A-SIT Technology
E-Government Innovation Center

28 March 2022 Study · Privacy & Crypto · Web & Cloud · 26 min read

# Cryptographic Access Protection: Transformation from Interactive to Non-Interactive Proxy Re-Encryption

**Archived.** This content is no longer updated. It remains available for reference.

**Version** 2022-03-25
**Source** https://technology2.a-sit.at/en/publications/cryptographic-access-protection-with-proxy-re-encryption/

Cloud services made it possible for users to conveniently store their data in the cloud and share it with other devices and other users. While such cloud services may not be fully trusted to handle sensitive data, cryptographic mechanisms can be used to achieve end-to-end confidentiality and enforce access control on a cryptographic level.

Initially, this work gives an overview of popular cryptographic mechanisms and compares them with regards to their flexibility in maintaining access rights over time, required effort for revocation procedures, trust requirements, etc. This overview includes approaches built upon traditional public-key encryption, attribute-based encryption, and (conditional) proxy re-encryption.

The second part focuses on applying key-policy conditional proxy re-encryption, which enforces fine-granular access control. Interactive proxy re-encryption schemes require private keys from both the sender and the receiver to enable data sharing. This is an issue, as users do not want to expose their private key just to receive access to re-encrypted data. In contrast, non-interactive schemes only require the receiver's public key to enable re-encryption and sharing of data. Therefore, we develop a generic mechanism to transform interactive into non-interactive proxy re-encryption schemes and evaluate its practical performance.

1. Data Sharing via the Cloud - 2 1.1. Public-Key Encryption to Protect Confidentiality - 2 1.2. Attribute-Based Encryption: Access-Control Based on Attributes and Policies - 3 1.3. Proxy Re-Encryption: Transforming Ciphertext for Authorized Receivers - 4 1.4. Conditional Proxy Re-Encryption: Enforcing Policies before Transformation - 4 2. Transforming Interactive into Non-Interactive Proxy Re-Encryption - 5 2.1. Definition: Interactive and Non-Interactive Key-Policy Conditional Proxy Re-Encryption - 6 2.1.1. Definition: Key-Policy Conditional Proxy Re-Encryption - 7 2.1.2. Correctness - 7 2.1.3. Oracles - 7 2.1.4. Security Definitions - 9 2.2. Transformation: Transforming Interactive into Non-

Interactive Proxy Re-Encryption - 10 2.3. Evaluation - 12 3. Conclusion - 13 4. Appendix: Sketch of Security Proof - 14 4.1. Proof Sketch of IND-CCA2 for Re-Encryptable Ciphertexts - 14 4.2. Proof Sketch of IND-CCA2 for Non-Re-Encryptable Ciphertexts - 15 1. Data Sharing via the Cloud Over the last decade, cloud-based storage services gained popularity due to their convenience. Users store their data in such cloud services, enabling them to 1) access it later from any of their devices and 2) also share it with other users. For example, popular services include Dropbox, Google Drive, Microsoft OneDrive, and Nextcloud.

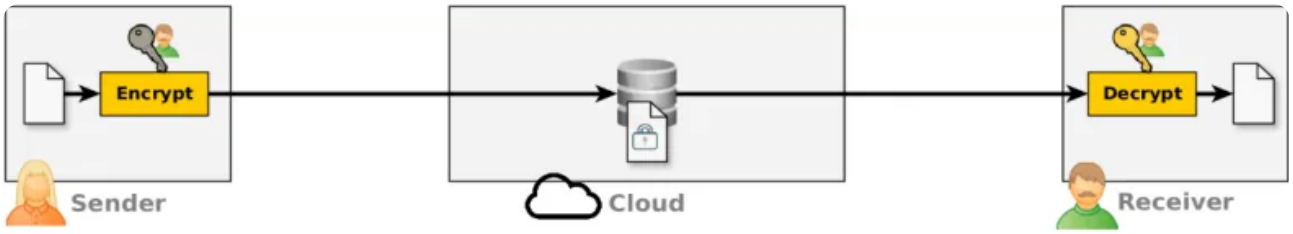
If the data is stored at the cloud service in plain text, this service, its employees, as well as parties operating the underlying cloud infrastructure are able to read all content. Users need to rely on the cloud services to perform access control according to the users' policies, and protect the data confidentiality as well as users' privacy. While users may reach the decision that convenience aspects outweigh privacy risks for low-sensitive data, they might not sufficiently trust cloud services to handle their highly-sensitive data, such as medical records or company secrets.

Fortunately, cryptographic mechanisms are available to enforce the users' access policies even if data are shared across not-fully-trusted infrastructure (e.g., the cloud). The next sections give an overview of cryptographic mechanisms for secure data sharing, particularly traditional public-key encryption, attribute-based encryption, and proxy re-encryption. These mechanisms place control into the users' hands.

Besides security, we also need to consider usability aspects, as approaches that are too tedious hinder adoption. Users may not wish to share their whole data set but rather subsets for different receivers, e.g., according to manual grouping or file hierarchies. Such fine-grained control over who has access to their data has been dynamic in that 1) files can easily be added and removed from groups, 2) access can be granted to further users, and – more interestingly – 3) access can be revoked from previously authorized users. These capabilities need to be available to users in a convenient way to preserve the initial value proposition of cloud-based data sharing services.

1.1. Public-Key Encryption to Protect Confidentiality Figure 1: Public-Key Encryption to Protect Data in the Cloud Public-Key Encryption In the simplest form, public-key encryption (PKE) can be used to protect access to the users' data when outsourced to the cloud. As shown in Figure 1, users encrypt their data for the public keys of authorized receivers before uploading the resulting ciphertexts to the cloud service. Consequently, the cloud cannot learn the encrypted data. Authorized receivers can download the ciphertexts from the cloud service and decrypt them with their private keys.

Hybrid Encryption In practice, hybrid encryption (Kaliski, 1998) combines the advantages of public-key encryption and symmetric encryption schemes. That is, the sender generates a symmetric key (i.e., the content encryption key), which is used to encrypt the data. This content encryption key is then encrypted with the public key of the authorized receiver. Both, the encrypted data and the encrypted content encryption key are uploaded to the cloud. Authorized receivers may download these two ciphertexts. They, firstly, use their private key to decrypt the content encryption key, before using it to decrypt the actual data. Symmetric encryption is used on the bulk of the data, as it is significantly faster than public-key encryption, while public-key encryption enables selecting authorized receivers.

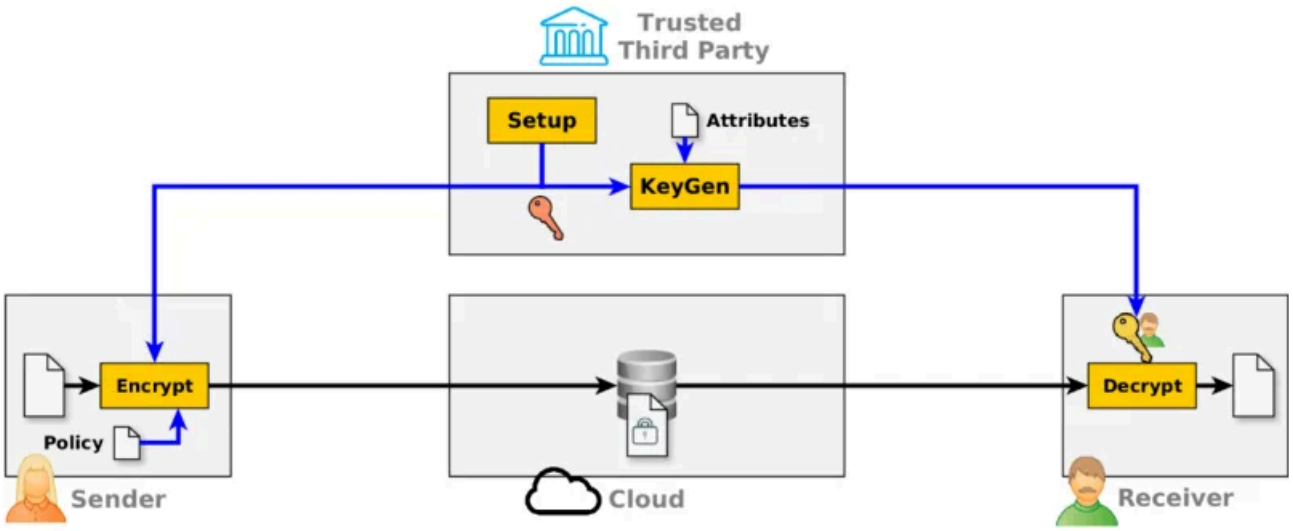



Lockboxes Going one step further, systems may use the concept of lockboxes (Fu, 1999). That is, lockboxes contain keys to access (or decrypt) a group of files or other lockboxes hierarchically. These lockboxes are encrypted for authorized receivers. Consequently, users are able to share access to files with different receivers hierarchically.

