Practical Revocation and Key Rotation

Steven Myers and Adam Shull(✉)

Indiana University, Bloomington, IN, USA
{samyers,amshull}@indiana.edu

Abstract. We consider the problems of data maintenance on untrusted clouds. Specifically, two important use cases: (i) using public-key encryption to enforce *dynamic* access control, and (ii) efficient key rotation.

Enabling access revocation is key to enabling dynamic access control, and proxy re-encryption and related technologies have been advocated as tools that allow for revocation on untrusted clouds. Regrettably, the literature assumes that data is encrypted directly with the primitives. Yet, for efficiency reasons hybrid encryption is used, and such schemes are susceptible to key-scraping attacks.

For key rotation, currently deployed schemes have insufficient security properties, or are computationally quite intensive. Proposed systems are either still susceptible to key-scraping attacks, or too inefficient to deploy.

We propose a new notion of security that is practical for both problems. We show how to construct hybrid schemes that are both resistant to key-scraping attacks and highly efficient in revocation or key rotation. The number of modifications to the ciphertext scales linearly with the security parameter and *logarithmically* with the file length.

1 Introduction

Data storage on the cloud is now a major business. Examples include both dynamic storage such as Dropbox, Box, Google Drive, and iCloud and static long term storage such as Amazon's Glacier, and Google's Coldline.

All of the dynamic services provide some degree of sharing and access control that allow one to share files with others, but they all come at the price that all of one's data is either (i) encrypted under a key that the cloud has access to or (ii) placed on the cloud in plaintext. This is necessary because the cloud provider must be able to provide the data to any of its users (as it may be shared), and therefore the cloud acts as an all-trusted reference monitor that decides who can access data. This makes data held by such cloud providers privy to insider and data exfiltration attacks that can put the data of large numbers of users at risk.

In a separate scenario, different regulatory agencies now require that certain sensitive data be encrypted under new keys over regular time intervals, a process dubbed key rotation. With some simplification, such rotation ensures that if keys are leaked, lost, or stolen without concurrent access to the encrypted data, then such keys have a limited useful lifespan: after the data is re-encrypted under a new key, the old key should no longer be useful.

N. P. Smart (Ed.): CT-RSA 2018, LNCS 10808, pp. 157–178, 2018.
https://doi.org/10.1007/978-3-319-76953-0_9

In both scenarios, we'd like the ability to re-encrypt data on the cloud (to revoke access to original recipients and/or provide access to new recipients in one case, and to rotate keys in the other), without trusting the cloud provider with access to the original unencrypted data, and thus not exposing the data's owners to exfiltration or insider attacks. Of course the original owner of the data could provide a newly encrypted copy of the data to the cloud in both cases, but in practice this is typically both expensive and operationally difficult. Similarly, the original ciphertexts can be re-encrypted under a new key on the cloud, but this has new associated costs.

Cryptography seemingly provides natural solutions to untrusted cloud access control; tools such as attribute-based and predicate encryption allow one to store data on a public cloud with cryptography enforcing access control functions. Further, to allow for re-encryption there are often corresponding proxy and delegated re-encrypted versions of these schemes, which would allow the cloud to re-encrypt data under new access schemes or for new recipients without having access to the original data. This theoretically provides solutions for both revocation and key rotation.

However, as detailed by Garrison III et al. [14], these cryptographic techniques are not yet well-suited for even relatively simple dynamic access control policies. Such re-keying is too slow for all but the smallest data, because of the expensive asymmetric operations that are necessary to be applied to the entire message payload. Yet, because changes to access policies can often affect large numbers of files, there is a need for extremely efficient revocation mechanisms. If one deploys hybrid re-encryption for speed gains, then the approach becomes problematic due to key-scraping attacks, where a user stores a large number of symmetric keys in order to maintain access to files even after revocation.

For the scenario of key rotation, there are similar issues. Existing approaches to key rotation include using very expensive asymmetric operations, such as the scheme by Boneh et al. employing key-homomorphic ciphers [7], or completely re-encrypting the data with a symmetric cipher. The technique currently used in constructions by Google and Amazon [1,15] is to use a long-term symmetric key to encrypt data and then encrypt that key under another symmetric key, providing a form of hybrid encryption; for rotation only the latter key is changed while the long-term key remains the same. These constructions have questionable and ill-defined security properties and are susceptible to key-scraping attacks. This latter point was concurrently observed by Everspaugh et al. [13].

Our Contributions. Our observation is that in both scenarios, the assumption should be that the adversary has a prior key to the encrypted material, and possibly some but not all of the original ciphertext (otherwise, an adversary that has both the prior key and full former ciphertext can already decrypt the data). The re-encryption in these scenarios should have the property that security is maintained assuming full access to the re-encrypted ciphertext, but no access to the new key. We propose new CPA and CCA definitions that properly capture this setting, and develop them for traditional and identity-based proxy re-encryption, as well as revocable attribute-based encryption (ABE). We also adapt the notion of UP-IND security for key rotation from [13] and show how it can be strengthened to address adversaries that have partial access to old ciphertexts. Next, we provide a construction that satisfies these definitions and shows exceptional performance. In particular, it only requires modifying a logarithmic number of the ciphertext bits, assuming the adversary can only see a fixed $(1 - \varepsilon)$ fraction of the original ciphertext. Finally, we discuss the implementation details, and show the relative benefits compared to a complete re-encryption with symmetric-key primitives.

Overview of our Construction. Here we present the construction in the case of it being used as a hybrid encryption mode for proxy re-encryption (PRE) schemes; the main ideas are the same for other settings. We make novel use of an All-or-Nothing Transform (AONT) and combine it with traditional ideas from hybrid encryption to produce a hybrid re-encryption process. The re-encrypted ciphertext grows slightly in size by an additive length of one public-key encryption, and thus in practice by several hundred to several thousand bits. However, for the use cases discussed above, storage is typically cheap, and so this ciphertext growth adds a negligible cost.

For those versed in the area, the main idea of our construction is to take a traditional hybrid construction, where a ciphertext consists of an asymmetric PRE encryption of the symmetric key and a symmetric-key encryption of the file in question. We then apply an AONT on top of the symmetric-key ciphertext. To re-encrypt we use the original proxy re-encryption scheme to update the asymmetric encryption to a new asymmetric key, and then pseudorandomly choose a number of locations in the AONT-transformed ciphertext to encrypt. We encrypt enough of the AONT's output that with high probability the adversary has not download some of the newly encrypted locations and thus cannot invert the AONT to decrypt. We then add a new asymmetric encryption of the symmetric key used to choose and encrypt the random bit locations, so that the appropriate decryptor can later invert all the operations and retrieve the appropriate locations. The number of locations to encrypt is roughly (i) proportional to the inverse of the fraction of the file the adversary does not look at, and (ii) proportional to the number of bits that need to be changed by the AONT, which ensures that with overwhelming probability the attacker cannot invert the AONT.

2 Background

Access Revocation. Consider a typical cryptographic access control scenario where a file is encrypted under a public key, and those that have read access are given the secret key. We stress that while in traditional PKI settings, only one person has a given secret key, in cryptographic access control settings this is not necessarily the case. This is further reflected in cryptographic systems more directly related to access control such as attribute-based encryption and predicate encryption, where a given set of credentials or a given access policy can result in multiple users being given the same corresponding key.

Now if a user's access is revoked from a file that is shared amongst many on an untrusted server, the typical cryptographic solution involves providing new secret keys to all users that should continue to have access to the file, and then re-encrypting the file. When the server is not trusted with the plaintext, but can be trusted to perform computation, proxy re-encryption or revocable encryption schemes can be used to re-encrypt the data on the cloud, without requiring a user that has a valid secret-key to retrieve, decrypt and re-encrypt the result. A re-encryption key is generated and sent to the cloud, which updates the ciphertext(s) to the new key.

Key Rotation. Key rotation is the process by which files encrypted and stored must be re-keyed on a timely basis. This ensures that if keys are accidentally leaked or otherwise revealed, the plaintext remains secure, assuming the adversary has not also already obtained a copy of the data encrypted under said key. Key rotation is recommended across a wide range of industries and organizations. For example, NIST [6] recommends regular and planned rotation, as does the Open Web Application Security Project (OWASP) [24], and the payment card industry [25] requires it on a periodic basis for customer data. Google [15] and Amazon [1] now provide partial support for such operations in their long term storage services, so that customers that are mandated to rotate keys can do so. However, as has been noted by Everspaugh et al. [13], the techniques used have questionable and undefined security.

Key-Scraping Attacks. Hybrid proxy re-encryption, revocable encryption, and key rotation schemes are all vulnerable to key-scraping attacks if the key used to encrypt the data itself is not changed during revocation and key rotation. A key-scraping attack occurs when a user—in order to maintain access to files even after a future revocation—downloads and stores a large number of symmetric keys in order to maintain access to files even after revocation.

