Key Rotation for Authenticated Encryption

Adam Everspaugh[1], Kenneth Paterson[2], Thomas Ristenpart[3], Sam Scott[4]

[1]University of Wisconsin–Madison, [2]Royal Holloway, University of London,
[3]Cornell Tech

Abstract. A common requirement in practice is to periodically rotate the keys used to encrypt stored data. Systems used by Amazon and Google do so using a hybrid encryption technique which is eminently practical but has questionable security in the face of key compromises and does not provide full key rotation. Meanwhile, symmetric updatable encryption schemes (introduced by Boneh *et al.* CRYPTO 2013) support full key rotation without performing decryption: ciphertexts created under one key can be rotated to ciphertexts created under a different key with the help of a re-encryption token. By design, the tokens do not leak information about keys or plaintexts and so can be given to storage providers without compromising security. But the prior work of Boneh *et al.* addresses relatively weak confidentiality goals and does not consider integrity at all. Moreover, as we show, a subtle issue with their concrete scheme obviates a security proof even for confidentiality against passive attacks.

This paper presents a systematic study of *updatable Authenticated Encryption (AE)*. We provide a set of security notions that strengthen those in prior work. These notions enable us to tease out real-world security requirements of different strengths and build schemes that satisfy them efficiently. We show that the hybrid approach currently used in industry achieves relatively weak forms of confidentiality and integrity, but can be modified at low cost to meet our stronger confidentiality and integrity goals. This leads to a practical scheme that has negligible overhead beyond conventional AE. We then introduce *re-encryption indistinguishability*, a security notion that formally captures the idea of fully refreshing keys upon rotation. We show how to repair the scheme of Boneh *et al.*, attaining our stronger confidentiality notion. We also show how to extend the scheme to provide integrity, and we prove that it meets our re-encryption indistinguishability notion. Finally, we discuss how to instantiate our scheme efficiently using off-the-shelf cryptographic components (AE, hashing, elliptic curves). We report on the performance of a prototype implementation, showing that fully secure key rotations can be performed at a throughput of approximately 116 kB/s.

1 Introduction

To cryptographically protect data while stored, systems use authenticated encryption (AE) schemes that provide strong message confidentiality as well as ciphertext integrity. The latter allows detection of active attackers who manipulate ciphertexts. When data is stored for long periods of time, good key management practice dictates that systems must support key rotation: moving encrypted data from an old key to a fresh one. Indeed, key rotation is mandated by regulation in some contexts, such as the payment card industry data security standard (PCI DSS) that dictates how credit card data must be secured [PCI16]. Key rotation can also be used to revoke old keys that are comprised, or to effect data access revocation.

Deployed approaches to key rotation. Systems used in practice typically support a type of key rotation using a symmetric key hierarchy. Amazon's Key Management Service [AWS], for example, enables users to encrypt a plaintext M under a fresh data encapsulation key via $C_{dem} = \mathsf{Enc}(K_d, M)$ and then wrap K_d via $C_{kem} = \mathsf{Enc}(K, K_d)$ under a long-term key K owned by the client. Here Enc is an authenticated encryption (AE) scheme. By analogy with the use of hybrid encryption in the asymmetric setting, we refer to such a scheme as a KEM/DEM construction, with KEM and DEM standing for key and data encapsulation mechanisms, respectively; we refer to the specific scheme as AE-hybrid.

The AE-hybrid scheme then allows a simple form of key rotation: the client picks a fresh K' and re-encrypts K_d as $C'_{kem} = \mathsf{Enc}(K', \mathsf{Dec}(K, C_{kem}))$. Note that the DEM key K_d does not change during key rotation. When deployed in a remote storage system, a client can perform key rotation just by fetching from the server the small, constant-sized ciphertext C_{kem}, operating locally on it to produce C'_{kem}, and then sending C'_{kem} back to the server. Performance is independent of the actual message length. The Google Cloud Platform [Goo] uses a similar approach to enable key rotation.

To our knowledge, the level of security provided by this widely deployed AE-hybrid scheme has never been investigated, let alone formally defined in a security model motivated by real-world security considerations. It is even arguable whether AE-hybrid truly rotates keys, since the DEM key does not change. Certainly it is unclear what security is provided if key compromises occur, one of the main motivations for using such an approach in the first place. On the other hand, the scheme is fast and requires only limited data transfer between the client and the data store, and appears to be sufficient to meet current regulatory requirements.