Tedious Maintenance Approaches that build upon traditional public-key encryption can be tedious to maintain. In a naïve system, the same file needs to be encrypted and uploaded once for each authorized receiver, leading to multiple versions per file. With hybrid encryption and lockboxes, only the symmetric keys (content encryption keys) need to be encrypted and uploaded for different receivers. However, as new files are added, they have to be tediously shared again or added to existing lockboxes. Furthermore, suppose receivers should no longer have access to some files. In that case, these lockboxes need to be re-organized, and ciphertext may need to be encrypted again for different content encryption keys. Overall, such approaches require heavy involvement of data owners to maintain groups of keys that correspond to groups of files for various users.

1.2. Attribute-Based Encryption: Access-Control Based on Attributes and Policies Figure 2: Access Control via Attribute-Based Encryption Attribute-based encryption (ABE) Attribute-based encryption (ABE) (Goyal, 2006) enables enforcing access control based on attributes and policies. In ciphertext-policy ABE, as shown in Figure 2, users define attributes when encrypting a file before uploading the ciphertext to the cloud service. The cloud service does not learn the encrypted data. A trusted third party generates decryption keys that grant decryption abilities to receivers according to attributes. Receivers are only able to decrypt ciphertext obtained from the cloud service if the receivers' attributes satisfy the policy defined by the sender. Alternatively, in key-policy ABE, users attach attributes to their data during encryption, while the trusted third party generates keys that have the power to decrypt based on policies. In practice, when using hybrid encryption, ABE is used to grant access to symmetric content encryption keys.

Challenges of ABE However, applying ABE also presents various challenges. Changes in the access rights are difficult to accommodate. To grant access to a new receiver, the user would have to update the policy of all relevant ciphertexts, i.e., re-generate and re-upload the ciphertexts. Furthermore, revoking access requires replacing all attributes the receiver holds in the entire system, which includes re-issuing decryption keys for all other receivers with replacement attributes and re-generating all ciphertexts for those replacement attributes. Additionally, as a trusted third party that assigns attributes to receivers in the traditional ABE setup, this party is able to decrypt and read all data.

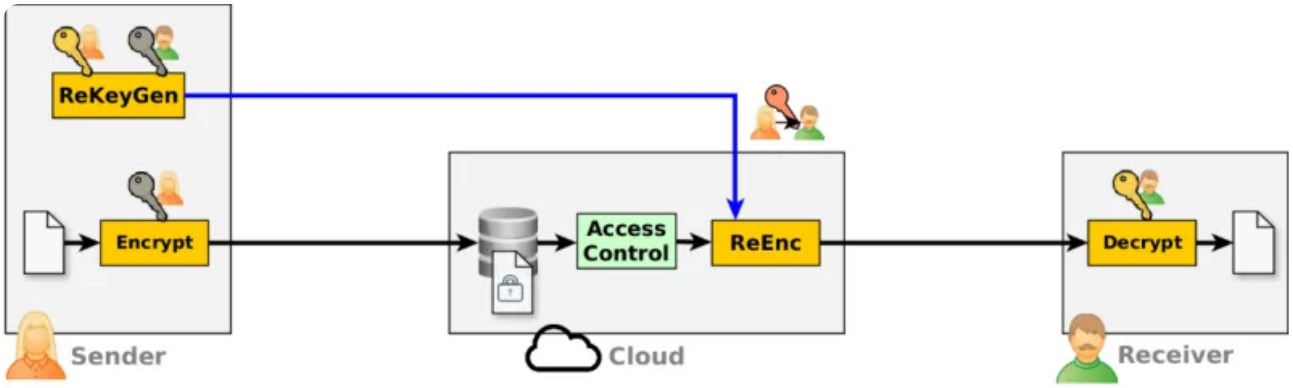


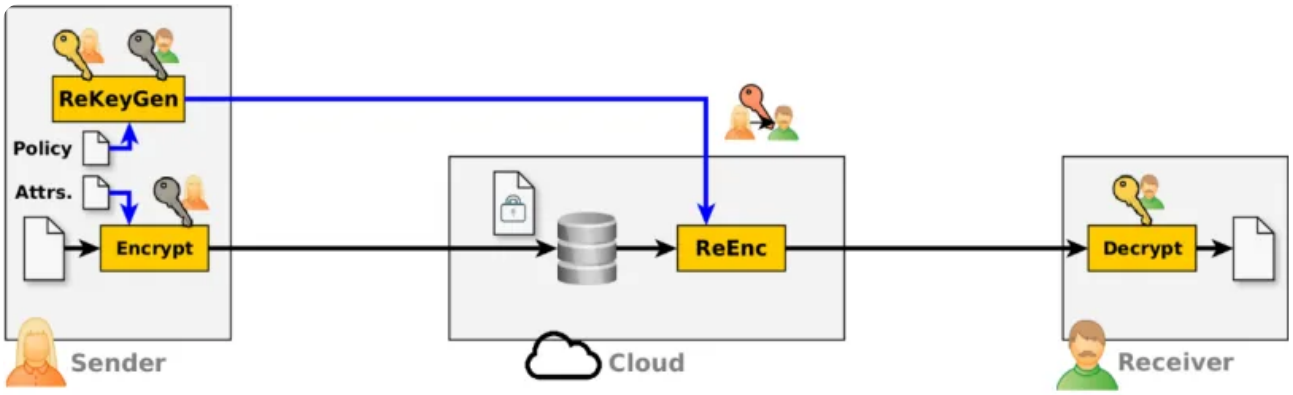

1.3. Proxy Re-Encryption: Transforming Ciphertext for Authorized Receivers Figure 3: Proxy Re-Encryption for Secure Data Sharing Proxy Re-Encryption (PRE) Proxy re-encryption (PRE) (Ateniese, Fu, Green, & Hohenberger, 2006) is an alternative approach to enable endto-end confidential data sharing, as shown in Figure 3. PRE enables a proxy to transform ciphertext encrypted for one user (e.g., data owner) into ciphertext for another user (e.g., intended receiver) without learning the plaintext in an intermediate step. The data owner controls the re-encryption process, as they have to generate a reencryption key from their private key towards a specified receiver.

Fine-Grained Access Control Not Yet Enforced However, PRE does not yet provide fine-grained access control on a cryptographic level. While the user-generated re-encryption key enforces which receivers may decrypt the users' data, the cloud service still needs to enforce which files it re-encrypts for the receiver, e.g., based on a more fine-grained access control policy of the user. Unfortunately, the cloud service could ignore the users' access control policies and use a re-encryption key to reencrypt any of her files for a receiver, rather than only sharing the intended subset.

1.4. Conditional Proxy Re-Encryption: Enforcing Policies before Transformation Figure 4: Conditional Proxy Re-Encryption to Enforce Access Control Conditional Proxy Re-Encryption (CPRE) Conditional proxy re-encryption (CPRE) (Weng, Deng, Ding, Chu, & Lai, 2009) extends proxy re-encryption with fine-grained access control, which is enforced on a cryptographic level. CPRE limits the transformation power of the proxy so that the ciphertext and re-encryption key also have to match w.r.t. their attached attributes and policies. Depending on where an attribute set and where a policy is defined, CPRE schemes are called key-policy (KP-CPRE) or ciphertext-policy (CP-CPRE).

The data flows when applying KP-CPRE (Fang, Susilo, Ge, & Wang, 2011) are shown in Figure 4. Users attach attributes to a ciphertext during encryption before uploading the ciphertext to the cloud service. At a later point in time, users may want to share subsets of their data with others. Therefore, they define an access policy and apply it when generating a re-encryption key towards an authorized receiver. This re-encryption key enables the cloud service to re-encrypt and therefore share specific subsets of the users' data with other users. The cloud




service is only able to successfully re-encrypt ciphertexts if the ciphertext's and the re-encryption key's attributes match.