To make the problem more concrete, consider the following scenario based on Garrison III et al. [14]: Content files are stored on a cloud and are hybrid-encrypted using a hybrid proxy re-encryption scheme with public-key encryption algorithm E and a symmetric-key encryption algorithm $\mathsf{E}^{\mathsf{Sym}}$. Alice has access to a large number of files $\{f_i\}_i$ that are encrypted on the cloud in ciphertexts $\left\{\left(\mathsf{E}(\mathsf{pk}_{Sub_0}, k_i), \mathsf{E}^{\mathsf{Sym}}(k_i, f_i)\right)\right\}_i$. Alice has the secret key, sk_{Sub_0}, corresponding to public key pk_{Sub_0}, as she belongs to an initial group of subscribers, and the subscribers all have access to sk_{Sub_0}, the secret key for this role.[1] She does not have the resources to download all of the content files she has access to. She is removed from the subscriber group, so the cloud proxy re-encrypts all data under a new public-key pk_{Sub_1}, denoting the new group of valid subscribers, and to which Alice does not have the key. The result is that the cloud now serves $\left\{\mathsf{E}(\mathsf{pk}_{Sub_1}, k_i), \mathsf{E}^{\mathsf{Sym}}(k_i, f_i)\right\}_i$, and Alice cannot directly access the content in the subscription service.

[1] We simplify [14] to keep the example simple.

However, while it may not be reasonable to assume that Alice can download all of the files she has access to on the cloud service while she is a subscriber, due to their collective size or rate limits on the outgoing service provider's network connection, it is more reasonable to assume that at some point Alice downloads and decrypts all of the symmetric keys $\{k_i\}_i$. Even for millions of files, this would require less than a gigabyte of storage/bandwidth, and she could use these keys to decrypt all of $\left\{\mathsf{E}^{\mathsf{Sym}}(k_i, f_i)\right\}_i$. Therefore, even if the symmetric keys are re-encrypted via proxy re-encryption, it is reasonable to assume that Alice would maintain the ability to decrypt the symmetric portion of the proxy hybrid re-encrypted files on the cloud. *One needs to ensure with hybrid re-encryption that ciphertexts are re-encrypted on both the public-key and symmetric-key ciphertext portions.*

While one could use the cloud to provide access control against scraping attacks, by for example monitoring a user who accesses the encrypted symmetric-keys portion of too many files, this has several downsides. First, it suggests that access control mechanisms of the cloud can't be circumvented by malicious actors or insiders, which is against the threat model of an untrusted cloud. Further, it implies that the cloud needs to have user accounts, and is aware of and actively records the history of such accesses, and implements access control denial when such occasions occur. The cloud thus monitors which files the users access, which portions, and how frequently, which for privacy, security, and anonymity reasons may be undesirable.

Consider the concrete use case of a subscription content service. With a traditional hybrid encryption scheme a malicious user may be tempted to download symmetric keys for the entire content service—performing a scraping attack—so that all the content could be accessed at a later time after stopping payments. Our proposed scheme would limit the user to the material they could download while paying for the service. Note that a service can easily limit the download rate to prevent mass download attacks without effect on legitimate use. For example, a library might limit downloads to a few tens of books of data a day, and similar a streaming service might limit a user to the equivalent of 24 h of video per day. This doesn't provide much limit on how much of the library content a legitimate user might actually access. However, a key-scraping attack under such a rate-limit would permit access to a significant fraction of all content of the service.

With respect to the scenario of key rotation of data stored on the cloud, our construction's ability to efficiently rotate keys lowers its cost, and this can allow for more efficient and less costly key rotations on large data stores, or alternately may permit for more frequent key rotations due to lowered costs. Other systems, such as those proposed by Boneh et al. [7] that permit updating of symmetric encryptions through key-homomorphisms, also fulfill this function, but their computational costs are significantly more expensive—requiring, for each "block" of the file exponentiations on cyclic groups where the discrete log problem is hard.

3 Related Work

Proxy re-encryption has a significant history (e.g., [4,16,17,22]) that involves the construction of a number of different variants and increasingly stringent security definitions and corresponding constructions in the public-key and identity-based encryption (IBE) spaces. Ateniese et al. [4] also provide a description of a secure file system scheme that uses proxy re-encryption. However, this scheme does not consider what happens when a user's access to a file is revoked.

Related is the notion of revocable encryption schemes. While most such schemes only revoke certificates/keys so that they cannot be used to decrypt ciphertexts encrypted in the future, the ABE scheme of Sahai et al. [28] also provides a mechanism for revoking access to previously encrypted ciphertexts by delegating the ciphertext to a later time. Since this scheme only delegates the ABE portion of the ciphertext and not the symmetric-key-encrypted portion, this scheme is susceptible to key-scraping attacks.

Watanabe and Yoshino [30] present a mechanism for efficiently updating symmetric keys. They also use an AONT to improve efficiency. However, their scheme is in the symmetric key setting, and it does not consider revocation, where the adversary previously had legitimate access to the file.

Li et al. [19] present a rekeying mechanism for encrypted deduplication storage and recognize its benefits for dynamic access control on the cloud and key rotation, but provide no formal analysis of security, and essentially note that their construction is susceptible to the key-scraping attack we describe and prevent.

Boneh et al. [7] show how to use key-homomorphic pseudorandom functions to implement symmetric-key proxy re-encryption, and address its use in key rotation. However, current constructions of key-homomorphic PRFs are far too inefficient to be used in practice, and their constructions would require asymmetric operations that scale directly with the length of the file being encrypted.

Everspaugh et al. [13] look at the issue of key rotations on untrusted clouds. They cite the problematic approaches being applied, and consider either simple solutions that are still susceptible to key-scraping attacks or solutions based on Boneh et al.'s [7] previously mentioned approach with its corresponding drawbacks.

Independently from our work, Bacis et al. [5] presented a technique for symmetric-key revocation similar to our approach of applying an AONT to the symmetric-key ciphertext and then re-encrypting only a small portion of it. Instead of using an AONT, they similarly apply the AES block cipher multiple times to different combinations of the bits of a ciphertext—or a portion of a ciphertext called a "macro-block"—to ensure that each bit affects every other bit of the macro-block. Their work differs from ours in several keys respects: (i) They provide no formal notion of security, and thus no formal argument of what security is achieved; but their security notion, for example, seems to presume the adversary has no knowledge of the underlying plaintext, and relatively high success rates of decryption with access to as little as 50% of the original ciphertext. (ii) The number of times the AES block cipher needs to be applied to encrypt or decrypt a file of length n in their scheme grows as $O(n \log n)$ whereas our scheme grows as $O(n)$ with applications of AES and SHA primitives. Our scheme only applies a symmetric-key encryption once and an AONT once to the file, regardless of its size. (iii) Lastly, we show how to incorporate our construction with public-key primitives, whereas their construction is solely symmetric-key.

4 Notation and Background Definitions

Given a string s over a given alphabet, we denote by $|s|$ the length of the string. A function μ is negligible if it grows slower than any inverse polynomial. Let $\mathcal{D}_1 = \{D_{1,i}\}_{i \in \mathbb{N}}$ and $\mathcal{D}_2 = \{D_{2,i}\}_{i \in \mathbb{N}}$ be two indexed sequences of distributions, then $\mathcal{D}_1 \approx \mathcal{D}_2$ denotes that the two sequences are computationally indistinguishable [18]. Let $[N]$ denote $\{1, \ldots, N\}$ and let $\left\{ {N \atop \ell} \right\}$ be the set of all ℓ-element subsets of $[N]$. For $y \in \{0,1\}^N$ and $L \in \left\{ {N \atop \ell} \right\}$, we use $[y]_L$ to denote the $N - \ell$ bits of y that are not in L. For a string t, let $t[j]$ represent the jth bit of s.

Let $\mathsf{Ind}(s, \ell^*)$ be a deterministic function that takes a seed s and produces a pseudorandom element of $\left\{ {N \atop \ell^*} \right\}$, i.e. a pseudorandom subset of $\{1, \ldots, N\}$ of size ℓ^*. Let $\mathsf{Ctr}(k, \ell^*)$ denote the keystream of length ℓ^* produced by a pseudorandom generator. Our notation envisions using counter mode encryption with key k and nonce 0, which is a known PRG. Note that if the underlying block cipher is secure, then $\mathsf{Ctr}(k, \ell^*)$ is pseudorandom.

Let $\mathsf{rInd}(\ell^*)$ denote a random element of $\left\{ {N \atop \ell^*} \right\}$, i.e. a random subset of $\{1, \ldots, N\}$ of size ℓ^*; and let $\mathsf{rStr}(\ell^*)$ be a random string of length ℓ^*. Let $[t]_{\mathsf{ind},\mathsf{str}}$ denote string t with the values of the bit positions specified by the indices in ind XORed with string str. For example, $T(x)_{\mathsf{ind}=\{1,3,4\},\mathsf{str}=101}$ would output $t[1] \oplus 1, t[2], t[3] \oplus 0, t[4] \oplus 1, t[5], \ldots$.

All-Or-Nothing Transforms

All-or-nothing transforms were introduced by Rivest [27] as a primitive function that has the property that without access to nearly the entire output, no party could retrieve any bit of the underlying input; but with the entire output the input is easily retrievable. The notion was formalized by Boyko [8] and Canetti et al. [9] in the random oracle and standard models respectively, with security against adaptive adversaries defined by Dodis et al. [12].