Updatable encryption. Boneh, Lewi, Montgomery, and Raghunathan (BLMR) [BLMR15] (the full version of [BLMR13]) introduced another approach to enabling key rotation that they call *updatable encryption*. An updatable encryption scheme is a symmetric encryption scheme that, in addition to the usual triple of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ algorithms, comes with a pair of algorithms $\mathsf{ReKeyGen}$ and ReEnc. The first, $\mathsf{ReKeyGen}$, generates a compact rekey token given the old and new secret keys and a target ciphertext, while the second, ReEnc, uses a rekey token output by the first to rotate the ciphertext without performing decryption. For example, AE-hybrid can be seen as an instance of an updatable encryption scheme in which the rekey token output by

$\mathsf{ReKeyGen}$ is C'_{kem} and where ReEnc simply replaces C_{kem} with C'_{kem}. BLMR introduced an IND-CPA-style security notion in which adversaries can additionally obtain some rekey tokens. Their definition is inspired by, but different from, those used for CCA-secure proxy re-encryption schemes [CH07]. Given its obvious limitations when it comes to key rotation, it is perhaps surprising that the AE-hybrid construction provably meets the BLMR confidentiality notion for updatable encryption schemes.

BLMR also introduced and targeted a second security notion for updatable encryption, called ciphertext independence. It demands that a ciphertext and its rotation to another key are identically distributed to a ciphertext and a rotation of another ciphertext (for the same message). The intuition is that this captures the idea that true key rotation should refresh all randomness used during encryption. This definition is *not* met by the AE-hybrid construction above. But it is both unclear what attacks meeting their definition would prevent, and, relatedly, whether more intuitive definitions exist.

BLMR gave a construction for an updatable encryption scheme and claimed that it provably meets their two security definitions. Their construction cleverly combines an IND-CPA KEM with a DEM that uses a key-homomorphic PRF [NPR99,BLMR15] to realize a stream cipher. This enables rotation of both the KEM and the DEM keys, though the latter requires a number of operations that is linear in the plaintext length. Looking ahead, their proof sketch has a bug and we provide strong evidence that it is unlikely to be fixable. Moreover, BLMR do not yet target or achieve any kind of authenticated encryption goal, a must for practical use.

Our contributions. We provide a systematic treatment of AE schemes that support key rotation without decryption, a.k.a. updatable AE.

Specifically, we provide a new security notion for confidentiality, UP-IND, that is strictly stronger than that of BLMR [BLMR15], a corresponding notion for integrity, UP-INT (missing entirely from BLMR but essential for practice), and a new notion called re-encryption indistinguishability (UP-REENC) that is strictly stronger and more natural in capturing the spirit of "true key rotation" than the ciphertext indistinguishability notion of BLMR.

Achieving our UP-REENC notion means that an attacker, having access to both a ciphertext and the secret key used to generate it, should not be able to derive any information that helps it attack a rotation of that ciphertext. Thus, for example, an insider with access to the encryption keys at some point in time but who is then excluded from the system cannot make use of the old keys to learn anything useful once key rotation has been carried out on the AE ciphertexts. Teasing out the correct form of this notion turns out to be a significant challenge in our work.

Armed with this set of security notions, we go on to make better sense of the landscape of constructions for updatable AE schemes. Table 1 summarises the security properties of the different schemes that we consider. Referring to this table, our security notions highlight the limitations of the AE-hybrid scheme: while it meets the confidentiality notion of BLMR, it only satisfies our UP-IND and UP-INT notions when considering a severely weakened adversary who has no access to any compromised keys. We propose an improved construction, KSS, that satisfies both notions for any number of compromised keys and which is easily deployable via small adjustments to AE-hybrid. KSS uses a form of secret sharing to embed key shares in the KEM and DEM components to avoid the issue of leaking the DEM key in the updating process, and adds a cryptographic hash binding the KEM and DEM components to prevent mauling attacks. These changes could easily be adopted by practitioners with virtually no impact on performance, while concretely improving security.

However, the improved scheme KSS cannot satisfy our UP-REENC notion, because it still uses a KEM/DEM-style approach in which the DEM key is never rotated. The BLMR scheme might provide UP-REENC security, but, as noted above, its security proof contains a bug which we consider unlikely to be fixable. Indeed, we show that proving the BLMR scheme confidential would imply that one could also prove circular security [BRS03,CL01] for a particular type of hybrid encryption scheme assuming only the key encapsulation is IND-CPA secure. Existing counter-examples of IND-CPA secure, but circular insecure, schemes [ABBC10,CGH12] do not quite rule out such a result. But the link to the very strong notion of circular security casts doubt on the security of this scheme. One can easily modify the BLMR scheme to avoid this issue, but even having done so the resulting encryption scheme is still trivially malleable and so cannot meet our UP-INT integrity notion.

