - On $(\mathsf{decrypt}, id, c)$ where (in the GUESS stage) $(id, c) \neq (id^*, c^*)$, check whether c is a *level-1* (non re-encrypted) or *level-2* (re-encrypted) ciphertext. In the GUESS stage, parse c^* as (S^*, A^*, B^*, C^*).

     For a level-1 ciphertext, $\mathcal{B}$ parses c as (S, A, B, C) and:

     (a) Looks up the value A in the $\mathcal{H}_4$ table, to obtain the tuple (σ, m, r). If A is not in the table, or if (in the GUESS stage) $A = A^*$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (b) Checks that $S = \mathcal{H}_3(id || \langle A, B, C \rangle)^r$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (c) Checks that $\sigma = B / e(g^a, \mathcal{H}_1(id)^r)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (d) Checks that $C = \mathcal{H}_5(\sigma) \oplus m$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
     (e) Otherwise, $\mathcal{B}$ returns m to $\mathcal{A}$.

     For a level-2 ciphertext, $\mathcal{B}$ parses c as (A, B, C, id_{src}, N) and:

     (a) Compute $\mathcal{H}_1(id_{src})$, $\mathcal{H}_1(id)$ to obtain (h_1, z_1, α_1), (h_2, z_2, α_2). If both $\alpha_1 = \alpha_2 = 0$ then abort the simulation. If $id = id^*$ then let $K = e(\mathcal{H}_1(id), (g^a)^{z_1})$. Otherwise, let $K = e(\mathcal{H}_1(id_{src}), (g^a)^{z_2})$.
     (b) Let $\sigma' = B / e(A, \mathcal{H}_2(K || id_{src} || id || N))$. Let $m' = C \oplus \mathcal{H}_5(\sigma')$, and let $r' = \mathcal{H}_4(\sigma', m')$. If $g^{r'} \stackrel{?}{=} A$ then return m' to $\mathcal{A}$.
     (c) If this approach fails, then look up A in the $\mathcal{H}_4$ table to obtain (r, σ, m). Check that $C = \mathcal{H}_5(\sigma) \oplus m$ and $B = e(g^a, \mathcal{H}_1(id_{src})^r) \cdot m / e(A, rk_{id_{src} \to id})$, where $rk_{id_{src} \to id}$ is the value computed by a $(\mathsf{extract}, \ldots)$ query using nonce N. If so, return m. Otherwise, return $\perp$.
   - On $(\mathsf{reencrypt}, id_1, id_2, c)$, $\mathcal{B}$ parses c as (S, A, B, C) and:

(a) Computes $H = \mathcal{H}_3(id_1||\langle A,B,C\rangle)$, and performs the ciphertext validity check $e(g,S) = e(H,A)$. If this check does not succeed, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.
(b) If (in the GUESS stage) $(id_1,A) = (id^*,A^*)$ and $c \neq c^*$ then $\mathcal{B}$ initiates a *forgery event* as described in the section below.
(c) $\mathcal{B}$ selects $N \xleftarrow{\$} \{0,1\}^n$, and:
   i. If (in the GUESS stage) $id_1 = id^*$ then $\mathcal{B}$ computes $\mathcal{H}_1(id_1)$ to obtain (h_1,y_1,z_1,α_1), and evaluates $\mathcal{H}_3(id_1||\langle A,B,C\rangle)$ to obtain (H,z_2,α_2). If $\alpha_2 = 0$ then $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ selects $t \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $\sigma' = B/\frac{e(A,H^{y_1 t})}{e(g^{y_1 t},S)}$. Finally, $\mathcal{B}$ outputs $c_{id_2} = (A, \sigma' \cdot e(A, \mathcal{H}_2(e(\mathcal{H}_1(id_1),(g^a)^{z_2})||id_1||id_2||N)), C, id_1, N)$.
   ii. If $id_1 \neq id^*$ then $\mathcal{B}$ extracts $rk_{id_1 \to id_2}$ by executing a query on $(\mathsf{rkgen}, id_1, id_2)$ and outputs $c_{id_2} = \mathsf{Reencrypt}(\mathsf{params}, rk_{id_1 \to id_2}, c)$.

(Note that in all cases where $\mathcal{B}$ does not abort, the values returned are correctly formed.)

At the end of the FIND phase, $\mathcal{A}$ outputs (id^*,m_0,m_1), with the condition that $\mathcal{A}$ has not previously issued $(\mathsf{extract}, id^*)$ or the pair $\langle(\mathsf{extract}, id_i),(\mathsf{rkextract}, id^*, id_i)\rangle$ for any $id_i \in \{0,1\}^*$.