Definition 1 (Adaptive AONT [12]). *A randomized polynomial time computable function $T : \{0,1\}^n \to \{0,1\}^N$ is an adaptive ℓ-AONT if it satisfies the following conditions:*

1. *T is efficiently invertible, i.e., there is a polynomial time machine I such that for any $x \in \{0,1\}^n$ and any $y \leftarrow T(x)$, we have $I(y) = x$.*
2. *For any $x_0, x_1 \in \{0,1\}^n$ and any PPT adversary $\mathcal{A}$ with oracle access to string $y = T(x_b)$ who can read at most $N - \ell$ bits of y, we have:*

$$\left| \Pr\left[\mathcal{A}^{T(x_0)}(x_0, x_1) = 1 \right] - \Pr\left[\mathcal{A}^{T(x_1)}(x_0, x_1) = 1 \right] \right| \leq \varepsilon(N)$$

for some negligible function ε.

Construction of AONTs. Boyko [8] showed that Optimal Asymmetric Encryption Padding (OAEP) satisfies a non-adaptive version of Definition 1 in the random oracle model. Extending the work of Canetti et al. [9] and Dodis et al. [12] we show that OAEP is also an adaptively secure AONT in the random oracle model. A proof for the following lemma is given in the full version [23].

Lemma 1. *Let $G : \{0,1\}^k \to \{0,1\}^n$, and $H : \{0,1\}^n \to \{0,1\}^k$ be random oracles. Define the probablistic function $f_{\mathrm{OAEP}} : \{0,1\}^n \to \{0,1\}^{n+k}$ as $f_{\mathrm{OAEP}}(x;r) = \langle G(r) \oplus x, H(G(r) \oplus x) \oplus r \rangle$, where $r \in_R \{0,1\}^k$. Let $\ell \leq k$, then f_{OAEP} is an adaptive 2ℓ-AONT, with security $q/2^{\ell-2}$ for an adversary that makes at most $q < 2^{\ell-1}$ adaptive queries to G or H.*

5 Updatable Encryption

We present a symmetric encryption mode with security properties that are stronger than those presented by Everspaugh et al. [13] in their UP-IND definition, but weaker than those presented in the UP-REENC definition. However, we get performance only slightly slower than known UP-IND constructions, and orders of magnitude faster performance than known UP-REENC constructions. Thus, we believe our construction has significant practical value for increasing deployed security in key-rotation settings.

5.1 Updatable Encryption Definition

To achieve key rotation, we borrow the notion of updatable encryption from [7,13]. This notion envisions KEM/DEM-type construction, where all the keys are symmetric keys. We use $\{\mathsf{sk}_i\}$ to denote the KEM keys that will be rotated, while $\{k_i\}$ will denote the DEM keys that may or may not be updated. Note, however, that these keys all come from the same symmetric-key encryption scheme and are identically distributed.

Definition 2 (Updatable Encryption). *An updatable encryption scheme Π_{upd} consists of five probabilistic polynomial time algorithms:*

$\mathsf{G}^{\mathsf{Upd}}(1^\lambda) \to (\mathsf{sk})$: *Key generation*
$\mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}, M) \to C = (\tilde{C}, \overline{C})$: *Symmetric hybrid encryption*
$\mathsf{D}^{\mathsf{Upd}}(\mathsf{sk}, C) \to M$: *Decryption, returns the underlying message or $\perp$.*
$\mathsf{RG}^{\mathsf{Upd}}(\mathsf{sk}_i, \mathsf{sk}_j, \tilde{C}) \to \Delta_{i,j,\tilde{C}}$: *Creates a re-encryption token that can transform a ciphertext encrypted under sk_i with header $\tilde{C}$ to a ciphertext encrypted under sk_j.*
$\mathsf{RE}^{\mathsf{Upd}}(\Delta_{i,j,\tilde{C}_i}, (\tilde{C}_i, \overline{C}_i)) \to C_j$: *Takes a re-encryption token $\Delta_{i,j,\tilde{C}_i}$ and a ciphertext encrypted under sk_i with header $\tilde{C}$, and translates it into a ciphertext encrypted under sk_j. $\mathsf{RE}^{\mathsf{Upd}}$ is required to be deterministic, as this simplifies the security definition.*

Correctness. For every message M and sequence of keys $\{\mathsf{sk}_u \leftarrow \mathsf{G}^{\mathsf{Upd}}(1^\lambda)\}_{u \in \{0,\ldots,r\}}$, let $C_0 = (\tilde{C}_0, \overline{C}_0) = \mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}_0, M)$. For $0 \leq u \leq r-1$ let $C_{u+1} = \mathsf{RE}^{\mathsf{Upd}}(\mathsf{RG}^{\mathsf{Upd}}(\mathsf{sk}_u, \mathsf{sk}_{u+1}, \tilde{C}_u), C_u)$. Then $\mathsf{D}^{\mathsf{Upd}}(\mathsf{sk}_r, C_r) = M$.

5.2 UP-IND Security for Updatable Encryption

We borrow the updatable encryption indistinguishability ($\mathsf{UP\text{-}IND}$ security) definition from Everspaugh et al. [13][2]. We make one small change to make our proofs easier to present: The adversary makes one query to the challenge oracle instead of a polynomial number of queries to a left-or-right oracle. Standard techniques show these equivalent up to a factor in the number of queries made to the left-or-right oracle.

Definition 3 (UP-IND Security Game). *The security game is given in Fig. 1 (p. 12). λ is the security parameter. Let adv. $\mathcal{A}$ be a poly-time oracle TM. The game creates $t+\kappa$ secret-keys: $t \geq 1$ uncorrupted and $\kappa \geq 0$ corrupted that are given to $\mathcal{A}$. The oracles are defined as follows:*

- **Encryption** $\mathcal{O}_{\mathsf{enc}}(i, M)$*: Output* $\mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}_i, M)$*.*
- **Re-Encryption Key Generation** $\mathcal{O}_{\mathsf{rkey}}(i, j, \tilde{C})$*: If j is corrupted and $(i, \tilde{C})$ is a challenge derivative, output $\perp$. Otherwise, output* $\Delta_{i,j,\tilde{C}} \leftarrow \mathsf{RG}^{\mathsf{Upd}}(\mathsf{sk}_i, \mathsf{sk}_j, \tilde{C})$*.*
- **Re-Encryption** $\mathcal{O}_{\mathsf{renc}}(i, j, (\tilde{C}, \overline{C}))$*: Compute* $\Delta_{i,j,\tilde{C}} \leftarrow \mathsf{RG}^{\mathsf{Upd}}(\mathsf{sk}_i, \mathsf{sk}_j, \tilde{C})$ *and* $C' = (\tilde{C}', \overline{C}') \leftarrow \mathsf{RE}^{\mathsf{Upd}}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$*. If j is corrupted and $(i, \tilde{C})$ is a challenge derivative, then output $\tilde{C}'$. Otherwise, output C'*
- **Challenge** $\mathcal{O}_{\mathsf{chal}}(M_0, M_1, i^*)$*: If i^* is corrupted, then output $\perp$. Otherwise, output* $C^* \leftarrow \mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}_{i^*}, M_b)$*.* The oracle can only be called once.

Define the concept of a challenge derivative (i, C) as follows:

- *$(i^*, \tilde{C}^*)$ is a challenge derivative if the challenge query was asked on secret-key index i^* and the response was $C^* = (\tilde{C}^*, \overline{C}^*)$.*
- *If $(i, \tilde{C})$ is a challenge derivative, and $\mathcal{A}$ has queried $\mathcal{O}_{\mathsf{renc}}(i, j, (\tilde{C}, \overline{C}))$ and received header $\tilde{C}'$ in response, then $(j, \tilde{C}')$ is a challenge derivative.*
- *If $(i, \tilde{C})$ is a challenge derivative, and $\mathcal{A}$ has queried $\mathcal{O}_{\mathsf{rkey}}(i, j, \tilde{C})$ and received $\Delta_{i,j,\tilde{C}}$ in response, then the header of $(j, \mathsf{RE}^{\mathsf{Upd}}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$ is a challenge derivative.*

Definition 4. *A updatable encryption scheme Π_{upd} is $\mathsf{UP\text{-}IND}$-secure if for all oracle PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that:*

$$\Pr[\mathsf{UP\text{-}IND}_{\mathcal{A},\Pi_{upd}}(1^\lambda, t, \kappa) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

5.3 $(1-\varepsilon)$-Exfiltration UP-IND Security for Updatable Encryption

We now provide our stronger definition, which demonstrates that an adversary that has a compromised key will be unable to break a key-rotated ciphertext unless it previously downloaded at least a $1-\varepsilon$ fraction of the former ciphertext before rotation.

[2] Everspaugh et al. [13] presents a security notion $\mathsf{UP\text{-}INT}$ that ensures integrity. However, at CRYPTO 2017 they noted a flaw in their constructions. Thus our scheme's improvement on their KSS scheme will also not have $\mathsf{UP\text{-}INT}$ security.