We therefore provide another new scheme, ReCrypt, meeting all three of our security notions: UP-IND, UP-INT and UP-REENC. We take inspiration from the previous constructions, especially that of BLMR: key-homomorphic PRFs provide the ability to fully rotate encryption keys; the KEM/DEM approach with secret sharing avoids the issue of leaking the DEM key in the updating process; and finally, adding a cryptographic hash to the KEM tightly binds the KEM and DEM portions and prevents ciphertext manipulation. We go on to instantiate the scheme using the Random Oracle Model (ROM) key-homomorphic PRF from [NPR99], having the form $H(M)^k$, where H is a hash function into a group in which DDH is hard. This yields a construction of an updatable AE scheme meeting all three of our security notions in the ROM under the DDH assumption. We report on the performance of an implementation of ReCrypt using elliptic curve groups, concluding that it is performant enough for practical use with short plaintexts. However, because of its reliance on exponentiation, ReCrypt is still orders of magnitude slower than our KSS scheme (achieving only UP-IND and UP-INT security). This, currently, is the price that must be paid for true key rotation in updatable encryption.

Summary. In summary, the main contributions of this paper are:

- To provide the first definitions of security for AE supporting key rotation without exposing plaintext.
- To explain the gap between existing, deployed schemes using the KEM/DEM approach and “full” refreshing of ciphertexts.

Scheme	**Section**	**Tokens**	**CT dep.**	UP-IND	UP-INT	UP-REENC
AE-hybrid[†]	4.1	uni-dir.	depend.	✗	✗	✗
KSS[*]	4.3	uni-dir.	depend.	✓	✓	✗
XOR-KEM[*]	A.1	bi-dir.	indep.	✓	✗	✗
BLMR	6	uni-dir.	depend.	✗	✗	✗
ReCrypt[*]	7	uni-dir.	depend.	✓	✓	✓

Table 1. Summary of schemes studied. † In-use by practitioners today. * Introduced in this work.

- To provide the first proofs of security for AE schemes using the KEM/DEM approach, namely AE-hybrid and KSS.
- To detail the first updatable AE scheme, ReCrypt, that fully and securely refreshes ciphertexts by way of key rotations without ever exposing plaintext data. We implement a prototype and report on microbenchmarks, showing that rotations can be performed in less than 9 μsper byte.

2 Updatable AE

We turn to formalizing the syntax and semantics of AE schemes supporting key rotation. Our approach extends that of Boneh et al. [BLMR15] (BLMR), the main syntactical difference being that we allow rekey token generation, re-encryption, and decryption to all return a distinguished error symbol $\perp$. This is required to enable us to later cater for integrity notions. We also modify the syntax so that ciphertexts include two portions, a header and a body. In our formulation, only the former is used during generation of rekey tokens (while in BLMR the full ciphertext is formally required).

Definition 1 (Updatable AE). *An* updatable AE scheme *is a tuple of algorithms $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{ReKeyGen}, \mathsf{ReEnc}, \mathsf{Dec})$ with the following properties:*

- $\mathsf{KeyGen}() \to k$. *Outputs a secret key k.*
- $\mathsf{Enc}(k, m) \to C$. *On input a secret key k and message m, outputs a ciphertext $C = (\tilde{C}, \overline{C})$ consisting of a ciphertext header $\tilde{C}$ and ciphertext body $\overline{C}$.*
- $\mathsf{ReKeyGen}(k_1, k_2, \tilde{C}) \to \Delta_{1,2,\tilde{C}}$. *On input two secret keys, k_1 and k_2, and a ciphertext header $\tilde{C}$, outputs a rekey token or $\perp$.*
- $\mathsf{ReEnc}(\Delta_{1,2,\tilde{C}}, (\tilde{C}, \overline{C})) \to C_2$. *On input a rekey token and ciphertext, outputs a new ciphertext or $\perp$. We require that ReEnc is deterministic.*
- $\mathsf{Dec}(k, C) \to m$. *On input a secret key k and ciphertext C outputs either a message or $\perp$.*

Of course we require that all algorithms are efficiently computable. Note that, in common with [BLMR15], our definition is *not* in the nonce-based setting that is widely used for AE. Rather, we will assume that Enc is randomised. We

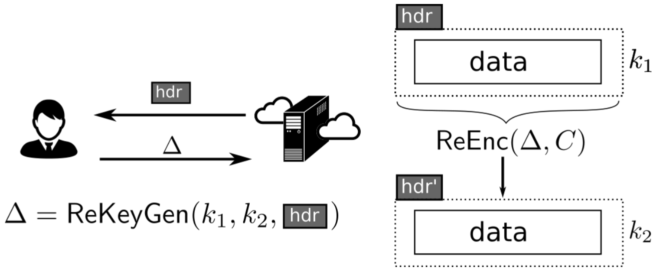

Fig. 1. Interaction between client and cloud during a ciphertext-dependent update. Client retrieves a small ciphertext header, and runs ReKeyGen to produce a compact rekey token Δ. The cloud uses this token to re-encrypt the data. At the end of the update, the data is encrypted using k_2, and cannot be recovered using only k_1.