At the end of the GUESS stage, $\mathcal{A}$ outputs its guess bit i'.

4. CHOICE AND CHALLENGE. At the end of the FIND stage, $\mathcal{A}$ submits (id^*,m_0,m_1). $\mathcal{B}$ forms the challenge ciphertext as follows:

   (a) Choose $\sigma \xleftarrow{\$} \mathbb{G}_T$, $p \xleftarrow{\$} \{0,1\}^n$, and insert (σ,p) into the $\mathcal{H}_5$ table. Insert $(\sigma,m_b,\cdot,g^b)$ into the $\mathcal{H}_4$ table. (If σ is already present in either table, choose a different σ.) Evaluate $\mathcal{H}_1(id^*)$ and obtain the recorded value z.
   (b) Compute $(A^*,B^*,C^*) = \langle g^b, \sigma \cdot T^z, p \oplus m_i\rangle$.
   (c) Select $z_1 \xleftarrow{\$} \mathbb{Z}_q^*$ and insert $(id||\langle A^*,B^*,C^*\rangle, z_1, g^{z_1}, \alpha_1 = 1)$ into the $\mathcal{H}_3$ table. (If $id||\langle A^*,B^*,C^*\rangle$ is already present in the table, choose a new σ and begin again.) Compute $S^* = (g^b)^{z_1}$.

   $\mathcal{B}$ outputs the challenge ciphertext $c^* = (S^*,A^*,B^*,C^*) = \left(g^{by}, g^b, \sigma \cdot T^z, p \oplus m_i\right)$ to $\mathcal{A}$, and begins the GUESS stage.

5. FORGERIES AND ABORT CONDITIONS. *Forgery events.* In the event that a forgery event occurs during the GUESS stage, $\mathcal{B}$ operates as follows. On values $id_1, id^*, c = (S,A,B,C)$, and $c^* = (S^*,A^*,B^*,C^*)$ (where $(id_1,A) = (id^*,A^*)$, and $c \neq c^*$), let $H = \mathcal{H}_3(id_1||\langle A,B,C\rangle)$ and extract the $\mathcal{H}_3$ table value y. Note that (except with negl. probability) $(id^*||\langle A^*,B^*,C^*\rangle) \neq (id_1||\langle A,B,C\rangle)$ when $c \neq c^*$.[3]

   By the definition of $\mathcal{H}_3$ it holds that $\Pr[H = g^{cz}] \geq \gamma$. In this case we substitute into the validity check to obtain $e(g,S) = e(g^{cz},g^b)$ which finds $S = g^{bcz}$. $\mathcal{B}$ computes $S' = S^{z^{-1}} = g^{bc}$ (computing CDH on g^b, g^c), and solves DBDH as follows:

$$i_{forge} \leftarrow \left(e(S',g^a) \stackrel{?}{=} T\right)$$

   $\mathcal{B}$ evaluates the conditions for abort (below) and if all conditions are satisfied, terminates the simulation and returns i_{forge} as the result.

[3] If $(id^*||\langle A^*,B^*,C^*\rangle) = (id_1||\langle A,B,C\rangle)$ then $H = \mathcal{H}_3(id^*||\langle A^*,B^*,C^*\rangle)$. However, this cannot be the case, since the validity check equation $e(g,S) = e(H,A)$ holds true iff $S = S^*$ (and thus $c = c^*$).

Conditions for Abort. Let α_i represent the value α generated by $\mathcal{H}_1(id_i)$. Prior to outputting a value, $\mathcal{B}$ verifies several conditions:

(a) The value α corresponding to id^* is 0.
(b) For each of $\mathcal{A}$'s queries $(\mathsf{extract}, id_i)$, $\alpha_i = 1$.
(c) For each of $\mathcal{A}$'s queries $(\mathsf{rkgen}, id_i, id_j)$, where $id_i \neq id^*$, $\alpha_i = 1$.

If any of the above conditions are false, $\mathcal{B}$ aborts the simulation and outputs a random bit. If a *forgery event* occurred during the GUESS phase, $\mathcal{B}$ outputs i_{forge}. Otherwise, if $i' = i$, $\mathcal{B}$ outputs 1, or 0 otherwise.

As in the previous proof, if the input to $\mathcal{B}$ is a DBDH tuple, then the challenge ciphertext c^* is a correct encryption of m_i under id^*— otherwise c^* is the encryption of a random element. Similarly, all elements given to $\mathcal{B}$— except for certain re-encryption keys and re-encrypted ciphertexts— have the correct distribution.