Ciphertext-policy schemes require that the data owners know exactly which categories of receivers should gain access to the data when encrypting and uploading it, while receivers may be placed into a category at a later point in time. Therefore, we focus on KP-CPRE.

Convenience of Revocation Compared to ABE Revocation of access is required to keep fine-grained access control up-to-date with future developments, e.g., if an employee leaves the company and therefore should not have access to the company's sensitive files anymore. When using ABE, revoking access requires a complex process. Only instructing the cloud service to not hand data to corrupted users anymore reverts to a non-cryptographic solution and is insufficient: Attacker could still steal ciphertext (from the cloud service, other users, or in transit) and apply their old existing decryption key. Instead, the user and system have to cooperate to 1) revoke a set of attributes, 2) replace them with new (versions of these) attributes to retain access for the remaining users, 3) distribute new decryption keys involving those new attributes, and 4) process all ciphertexts that include impacted attribute sets.

With KP-CPRE, minimal user involvement is required: The data owner revokes access rights by simply instructing the proxy to not apply the re-encryption key towards a corrupted user anymore. This corrupted user gains no benefit by stealing ciphertext in transit or from other users, as the ciphertext has to be transformed with KP-CPRE specifically for this corrupted user to enable them to decrypt with their (now obsolete) key material.

2. Transforming Interactive into Non-Interactive Proxy Re-Encryption Remaining Challenge: Interactivity When applying conditional proxy re-encryption to achieve fine-grained and cryptographically-enforced access control, we face a remaining challenge: the required interactivity.

The required interactivity is an important property of schemes implementing conditional proxy re-encryption (and proxy re-encryption in general). Such schemes can be interactive or non-interactive. (1) Interactive PRE requires the private keys of both sender and receiver to generate a re-encryption key, potentially in an interactive protocol. (2) In contrast, in non-interactive PRE, only the sender uses their private key and only requires the receiver's public key to generate a re-encryption key. The required interactivity is dictated by how the scheme is constructed.

When considering a cloud-based data sharing service, PRE schemes that are non-interactive can easily be applied. If the data owner (for whom the data is encrypted) wants to share it, they generate a re-encryption key by themselves with their private key. They only need to obtain the receiver's public key. However, using interactive PRE schemes represents an issue: Neither the sender nor the receiver wants to expose their private keys to another party. Even if they would enter their private data in a secure interactive protocol, they would still need to be available at the same time to perform (sensitive) computations. Consequently, non-interactive schemes are significantly more convenient to use than interactive PRE.

When selecting schemes with desirable features, a requirement of non-interactivity limits the set of eligible PRE schemes, as interactive schemes need to be excluded even if they have other highly-desired features. Such a limitation can lead to a conflict, as for some categories of schemes (e.g., exhibiting a desired or required property), there might not be a scheme that satisfies non-interactivity in addition to the other desired properties.

Also, it is non-trivial to adapt interactive schemes to turn them into non-interactive schemes. Schemes need to adhere to complicated security guarantees in a complex setup of senders, delegators and delegates within a PRE ecosystem. For each scheme, a security proof has to show how the guarantees are met. If such schemes were to be modified on top, e.g., in ad-hoc attempts to resolve the issue of interactivity, it would be unclear if the security guarantees still hold.

Transforming Interactive into Non-Interactive PRE This section presents a transformation from interactive into non-interactive PRE schemes. While we focus on keypolicy conditional PRE, the same transformation can also be applied to more generic PRE as well. We perform the following steps:

1. Formal Model and Security Definition: Initially, we formally define the algorithms of non-interactive key-policy CPRE (NI-KP-CPRE) as well as correctness and game-based security notions of various strengths for first- and second-level ciphertexts.

2. Modular Construction: We present the first non-interactive construction of KP-CPRE. Our construction has been designed to be modular so that its components can be easily replaced to accommodate different schemes. It is based on interactive KP-CPRE (I-KP-CPRE) and public-key encryption (PKE): The construction uses I-KP-CPRE for fine-grained access control, i.e., to attach attributes to ciphertexts and policies to the reencryption keys, which have to match during the re-encryption operation. To make our construction noninteractive, the construction uses PKE to share a

newly generated I-KP-CPRE key pair with the receiver. Reencryption keys are generated towards this new I-KP-CPRE key pair. We sketch a proof of IND-CCA2-security for this construction under the assumption that the used PKE and I-KP-CPRE satisfy their respective IND-CCA notions.

3. Implementation and Evaluation: Finally, we showcase the feasibility and practical efficiency of our transformer construction. We implement our modular construction based on Fang et al.'s interactive KP-CPRE scheme (Fang, Susilo, Ge, & Wang, 2011) and run benchmarks for various policies and attribute sets on both a PC as well as a mobile phone.

2.1. Definition: Interactive and Non-Interactive Key-Policy Conditional Proxy Re-Encryption This section provides a formal definition for non-interactive (NI-KP-CPRE) and interactive key-policy CPRE (I-KP-CPRE), which is illustrated in Figure 5. We adapt and extend the definition of Fang et al. (Fang, Susilo, Ge, & Wang, 2011) by also considering non-interactive schemes, i.e., so that the intended receiver only needs to provide their public key instead of a private key. We elaborate on the correctness property and define oracles as well as security games of various strengths.

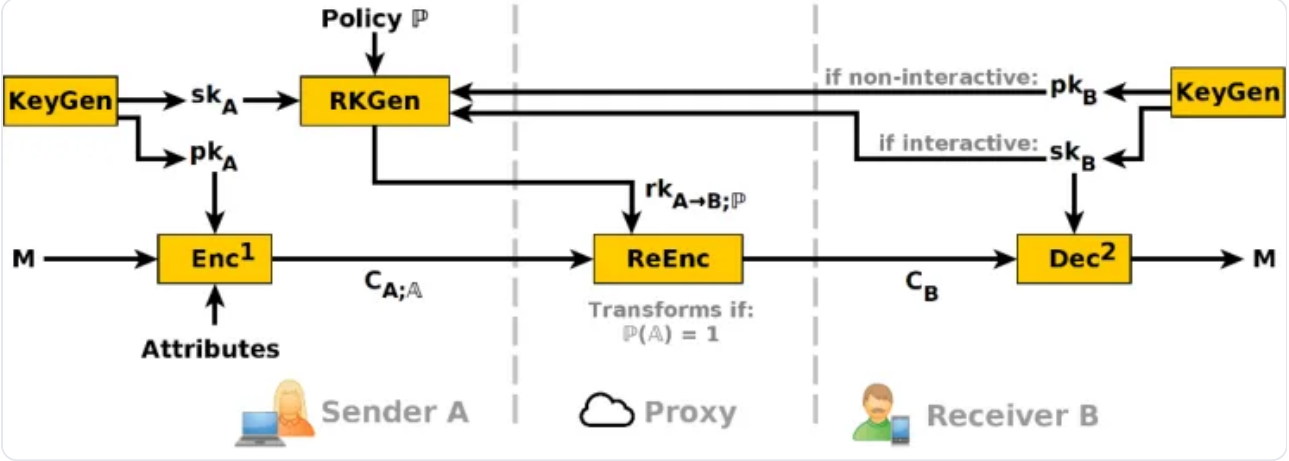



Figure 5: Algorithms and Dataflows in Interactive and Non-Interactive Key-Policy Conditional Proxy Re-Encryption

2.1.1. Definition: Key-Policy Conditional Proxy Re-Encryption A fine-grained key-policy conditional proxy re-encryption scheme consists of the following algorithms.

2.1.2. Correctness Correctness requires that 1) every first-level ciphertext can be decrypted with the respective private key, and 2) when also considering re-encryptable and re-encrypted ciphertexts, requires that second-level ciphertexts can be re-encrypted with a suitable re-encryption key and then decrypted using the (delegatee's) respective private key.