consider this sufficient for a first treatment of updatable AE; it also reflects common industry practice as per the schemes currently used by Amazon [AWS] and Google [Goo]. We relegate the important problem of developing a parallel formulation in the nonce-based setting to future work. Similarly, we assume that all our AE schemes have single decryption errors, cf. [BDPS14], and we do not consider issues such as release of unverified plaintext, cf. [ABL$^+$14], tidiness, cf. [NRS14] and length-hiding, cf. [PRS11].

Correctness. An updatable AE scheme is *correct* if decrypting a legitimately generated ciphertext reproduces the original message. Of course, legitimate ciphertexts may be rotated through many keys, complicating the formalization of this notion.

Definition 2 (Correctness). *Fix an updatable AE scheme Π. For any message m and any sequence of secret keys $k_1, \ldots k_T$ output by running KeyGen T times, let $C_1 = (\tilde{C}_1, \overline{C}_1) = \mathsf{Enc}(k_1, m)$ and recursively define for $1 \leq t < T$*

$$C_{t+1} = \mathsf{ReEnc}(\mathsf{ReKeyGen}(k_t, k_{t+1}, \tilde{C}_t), C_t).$$

Then Π is correct if $\mathsf{Dec}(k_T, C_T) = m$ with probability 1.

Compactness. We say that an updatable AE scheme is compact if the size of both ciphertext headers and rekeying tokens are independent of the length of the plaintext. In practice the sizes should be as small as possible, and for the constructions we consider these are typically a small constant multiple of the key length.

Compactness is important for efficiency of key rotation. Considering the abstract architecture in Figure 1, header values must be available to the key server when rekey tokens are generated. Typically this will mean having to fetch them from storage. Likewise, the rekey token must be sent back to the storage system. Note that there are simple constructions that are not compact, such as the one that sets $\tilde{C}$ to be a standard authenticated encryption of the message and in which ReKeyGen decrypts $\tilde{C}$, re-encrypts it, and outputs a “rekeying token” as the new ciphertext.

Ciphertext-dependence. As formulated above, updatable AE schemes require part of the ciphertext, the ciphertext header $\tilde{C}$, in order to generate a rekey token. We will also consider schemes for which $\tilde{C}$ is the empty string, denoted ε. We will restrict attention to schemes for which encryption either always outputs $\tilde{C} = \varepsilon$ or never does. In the former case we call the scheme ciphertext-independent and, in the latter case, ciphertext-dependent. When discussing ciphertext-independent schemes, we will drop $\tilde{C}$ from notation, e.g., writing $\Delta_{i,j}$ instead of $\Delta_{i,j,\tilde{C}}$.

However, we primarily focus on ciphertext-dependent schemes which appear to offer more flexibility and achieve stronger security guarantees (though it is an open question whether a ciphertext-independent scheme can achieve our strongest security notion). We do propose a very lightweight ciphertext-independent scheme included in Appendix A.1, but we show it achieves strictly weaker confidentiality and integrity notions. One can generically convert a ciphertext-independent scheme into a ciphertext-dependent one, simply by deriving a ciphertext-specific key using some unique identifier for the ciphertext. We omit the formal treatment of this trivial approach.

Directionality of rotations. Some updatable AE schemes are bidirectional, meaning rekey tokens can be used to go forwards or backwards.

We only consider bi-directionality to be a feature of ciphertext-independent schemes. Formally, we say that a scheme is *bidirectional* if there exists an efficient algorithm $\mathsf{Invert}(\cdot)$ that produces a valid rekey token $\Delta_{j,i}$ when given $\Delta_{i,j}$ as input.

Schemes that are not bidirectional might be able to ensure that an adversary cannot use rekey tokens to "undo" a rotation of a ciphertext. We will see that ciphertext-dependence can help in building such unidirectional schemes, whereas ciphertext-independent schemes seem harder to make unidirectional. This latter difficulty is related to the long-standing problem of constructing unidirectional proxy re-encryption schemes in the public key setting.