Definition 5 ($(1-\varepsilon)$-Exfiltration UP-IND Security Game). *We define game* $(1-\varepsilon)\text{-}\mathsf{Exfil\text{-}UP\text{-}IND}_{A,\Pi_{upd}}(1^\lambda)$ *as being identical to* $\mathsf{UP\text{-}IND}_{\mathcal{A},\Pi_{upd}}(1^\lambda)$ *except that the challenge oracle is called as* $\mathcal{O}_{\mathsf{chal}}(M_0, M_1, [i_0^*, \ldots, i_r^*], j^*, \mathsf{bitPos})$ *and works as follows:*

The adversary can call $(M_0,\ M_1,\ [i_0^*, \ldots, i_r^*],\ j^*,\ \mathsf{bitPos})$ *for* any *values* $[i_0^*, \ldots, i_r^*]$ *such that* $i_u^* \neq i_{u+1}^*$ *for* $0 \leq u \leq r-1$*. These values represent the keys, prior to the current key, through which the challenge ciphertext is updated. These keys may be corrupted, to model the fact that an adversary may have obtained the old keys. However, now* j^* *must be an uncorrupted index distinct from* i_r^**. The input* bitPos *will be used to indicate the bits of ciphertexts created prior to key rotation that the adversary receives. The challenger computes* $\{C_u^* = (\tilde{C}_u^*, \overline{C}_u^*)\}_{0 \leq u \leq r}$ *where* $C_0^* = \mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}_{i_0^*}, M_b)$*, and for* $u > 0$*,*

$$\left(\tilde{C}_u^*, \overline{C}_u^*\right) = \mathsf{RE}^{\mathsf{Upd}}\left(\mathsf{RG}^{\mathsf{Upd}}\left(\mathsf{sk}_{i_{u-1}^*}, \mathsf{sk}_{i_u^*}, \tilde{C}_u^*\right), \left(\tilde{C}_{u-1}^*, \overline{C}_{u-1}^*\right)\right).$$

The challenger also computes

$$C^{**} = \left(\tilde{C}^{**}, \overline{C}^{**}\right) = \mathsf{RE}^{\mathsf{Upd}}\left(\mathsf{RG}^{\mathsf{Upd}}\left(\mathsf{sk}_{i_r^*}, \mathsf{sk}_{j^*}, \tilde{C}_r^*\right), \left(\tilde{C}_r^*, \overline{C}_r^*\right)\right).$$

Here each C_u^* *represents a ciphertext before key rotation and* C^{**} *represents the ciphertext after key rotation. Let* $N' = \min_{0 \leq u \leq r} |C_u^*|$*.*

In this definition, only derivatives of $(j^{**}, \tilde{C}^{**})$*—not* $(i_0^*, \tilde{C}_0^*)$ *through* $(i_r^*, \tilde{C}_r^*)$*—are considered challenge derivatives for purposes of the* $\mathcal{O}_{\mathsf{rkey}}$ *and* $\mathcal{O}_{\mathsf{renc}}$ *oracles.*

The challenge oracle is stateful. The adversary selects bitPos *one pair* (u, v) *at a time and receives the* v*th bit of ciphertext* C_u^**, so it can choose each pair based on the previous bits it received. Once the adversary has received* $(1-\varepsilon)N'$ *total bits of* $\{C_u^*\}_{0 \leq u \leq r}$*, the oracle outputs* C^{**}*. After this it refuses to respond. Similarly, the oracle refuses to respond if queries change any of the calling values other than* bitPos*.*

Note that it is possible to be secure in the previous game without actually achieving $\mathsf{UP\text{-}IND}$ security, so the definition of security requires both notions.

Definition 6. *An updatable encryption scheme* Π_{upd} *is* $(1-\varepsilon)$*-Exfiltration* $\mathsf{UP\text{-}IND}$*-Secure if for all oracle PPT adversaries* $\mathcal{A}$*, there exists a negligible function* negl *s.t. both hold:*

1. $\Pr[(1-\varepsilon)\text{-}\mathsf{Exfil\text{-}UP\text{-}IND}_{\mathcal{A},\Pi_{upd}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$
2. $\Pr[\mathsf{UP\text{-}IND}_{\mathcal{A},\Pi_{upd}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

5.4 Construction

The basis of our construction is the KSS updatable authenticated encryption scheme of Everspaugh et al. [13], based on a symmetric encryption primitive $\Pi_{sym} = (\mathsf{G}^{\mathsf{Sym}}, \mathsf{E}^{\mathsf{Sym}}, \mathsf{D}^{\mathsf{Sym}})$. This scheme uses a key encapsulation mechanism

(KEM) and a data encapsulation mechanism (DEM), both based on a symmetric authenticated encryption scheme. In this scheme, the KEM key is updated while the DEM key is not. The ciphertext header contains a share of the DEM key encrypted under the KEM key. The ciphertext body contains the other share of the DEM key and the message encrypted under the DEM key. When the ciphertext is updated, the DEM key is split into new shares, and the new ciphertext header is encrypted under the new KEM key. Note that the KSS scheme also includes an encrypted hash of the message in the header—designed to ensure integrity—but we exclude it from our scheme because it is insufficient to provide integrity and is not needed for our security definitions.

In addition to all this, our scheme applies an AONT, T, to the encrypted message. An initial (never updated) ciphertext has the form $(\tilde{C}, (y, C^T))$, where the components are:

- $\tilde{C} = \mathsf{E}^{\mathsf{Sym}}(\mathsf{sk}, \chi)$ is an encryption under the KEM key sk of a share χ of the DEM key x.
- y is the other share of the DEM key x.
- $C^T = T\left(\mathsf{E}^{\mathsf{Sym}}(x, M)\right)$ is the AONT applied to the encryption under the DEM key x of the message M.

When the ciphertext is updated, the same actions are taken as in the KSS scheme. Additionally, the updater re-encrypts a randomly selected set of bits of C^T, on top of any previous re-encryptions of bits of C^T. To allow decryption, the locations of the re-encrypted bits and the key used to encrypt them are also stored in the ciphertext header. As a result, the ciphertext header will grow linearly each time the ciphertext is updated; however, the header size remains independent of the length of the message.

A ciphertext updated r times has the form $(\tilde{C}, (y, C^T))$, where the components are:

- $\tilde{C} = \mathsf{E}^{\mathsf{Sym}}(\mathsf{sk}, (\chi, (s_1, k_1), \ldots, (s_r, k_r))$ is an encryption under the KEM key sk of a share χ of the DEM key x and all the seeds and keys used to re-encrypt bits of C^T.
- y is the other share of the DEM key x.
- $C^T = T\left(\mathsf{E}^{\mathsf{Sym}}(x, M)\right)$ is the AONT applied to the encryption under the DEM key x of the message M, with bits re-encrypted as specified by $(s_1, k_1), \ldots, (s_r, k_r)$.

