On Efficient Access Control Mechanisms in Hierarchy using Unidirectional and Transitive Proxy Re-encryption Schemes

Gaurav Pareek and Purushothama B. R.
National Institute of Technology Goa, Ponda, India

Keywords: Cryptographic Access Control, Proxy Re-encryption, Access Control in Hierarchy, Unidirectional-transitive, Key Management.

Abstract: Proxy re-encryption is a cryptographic primitive used to transform a ciphertext under one public key such that it becomes a ciphertext under another public key using a re-encryption key. Depending on the properties featured by a proxy re-encryption scheme, it can be applied to a variety of applications. In this paper, we target one such application of proxy re-encryption – access control in hierarchy, to highlight an important research gap in its design. We study how a proxy re-encryption scheme that is both unidirectional and transitive can be useful for enforcing hierarchical access control with constant computation and storage overhead on its users irrespective of the depth of the hierarchy. Also, we present improvements on the existing re-encryption schemes to make it applicable to hierarchical key assignment and achieve performance closer to that in case of a unidirectional transitive proxy re-encryption scheme.

1 INTRODUCTION

Cloud computing is gaining importance as more and more enterprises are switching to cloud for providing storage and computing services to its users. To maintain confidentiality of the data, it is encrypted. A cloud customer willing to enforce cryptographic access control (Vimercati et al., 2010) uses proxy re-encryption (Blaze et al., 1998) to delegate decryption rights of a data item to any other party. This delegation of decryption rights (or simply delegation) requires re-encryption to be done using a special key called re-encryption key (or delegation key). Anyone can use a re-encryption key $rk_{A \to B}$ to transform message encrypted for A such that can be decrypted by B. The re-encryption procedure does not reveal anything about the underlying plaintext or secret keys A and B. First proposed by Blaze et al. (Blaze et al., 1998), revisited by Dodis et al.(Dodis and Fazio, 2003), the desirable properties of proxy re-encryption were first presented by Ateniese et al. (Ateniese et al., 2006). The properties include unidirectionality, transitivity, collusion safety, proxy invisibility, key optimality, temporary delegation and non-transferability. Various applications of proxy re-encryption require different combinations of these properties to be satisfied.
Consider an application scenario where cryptographic access control in a hierarchy of security classes is provided using key management (Atallah et al., 2009). The set of users is divided into a disjoint collection of classes depending on the security clearance of the users. The hierarchy of classes forms a POSET (partially ordered set) under the partial order $\preceq$. Here, $C_j \preceq C_i$ means that users in a class C_i have access to data items encrypted for all the classes C_j in addition to the data items directly encrypted for users in C_i.

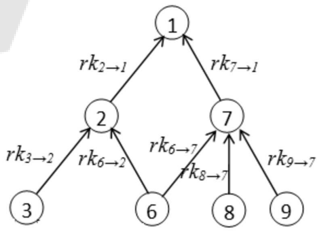

Figure 1: Hierarchical Key Assignment using Proxy Re-encryption.

Proxy re-encryption can be applied for managing access in this scenario by assigning re-encryption keys $rk_{i \to j}$ to each edge in the access hierarchy as shown in Figure 1. Suppose users in C_1 wish to access ciphertext CT_2 of class C_2. Users in C_1 can get CT_2 re-encrypted using $rk_{2\to1}$. But if users in C_1 want to access ciphertext CT_6, due to unavailability of re-encryption key $rk_{6\to1}$, CT_6 must be first transformed into CT_2 using $rk_{6\to2}$ and then into CT_1 using $rk_{2\to1}$.

Pareek, G. and R., P.
On Efficient Access Control Mechanisms in Hierarchy using Unidirectional and Transitive Proxy Re-encryption Schemes.
DOI: 10.5220/0006466405190524
In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT*, pages 519-524
ISBN: 978-989-758-259-2