Probability of abort. A variety of conditions in the above simulation can lead the simulator to abort. Gentry and Silverberg in [14] provide a technique for computing the value γ used in simulating the random oracle $\mathcal{H}_1$, and for placing bounds on the abort probability. We refer the reader to this discussion, and provide a detailed argument in the full version. □

Invalid Re-encryption Keys/Re-encryptions. In the simulation above, every re-encryption key $rk_{id^* \to id_i}$ (for all id_i) is incorrectly formed. As in the proof of Appendix A, we make a separate argument that no adversary $\mathcal{A}$ can distinguish our simulation from a "real-world" interaction in which all values have the correct form. Since these keys are also used to compute the result of $(\mathsf{reencrypt}, \ldots)$ queries, we extend our argument to these results. Incorrectly-formed re-encryption keys are of the form $rk_{invalid} = g^{az_1} \cdot W$, while a correctly-formed re-encryption key $rk_{id^* \to id_i} \in \mathbb{G}_1$ should have the form $rk_{valid} = g^{caz_1} \cdot W$, and W is:

$$W = \mathcal{H}_2(e(\mathcal{H}_1(id^*), \mathcal{H}_1(id_1)^a) || id^* || id_1 || N)$$

Note that by the rules of the game, $\mathcal{A}$ cannot possess a re-encryption key $rk_{id^* \to id_i}$ and also hold sk_{id_i}. Similarly, $\mathcal{A}$ can never extract sk_{id^*}. Because the value W is randomly distributed, for $\mathcal{A}$ to distinguish the keys requires that it be capable of solving the computational problem of determining the value $K = e(\mathcal{H}_1(id^*), \mathcal{H}_1(id_1)^a)$. We formalize the statement via the following Lemma, leave the complete proof for the full version of this work:

Lemma B.1 (Indistinguishability of simulations) If there exists a *p.p.t.* algorithm $\mathcal{A}'$ with non-negligible advantage ε' at distinguishing the simulation above from a "correct" simulation (in which all values are correctly-formed), then we can construct an algorithm $\mathcal{B}'$ that solves the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_T)$ with non-negligible advantage.

Proof Sketch. Let $P_1 = \Pr[i = i']$ when $\mathcal{A}'$ is given correctly-distributed values as in the real attack, and $P_2 = \Pr[i = i']$ when $\mathcal{A}'$ is given one or more invalid re-encryption key(s) of the form described above. Let $\mathcal{A}'$ be an algorithm such that $|P_1 - P_2| > \nu(k)$: *i.e.,* $\mathcal{A}'$ has this *advantage* at distinguishing the simulations. We use distinguishing $\mathcal{A}'$ to construct $\mathcal{B}'$ that solves DBDH with probability $> \nu(k)$.

We do not specify the full details of $\mathcal{B}'$'s simulation, but instead re-use much of the simulation above. Given a tuple (g, g^a, g^b, g^c, T), let (g, g^a) be the public key given to $\mathcal{A}'$. We contrive the $\mathcal{H}_1$ oracle to (probabilistically) output g^{bz_i} (for some known z_i) on input $\mathcal{H}_1(id_i)$, and g^{cz_1} (for known z_1) on input $\mathcal{H}_1(id^*)$. When $\mathcal{A}'$ extracts a single re-encryption key $rk_{id^* \to id_i}$, $\mathcal{B}'$ computes the key as $g^{az_1} \cdot W$ where:

$$W = \mathcal{H}_2(T^{z_1 z_i} || id^* || id_1 || N)$$

Note that (z_1, z_i) are randomly distributed and outside of the view of $\mathcal{A}'$. Note also that when $T = e(g,g)^{abc}$ then $T^{z_1 z_i}$ is correctly distributed; otherwise $T^{z_1 z_i}$ is a random element.

When T is random, then $\mathcal{A}'$ has a negligible probability of issuing an oracle call on $\mathcal{H}_2(T^{z_1 z_i} || id^* || id_1 || N)$. Conversely, when $T = e(g,g)^{abc}$, it must be the case that $\mathcal{A}'$ issues a call on $\mathcal{H}_2(T^{z_1 z_i} || id^* || id_1 || N)$ with probability $> \nu(k)$. $\mathcal{B}'$ examines the trace to find such a call, and if it is present, outputs 1, and otherwise outputs 0. Using standard arguments, we may extend this claim to address a case where $\mathcal{A}'$ extracts multiple incorrectly-formed re-encryption keys.