5.5 Updatable Encryption Scheme

We now give the formal description of our $(1-\varepsilon)$-Exfil-UP-IND-secure updatable encryption scheme $\Pi_{upd} = \left(\mathsf{G}^{\mathsf{Upd}}, \mathsf{E}^{\mathsf{Upd}}, \mathsf{RG}^{\mathsf{Upd}}, \mathsf{RE}^{\mathsf{Upd}}, \mathsf{D}^{\mathsf{Upd}}\right)$. Let N be the output length of T, and let $\ell^* \leq N$ with $\ell^* = \omega(\log(\lambda))$ be the number of bits of the AONT output that are re-encrypted. The value of ℓ^* will depend on the security of the AONT and how small ε is (i.e., how much of the file we assume the adversary will download). $\mathsf{G}^{\mathsf{Upd}}\left(1^\lambda\right) = \mathsf{G}^{\mathsf{Sym}}\left(1^\lambda\right)$, with the remaining algorithms defined in Fig. 1.

```
$\mathsf{UP\text{-}IND}_{\mathcal{A},\Pi_{upd}}(1^\lambda, t, \kappa)$

$b \leftarrow \{0,1\}$
$\mathsf{sk}_1, \ldots, \mathsf{sk}_{t+\kappa} \leftarrow \mathsf{G}^{\mathsf{Upd}}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{enc}}, \mathcal{O}_{\mathsf{rkey}}, \mathcal{O}_{\mathsf{renc}}, \mathcal{O}_{\mathsf{chal}}}(\mathsf{sk}_{t+1}, \ldots, \mathsf{sk}_{t+\kappa})$
Output 1 iff $b = b'$
```

```
$\mathsf{E}^{\mathsf{Upd}}(\mathsf{sk}, M)$

$x, y \leftarrow \mathsf{G}^{\mathsf{Sym}}(1^\lambda)$
$\chi = x \oplus y$
$C^T \leftarrow T\big(\mathsf{E}^{\mathsf{Sym}}(x, M)\big)$
$\tilde{C} \leftarrow \mathsf{E}^{\mathsf{Sym}}(\mathsf{sk}, \chi)$
return $C = \Big(\tilde{C}, \big(y, C^T\big)\Big)$
```

```
$\mathsf{RG}^{\mathsf{Upd}}(\mathsf{sk}_i, \mathsf{sk}_j, \tilde{C})$

$(\chi, (s_1, k_1), \ldots, (s_r, k_r)) \leftarrow \mathsf{D}^{\mathsf{Sym}}(\mathsf{sk}_i, \tilde{C})$
if $(\chi, (s_1, k_1), \ldots, (s_r, k_r)) = \perp$ then
    return $\perp$
$y', k_{r+1} \leftarrow \mathsf{G}^{\mathsf{Sym}}(1^\lambda)$
Choose seed $s_{r+1}$ uniformly
return $\Delta_{i,j,\tilde{C}} = (y', s_{r+1}, k_{r+1},$
  $\mathsf{E}^{\mathsf{Sym}}(\mathsf{sk}_j, (\chi \oplus y', (s_1, k_1), \ldots, (s_{r+1}, k_{r+1}))))$
```

```
$\mathsf{RE}^{\mathsf{Upd}}\Big(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C})\Big)$

$(y', s_{r+1}, k_{r+1}, \tilde{C}') = \Delta_{i,j,\tilde{C}}$
$(y, C_r^T) = \overline{C}$
$\mathsf{ind}_{r+1} \leftarrow \mathsf{Ind}(s_{r+1}, \ell^*)$
$\mathsf{str}_{r+1} \leftarrow \mathsf{Ctr}(k_{r+1}, \ell^*)$
$w = 0$
for $v \leftarrow 1, \ldots, N$ do
    if $v \in \mathsf{ind}_{r+1}$ then
        $C_{r+1}^T[v] \leftarrow C_r^T[v] \oplus \mathsf{str}_{r+1}[w]$
        $w \leftarrow w + 1$
    else
        $C_{r+1}^T[v] \leftarrow C_r^T[v]$
return $\Big(\tilde{C}, \overline{C} = \big(y + y', C_{r+1}^T\big)\Big)$
```

```
$\mathsf{D}^{\mathsf{Upd}}(\mathsf{sk}, (\tilde{C}, \overline{C}))$

$(\chi, (s_1, k_1), \ldots, (s_r, k_r)) \leftarrow \mathsf{D}^{\mathsf{Sym}}(\mathsf{sk}, \tilde{C})$
if $(\chi, (s_1, k_1), \ldots, (s_r, k_r)) = \perp$ then
    return $\perp$
$(y, C_r^T) = \overline{C}$
for $u \leftarrow r, \ldots, 1$ do
    $\mathsf{ind}_u, \mathsf{str}_u \leftarrow \mathsf{Ind}(s_u, \ell^*), \mathsf{Ctr}(k_u, \ell^*)$
    $w = 0$
    for $v \leftarrow 1, \ldots, N$ do
        if $v \in \mathsf{ind}_u$ then
            $C_{u-1}^T[v] \leftarrow C_u^T[v] \oplus \mathsf{str}_u[w]$
            $w \leftarrow w + 1$
        else
            $C_{u-1}^T[v] \leftarrow C_u^T[v]$
$(x, M) \leftarrow (\chi \oplus y, \mathsf{D}^{\mathsf{Sym}}(x, T^{-1}(C_0^T)))$
return $M$
```

Fig. 1. UP-IND security experiment and Π_{upd} algorithms

5.6 Security of Our Scheme

Since our scheme Π_{upd} is essentially the KSS scheme from [13] with the AONT added on top, the proof of UP-IND security of KSS in Theorem 6 of [13] also applies to Π_{upd}. Note that the proof of UP-IND security only requires the underlying scheme to be IND-CPA-secure encryption, not full authenticated encryption. Thus we have:

Theorem 1. *Assume the existence of an* IND-CPA*-secure symmetric-key encryption scheme* $\Pi_{sym} = \left(\mathsf{G}^{\mathsf{Sym}}, \mathsf{E}^{\mathsf{Sym}}, \mathsf{D}^{\mathsf{Sym}}\right)$ *and an all-or-nothing transform* T*. Then the construction of* Π_{upd} *in Sect. 5.5 is* UP-IND*-secure.*

The following theorem claims that our scheme Π_{upd} also has $(1-\varepsilon)$-Exfil-UP-IND-security, meaning it satisfies Definition 6.

Theorem 2. *Assume the existence of an* IND-CPA*-secure symmetric-key encryption scheme* $\Pi_{sym} = \left(\mathsf{G}^{\mathsf{Sym}}, \mathsf{E}^{\mathsf{Sym}}, \mathsf{D}^{\mathsf{Sym}}\right)$ *and an adaptive* ℓ*-AONT* T*. Suppose that for the construction of* Π_{upd} *from Sect. 5.5,* C^T *comprises at least a fraction* $1-\delta$ *of the total size of each ciphertext. Then for any* $\varepsilon < 1$ *with* $\varepsilon > \delta$ *and any* $\ell^* > \frac{\ell}{\varepsilon-\delta}$*, this construction is* $(1-\varepsilon)$-Exfil-UP-IND*-secure.*

We provide a brief sketch that discusses the important ideas.

Proof (very brief sketch — full proof to appear in an upcoming paper on the Cryptology ePrint Archive). We consider a series of hybrid games that remove the challenge ciphertext's dependence on any encapsulated keys that are encrypted with uncorrupted secret keys in the experiment. The IND-CPA security of the underlying scheme enables this. In several other games we then exchange the pseudo-random subsets of encrypted bits in the challenge ciphertext's AONT with completely random subsets, encrypted with a one-time-pad. Finally, we argue that with overwhelming probability that a logarithmic number of encrypted bits were in the ε fraction of T's output. Therefore, the adversary is without knowledge of these bits of T's output, and cannot invert T by the security of the AONT.

6 CPA-Secure Hybrid Public-Key Proxy Re-Encryption Scheme

In this section we show how a public-key proxy re-encryption scheme can be updated with a similar hybrid encryption scheme as depicted in the last section for updatable encryption. This update allows for efficient revocation of ciphertext access privileges in dynamic access control schemes, as well as fast key rotation for files that are stored with a public- and symmetric-key hybrid encryption scheme.

We begin with a unidirectional multi-hop proxy re-encryption (PRE) public-key encryption scheme, such as the one described in [26]. Unidirectionality implies one cannot use a re-encryption key to go backwards (i.e., you cannot produce $r_{j\to i}$ given $r_{i\to j}$), and multi-hop means that the re-encryption scheme can be applied an unlimited number of times. Our results apply to bidirectional and/or single-hop schemes as well, with the resulting scheme inheriting the properties of the underlying PRE scheme, but for our application the selected properties seem most appropriate.

Definition 7 (Public-Key Proxy Re-Encryption). *A proxy public-key re-encryption scheme Π consists of five probabilistic polynomial time algorithms, the first three of which form a standard public-key encryption primitive (i)* $\mathsf{G}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$ *(key generation); (ii)* $\mathsf{E}(\mathsf{pk}, M) \rightarrow C$ *(public-key encryption); and (iii)* $\mathsf{D}(\mathsf{sk}, C) \rightarrow M$ *(decryption). The last two are: (iv)* $\mathsf{RG}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j, \mathsf{sk}_j) \rightarrow \mathsf{rk}_{i \rightarrow j}$ *(generating re-keying keys), which takes a source, i, and destination key, j, pair and creates a re-encryption key; and (v)* $\mathsf{RE}(\mathsf{rk}_{i \rightarrow j}, C_i) \rightarrow C_j$ *(re-encryption), which takes a re-encryption key and a ciphertext, and produces a re-encryption of it under the destination key j.*

Correctness. For every message M, set of key pairs $\{(\mathsf{pk}_{i_u}, \mathsf{sk}_{i_u}) \leftarrow \mathsf{G}\}_{u \in \{0,\ldots,r\}}$, and set of re-encryption keys $\{\mathsf{rk}_{i_u \rightarrow i_{u+1}} \leftarrow \mathsf{RG}(\mathsf{pk}_{i_u}, \mathsf{sk}_{i_u}, \mathsf{pk}_{i_{u+1}}, \mathsf{sk}_{i_{u+1}})\}_{u \in \{0,\ldots,r-1\}}$, we have $\mathsf{D}\left(\mathsf{sk}_{i_r}, \mathsf{RE}\left(\mathsf{rk}_{i_{r-1} \rightarrow i_r}, \ldots \mathsf{RE}\left(\mathsf{rk}_{i_0 \rightarrow i_1}, \mathsf{E}\left(\mathsf{pk}_{i_0}, M\right)\right) \ldots\right)\right) = M$.

6.1 PRE-CPA-Security (Unidirectional and Multi-Hop)

The security game allows the adversary to query public keys for which it will get the corresponding secret key—in which case we say that the index of the public key is corrupted—and public keys for which it will not get the secret key—in which case the index is uncorrupted. The challenge ciphertext must be encrypted under a key with an uncorrupted index. The adversary can query any re-encryption or re-encryption key that does not go from an uncorrupted to a corrupted index.