Formally, this means:

2.1.3. Oracles This section defines oracles, which aim to encapsulate the complexity of a KP-CPRE system. These oracles are used to simplify the security experiments and their proofs. The oracles have been designed in line with the security definition by Fang et al. (Fang, Susilo, Ge, & Wang, 2011), who concentrate on an interactive scheme. The environment maintains initially empty lists of challenge ( `CH` ), corrupt ( `CU` ), and honest users ( `HU` ), as well as initially empty sets for private ( `SK` ) and public keys ( `PK` ). Additionally, the environment has an initially empty set `ReMap` to track re-encryptions of the challenge ciphertext.

$\mathsf{Setup}(1^\kappa) \to \mathsf{par}$: On input of a security parameter $\kappa$, this algorithm outputs the public parameters $\mathsf{par}$. These public parameters $\mathsf{par}$ are the first input to all other algorithms, but we omit them in the remainder of this work for brevity.

$\mathsf{KeyGen}() \to (\mathsf{sk}, \mathsf{pk})$: This algorithm outputs a secret and public key $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{RKGen}(\mathsf{sk}_A, \mathbb{P}, \mathsf{pk}_B \text{ or } \mathsf{sk}_B) \to \mathsf{rk}_{A \to B;\mathbb{P}}$: This algorithm takes a secret key $\mathsf{sk}_A$ of user $A$, key material of user $B$ and a policy $\mathbb{P}$, which limits the re-encryption power of the resulting re-encryption key $\mathsf{rk}_{A \to B;\mathbb{P}}$. If the KP-CPRE scheme is interactive, user $B$ provides the private key $\mathsf{sk}_B$, while if the KP-CPRE scheme is non-interactive, user $B$ only provides the public key $\mathsf{pk}_B$.

$\mathsf{Enc}^l(\mathsf{pk}, \mathbb{A}, M) \to C_A$: On input of a public key $\mathsf{pk}$, a message $M \in \mathcal{M}$, and an attribute set $\mathbb{A}$, the algorithm outputs a level-$l$ ciphertext $C^l$. First-level ciphertext $C^1$ can be re-encrypted, while second-level ciphertext $C^2$ cannot be (further) re-encrypted.

$\mathsf{ReEnc}(\mathsf{rk}_{A \to B;\mathbb{P}}, C_A^1) \to C_B^2$: A ciphertext $C_A^1$ conditioned with an attribute set $\mathbb{A}$ can only be translated for another user to ciphertext $C_B^1$, if the re-encryption key $\mathsf{rk}_{A \to B;\mathbb{P}}$ was created for the same condition $\mathbb{P}(\mathbb{A}) = 1$.

$\mathsf{Dec}^l(\mathsf{sk}, C^l) \to M$: On input of a secret key $\mathsf{sk}$ and ciphertext $C^l$, the algorithm outputs the underlying message $M \in \mathcal{M}$ or $\{\perp\}$.

| for | it holds that |
|---|---|
| all security parameters $\kappa \in \mathbb{N}$ | $\mathsf{Dec}^1(\mathsf{sk}_u, \mathsf{Enc}^1(\mathsf{pk}_u, \mathbb{A}, M)) = M$ |
| all public parameters $\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$ | $\mathsf{Dec}^2(\mathsf{sk}_u, \mathsf{Enc}^2(\mathsf{pk}_u, M)) = M$ |
| any number of users $U \in \mathbb{N}$ | $\mathsf{Dec}^2(\mathsf{sk}_{u'}, \mathsf{ReEnc}(\mathsf{rk}_{u \to u';\mathbb{P}}$ |
| all keys $(\mathsf{pk}_u, \mathsf{sk}_u)_{u \in [U]}$ by $\mathsf{KeyGen}(1^\kappa)$ | $\mathsf{Enc}^1(\mathsf{pk}_u, \mathbb{A}, M))) = M$ |
| any $u \in [U]$ | |
| all $u' \in [U], u \neq u'$ | |
| any attribute set $\mathbb{A}$ | |
| any policy $\mathbb{P}$ where $\mathbb{P}(\mathbb{A}) = 1$ | |
| any re-encryption $\mathsf{rk}_{u \to u';\mathbb{P}}$ by RKGen | |
| all messages $M \in \mathcal{M}$ | |

Adding Honest and Corrupted Users Oracle ( `AHU`, `ACU` ) Adversaries can add new users to the system, apart from the challenge user. These users can either be honest ( `AHU` ), where the adversary only learns the public key, or corrupt ( `ACU` ), where the adversary additionally learns the private key.

Re-Encryption Key Generator Oracle ( `RKG` ) Additionally, adversaries may obtain re-encryption keys ( `RKG` ). This oracle is only limited so that an adversary does not obtain a re-encryption key from the challenge user to a corrupt user, which satisfies the challenge attribute set (i.e., $\mathbb{P}$ ( $\mathbb{A}$ *) = 1). With such a key, the adversary

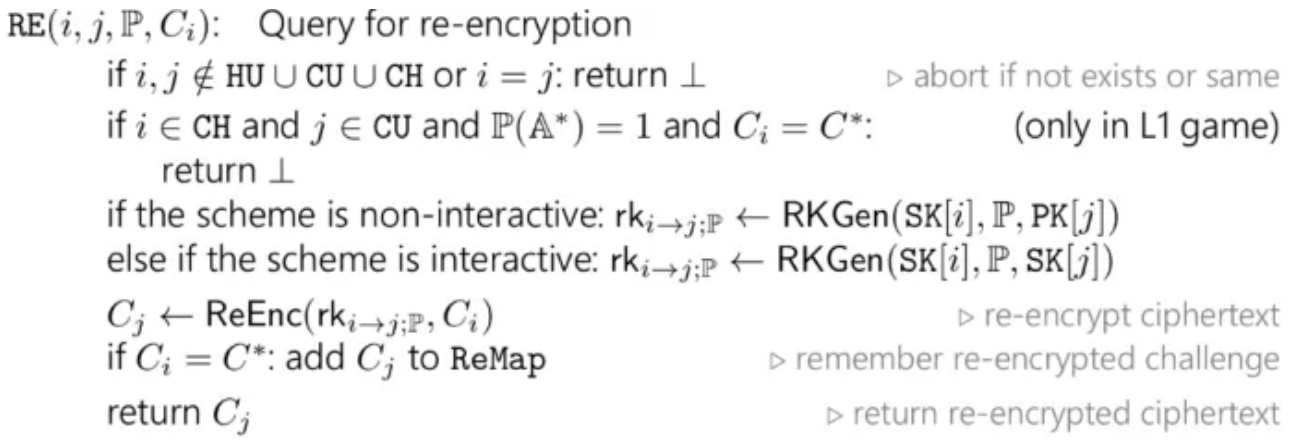
could re-encrypt the challenge ciphertext for a corrupted user, decrypt it, and therefore trivially break the game. Note that we still allow re-encryption keys from honest to corrupted users with any policy (cannot decrypt on her own or with decrypt oracle), or also keys from challenge to corrupted users with a too weak policy (i.e., $\mathbb{P}(\mathbb{A}^*) = 0$). Please also note that re-encrypted ciphertexts cannot be further re-encrypted, and therefore, we do not have to consider scenarios where a challenge ciphertext is reencrypted for an honest user and further re-encrypted for a corrupt user. Re-encryption keys from a corrupt user to any other user do not need to be covered, as the adversary can generate these keys on their own given the corrupt user's private key material.

Re-Encryption Oracle ( RE ) Similarly, the adversary may obtain re-encryptions of ciphertexts ( RE ). We weaken the previous limitation and thus provide a more powerful oracle to the adversary: A ciphertext from the challenge user cannot be re-encrypted for a corrupted user if the policy $\mathbb{P}$ fulfills the challenge ciphertext's attributes $\mathbb{A}$ - (i.e., $\mathbb{P}(\mathbb{A}^*) = 1$) and furthermore the input ciphertext is the challenge ciphertext. Note that this oracle ( RE ) is more powerful than the re-encryption key oracle ( RKG ), as ciphertext of the challenge user can be re-encrypted for a corrupt user even under the challenge attribute set (i.e., $\mathbb{P}(\mathbb{A}^*) = 1$), which can then be decrypted by the adversary, as long as it is not the challenge ciphertext.