Relationship to proxy re-encryption. Proxy re-encryption targets a different setting than updatable encryption (or AE): the functional ability to allow a ciphertext encrypted under one key to be converted to a ciphertext decryptable by another key. The conversion should not leak plaintext data, but, unlike key rotation, it is not necessarily a goal of proxy re-encryption to remove all dependency on the original key, formalised as indistinguishability of re-encryptions (UP-REENC security) in our work. For example, previous work [CK05,ID03] suggests twice encrypting plaintexts under different keys. To rotate, the previous outer key and a freshly generated outer key is sent to the proxy to perform conversion, but the inner key is never modified. Such an approach does not satisfy the goals of key rotation.

That said, any bidirectional, ciphertext-independent updatable AE ends up also being usable as a symmetric proxy re-encryption scheme (at least as formalized by [BLMR15]).

3 Confidentiality and Integrity for Updatable Encryption

Updatable AE should provide confidentiality for messages as well as integrity of ciphertexts, even in the face of adversaries that obtain rekey tokens and re-encryptions, and that can corrupt some number of secret keys. Finding definitions that rule out trivial wins — e.g., rotating a challenge ciphertext to a compromised key, or obtaining sequences of rekey tokens that allow such rotations — is delicate. We provide a framework for doing so.

Our starting point will be a confidentiality notion which improves significantly upon the previous notion of BLMR by including additional attack vectors, and strengthening existing ones.

For ciphertext integrity, we develop a new definition, building on the usual INT-CTXT notion for standard AE [BN00]. Looking ahead, we will target unidirectional schemes that simultaneously achieve both UP-IND and UP-INT security.

We will follow a concrete security approach in which we do not strictly define security, but rather measure advantage as a function of the resources (running time and number of queries) made by an adversary. Informally, schemes are secure if no adversary with reasonable resources can achieve advantage far from zero.

3.1 Message Confidentiality

The confidentiality game UP-IND is shown in the leftmost column of Figure 2. The adversary's goal is to guess the bit b. Success implies that a scheme leaks partial information about plaintexts. We paramaterise the game by two values t and κ. The game initialises $t+\kappa$ secret keys, κ of which are given to the adversary, and t are kept secret for use in the oracles. We label the keys by $k_1, \ldots, k_t$ for the uncompromised keys, and by $k_{t+1}, \ldots k_{t+\kappa}$ for the compromised keys. We require at least one uncompromised key, but do not necessarily require any compromised keys, i.e. $t \geq 1$ and $\kappa \geq 0$. We leave consideration of equivalences between models with many keys and few keys and between models with active and static key compromises as interesting problems for future work.

The game relies on two subroutines $\mathsf{Invalid}_{\mathsf{RK}}$ and $\mathsf{Invalid}_{\mathsf{RE}}$ to determine if a re-keygen and re-encryption query, respectively, should be allowed. These procedures are efficiently computed by the game as a function of the adversarial queries and responses. This reliance on the transcript we leave implicit in the notation to avoid clutter. Different choices of invalidity procedures gives rise to distinct definitions of security, and we explain two interesting ones in turn. Note that an invalid query (as determined by $\mathsf{Invalid}_{\mathsf{RE}}$) still results in the adversary learning the ciphertext header, giving greater power to the adversary. We believe this to be an important improvement both in practice and theoretically over previous models, which consider only a partial compromise. The full compromise of a client results in the adversary playing the role of the client in the key update procedure, during which the server will return the ciphertext header. In practice,

```
UP-IND
$b \leftarrow_{\$} \{0,1\}$
$k_1, \ldots, k_{t+\kappa} \leftarrow_{\$} \mathsf{KeyGen}()$
$b' \leftarrow_{\$} \mathcal{A}^{O}(k_{t+1}, \ldots, k_{t+\kappa})$
return $(b' = b)$

Enc$(i, m)$
return $\mathsf{Enc}(k_i, m)$

ReKeyGen$(i, j, \tilde{C})$
if $\mathsf{Invalid}_{\mathsf{RK}}(i, j, \tilde{C})$ then return $\perp$
$\Delta_{i,j,\tilde{C}} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
return $\Delta_{i,j,\tilde{C}}$

ReEnc$(i, j, (\tilde{C}, \overline{C}))$
$\Delta_{i,j,\tilde{C}} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
$C' = (\tilde{C}', \overline{C}') \leftarrow \mathsf{ReEnc}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$
if $\mathsf{Invalid}_{\mathsf{RE}}(i, j, \tilde{C})$ then return $\tilde{C}'$
else return $C'$

LR$(i, m_0, m_1)$
if $i > t$ then return $\perp$
$C \leftarrow_{\$} \mathsf{Enc}(k_i, m_b)$
return $C$
```

```
UP-INT
$\mathsf{win} \leftarrow \mathsf{false}$
$k_1, \ldots, k_{t+\kappa} \leftarrow_{\$} \mathsf{KeyGen}()$
$\mathcal{A}^{O}(k_{t+1}, \ldots, k_{t+\kappa})$
return $\mathsf{win}$

Enc$(i, m)$
return $\mathsf{Enc}(k_i, m)$

ReKeyGen$(i, j, \tilde{C})$
return $\mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$

ReEnc$(i, j, (\tilde{C}, \overline{C}))$
$\Delta_{i,j,\tilde{C}} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
$C' \leftarrow \mathsf{ReEnc}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$
return $C'$

Try$(i, C)$
if $\mathsf{InvalidCTXT}(i, C)$ then return $\perp$
$M \leftarrow \mathsf{Dec}(k_i, C)$
if $M = \perp$ then return $\perp$
$\mathsf{win} \leftarrow \mathsf{true}$
return $M$
```