Definition 8 (PRE-CPA-Security Game [3]). *Let λ be the security parameter. Let adversary $\mathcal{A}(\lambda)$ be a poly-time oracle TM. The* PRE-CPA *game consists of an execution of $\mathcal{A}$ in two phases, as described in Fig. 2 (p. 16). Within each phase, $\mathcal{A}$ has access to oracles (described below) that can be queried in any order arbitrarily many times unless otherwise specified.*

Phase 1: *There are two oracles. On the ith query to either of the oracles, we compute* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{G}$ *and then depending on the query:*
Uncorrupted Key Generation $\mathcal{O}_{\mathsf{ukey}}$*: Output* pk_i*; note i is uncorrupted.*
Corrupted Key Generation $\mathcal{O}_{\mathsf{ckey}}$*: Output* $(\mathsf{pk}_i, \mathsf{sk}_i)$*; note i is corrupted.*

Phase 2: *There are oracles producing re-encryption keys and re-encryptions of ciphertexts, as well as the challenge oracle. Note that the indices correspond to those of the keys produced in Phase 1.*
Re-Encryption Key Generation $\mathcal{O}_{\mathsf{rkey}}(i, j)$*: If $i = j$, or if i is uncorrupted and j is corrupted, then output $\perp$. Otherwise, output* $\mathsf{rk}_{i \rightarrow j} \leftarrow \mathsf{RG}(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j, \mathsf{sk}_j)$.
Re-Encryption $\mathcal{O}_{\mathsf{renc}}(i, j, C)$*: If $i = j$, or if i is uncorrupted and j is corrupted, then output $\perp$. Otherwise, output* $\mathsf{RE}(\mathsf{rk}_{i \rightarrow j}, C)$ *where* $\mathsf{rk}_{i \rightarrow j} \leftarrow \mathsf{RG}(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{pk}_j)$.
Challenge $\mathcal{O}_{\mathsf{chal}}(M_0, M_1, i^*)$*: If i^* is corrupted, output $\perp$. Otherwise, output* $C^* \leftarrow \mathsf{E}(\mathsf{pk}_{i^*}, M_b)$*. The oracle can only be called once.*

Definition 9. *A Proxy Re-Encryption scheme Π is Unidirectional, Multi-Hop, PRE CPA-Secure if for all oracle PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *such that:* $\Pr[\mathsf{PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

6.2 $(1-\varepsilon)$-Revocable PRE-CPA-Security

We modify the above security definition of traditional PRE security to incorporate abilities that adversaries have in practice in the revocation and re-keying scenarios: initial access to files and their decryption keys, but a lack of inclination or capability to download all of these files. In particular, they may download the symmetric keys used in a file's hybrid encryption. The goal is now that after a file is re-encrypted the adversary cannot, at this point, decrypt the ciphertext. The new definition modifies Definition 8 similarly to how Definition 5 modifies Definition 6.

Definition 10 ($(1-\varepsilon)$-Revocable PRE-CPA Security Game). *Security game* $(1-\varepsilon)$-$\mathsf{Revoke\text{-}PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda)$ *is identical to* $\mathsf{PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda)$ *given in Definition 8 except that the challenge oracle is called as* $\mathcal{O}_{\mathsf{chal}}(M_0, M_1, [i_0^*, \ldots, i_r^*], j^*, \mathsf{bitPos})$ *and works as follows:*

The adversary can call $(M_0, M_1, [i_0^*, \ldots, i_r^*], j^*, \mathsf{bitPos})$ *for* any *values* $[i_0^*, \ldots, i_r^*]$ *such that* $i_u^* \neq i_{u+1}^*$ *for* $0 \leq u \leq r-1$. *However,* j^* *must be an uncorrupted index distinct from* i_r^*. *The input* bitPos *will be used to indicate the bits of ciphertexts created prior to revocation that the adversary receives. The challenger computes the following:*

- $\{C_u^*\}_{0 \leq u \leq r}$ *where* $C_0^* = \mathsf{E}(\mathsf{pk}_{i_0^*}, M_b)$, *and for* $u > 0$, $C_u^* = \mathsf{RE}(\mathsf{rk}_{i_{u-1}^* \to i_u^*}, C_{u-1}^*)$
- $C^{**} = \mathsf{RE}(\mathsf{rk}_{i_r^* \to j^*}, C^*)$ *for* $\mathsf{rk}_{i_r^* \to j^*} = \mathsf{RG}(\mathsf{pk}_{i_r^*}, \mathsf{sk}_{i_r^*}, \mathsf{pk}_{j^*}, \mathsf{sk}_{j^*})$

Here each C_u^* *represents a ciphertext before revocation and* C^{**} *represents the ciphertext after revocation. Let* $N' = \min_{0 \leq u \leq r} |C_u^*|$.

The challenge oracle is stateful. The adversary selects bitPos *one pair* (u, v) *at a time and receives the* v*th bit of ciphertext* C_u^*, *so it can choose each pair based on the previous bits it received. Once the adversary has received* $(1-\varepsilon)N'$ *total bits of* $\{C_u^*\}_{0 \leq u \leq r}$, *the oracle outputs* C^{**}. *After this it refuses to respond. Similarly, the oracle refuses to respond if queries change any of the calling values other than* bitPos. *In the* static *game all* $(1-\varepsilon)N'$ *queries are made in parallel.*

Definition 11. *A proxy re-encryption scheme Π is $(1-\varepsilon)$-Revocable-PRE-CPA-Secure if for all oracle PPT adversaries $\mathcal{A}$, there exists a negligible function* negl *s.t.:*

1. $\Pr[(1-\varepsilon)\text{-}\mathsf{Revoke\text{-}PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$, *and*
2. $\Pr[\mathsf{PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Note the scheme needs to satisfy both the traditional and revocable definitions (Definitions 10 and 11), as it is possible to construct revocation schemes that produces secure re-keyed ciphertexts, but where the originals are insecure.

6.3 Proxy Re-Encryption Construction

The basis of our construction is a standard hybrid encryption scheme with an AONT applied to the symmetric ciphertext portion of the hybrid ciphertext. That is, an initial ciphertext has the form $\big(C^{pk} = \mathsf{E}(\mathsf{pk}, k_0),$ $C^T = T\left(\mathsf{E}^{\mathsf{Sym}}(k_0, M)\right)\big)$, where the components of the ciphertext are a standard encryption of a symmetric-key and an AONT applied to a symmetric-key encryption of the message M.

For each proxy re-encryption, there is a traditional proxy re-encryption of the symmetric key followed by re-encryption a random subset of bits of C^T. This makes inverting the AONT impossible unless the adversary was lucky enough to have previously queried and stored all of the encrypted bits, and since they are randomly distributed this is incredibly unlikely. However, to allow decryption, the proxy needs to store the locations of the re-encrypted bits and the key used to encrypt them. This is done by producing a new public-key encryption of the seed used to select the positions and encrypt the bits, and adding this to the ciphertext. As a result, the ciphertext size and encryption time grow additively with the number of re-encryptions, where the summand is the size of a proxy ciphertext.

A ciphertext that was re-encrypted r times has the form $\big(C^{pk},$ $\left[C_1^{bks}, \ldots, C_r^{bks}\right]$, $C^T\big)$, where C^{pk} is as before. Each C_i^{bks} of $C_1^{bks}, \ldots, C_r^{bks}$ is an encryption of a random subset of bit positions that we encrypted in the ith re-encryption, along with the one-time pad used for encryption. Finally, C^T is as before, but with all of the bits defined in the C_i^{bks} encrypted with the corresponding one-time-pads. To keep the notation consistent, we write an initial ciphertext as $\left(C^{pk}, [\,], C^T\right)$.

Our proxy re-encryption scheme is the five-tuple ($\mathsf{G}^{\mathsf{Hyb}}$, $\mathsf{E}^{\mathsf{Hyb}}$, $\mathsf{D}^{\mathsf{Hyb}}$, $\mathsf{RG}^{\mathsf{Hyb}}$, $\mathsf{RE}^{\mathsf{Hyb}}$), where $\mathsf{G}^{\mathsf{Hyb}}\left(1^\lambda\right) = \mathsf{G}\left(1^\lambda\right)$, $\mathsf{E}^{\mathsf{Hyb}}$ is defined in Fig. 2, $\mathsf{D}^{\mathsf{Hyb}}$ is defined in Fig. 2, $\mathsf{RG}^{\mathsf{Hyb}}\left(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j, \mathsf{sk}_j\right) = \left(\mathsf{pk}_j, \mathsf{rk}_{i\to j} = \mathsf{RG}\left(\mathsf{pk}_i, \mathsf{sk}_i, \mathsf{pk}_j, \mathsf{sk}_j\right)\right)$, and $\mathsf{RE}^{\mathsf{Hyb}}$ is defined in Fig. 2. We use the same notation as in Sect. 5.5.

6.4 Security of Our Scheme

In [23] we provide a proof to the following theorem showing basic PRE-CPA security.