$\mathtt{AHU}(i)$: Query to add honest user
- if $i \in \mathtt{HU} \cup \mathtt{CU} \cup \mathtt{CH}$: return $\perp$ ▷ user does not yet exist
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa)$ ▷ generate key material
- $\mathtt{HU} \leftarrow \mathtt{HU} \cup \{i\}$, $\mathtt{SK}[i] \leftarrow \mathsf{sk}_i$, $\mathtt{PK}[i] \leftarrow \mathsf{pk}_i$ ▷ remember user
- return $\mathsf{pk}_i$ ▷ return only public key

$\mathtt{ACU}(i)$: Query to add corrupt user
- if $i \in \mathtt{HU} \cup \mathtt{CU} \cup \mathtt{CH}$: return $\perp$ ▷ user does not yet exist
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(1^\kappa)$ ▷ generate key material
- $\mathtt{CU} \leftarrow \mathtt{CU} \cup \{i\}$, $\mathtt{SK}[i] \leftarrow \mathsf{sk}_i$, $\mathtt{PK}[i] \leftarrow \mathsf{pk}_i$ ▷ remember user
- return $(\mathsf{sk}_i, \mathsf{pk}_i)$ ▷ return public and private key

$\mathtt{RKG}(i, \mathbb{P}, j)$: Query for re-encryption key generation
- if $i, j \notin \mathtt{HU} \cup \mathtt{CU} \cup \mathtt{CH}$ or $i = j$: return $\perp$ ▷ abort if not exists or same
- if $i \in \mathtt{CH}$ and $j \in \mathtt{CU}$ and $\mathbb{P}(\mathbb{A}^*) = 1$: return $\perp$ (only in L1 game)
- if the scheme is non-interactive:
  - return $\mathsf{RKGen}(\mathtt{SK}[i], \mathbb{P}, \mathtt{PK}[j])$ ▷ return re-encryption key
- else if the scheme is interactive:
  - return $\mathsf{RKGen}(\mathtt{SK}[i], \mathbb{P}, \mathtt{SK}[j])$ ▷ return re-encryption key




Decryption Oracle ( D ) Finally, adversaries may also obtain decryptions of ciphertexts that were encrypted for the challenge, honest and corrupt users. This oracle aborts if the challenge ciphertext or a re-encryption of the challenge ciphertext is submitted. Otherwise, the adversary could directly decrypt the challenge ciphertext or a re-encryption of the challenge ciphertext for an honest user and therefore trivially win the game.

Oracles for IND-RCCA To support the notion of IND-RCCA, we adapt the re-encryption and decryption oracles ( RE and D ): Instead of aborting only for the exact (re-encryption of the) challenge ciphertext, the oracles abort if their input ciphertexts would decrypt to the challenge messages, i.e., m0 or m1. That is, the RE [IND-RCCA] checks for Dec(SK[i], ci) ∈ {m0, m1}, while the D [IND-RCCA] aborts if the decrypted m ∈ {m0, m1}.

2.1.4. Security Definitions Our security definition is based on the work of Fang et al. (Fang, Susilo, Ge, & Wang, 2011), which we extend 1) from interactive to non-interactive schemes and 2) from IND-CCA to additionally cover various strengths of security games.

(Red marks differences between experiments.) Figure 6 defines the security experiments on first- (L1) and second-level ciphertexts (L2) for various types T ∈ { IND-CPA, IND-CCA1, IND-CCA2, IND-RCCA}. The two experiments for first- and second-level ciphertexts use different algorithms to encrypt the challenge ciphertext, where second-level ciphertexts make the challenge attribute set $\mathbb{A}$ - obsolete. The type of security T determines which oracles are available in the phases before and after the challenge parameters have been fixed, as detailed in the included table. For type T security, we require that an adversary is not able to win the security experiments for first- and second-level ciphertexts with non-negligible advantage. These security experiments directly apply to both interactive as well as non-interactive KP-CPRE schemes, as only the used oracles for re-encryption key generation and re-encryption oracles require slight modifications.

$\mathtt{D}^{1\&2}(i, C_i)$: Query for decryption

if $i \notin \mathtt{HU} \cup \mathtt{CU} \cup \mathtt{CH}$: return $\perp$ ▷ abort if user does not exist

if $C_i = C^*$ or $C_i \in \mathtt{ReMap}$: return $\perp$ ▷ abort

return $M \leftarrow \mathsf{Dec}^{1\&2}(\mathtt{SK}[i], C_i)$ ▷ return decrypted ciphertext

Figure 7: Overview of Transformation from Interactive to Non-Interactive Proxy Re-Encryption Initial Attempt based on ABE In a naive approach to build non-interactive KP-CPRE, we could generate a key-policy attribute-based encryption (KP-ABE) decryption key and use public-key encryption (PKE) to distribute this key to the intended recipient. Consequently, the recipient would be able to decrypt any ciphertext, as long as the ABE decryption key's policy satisfies the ciphertext's attributes. While this approach seems to fulfill the ambition of KP-CPRE, we encounter a challenge when aiming for IND-CCA security: Once the adversary obtains one decryption of a sufficiently strong ABE decryption key, they can also use it on other ciphertexts from the same origin. In the context of a IND-CCA security game, this would mean applying the decryption oracle ( `D` ) on the tuple of PKE-encrypted ABE decryption key and ABE-encrypted payload, and then using the obtained ABE decryption key on the challenge ciphertext. Unfortunately, this would allow the adversary to win the game trivially, and therefore such a construction would not satisfy the IND-CCA security property.

Experiment $\mathsf{Exp}^{T,\mathsf{L1}}_{\mathsf{KP\text{-}CPRE},\mathcal{A}}(1^\kappa)$

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$

$(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KeyGen}()$

$(\mathbb{A}^*, \mathsf{st}) \leftarrow \mathcal{A}()$

$\mathtt{CH} \leftarrow \{0\}$, $\mathtt{SK}[0] \leftarrow \mathsf{sk}^*$, $\mathtt{PK}[0] \leftarrow \mathsf{pk}^*$

$(M_0, M_1, \mathsf{st}') \leftarrow \mathcal{A}^{\mathcal{O}_1}(\mathsf{st}, \mathsf{par}, \mathsf{pk}^*)$

if $|M_0| \neq |M_1|$: abort

$b \xleftarrow{R} \{0,1\}$, $C^* = \mathsf{Enc}^1(\mathsf{pk}^*, \mathbb{A}^*, M_b)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_2}(\mathsf{st}', C^*)$

if $b = b'$: return 1, else: return 0

Experiment $\mathsf{Exp}^{T,\mathsf{L2}}_{\mathsf{KP\text{-}CPRE},\mathcal{A}}(1^\kappa)$

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\kappa)$

$(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KeyGen}()$

$\mathtt{CH} \leftarrow \{0\}$, $\mathtt{SK}[0] \leftarrow \mathsf{sk}^*$, $\mathtt{PK}[0] \leftarrow \mathsf{pk}^*$

$(M_0, M_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}_1}(\mathsf{par}, \mathsf{pk}^*)$

if $|M_0| \neq |M_1|$: abort

$b \xleftarrow{R} \{0,1\}$, $C^* = \mathsf{Enc}^2(\mathsf{pk}^*, M_b)$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_2}(\mathsf{st}, C^*)$

if $b = b'$: return 1, else: return 0

| | $\mathcal{O}_1$ contains | | | | | $\mathcal{O}_2$ contains | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Oracles | AHU | ACU | RKG | RE | D | AHU | ACU | RKG | RE | D |
| Experiment Levels | both | both | both | only L1 | both | both | both | both | only L1 | both |
| $T = \mathsf{IND\text{-}CPA}$ | | | | | | ✓ | ✓ | ✓ | ✓ | |
| $T = \mathsf{IND\text{-}CCA1}$ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| $T = \mathsf{IND\text{-}CCA2}$ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| $T = \mathsf{IND\text{-}RCCA}$ | ✓ | ✓ | ✓ | weak | weak | ✓ | ✓ | ✓ | weak | weak |

Figure 6: Security Experiments of Key-Policy Conditional Proxy Re-Encryption.