Fig. 2. Confidentiality and integrity games for updatable encryption security.

it is likely that an adversary who has initially breached the client would use this access to query related services.

Invalidity procedures. For the invalidity constraints used in UP-IND, we target a strong definition, while preventing the adversary from trivially receiving a challenge ciphertext re-encrypted to a compromised key.

We use the ciphertext headers to determine whether a ciphertext has been derived from a challenge ciphertext. It is natural to use only the headers since these will be processed by the client when performing an update. We define a procedure $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C})$ that outputs true should $\tilde{C}$ have been derived from the ciphertext header returned by an LR query.

Definition 3 (LR-derived headers). *We recursively define function* $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C})$ *to output* true *iff any of the following conditions hold:*

- $\tilde{C}$ *was the ciphertext header output in response to a query* $\mathrm{LR}(i, m_0, m_1)$
- $\tilde{C}$ *was the ciphertext header output in response to a query* $\mathrm{ReEnc}(j, i, C')$ *and* $\mathsf{Derived}_{\mathsf{LR}}(j, \tilde{C}') = \mathsf{true}$
- $\tilde{C}$ *is the ciphertext header output by running* $\mathsf{ReEnc}(\Delta_{j,i,\tilde{C}'}, C')$ *where* $\Delta_{j,i,\tilde{C}'}$ *is the result of a query* $\mathrm{ReKeyGen}(j, i, \tilde{C}')$ *for which* $\mathsf{Derived}_{\mathsf{LR}}(j, \tilde{C}') = \mathsf{true}$*.*

The predicate $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C})$ is efficient to compute and can be computed locally by the adversary. The most efficient way to implement it is to grow a look-up table $\mathtt{T}$ indexed by a key identifier and a ciphertext header and whose entries are sets of ciphertexts. Any query to $\mathrm{LR}(i, m_0, m_1)$ updates the table by adding the returned ciphertext to the set $\mathtt{T}[i, \tilde{C}]$ where $\tilde{C}$ is the oracle's returned ciphertext header value. For a query $\mathrm{ReEnc}(j, i, C')$, if $\mathtt{T}[j, \tilde{C}']$ is not empty, then it adds the returned ciphertext to the set $\mathtt{T}[i, \tilde{C}^*]$ for $\tilde{C}^*$ the returned ciphertext header. For a query $\mathrm{ReKeyGen}(j, i, \tilde{C}')$ with return value $\Delta_{j,i,\tilde{C}'}$, apply $\mathsf{ReEnc}(\Delta_{j,i,\tilde{C}'}, C)$ for all ciphertexts C found in entry $\mathtt{T}[j, \tilde{C}']$ and add appropriate new entries to the table. In this way, one can maintain the table in worst-case time that is quadratic in the number of queries, and compute in constant time $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C})$ by simply checking if $\mathtt{T}[i, \tilde{C}]$ is non-empty. If any call to $\mathsf{ReKeyGen}$ or ReEnc in $\mathsf{Derived}_{\mathsf{LR}}$ or the main oracle procedure returns $\perp$, then the entire procedure returns $\perp$.

Note that $\mathsf{Derived}_{\mathsf{LR}}$ relies on ReEnc being deterministic, a restriction we made in Section 2. To complete the definition, we specify the invalidity procedures that use $\mathsf{Derived}_{\mathsf{LR}}$ as a subroutine:

- $\mathsf{Invalid}_{\mathsf{RK}}(i, j, \tilde{C})$ outputs true if $j > t$ and $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C}) = \mathsf{true}$. In words, the target key is compromised and $i, \tilde{C}$ derives from an LR query.
- $\mathsf{Invalid}_{\mathsf{RE}}(i, j, \tilde{C})$ outputs true if $j > t$ and $\mathsf{Derived}_{\mathsf{LR}}(i, \tilde{C}) = \mathsf{true}$. In words, the target key is compromised and $i, \tilde{C}$ derives from an LR query.