Theorem 3. *Assume there exists a* PRE-CPA*-secure public-key proxy re-encryption scheme* $\Pi = (\mathsf{G}, \mathsf{RG}, \mathsf{E}, \mathsf{RE}, \mathsf{D})$*, an* IND-CPA*-secure symmetric-key encryption scheme* $\Pi_{sym} = \left(\mathsf{G}^{\mathsf{Sym}}, \mathsf{E}^{\mathsf{Sym}}, \mathsf{D}^{\mathsf{Sym}}\right)$*, and an all-or-nothing transform* T*. Then the construction of* Π_{hyb} *in Sect. 6.3 is* PRE-CPA*-secure.*

The next theorem establishes the $(1-\varepsilon)$-revocable security of our scheme. We note that we require a minor additional property of the underlying PRE scheme, which we call *re-encryption history independence.* It requires the distribution of a re-encrypted ciphertext does not depend on the keys used in encryption and re-encryption prior to the current key (though it may depend on the number

```
$\mathsf{PRE\text{-}CPA}_{\mathcal{A},\Pi}(1^\lambda)$

$\sigma \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{ukey}},\mathcal{O}_{\mathsf{ckey}}}(1^\lambda)$    ▷ (Phase 1)
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{rkey}},\mathcal{O}_{\mathsf{renc}},\mathcal{O}_{\mathsf{chal}}}(\sigma)$    ▷ (Phase 2)
Output 1 iff $b = b'$
```

```
$\mathsf{E}^{\mathsf{Hyb}}(\mathsf{pk}, M)$

$k_0 \leftarrow \mathsf{G}^{\mathsf{Sym}}(1^\lambda)$
$C^{pk} \leftarrow \mathsf{E}(\mathsf{pk}, k_0)$
$C^T \leftarrow T(\mathsf{E}^{\mathsf{Sym}}(k_0, M))$
return $C = (C^{pk}, [\,], C^T)$
```

```
$\mathsf{RE}^{\mathsf{Hyb}}((\mathsf{pk}_j, \mathsf{rk}'_{i\to j}), (C^{pk}, [C_1^{bks}, \ldots, C_r^{bks}], C_r^T)$

$\widetilde{C^{pk}} \leftarrow \mathsf{RE}(\mathsf{rk}'_{i\to j}, C^{pk})$
for $u \leftarrow 1, \ldots, r$ do
    $\widetilde{C_u^{bks}} \leftarrow \mathsf{RE}(rk'_{i\to j}, C_u^{bks})$
Choose $s_{r+1}, k_{r+1}$ uniformly.
$C_{r+1}^{bks} \leftarrow \mathsf{E}(\mathsf{pk}_j, (s_{r+1}, k_{r+1}))$
$\mathsf{ind}_{r+1} \leftarrow \mathsf{Ind}(s_{r+1}, \ell^*)$
$\mathsf{str}_{r+1} \leftarrow \mathsf{Ctr}(k_{r+1}, \ell^*)$
$w = 0$
for $v \leftarrow 1, \ldots, N$ do
    if $v \in \mathsf{ind}_{r+1}$ then
        $C_r^T[v] \leftarrow C^T[v] \oplus \mathsf{str}_{r+1}[w]$
        $w \leftarrow w + 1$
    else
        $C_r^T[v] \leftarrow C^T[v]$
return $\widetilde{C^{pk}}, [\widetilde{C_1^{bks}}, \ldots, \widetilde{C_r^{bks}}, C_{r+1}^{bks}], C_r^T)$
```

```
$\mathsf{D}^{\mathsf{Hyb}}(\mathsf{sk}, (C^{pk}, [C_1^{bks}, \ldots, C_r^{bks}], C_r^T))$

for $u \leftarrow r, \ldots, 1$ do
    $(s_u, k_u) \leftarrow \mathsf{D}(\mathsf{sk}, C_u^{bks})$
    $\mathsf{ind}_u \leftarrow \mathsf{Ind}(s_u, \ell^*)$
    $\mathsf{str}_u \leftarrow \mathsf{Ctr}(k_u, \ell^*)$
    $w = 0$
    for $v \leftarrow 1, \ldots, N$ do
        if $v \in \mathsf{ind}_u$ then
            $C_{u-1}^T[v] \leftarrow C_u^T[v] \oplus \mathsf{str}_u[w]$
            $w \leftarrow w + 1$
        else
            $C_{u-1}^T[v] \leftarrow C_u^T[v]$
$k_0 \leftarrow \mathsf{D}(\mathsf{sk}, C^{pk})$
return $M \leftarrow \mathsf{D}^{\mathsf{Sym}}(k_0, T^{-1}(C_0^T))$
```

Fig. 2. PRE-CPA security experiment and Π_{hyb} algorithms

of previous re-encryptions). Although PRE schemes do not need to have this property to be PRE-CPA-secure, it is a natural property to have. It does follow from re-encryption key privacy, an additional security property found in the schemes of [2,3,26]. Every PRE scheme we looked at [2–4,10,16,21,26] has re-encryption history independence.

Definition 12 (Re-Encryption History Independence). *A public-key proxy re-encryption scheme* $\Pi = (\mathsf{G}, \mathsf{RG}, \mathsf{E}, \mathsf{RE}, \mathsf{D})$ *has re-encryption history independence if for every set of public/secret key pairs* $\{(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}'_0, \mathsf{sk}'_0), \ldots, (\mathsf{pk}_{r-1}, \mathsf{sk}_{r-1}), (\mathsf{pk}'_{r-1}, \mathsf{sk}'_{r-1}), (\mathsf{pk}_r, \mathsf{sk}_r)\}$ *with re-encryption keys* $\mathsf{rk}_{u\to u+1} \leftarrow \mathsf{RG}(\mathsf{pk}_u, \mathsf{sk}_u, \mathsf{pk}_{u+1}, \mathsf{sk}_{u+1})$, $\mathsf{rk}'_{u\to u+1} \leftarrow \mathsf{RG}(\mathsf{pk}'_u, \mathsf{sk}'_u, \mathsf{pk}'_{u+1}, \mathsf{sk}'_{u+1})$ *for* $u \in [0, \ldots, r-2]$ *and* $\mathsf{rk}_{r-1\to r} \leftarrow \mathsf{RG}(\mathsf{pk}_{r-1}, \mathsf{sk}_{r-1}, \mathsf{pk}_r, \mathsf{sk}_r)$, $\mathsf{rk}'_{r-1\to r} \leftarrow \mathsf{RG}(\mathsf{pk}'_{r-1}, \mathsf{sk}'_{r-1}, \mathsf{pk}_r, \mathsf{sk}_r)$ *and every message M:* $\mathsf{RE}(\mathsf{rk}_{r-1\to r}, \ldots \mathsf{RE}(\mathsf{rk}_{0\to 1}, \mathsf{E}(\mathsf{pk}_0, M)) \ldots)$ *is indistinguishable from* $\mathsf{RE}(\mathsf{rk}'_{r-1\to r}, \ldots \mathsf{RE}(\mathsf{rk}'_{0\to 1}, \mathsf{E}(\mathsf{pk}'_0, M)) \ldots)$.

Theorem 4. *Assume there exists a* PRE-CPA*-secure public-key proxy re-encryption scheme* $\Pi = (\mathsf{G}, \mathsf{RG}, \mathsf{E}, \mathsf{RE}, \mathsf{D})$ *with re-encryption history* *independence, a* $\mathsf{IND\text{-}CPA}$*-secure symmetric-key encryption scheme* $\Pi_{sym} = (\mathsf{G^{Sym}}, \mathsf{E^{Sym}}, \mathsf{D^{Sym}})$*, and an adaptive* ℓ*-AONT* T*. Suppose that for the construction of* Π_{hyb} *from Sect. 6.3,* C^T *comprises at least a fraction* $1 - \delta$ *of the total size of each ciphertext. Then for any* $\varepsilon < 1$ *with* $\varepsilon > \delta$ *and any* $\ell^* > \frac{\ell}{\varepsilon - \delta}$*, this construction is* $(1 - \varepsilon)$-$\mathsf{Revoke\text{-}PRE\text{-}CPA}$*-secure.*

See [23] for the full proof.

7 Extensions to IBE and ABE, and RCCA Security

It is difficult to present a unified theorem that shows our construction immediately lifts to all proxy primitives. This is similar to how it is difficult to have a generic hybrid encryption theorem that covers traditional PKE, IBE, and ABE. Due to space limitations and the definition's relative simplicity, herein we only provide the results for a CPA secure PRE scheme. However, the hybrid construction that we demonstrate naturally ports to both identity-based PRE and revocable-storage ABE, which are important primitives for expressive cloud-based access control schemes (cf., identity-based proxy re-encryption [16,20,29] and revocable-storage ABE [28]). Results for those primitives are presented in [23].

Finally, RCCA security is an important requirement for many real-world scenarios. We note that based on this construction it is possible to extend it to such security. We demonstrate a more stringent RCCA-secure system for public-key proxy re-encryption systems in the Non-Programmable Random Oracle Model in an upcoming paper on the Cryptology ePrint Archive.

8 Implementation Issues and Efficiency

Implementation. A standard cryptographic hash and block-cipher are all that are necessary to implement the hybrid portion of our schemes. Given the frequent in silico inclusion of AES and SHA-256, this allows for incredibly efficient computational implementations of our scheme.

From a practical perspective our construction allows certain overhead computations to be moved to the cloud, where they may be more palatable. For example, a thin client need not compute the AONT on the symmetric ciphertext—this computation does not rely on any secret data. Thus a thin client can upload an appropriate traditional hybrid encryption $(\mathsf{E}(\mathsf{pk}, k), \mathsf{E^{Sym}}(k, m))$, where E is part of a proxy re-encryption scheme, and the cloud can compute $T(\mathsf{E^{Sym}}(k, m))$ for the AONT T—the cloud covers the extra encryption costs. Similarly, if a hybrid ciphertext has not been proxy re-encrypted, the cloud can remove the AONT, reducing the decryption cost to that of traditional hybrid encryption. Alternately, the application of an OAEP AONT, if implemented correctly, allows for a streaming implementation that could complement appropriate streaming (one-pass) authenticated encryption encryption schemes, resulting in the entire transform being implemented in one pass. Depending on the file access bottleneck, it is possible that in some settings the additional AONT for encryption in our setting will not actually add extra time to initial encryption.