Definition: Type T Security. A non-interactive or interactive KP-CPRE scheme is T-secure, with T ∈ {IND-CPA, IND-CCA1, IND-CCA2, IND-RCCA}, if for any PPT adversary A and for both levels L ∈ {L1, L2} there exists a negligible function ε such that 2.2. Transformation: Transforming Interactive into Non-Interactive Proxy Re-Encryption Figure 8 presents a transformation from interactive KP-CPRE to a non-interactive construction. This transformation can also be applied to traditional PRE schemes. A visual overview is shown in Figure 7.

Solution with I-KP-CPRE To solve this challenge, the re-encryption operation needs to re-key the ciphertext, which our construction achieves by using I-KP-CPRE. For example, Fang et al. (Fang, Susilo, Ge, & Wang, 2011) extend Goyal et al.'s KP-ABE (Goyal, 2006) so that they shift the individual shares of the tree over linear secret sharing parts to another key.

To make I-KP-CPRE non-interactive, we introduce a PKE key pair per user, i.e., KeyGen of our construction generates a PKE key pair besides the I-KP-CPRE key pair. ReKeyGen generates an ephemeral I-KP-CPRE key pair, encrypts it with the other user's PKE key pair, and uses the ephemeral private key to generate the interactive I-KP-CPRE's re-encryption key. This re-encryption key enables re-encryption in ReEnc. To decrypt a re-encrypted ciphertext, Dec decrypts the ephemeral key with PKE, and then uses this ephemeral key to decrypt the CPRE ciphertext. First-level ciphertexts or ciphertexts that cannot be re-encrypted are directly generated and decrypted by the I-KP-CPRE scheme. This transformation is detailed in Figure 8.

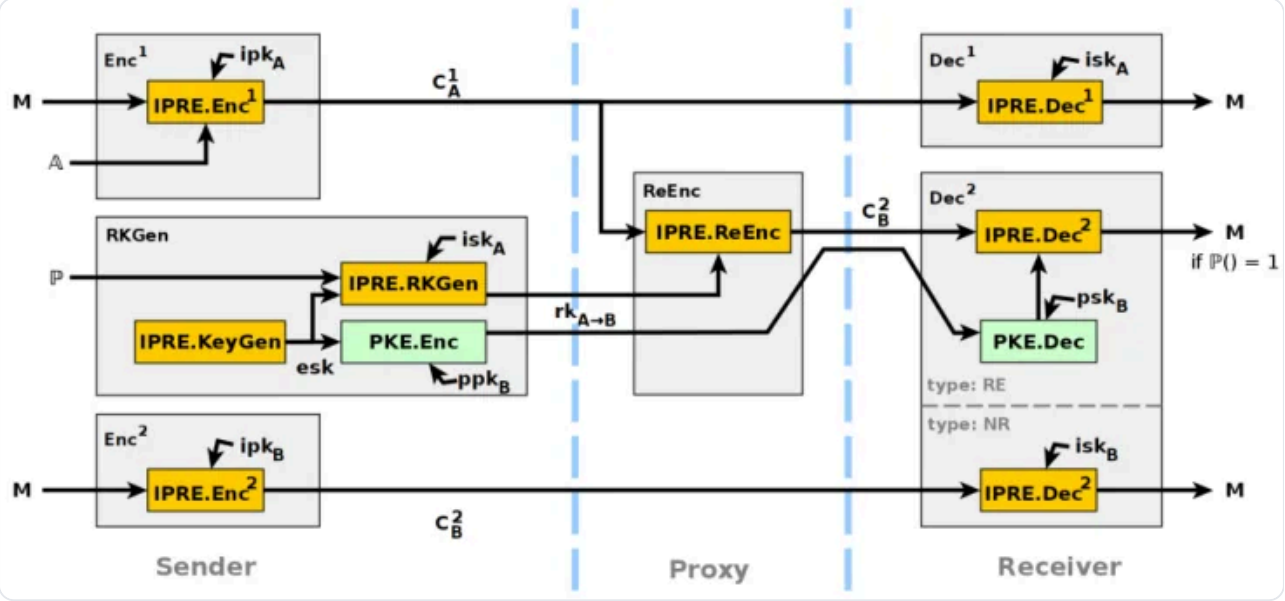



Figure 8: Transformation from Interactive to Non-Interactive Proxy Re-Encryption Generalization to other PRE Types While this transformation is shown on complex KP-CPRE, the transformation also applies to traditional PRE schemes. Traditional PRE can be seen as a special case of KP-CPRE: By requiring that the attribute set $\mathbb{A}$ is empty and that the policy $\mathbb{P}$ that is always satisfied, we reach a semantically-equal definition.

The transformed scheme is IND-CCA2-secure, if the used PKE is IND-CCA-secure and the used I-KP-CPRE is also IND-CCA-secure. A security proof is sketched in the Appendix.

As public parameters, fix a I-KP-CPRE scheme $\mathcal{I}$ and a PKE scheme $\mathcal{P}$.

$\mathsf{KeyGen}(1^\kappa) \to (\mathsf{sk}, \mathsf{pk})$:
- $(\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathcal{I}.\mathsf{KeyGen}(1^\kappa)$ ▷ generate I-KP-CPRE key pair
- $(\mathsf{psk}, \mathsf{ppk}) \leftarrow \mathcal{P}.\mathsf{KeyGen}(1^\kappa)$ ▷ generate PKE key pair
- return $\mathsf{sk} \leftarrow (\mathsf{isk}, \mathsf{psk})$, $\mathsf{pk} \leftarrow (\mathsf{ipk}, \mathsf{ppk})$

$\mathsf{RKGen}(\mathsf{sk}_A, \mathbb{P}, \mathsf{pk}_B) \to \mathsf{rk}_{A\to B;\mathbb{P}}$:
- $(\mathsf{esk}_B, \mathsf{epk}_B) \leftarrow \mathcal{I}.\mathsf{KeyGen}(1^\kappa)$ ▷ generate ephemeral key
- $\mathsf{rk}_{A\to B} \leftarrow \mathcal{I}.\mathsf{RKGen}(\mathsf{isk}_A, \mathbb{P}, \mathsf{esk}_B)$ ▷ use ephemeral key for rk
- $C^{\mathcal{P}}_{B,\mathsf{esk}} \leftarrow \mathcal{P}.\mathsf{Enc}(\mathsf{ppk}_B, \mathsf{esk}_B)$ ▷ encrypt ephemeral key for receiver
- return $(\mathsf{rk}_{A\to B}, C^{\mathcal{P}}_{B,\mathsf{esk}})$

$\mathsf{Enc}^1(\mathsf{pk}_A, \mathbb{A}, M) \to C_A$: // re-encryptable
- return $C^{\mathcal{I}}_{A,M} \leftarrow \mathcal{I}.\mathsf{Enc}^1(\mathsf{ipk}_A, \mathbb{A}, M)$ ▷ encrypt message

$\mathsf{Enc}^2(\mathsf{pk}_A, M) \to C_A$: // not re-encryptable
- $C^{\mathcal{I}}_{A,M} \leftarrow \mathcal{I}.\mathsf{Enc}^2(\mathsf{ipk}_A, M)$ ▷ encrypt message
- return $(\mathtt{NR}, C^{\mathcal{I}}_{A,M})$

$\mathsf{ReEnc}^{1\to 2}(\mathsf{rk}_{A\to B;\mathbb{P}}, C_A) \to C_B$:
- parse $\mathsf{rk}_{A\to B;\mathbb{P}}$ as $(\mathsf{rk}_{A\to B}, C^{\mathcal{P}}_{B,\mathsf{esk}})$ and $C_A$ as $C^{\mathcal{I}}_{A,M}$
- $C^{\mathcal{I}}_{B,M} \leftarrow \mathcal{I}.\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, C^{\mathcal{I}}_{A,M})$ ▷ re-encrypt ciphertext
- return $(\mathtt{RE}, C^{\mathcal{I}}_{B,M}, C^{\mathcal{P}}_{B,\mathsf{esk}})$ ▷ and forward encrypted ephemeral key

$\mathsf{Dec}^1(\mathsf{sk}_A, C_A) \to M$:
- parse $C_A$ as $C^{\mathcal{I}}_{A,M}$
- return $M \leftarrow \mathcal{I}.\mathsf{Dec}^1(\mathsf{isk}_A, C^{\mathcal{I}}_{A,M})$ ▷ decrypt directly