In general, the number of re-encryptions required is at most equal to the depth of the hierarchy. Also, schemes that support repeated re-encryption and only in one direction i.e., *unidirectional and multi-hop re-encryption schemes* (Shao et al., 2011; Chandran et al., 2014) are computationally very costly to realize and require extremely costly operations for each re-encryption. But if the re-encryption scheme is transitive and unidirectional, then given $rk_{6\to2}$ and $rk_{2\to1}$, it is possible to derive $rk_{6\to1}$. So, it avoids repeated re-encryptions and requires only one re-encryption followed by one decryption.
However, we note that there is no proxy re-encryption scheme that satisfies both unidirectionality and transitivity. In this paper, we open an interesting problem of designing a unidirectional-transitive proxy re-encryption scheme. Our claims are based on the existing security requirements and state-of-the art design of proxy re-encryption schemes. For this, we capture high level design goals in the form of functions. We also propose the ways by which existing re-encryption schemes can be modified to obtain performance close to the case with unidirectional-transitive proxy re-encryption.
The remainder of the paper is organized as follows. Section 2 presents system definitions for hierarchical access control. Section 3 defines a unidirectional and transitive proxy re-encryption scheme along with its security requirements. Section 4 discusses our mathematical claims regarding the possibility of design of a unidirectional-transitive proxy re-encryption.

2 ACCESS CONTROL IN HIERARCHY

We present a formal model for access control in hierarchy. This includes definitions of procedures involved and security definition.

2.1 System Model

Let $C_1, C_2, \ldots, C_n$ be a disjoint partition (security classes or simply classes) of the set of users $\mathcal{U}$ in the system. Let $S = \{C_1, C_2, \ldots, C_n\}$ be partially ordered by the binary relation $\preceq$. If $C_j \preceq C_i$, then users in class C_j have the right to access information meant for users in class C_i. However, the reverse is not allowed. We use following notations in context of access control in hierarchy:

- *parent(C_i)/ances(C_i)* denotes the set of classes $C_j \in S$ such that $C_i \preceq C_j$ and C_j is one/more than one level higher in the hierarchy than C_i.
- *child(C_i)/desc(C_i)* denotes the set of classes $C_j \in S$ such that $C_j \preceq C_i$ and C_j is more than one/more than one level lower in the hierarchy than C_i.

All the users in a class C_i share a class secret key, CSK_i and the corresponding class encryption key CEK_i is published. All these users in a security class C_i communicate with each other through messages encrypted using a symmetric secret key K_i. Users in a class C_j may want to access data items encrypted for a class C_i where $C_i \preceq C_j$. For this, symmetric secret key K_i has to be obtained by users of C_j. A class addition or deletion from the hierarchy may take place. There exists a central trusted Key Generation Center (KGC) which generates the CSK and CEK for every class.

2.2 Definitions

Any hierarchical key assignment scheme can be defined as a collection of the following algorithms:

1. **Set:** This algorithm takes security parameter as input and computes the public system parameters.
2. **Gen:** In this algorithm, each class C_i is assigned CSK_i to be kept secret by the class members and a CEK_i which is made public. This phase also generates public parameters corresponding to the edges in the hierarchy tree which may be used for key derivation.
3. **Derive:** This procedure enables users of a class $C_j \in \{ances(C_i) \cup parent(C_i)\}$ to derive decryption key of class C_i.

2.3 Security

Security definition for key management for hierarchy captures the inability of a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ to have a non-negligible advantage over the challenger $\mathcal{C}$ in the following game:

- *Initiate:* $\mathcal{C}$ executes **Set** and **Gen** and gives all the public parameters and CEK_i to the adversary.
- *Query-1:* $\mathcal{A}$ can request K_i for the set of corrupted security classes for number of times polynomial in the security parameter k.
- *Challenge:* When $\mathcal{A}$ decides that the query procedure is over, she can specify a target security class C_{i^*} such that $C_{i^*} \notin \{desc(C_i) \cup child(C_i)\}$ $\forall C_i$ asked in the above query phase. $\mathcal{C}$ now picks a bit h randomly such that $h \in_R \{0,1\}$ and sends a random output (other than the key) $\overline{K_{i^*}}$ if $h = 0$ and the actual key K_{i^*} if $h = 1$.
- *Query-2:* $\mathcal{A}$ can issue more queries for classes C_j such that $C_j \notin \{ances(C_{i^*}) \cup parent(C_{i^*})\}$.
- *Guess:* Now $\mathcal{A}$ outputs her best guess h' and $\mathcal{A}$ is declared winner of the game if $h = h'$.

$\mathcal{A}$'s advantage in the game above $= |Pr[h = h'] - 1/2|$. In addition to the above security definition, hierarchical key management must also support forward and backward secrecy. *Backward Secrecy* requires that if a new security class C_l joins in, the users in C_l must not be able to obtain data encrypted for all the classes C_k such that $C_k \preceq C_l$ prior to C_l's joining. To maintain *Forward Secrecy*, a class $C_{l'}$ that leaves the system must not be able to access data encrypted for all classes C_k such that $C_k \preceq C_{l'}$ after $C_{l'}$ leaves.