We denote the game defined by using these invalidity procedures by UP-IND. We associate to an UP-IND adversary $\mathcal{A}$ and scheme Π the advantage measure:

$$\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) = 2 \cdot \Pr\left[\text{UP-IND}^{\mathcal{A}}_{\Pi,\kappa,t} \Rightarrow \mathsf{true}\right] - 1 \ .$$

This notion is very strong and bidirectional schemes cannot meet it.

Theorem 1. *Let Π be a bidirectional updatable encryption scheme. Then there exists an* UP-IND *adversary $\mathcal{A}$ that makes* 2 *queries and for which*

$$\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) = 1$$

for any $\kappa \geq 1$ and $t \geq 1$.

Proof. We explicitly define the adversary $\mathcal{A}$. It makes a query to $C_1 = \text{LR}(1, m_0, m_1)$ for arbitrary messages $m_0 \neq m_1$ and computes locally $C_{t+1} = \mathsf{Enc}(k_{t+1}, m_1)$. It then makes a query $\Delta_{t+1,1,\tilde{C}_{t+1}} = \text{ReKeyGen}(t+1, 1, \tilde{C}_{t+1})$. It runs $C' = \mathsf{ReEnc}(\mathsf{Invert}(\Delta_{t+1,1,\tilde{C}_{t+1}}, C_{t+1}, C_1), C_1)$ locally and then decrypts C' using k_{t+1}. It checks whether the result is m_0 or m_1 and returns the appropriate bit.

BLMR confidentiality. In comparison, we define invalidity procedures corresponding to those in BLMR's security notion.

- $\mathsf{InvalidBLMR}_{\mathsf{RK}}(i, j, \tilde{C})$ outputs true if $i \leq t < j$ or $j \leq t < i$ and outputs false otherwise. In words, the query is not allowed if exactly one of the two keys is compromised.
- $\mathsf{InvalidBLMR}_{\mathsf{RE}}(i, j, \tilde{C})$ outputs true if $j > t$ and false otherwise. In words, the query is not allowed if the target key k_j is compromised.

We denote the game defined by using these invalidity procedures by UP-IND-BI (the naming will become clear presently). We associate to an UP-IND-BI adversary $\mathcal{A}$, scheme Π, and parameters κ, t the advantage measure:

$$\mathsf{Adv}^{\text{up-ind-bi}}_{\Pi,\kappa,t}(\mathcal{A}) = 2 \cdot \Pr\left[\text{UP-IND-BI}^{\mathcal{A}}_{\Pi,\kappa,t} \Rightarrow \mathsf{true}\right] - 1\,.$$

A few observations are in order. First, it is apparent that the invalidity procedures for the BLMR notion are significantly stronger than ours, leading to a weaker security notion: the BLMR procedures are not ciphertext-specific but instead depend only on the compromise status of keys. We will show that this difference is significant. In addition, the corresponding BLMR definition did not consider leakage of the ciphertext header when $\mathsf{InvalidBLMR}_{\mathsf{RE}}$ returns true. Second, for ciphertext-independent schemes in which $\tilde{C} = \varepsilon$ always, the BLMR definition coincides with symmetric proxy re-encryption security (as also introduced in their paper [BLMR15]). Third, the BLMR confidentiality notion does not require unidirectional security of rekey tokens because it has the strong restriction of disallowing attackers from obtaining rekey tokens $\Delta_{i,j,\tilde{C}}$ with $i > t$ (so the corresponding key is compromised), but with $j < t$ (for an uncompromised key). Thus, in principle, bidirectional schemes could meet this notion, explaining our naming convention for the notion. Finally, the BLMR notion does not require ciphertext-specific rekey tokens because the invalidity conditions are based only on keys and not on the target ciphertext.

Detailed in Appendix A.1 is a bidirectional scheme that is secure in the sense of UP-IND-BI. This result and the negative result that no bidirectional scheme can achieve UP-IND given above (Theorem 1) yields as a corollary that UP-IND-BI security is strictly weaker than UP-IND security. This illustrates the enhanced strength of our UP-IND security notion compared to the corresponding BLMR notion, UP-IND-BI.

Given that bidirectional, ciphertext-independent schemes have certain advantages in terms of performance and deployment simplicity, practitioners may prefer them in some cases. For that flexibility, one trades off control over the specificity of rekey tokens, which could be dangerous to confidentiality in some compromise scenarios.