$\mathsf{Dec}^2(\mathsf{sk}_B, C_B) \to M$:
- if type of $C_B$ is $\mathtt{NR}$: ▷ ciphertext has not been re-encypted
  - parse $C_B$ as $(\mathtt{NR}, C^{\mathcal{I}}_{B,M})$
  - return $M \leftarrow \mathcal{I}.\mathsf{Dec}^2(\mathsf{isk}_B, C^{\mathcal{I}}_{B,M})$ ▷ decrypt directly
- else if type of $C_B$ is $\mathtt{RE}$: ▷ ciphertext has been re-encypted
  - parse $C_B$ as $(\mathtt{RE}, C^{\mathcal{I}}_{B,M}, C^{\mathcal{P}}_{B,\mathsf{esk}})$
  - $\mathsf{esk}_B \leftarrow \mathcal{P}.\mathsf{Dec}(\mathsf{psk}_B, C^{\mathcal{P}}_{B,\mathsf{esk}})$ ▷ recover ephemeral key
  - return $M \leftarrow \mathcal{I}.\mathsf{Dec}^2(\mathsf{esk}_B, C^{\mathcal{I}}_{B,M})$ ▷ decrypt with eph. key

2.3. Evaluation In this section, we evaluate the practical efficiency of our transformation (cf. Figure 8) by implementing our modular construction, which we then benchmark for various parameters.

Proof-of-Concept Implementation We instantiate our transformation (cf. Figure 8) with Fang et al.'s I-KP-CPRE scheme (Fang, Susilo, Ge, & Wang, 2011), SHA-256 as hash, and ECIES as PKE. For these underlying schemes, we have selected parameters to achieve 128bit security according to recommendations from NIST (NIST, 2016). Groups for pairing-based curves are chosen following recommendations by Menezes et al. (Menezes, Sarkar, & Singh, 2016) and Barbulescu et al. (Barbulescu & Duquesne, 2019). Our implementation builds upon the RELIC toolkit (Aranha, Gouvêa, Markmann, Wahby, & Liao, 2022) and currently only uses a single thread, which leaves room for future performance improvements.

Methodology Our benchmark focuses on the data flow involving re-encryption of first-level ciphertexts into re-encrypted second-level ciphertexts. Initially, our benchmark generates key pairs for two participants and a re-encryption key between them, with a policy $\mathbb{P}$ . Next, the benchmark takes a 128bit AES key as payload, encrypts the payload with an attribute set $\mathbb{A}$ for the first participant, re-encrypts it with the re-encryption key for the second participant, and finally decrypts the payload with the second participant's private key. The policy $\mathbb{P}$ is shaped as a binary tree of AND gates. As our implementation finds a minimal but sufficient match between ciphertext attributes and policy components before resolving the hierarchical secret sharing system, OR gates only have a negligible impact on the execution time. The attribute set $\mathbb{A}$ consists of the leaves L( $\mathbb{P}$ ) of the policy $\mathbb{P}$, so that $\mathbb{P}$ ( $\mathbb{A}$ ) = 1. We repeat the benchmarks for various sizes of policies and attribute sets.

Direct decryption of first-level ciphertexts or encryption and decryption of non-re-encrypted ciphertexts has similar performance characteristics as the evaluated Enc [1] and Dec [2] on re-encrypted ciphertexts: Dec [1] requires an additional hash per ciphertext attribute. Dec [2] on non-re-encrypted ciphertexts does not require a PKE decryption.

Evaluation Results Figure 9 collects the average execution times in milliseconds as well as standard deviations of 100 runs for two platforms: a PC and a mobile phone. KeyGen and Dec [2] require constant execution times. The time for RKGen, Enc [1], and ReEnc show linear dependencies: ReKeyGen depends on the policy's size; Enc [1] depends on the attribute set's size; ReEnc depends on the size of the minimal allocation between ciphertext attributes that satisfy the policy components, which is for our policy shape again $|\mathbb{A}|$.

| $\lvert\mathbb{P}\rvert$ | $\lvert\mathbb{A}\rvert$ | KeyGen | RKGen | Enc[1] | ReEnc | Dec[2] |
|---|---|---|---|---|---|---|
| PC (Intel i7-4790, 3.6 GHz, 16GB RAM) | | | | | | |
| 3 | 2 | 2.08 (±0.13) | 4.30 (±0.35) | 2.42 (±0.20) | 2.90 (±0.20) | 2.19 (±0.23) |
| 7 | 4 | 2.08 (±0.06) | 6.29 (±0.16) | 3.00 (±0.13) | 3.90 (±0.08) | 2.18 (±0.07) |
| 15 | 8 | 2.06 (±0.04) | 10.24 (±0.18) | 4.16 (±0.13) | 5.90 (±0.10) | 2.17 (±0.07) |
| 31 | 16 | 2.07 (±0.04) | 18.26 (±0.19) | 6.58 (±0.22) | 9.91 (±0.09) | 2.17 (±0.06) |
| Phone (OnePlus 6T) | | | | | | |
| 3 | 2 | 23.02 (±0.50) | 49.64 (±1.31) | 27.92 (±1.20) | 33.20 (±0.69) | 24.25 (±0.69) |
| 7 | 4 | 22.54 (±0.27) | 72.10 (±0.86) | 34.63 (±0.90) | 44.83 (±0.06) | 23.66 (±0.31) |
| 15 | 8 | 22.54 (±0.25) | 119.70 (±1.22) | 50.15 (±1.47) | 69.59 (±0.35) | 23.68 (±0.37) |
| 31 | 16 | 22.51 (±0.24) | 214.26 (±1.93) | 80.28 (±2.02) | 119.05 (±0.17) | 23.72 (±0.34) |

Figure 9: Execution Times (in Milliseconds) and Standard Deviation of our Implementation of the Transformer. The attribute set $\mathbb{A}$ consists of the leaf values L(P) within the policy $\mathbb{P}$, which is shaped as a binary tree of AND gates.

3. Conclusion Cryptography enables users to enforce access control when sharing data via not-fully-trusted infrastructures, such as the cloud. The first part of this article gave an overview of various cryptographic mechanisms for secure data sharing, including traditional public-key encryption, attribute-based encryption, and (conditional) proxy reencryption. These mechanisms differ in the

effort required to maintain and update sharing permissions over time, trust requirements, revocation procedures, etc. We focused on key-policy conditional proxy re-encryption as a flexible and secure data sharing mechanism.

The second part addressed a remaining challenge when aiming to employ (key-policy conditional) proxy reencryption: We presented a generic transformation to turn interactive into (more convenient) non-interactive proxy re-encryption schemes. Our evaluation highlights the feasibility and practical efficiency of the transformation.

4. Appendix: Sketch of Security Proof The proof for Theorem 1 (i.e., IND-CCA2-security of our Transformation) consists of two parts, targeted at firstand second-level ciphertext. Firstly, we prove IND-CCA2 security for re-encryptable ciphertexts in Lemma 1. We can then simplify the first proof to show IND-CCA2 security for non-re-encryptable ciphertexts in Lemma 2.

4.1. Proof Sketch of IND-CCA2 for Re-Encryptable Ciphertexts Lemma 1. Scheme 1 is IND-CCA2-L1-secure if I-KP-CPRE is IND-CCA secure and PKE is IND-CCA-secure.

Proof Sketch. We aim to achieve a reduction based on the very similar definition of IND-CCA for I-KP-CPRE. This reduction simulates the PKE keys and operations for all users. Furthermore, the reduction's oracles, or their underlying NI-KP-CPRE algorithms, place queries to the oracles of I-KP-CPRE whenever necessary, e.g., if private key material is not available of the challenge or an honest user.

However, we face a challenge when generating a re-encryption key: We do not have access to the private key of the challenger and honest user, which are required of both the delegator i as well as the delegatee j in I-KP-CPRE. To instead use the provided oracle RKG [I-KP-CPRE], the delegatee also has to be registered with the system. Therefore, we set up a new ephemeral user j' with oracles AHU [I-KP-CPRE] or ACU [I-KP-CPRE], and then query a re-encryption key towards this user at oracle RKG [I-KP-CPRE] . We apply the same approach for the re-encryption oracle RE.