3 UNIDIRECTIONAL AND TRANSITIVE PROXY RE-ENCRYPTION

A proxy re-encryption scheme uses a re-encryption key $rk_{A\to B}$ for transformation of ciphertext under a public key pk_A such that it becomes a ciphertext under pk_B and is decrypted by sk_B. While a unidirectional proxy re-encryption prevents delegation in reverse direction, transitivity makes it possible for anyone to derive a direct re-encryption key using various single-hop re-encryption keys.

3.1 Definition

We define a proxy re-encryption scheme which is same as in (Ateniese et al., 2006) except that it has an additional procedure for direct re-encryption key derivation.

Definition 1. *A proxy re-encryption scheme can be defined as a collection Π =(Setup, KeyGen, Enc$_1$, Dec$_1$, Enc$_2$, Dec$_2$, ReKeyGen, ReEnc, Derive) of algorithms defined as follows:*

- ***Setup(1^λ)**$\to$ param: This algorithm inputs security parameter 1^λ to produce system parameters.*
- ***KeyGen(i)**$\to$ (sk_i, pk_i): Public-secret key pair (sk_i, pk_i) can be generated using this procedure.*
- ***Enc$_1$(pk_i, m)**$\to$ CT_i: This produces a first-level ciphertext CT_i that can be decrypted using sk_i.*
- ***Enc$_2$(pk_i, m)**$\to$ CT_i': This produces a second-level ciphertext CT_i' which can be re-encrypted using a valid re-encryption key.*
- ***Dec$_1$(sk_i, CT_i)**$\to$ m: The first-level ciphertext CT_i can be decrypted by this procedure using sk_i.*
- ***Dec$_2$(sk_i, CT_i')**$\to$ m: This algorithm decrypts the second-level ciphertext under pk_i using sk_i.*
- ***ReKeyGen(sk_i, pk_j)**$\to$ $rk_{i\to j}$: This algorithm generates a re-encryption key $rk_{i\to j}$ that transforms CT_i' into CT_j.*
- ***ReEnc**($rk_{i\to j}, CT_i'$) $\to CT_j$: Given a valid $rk_{i\to j}$, anyone can transform CT_i' into CT_j.*
- ***Derive$_{\mathbf{RK}}$**($\{rk_{1\to 2}, \ldots, rk_{(n-1)\to n}\}$) $\to$ $rk_{1\to n}$: Given a sequence of all the one-hop re-encryption keys on the path from first delegator to the last delegatee, a direct re-encryption key can be derived using this procedure.*

3.2 Correctness

1. A first-level ciphertext must decrypt correctly.
   $\mathbf{Dec_1}(sk_i, \mathbf{Enc_1}(pk_i, m)) = m$
2. A second-level ciphertext must decrypt correctly.
   $\mathbf{Dec_2}(sk_i, \mathbf{Enc_2}(pk_i, m)) = m$
3. A re-encrypted ciphertext must decrypt correctly.
   $\mathbf{Dec_1}(sk_j, \mathbf{ReEnc}(\mathbf{ReKeyGen}(sk_i, pk_j), \mathbf{Enc_2}(pk_i, m))) = m$
4. Derivation of direct re-encryption key given all the intermediate keys, must be correct. That is,
   $\mathbf{Derive_{RK}}(\mathbf{ReKeyGen}(sk_1, pk_2), \ldots, \mathbf{ReKeyGen}(sk_{n-1}, pk_n)\}) \to rk_{1\to n}$

3.3 Security

We define security requirements for the proposed proxy re-encryption scheme against a PPT adversary ($\mathcal{A}$) that tries to distinguish two second-level ciphertexts as under:

$$Pr[(pk_{i^*}, sk_{i^*}) \leftarrow \mathbf{KeyGen}(i^*), \{(pk_B, sk_B) \leftarrow \mathbf{KeyGen}(B)\}, \{(pk_X, sk_X) \leftarrow \mathbf{KeyGen}(X)\}, \{rk_{B\to i^*} \leftarrow \mathbf{ReKeyGen}(sk_B, pk_{i^*})\}, \{rk_{X\to i^*} \leftarrow \mathbf{ReKeyGen}(sk_X, pk_{i^*})\}, \{rk_{X\to B} \leftarrow \mathbf{ReKeyGen}(sk_X, pk_B)\}, (m_0, m_1, \alpha) \leftarrow \mathcal{A}(pk_{i^*}, \{pk_B\}, \{(pk_X, sk_X)\}, \{rk_{B\to i^*}\}, \{rk_{X\to i^*}\}, \{rk_{X\to B}\}, CT_{i^*}, b \leftarrow \{0,1\}, b' \leftarrow \mathcal{A}_k(\alpha, \mathbf{Enc_2}(pk_i, m_b))) : b = b'] < \frac{1}{2} + \frac{1}{poly(k)}$$

Set of key-pairs of the honest users as $\{(sk_B, pk_B)\}$, set of key-pairs of set of corrupted users as $\{(sk_X, pk_X)\}$ and key-pair of the target user as (sk_{i^*}, pk_{i^*}). The definition means that a colluding adversary is able to distinguish two direct ciphertexts despite the availability of all the re-encryption keys, with negligible advantage.

3.4 Hierarchical Key Assignment

We define hierarchical key assignment (**Set**, **Gen** and **Derive**) as an application of a unidirectional transitive proxy re-encryption scheme. (See Table 1):

1. **Set**: It uses **SetUp** defined in Section 3.1 to generate system parameters.
2. **Gen**: It calls **KeyGen** defined in Section 3.1 and generates key pairs ($CEK_i = pk_i, CSK_i = sk_i$) $\forall C_i \in S$. Re-encryption keys $rk_{i\to j}, \forall C_i, C_j \in S$

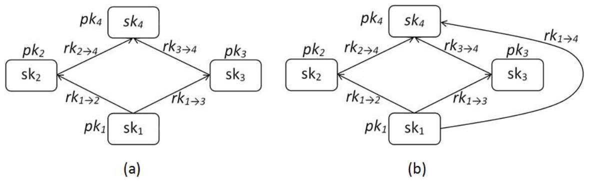

Figure 2: (a) Key assignment and (b) re-encryption key derivation in a unidirectional-transitive proxy re-encryption.

Table 1: Operation of access control in hierarchy using unidirectional and transitive proxy re-encryption scheme.

HKA	PRE procedures
Set	SetUp(1^λ)$\rightarrow$ *params*
Gen	KeyGen(i)$\rightarrow (CSK_i = sk_i, CEK_i = pk_i)$ ReKeyGen(CSK_i, CEK_j)$\rightarrow rk_{i\rightarrow j}$ Enc$_2$(CEK_i, K_i)$\rightarrow CT_i'$
Derive	*If $C_i \in child(C_j)$:* ReEnc($rk_{i\rightarrow j}, CT_i'$)$\rightarrow CT_j$ Dec$_1$(CSK_j, CT_j)$\rightarrow K_i$
	If $C_i \in desc(C_j)$: Derive$_{\text{RK}}$ ReEnc($rk_{i\rightarrow j}, CT_i'$)$\rightarrow CT_j$ Dec$_1$(CSK_j, CT_j)$\rightarrow K_i$

such that $C_i \in child(C_j)$ are generated and assigned to the corresponding edges by **ReKeyGen**(CSK_i, CEK_j). $\forall C_i \in S$. K_i encrypted using $\mathbf{Enc_2}(CEK_i, K_i)$ to produce CT_i' is published.

3. **Derive**: When any user from C_j wishes to access K_i of class C_i, following two cases are possible:
   $\underline{C_i \in child(C_j)}$: Re-encryption key $rk_{i\rightarrow j}$ should be available as a result of the procedure **Gen** defined above. For users in C_j to access K_i, anyone can compute $\mathbf{ReEnc}(rk_{i\rightarrow j}, CT_i')$ to get CT_j. This can be decrypted using $\mathbf{Dec_1}(CSK_j, CT_j)$.
   $\underline{C_i \in desc(C_j)}$: In this case, $rk_{i\rightarrow j}$ would not be available. The procedure $\mathbf{Derive_{RK}}$ is used with all the one-hop re-encryption keys on the path from C_i to C_j to compute $rk_{i\rightarrow j}$ (see Figure 2). This $rk_{i\rightarrow j}$ can now be used for re-encryption as in the previous case (where $C_i \in child(C_j)$) to produce CT_j. CT_j can be decrypted as $\mathbf{Dec_1}(CSK_j, CT_j)$ to obtain K_i.