Efficiency. In comparing efficiency, we first need to consider the security we provide. We provide less security than notions similar to ciphertext independence in [7] and UP-REENC-security in [13]. However, in practice it is unclear what attacks they prevent that are not similarly prevented by our definition with a small value of ε. Everspaugh et al. [13] performed a sample single-core implementation on a modern machine on a 1 GB file, and had run-times of approximately 2.5 h for each of Encrypt, ReEnc and Decrypt, comparing to roughly 10 ms for a similar approach with AES-GCM to encrypt the same file. This is of course because currently known UP-REENC constructions require the entire plaintext to be encrypted with asymmetric encryption primitives. Regardless, while our times will be more than the AES times, they will not be substantially more.

Due to differences in implementation, in silico support, disk types, and other performance parameters, we felt the best comparison would be in terms of the numbers of calls to a block cipher and hash function (for costing an OAEP construction of an AONT) that are needed in our construction. We compare the efficiency of the hybrid portion of our scheme to a naive hybrid proxy re-encryption. In the naive approach, to re-encrypt the proxy re-encrypts the public-key–encrypted symmetric key using the public-key proxy re-encryption algorithm, creates a new symmetric key and encrypts it under the new public-key, and re-encrypts the already encrypted message under the new symmetric key. We note that this naive solution does not achieve our security definition, because the adversary can perform a key-scraping attack for any reasonable value of ε. Regardless, it provides a reasonable benchmark system.

The AONT is only used in our scheme for encryption and decryption. Computing the AONT on an N-bit message, as well as inverting the AONT, requires computing two hash functions. Computing OAEP requires the SHA-256 compression function to run a total of $3N/512$ times, e.g., 50k times for a 1 MiB file and 50 mil. times for a 1 GiB file. As noted above, it is possible that in some instances and implementations these costs will be overshadowed by the overhead of file access. Regardless, the costs are fixed for any encryption and decryption of our file, and are fairly small. For example, common package Crypto++ [11] gives benchmarks of SHA-256 hashing 223 Mib/S on a modern Intel Skylake processor without in silico support.

Table 1 compares the number of times the AES block cipher is run for each operation. This depends on ℓ^*, the number of bits that are encrypted in each re-encryption, which in turn depends on several parameters: ℓ, the minimum number of missing bits for the AONT to be secure; ε, the minimum fraction of the ciphertext not downloaded by the adversary; and δ, is the maximum fraction of the ciphertext comprised by the public-key portion. Let ε^* denote $\varepsilon - \delta$, the minimum fraction of the symmetric-key portion of the ciphertext that the adversary has not downloaded. OAEP implemented with SHA-256 as described above with $\ell = 260$ will have 128 bits of security as an adaptive ℓ-AONT (Lemma 1). So we use 260 as our value for ℓ. Similarly, we assume that the pseudo-random index-selection and one-time pad are computed by AES, and count the number of invocations that are necessary.

Table 1. Instances of the AES block cipher required for each operation in the naive approach and in our scheme, where r is the number of re-encryptions

File size	ε^*	ℓ^*	Encryption	Re-enc.	Decryption		
					$r = 1$	$r = 10$	$r = 100$
1 GiB $N = 2^{33}$	Naive		6.711×10^7	6.711×10^7	1.342×10^8	7.382×10^8	6.778×10^9
	0.5	926	6.711×10^7	4.847×10^2	6.711×10^7	6.711×10^7	6.716×10^7
	0.25	2325	6.711×10^7	1.217×10^3	6.711×10^7	6.712×10^7	6.723×10^7
	0.1	8875	6.711×10^7	4.646×10^3	6.711×10^7	6.716×10^7	6.757×10^7

Figure 3 shows the effect that file size has on the cost of re-encryption, comparing naive re-encryption and our scheme with various values of ε^*. Figure 4 shows the effect that the number of previous re-encryptions has on the cost of decryption (for AES).

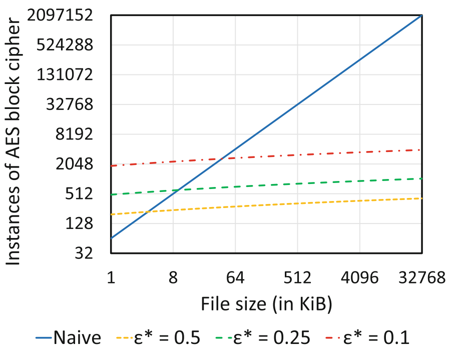

Fig. 3. Cost of re-encs vs. file size

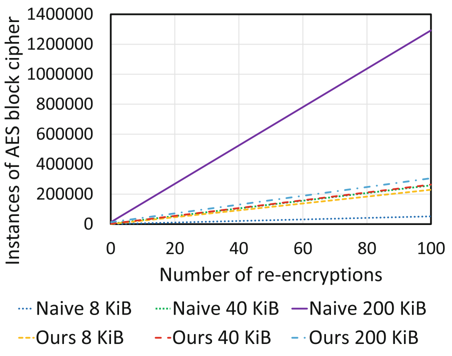

Fig. 4. Cost of decs vs. prev. re-encs.

While our scheme is slower for encryption (due to computing the AONT), in practice this will only occur once for each file. Re-encryption costs are more significant because re-encryption can occur for a large number of files at the same time. In this case, our scheme is several orders of magnitude faster than the naive approach, with the difference greater for larger files.

Acknowledgements. This work was supported by the National Science Foundation under awards CNS–1111149 and CNS–156375.

References

1. Amazon Web Services. Rotating customer master keys, September 2017. https://goo.gl/Ym9WeM
2. Aono, Y., Boyen, X., Phong, L.T., Wang, L.: Key-private proxy re-encryption under LWE. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 1–18. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03515-4_1

3. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00862-7_19
4. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (2006)
5. Bacis, E., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Rosa, M., Samarati, P.: Mix&slice: efficient access revocation in the cloud. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 217–228. ACM (2016)
6. Barker, E.: SP 800–57. Recommendation for key management, Part 1: General (revision 4). Technical report, NIST, January 2016
7. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_23
8. Boyko, V.: On the security properties of OAEP as an all-or-nothing transform. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 503–518. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_32
9. Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 453–469. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_33
10. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: CCS 2007, pp. 185–194 (2007)
11. Crypto++: Crypto++ 5.6.5 benchmarks, September 2017. https://goo.gl/xxSyU9
12. Dodis, Y., Sahai, A., Smith, A.: On perfect and adaptive security in exposure-resilient cryptography. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 301–324. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_19
13. Everspaugh, A., Paterson, K., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 98–129. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_4
14. Garrison III, W.C., Shull, A., Myers, S., Lee, A.J.: On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In: IEEE Proc. S&P (2016)
15. Google: Managing data encryption, September 2017. https://goo.gl/5UidnU
16. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72738-5_19
17. Ivan, A., Dodis, Y.: Proxy cryptography revisited. In: NDSS 2003. The Internet Soc. (2003)
18. Katz, J., Lindell, Y.: Intro to Modern Cryptography. Chapman & Hall/CRC, Boca Raton (2007)
19. Li, J., Qin, C., Lee, P.P.C., Li, J.: Rekeying for encrypted deduplication storage. In: DSN 2016, pp. 618–629. IEEE Computer Society (2016)
20. Liang, X., Cao, Z., Lin, H., Shao, J.: Attribute based proxy re-encryption with delegating capabilities. In: ASIACCS 2009, pp. 276–286 (2009)

21. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78440-1_21
22. Mambo, M., Okamoto, E.: Proxy cryptosystems: delegation of the power to decrypt ciphertexts. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **80**, 54–63 (1997)
23. Myers, S., Shull, A.: Efficient hybrid proxy re-encryption for practical revocation and key rotation. Cryptology ePrint Archive, Report 2017/833 (2017). http://eprint.iacr.org/2017/833
24. Open Web Application Security Project. Cryptographic storage cheat sheet, August 2016. https://goo.gl/MwKL8T
25. Payment Card Industry Security Standards Council. Payment card industry (PCI) data security standard, v3.2, April 2016
26. Phong, L.T., Wang, L., Aono, Y., Nguyen, M.H., Boyen, X.: Proxy re-encryption schemes with key privacy from LWE. Cryptology ePrint Archive, Report 2016/327 (2016). http://eprint.iacr.org/2016/327
27. Rivest, R.L.: All-or-nothing encryption and the package transform. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 210–218. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052348
28. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_13
29. Wang, H., Cao, Z., Wang, L.: Multi-use and unidirectional identity-based proxy re-encryption schemes. Inf. Sci. **180**(20), 4042–4059 (2010)
30. Watanabe, D., Yoshino, M.: Key update mechanism for network storage of encrypted data. In: CloudCom 2013, pp. 493–498 (2013)