As oracle AHU [I-KP-CPRE] does not return private keys for (ephemeral) honest users, the reduction needs to be amended. In a sequence of games, we replace the ephemeral private key esk with a random r from the same domain to achieve a slightly modified game G1. By remembering this r, the decrypt oracle can recall the ephemeral user j' and use oracle D [I-KP-CPRE] to decrypt the re-encrypted ciphertext. Since this esk is encrypted with an IND-CCA secure PKE scheme, the change should be indistinguishable.

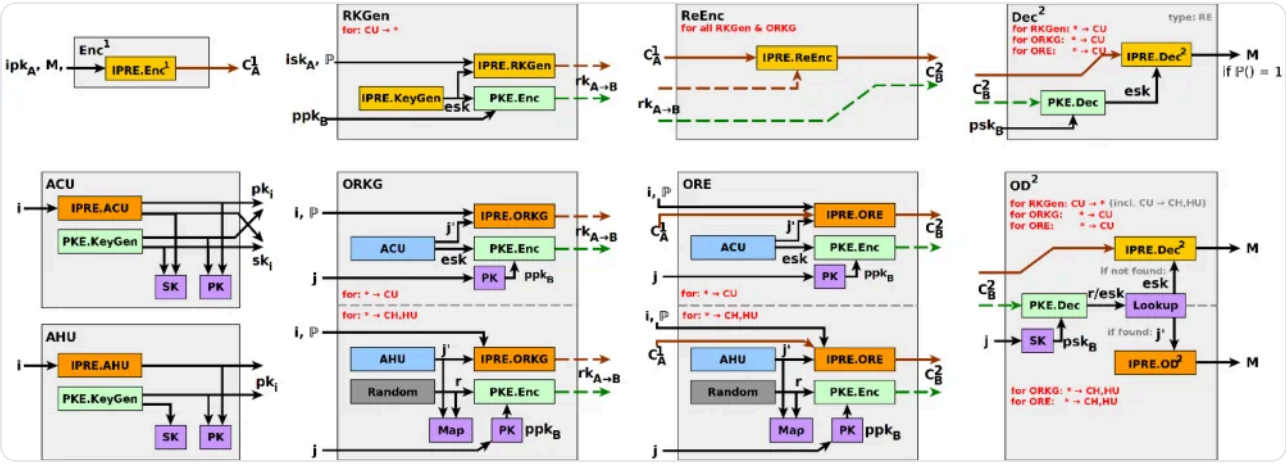
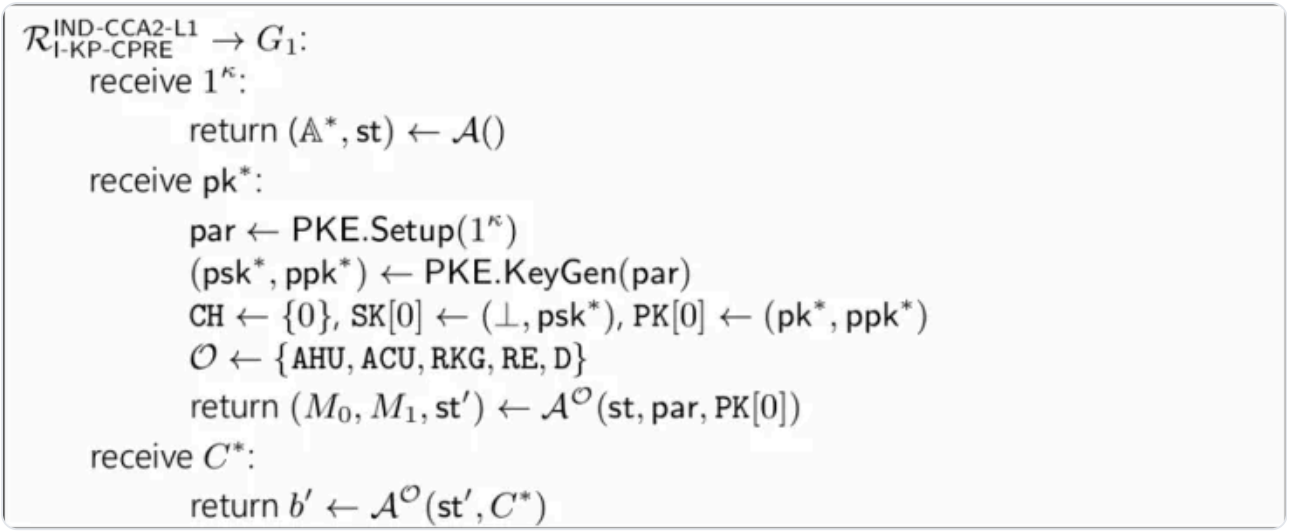



Figure 10: Visual Overview of Adapted Oracles for the Security Proof Figure 11: Reduction for Re-Encryptable Ciphertext



$$\mathcal{R}^{\text{IND-CCA2-L1}}_{\text{I-KP-CPRE}} \rightarrow G_1:$$

receive $1^\kappa$:

return $(\mathbb{A}^*, \text{st}) \leftarrow \mathcal{A}()$

receive $\text{pk}^*$:

$\text{par} \leftarrow \text{PKE.Setup}(1^\kappa)$

$(\text{psk}^*, \text{ppk}^*) \leftarrow \text{PKE.KeyGen}(\text{par})$

$\text{CH} \leftarrow \{0\}$, $\text{SK}[0] \leftarrow (\perp, \text{psk}^*)$, $\text{PK}[0] \leftarrow (\text{pk}^*, \text{ppk}^*)$

$\mathcal{O} \leftarrow \{\text{AHU}, \text{ACU}, \text{RKG}, \text{RE}, \text{D}\}$

return $(M_0, M_1, \text{st}') \leftarrow \mathcal{A}^{\mathcal{O}}(\text{st}, \text{par}, \text{PK}[0])$

receive $C^*$:

return $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\text{st}', C^*)$

Figure 12: Reduction for Non-Re-Encryptable Ciphertext References Aranha, Gouvêa, Markmann, Wahby, & Liao. (2022). RELIC is an Efficient LIbrary for Cryptography. Retrieved from https://github.com/relic-toolkit/relic Ateniese, G., Fu, K., Green, M., & Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Information System Security. Barbulescu, R., & Duquesne, S. (2019). Updating key size estimations for pairings. J. Cryptology. Fang, L., Susilo, W., Ge, C., & Wang, J. (2011). Interactive conditional proxy re-encryption with fine. J. Syst. Softw. Fu, K. E. (1999). Group Sharing and Random Access in Cryptographic Storage File Systems. Massachusetts Institute of Technology. Goyal, V. P. (2006). Attribute-based encryption for fine-grained access control of encrypted data. ACM Conference on Computer and Communications Security. ACM. Kaliski, B. (1998). PKCS #7: Cryptographic Message Syntax Version 1.5. RFC 2315. Menezes, A., Sarkar, P., & Singh, S. (2016). Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. MCRYPT. NIST. (2016). SP 800-57. Recommendation for Key Management, Part 1: General (Rev 4). Weng, J., Deng, R. H., Ding, X., Chu, C.-K., & Lai, J. (2009). Conditional proxy re-encryption secure against chosen ciphertext attack. ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009. ACM.

### 4.2. Proof Sketch of IND-CCA2 for Non-Re-Encryptable Ciphertexts Lemma 2. Scheme 1 is IND-CCA2-L2-secure, if I-KP-CPRE is IND-CCA secure and PKE is IND-CCA-secure.

Proof Sketch. The proof for second-level ciphertexts follows the same approach as the proof for first-level ciphertexts. Again, we follow the same sequence of games as before, which replaces private key material of honest and challenge users with random values, remembers these random values (in oracles RKG and

RE), and uses them to place appropriate queries to oracles when necessary (in oracle D). Next, we build an efficient adversary B via the above reduction, which shows that an adversary A winning G1 (i.e., the modified IND-CCA2-L2 game of NI-KP-CPRE) would be able to break IND-CCA2-L2 of I-KP-CPRE. This reduction reuses a simplified version of the oracle implementation from the previous proof of IND-CCA2-L1 for NI-KP-CPRE. As the challenge ciphertext cannot be re-encrypted, various checks become obsolete, e.g., whether a no-longer-used challenge attribute set A [∗] satisfies an input policy.