3.4.1 Dynamic Access Hierarchy

To preserve backward secrecy, K_i and the individual key pair (CSK_i, CEK_i) for all classes C_i such that $C_i \in \{desc(C_l) \cup child(C_l)\}$ are updated to K_i' and (CSK_i', CEK_i') respectively. The new symmetric secret key K_i' is now encrypted using $\mathbf{Enc_2}(CEK_i', K_i')$. As a result of changing the individual key pairs of every class $C_i \in \{desc(C_l) \cup child(C_l)\}$, corresponding

Table 2: Components stored by a security class C_i in the hierarchy with storage requirement (public/private) and size.

Key Component	Type	No.s
CEK_i	public	1
CSK_i	private	1
K_i	private	1
$rk_{i\rightarrow j}, \forall C_j \in parent(C_i)$	public	$\lvert parent(C_i)\rvert$
$CT_i' = \text{Enc}_2(pk_i, K_i)$	public	1

re-encryption keys $rk_{i'\rightarrow j}$ where $C_j \in parent(C_i)$ are also updated using **ReKeyGen**(CSK_i, CEK_j). Similar procedure is followed in case a class is removed from the hierarchy to preserve forward secrecy.

3.4.2 Performance Analysis

Storage cost: As shown in Table 2, users in C_i have to store only one CSK_i and one symmetric secret key K_i irrespective of the depth d_i of C_i.
Computation cost: As can be seen in Table 1, to derive symmetric secret key K_i of any class $C_i \in \{desc(C_j) \cup child(C_j)\}$ it requires the users in class C_j to perform only one decryption. In case of a change in the access hierarchy, total computation cost for a class C_j equals that of one encryption and generation of $\lvert parent(C_j)\rvert$ re-encryption keys.

3.4.3 Security

Since K_i is encrypted and re-encrypted using algorithms defined in Section 3.1, security of the hierarchical access control depends on security of the unidirectional transitive proxy re-encryption defined in Section 3.3.

4 DESIGNING DESIRED PROXY RE-ENCRYPTION SCHEME

So far, it has been discussed that using a unidirectional and transitive proxy re-encryption scheme for key management in hierarchy, we can achieve:

1. Constant key derivation cost for users of any class,
2. Constant storage overhead on the users,
3. Efficient forward an backward secrecy,
4. No repeated transformation of ciphertext.

We note, however, that there is no proxy re-encryption scheme in the existing literature that is both unidirectional and transitive. We present claims that have the potential to establish that transitivity has to be compromised while achieving unidirectionality and vice versa. We also present improvements by which unidirectionality and transitivity can be "closely" imitated.

4.1 Assumptions

Following are the assumptions subject to which we state and prove our claims regarding transitivity and unidirectionality of a proxy re-encryption scheme:

1. Secret keys required for re-encryption key generation are a member of a finite group of prime order (say $\mathbb{Z}_p$) and resulting re-encryption keys are a member of finite cyclic group of prime order ($\mathbb{G}_1$)
2. A re-encryption key $rk_{i \to j}$ is a function $f : \mathbb{Z}_p^2 \to \mathbb{G}_1$ i.e., $rk_{i \to j} = f(sk_i, sk_j)$. The order of appearance of the function input is important as $f(sk_j, sk_i) = rk_{j \to i}$. Also, $f(x,x) = e$, where e is the identity element of $\mathbb{G}_1$ with respect to the operation defined by the function f' defined next.
3. There exists a function $f' : \mathbb{G}_1^2 \to \mathbb{G}_1$ such that $f'(f(u,v),(w,x)) = f'(f(u,x), f(v,w))$. High level intuition of re-encryption key derivation for transitive re-encryption keys is captured by f'.
4. A unidirectional re-encryption key $f(sk_i, sk_j)$ is a trapdoor function with no trapdoor information to recover either sk_i or sk_j.

4.2 Desirable Conditions for $\mathbf{Derive_{RK}}$

1. Unidirectionality of the re-encryption key, that is, given $rk_{i \to j}$, it is not possible to compute $rk_{j \to i}$.
2. No secret keys are revealed during the process.
3. $\mathbf{Derive_{RK}}(rk_{i \to j}, rk_{i \to j})$ must correctly produce $rk_{i \to k}$. That is:
   $\mathbf{Derive_{RK}}(f(sk_i, sk_j), f(sk_j, sk_k)) = f(sk_i, sk_k)$

We can think of the procedure $\mathbf{Derive_{RK}}$ as the function f' defined in the assumptions above. So,
$f'(f(sk_i, sk_j), f(sk_j, sk_k)) = f'(f(sk_i, sk_k), f(sk_j, sk_j)) = f'(f(sk_i, sk_k), e) = f(sk_i, sk_k) = rk_{i \to k}$.

4.3 Our Claims

In this section, we state and argue for our claims which establish that given the current state-of-the-art design for a proxy re-encryption scheme, it is hard to satisfy both unidirectionality and transitivity.

Claim 1. *To achieve transitivity in the re-encryption key derivation, unidirectionality has to be sacrificed.*

Argument: As mentioned earlier, the $\mathbf{Derive_{RK}}$ function defined in Section 3.1 can be modelled using function f' defined in Section 4.1.
$\mathbf{Derive_{RK}}(rk_{j \to k}, rk_{i \to j}) = f'(f(sk_j, sk_k), f(sk_i, sk_j)) = f'(f(sk_j, sk_j), f(sk_k, sk_i)) = f'(e, f(sk_k, sk_i)) = rk_{k \to i}$.
So, using $\mathbf{Derive_{RK}}$ defined using function f', we may obtain a reverse re-encryption key $rk_{k \to i}$.

Claim 2. *To achieve unidirectionality in re-encryption key generation, transitivity in key derivation has to be sacrificed.*

Argument: Given $rk_{i \to j}$ and $rk_{j \to k}$, definition of $\mathbf{Derive_{RK}}$ using function f' in Section 4.2 requires that the secret key sk_j be separated from the re-encryption key defined using the function f. This is in contradiction to the assumptions we made in Section 4.1 which states inseparability of the secret keys of the communicating parties from the proxy re-encryption keys.

4.4 Alternate Methods

We present the idea of achieving direct re-encryption key derivation using the existing properties of proxy re-encryption. First we exploit the *transferability* and then we present another potential method by deriving a decryption key of the class lower down the hierarchy which can be used to obtain direct re-encryption key.

4.4.1 Bottom-up: Exploiting a transferable PRE

Almost all the initial proxy re-encryption schemes were transferable (Blaze et al., 1998; Ateniese et al., 2006; Canetti and Hohenberger, 2007; Green and Ateniese, 2007; Libert and Vergnaud, 2008). For a transferable proxy re-encryption scheme, it is possible to generate $rk_{i \to k}$ given $rk_{i \to j}$, $rk_{j \to k}$ and secret keys sk_j, sk_k. For example, re-encryption key in (Ateniese et al., 2006) takes the form $rk_{i \to j} = g^{sk_j/sk_i}$ where $g \in \mathbb{G}_1$ is one of the generators of a cyclic group of prime order $\mathbb{G}_1$. One can obtain the decryption key $dk_i = (rk_{i \to j})^{sk_j^{-1}} = g^{1/sk_i}$. This decryption key can be used to generate $rk_{i \to k}$ using $(g^{1/sk_i})^{sk_k}$. Therefore, the task of direct re-encryption key derivation can be reduced to making the decryption key g^{1/sk_i} available to the user with public key pk_k securely. More formally, suppose $pk_1, pk_2, \ldots, pk_n$ is a collection of public keys and $rk_{1 \to 2}, rk_{2 \to 3}, \ldots, rk_{(n-1) \to n}$ are the re-encryption keys. A direct re-encryption key $rk_{1 \to n}$ can be derived using all these re-encryption keys as well as the secret keys $sk_2, sk_3, \ldots, sk_n$. As can be seen, this approach demands active participation of each intermediate user which is unrealistic.

4.4.2 Top-down: Transferring Decryption Keys

The unrealistic approach described in the previous section works bottom-up. The top-down approach discussed here is more practical and requires participation of only one secret key thereby making it more suitable for access control in hierarchy. We assume existence of a light-weight encryption scheme $\mathcal{E}$ that involves (at most) one modular exponentiation or one multiplication such that $\mathcal{E}(x,y)$ gives the encryption of y using x as the key. Decryption of this encryption can be done using another procedure $\mathcal{D}(x,\mathcal{E}(x,y))$ to obtain y. Consider the same collection of public keys $pk_1, pk_2, \ldots, pk_n$ with the sequence of re-encryption keys being $rk_{1\to 2}, rk_{2\to 3}, \ldots, rk_{(n-1)\to n}$. As has been discussed in the bottom-up approach, decryption key dk_i can be obtained using secret key sk_{i+1} and $rk_{i\to(i+1)}$. Now if decryption key dk_{i-1} is encrypted using $\mathcal{E}(dk_i, dk_{i-1})$ and stored as additional component of re-encryption key (as shown in Table 3) $rk_{(i-1)\to i}$, then using $rk_{(i-1)\to i}, rk_{i\to(i+1)}$ and sk_{i+1}, one can obtain not only dk_i but also dk_{i-1} using $\mathcal{D}(dk_i, \mathcal{E}(dk_i, dk_{i-1}))$. Therefore a direct re-encryption key $rk_{(i-1)\to(i+1)}$ can be derived using $dk_{(i-1)}$ and $sk_{(i+1)}$. Note that in this process, only one secret key (sk_{i+1}) is involved. So, participation of only one user is required. This would facilitate derivation of dk_1 using sk_i alone in the general case where $rk_{1\to 2}, \ldots, rk_{(n-1)\to n}$ and $pk_1, \ldots, pk_n$ are given as sequences of re-encryption keys and public keys respectively. Since this procedure requires chain of dependent computations for obtaining subsequent decryption keys, these computations cannot be outsourced. However, the direct re-encryption key obtained as a result of this process can be given to anyone to carry out re-encryption.

Table 3: Public, secret and re-encryption key components and their values for approach defined in Section 4.4.2.

Component name	**Content**
Public keys	pk_i
Secret keys	$sk_i, dk_i = f'(sk_i)$
Re-encryption key ($rk_{(i-1)\to i}$)	$f(dk_{i-1}, ek_i)$
	$\mathcal{E}(dk_{i-1}, dk_i)$

5 CONCLUSIONS

In this paper, we have proposed an idea of using proxy re-encryption satisfying special properties for hierarchical key management. We observed that key derivation cost in hierarchy can be reduced to a constant (independent of depth) with all the computations to be done on re-encryption key level if a proxy re-encryption scheme is unidirectional and transitive. We note that there is no re-encryption scheme in literature that is both unidirectional and transitive. This leads us to prove the concrete requirements for such a re-encryption scheme. We claim that given the state-of-the-art design of proxy re-encryption scheme, both unidirectionality and transitivity cannot be achieved simultaneously. We also suggest improvements on the existing re-encryption schemes to achieve efficiency comparable to the case with transitive-unidirectional proxy re-encryption.

REFERENCES

Atallah, M. J., Blanton, M., Fazio, N., and Frikken, K. B. (2009). Dynamic and efficient key management for access hierarchies. *ACM Transactions on Information and System Security*, 12(3):18:1–18:43.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security, (TISSEC'06)*, 9(1):1–30.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology-EUROCRYPT'98*, pages 127–144. Springer.

Canetti, R. and Hohenberger, S. (2007). Chosen-ciphertext secure proxy re-encryption. In *ACM Conference on Computer and Communications Security, (CCS'07)*, pages 185–194. ACM.

Chandran, N., Chase, M., Liu, F.-H., Nishimaki, R., and Xagawa, K. (2014). *Re-encryption, Functional Re-encryption, and Multi-hop Re-encryption: A Framework for Achieving Obfuscation-Based Security and Instantiations from Lattices*, pages 95–112. Springer, Berlin, Heidelberg.

Dodis, Y. and Fazio, N. (2003). Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, PKC '03, pages 100–115, London, UK. Springer-Verlag.

Green, M. and Ateniese, G. (2007). Identity-based proxy re-encryption. In *5th International Conference on Applied Cryptography and Network Security, (ACNS'07)*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer.

Libert, B. and Vergnaud, D. (2008). Unidirectional chosen-ciphertext secure proxy re-encryption. In *11th International Workshop on Practice and Theory in Public-Key Cryptography*, pages 360–379. Springer Berlin Heidelberg.

Shao, J., Liu, P., Cao, Z., and Wei, G. (2011). Multi-use unidirectional proxy re-encryption. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5.

Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. (2010). Encryption policies for regulating access to outsourced data. *ACM Transactions on Database Systems*, 35(2):12:1–12:46.