3.2 Ciphertext Integrity

We now turn to a notion of integrity captured by the game UP-INT shown in Figure 2. The adversary's goal is to submit a ciphertext to the Try oracle that decrypts properly. Of course, we must exclude the adversary from simply resubmitting valid ciphertexts produced by the encryption oracle, or derived from such an encryption by way of re-encryption queries or rekey tokens.

In a bit more detail, in the Try oracle, we define a predicate $\mathsf{InvalidCTXT}$ which captures whether the adversary has produced a trivial derivation of a ciphertext obtained from the encryption oracle. This fulfills a similar role to that of the $\mathsf{Invalid_{RE}}$ and $\mathsf{Invalid_{RK}}$ subroutines in the UP-IND game.

For the unidirectional security game UP-INT, we define $\mathsf{InvalidCTXT}(i, C = (\tilde{C}, \overline{C}))$ inductively, outputting true if any of the following conditions hold:

- $i > t$, i.e. k_i is known to the adversary
- $(\tilde{C}, \overline{C})$ was output in response to a query $\mathrm{Enc}(i, m)$
- $(\tilde{C}, \overline{C})$ was output in response to a query $\mathrm{ReEnc}(j, i, C')$ and $\mathsf{InvalidCTXT}(j, C') = \mathsf{true}$
- $(\tilde{C}, \overline{C})$ is the ciphertext output by running $\mathsf{ReEnc}(\Delta_{j,i,\tilde{C}'}, C')$ for $C' = (\tilde{C}', \overline{C}')$ where $\Delta_{j,i,\tilde{C}'}$ was the result of a query $\mathrm{ReKeyGen}(j, i, \tilde{C}')$ and $\mathsf{InvalidCTXT}(j, C')) = \mathsf{true}$.

This predicate requires the transcript of queries thus far; to avoid clutter we leave the required transcript implicit in our notation. The definition of $\mathsf{InvalidCTXT}$ is quite permissive: it defines invalid ciphertexts as narrowly as possible, making our security notion stronger. Notably, the adversary can produce any ciphertext (valid or otherwise) using a corrupted key k_i, and use the ReKeyGen oracle to learn a token to update this ciphertext to a non-compromised key. Only the direct re-encryption of the submitted ciphertext is forbidden.

We associate to an updatable encryption scheme Π, an UP-INT adversary $\mathcal{A}$, and parameters κ, t the advantage measure:

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) = \Pr\left[\text{UP-INT}^{\mathcal{A}}_{\Pi,\kappa,t} \Rightarrow \mathsf{true}\right] .$$

4 Practical Updatable AE Schemes

We first investigate the security of updatable AE schemes built using the KEM/DEM approach sketched in the introduction. Such schemes are in widespread use at present, for example in AWS's and Google's cloud storage systems, yet have received no formal analysis to date. We produce the AE-hybrid construction as a formalism of this common practice.

Using the confidentiality and integrity definitions from the previous section, we discover that this construction offers very weak security against an adversary capable of compromising keys. Indeed, we are only able to prove security when the number of compromised keys κ is equal to 0. Given the intention of key rotation this is a somewhat troubling result.

On a positive note, we show a couple of simple tweaks to the AE-hybrid which fix these issues. The resultant scheme, named KSS, offers improved security at little additional cost.

We leave to the appendix our bidirectional, ciphertext-independent scheme XOR-KEM which does not offer strong integrity guarantees but may be of interest for other applications.

4.1 Authenticated Encryption

In the following constructions we make use of authenticated encryption (AE) schemes which we define here.

Definition 4 (Authenticated encryption). *An authenticated encryption scheme π is a tuple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. $\mathcal{K}$ is a randomised algorithm outputting keys. We denote by $\mathcal{E}_k(\cdot)$ the randomised algorithm for encryption by key k, and by $\mathcal{D}_k(\cdot)$ decryption. Decryption is a deterministic algorithm and outputs the distinguished symbol $\perp$ to denote a failed decryption.*

In keeping with our definitional choices for updatable AE, we consider randomised AE schemes rather than nonce-based ones.

We use the all-in-one authenticated encryption security definition from [RS06].

Definition 5 (Authenticated Encryption Security). *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme. Let* Enc*,* Dec *be oracles whose behaviors depends on hidden values $b \in \{0, 1\}$ and key $k \leftarrow_\$ \mathcal{K}$.* Enc *takes as input a bit string m and produces $\mathcal{E}_k(m)$ when $b = 0$, and produces a random string of the same length otherwise.* Dec *takes as input a bit string C and produces $\mathcal{D}_k(C)$ when $b = 0$, and produces $\perp$ otherwise.*

Let $\text{AE-ROR}_\pi^{\mathcal{A}}$ *be the game in which an adversary $\mathcal{A}$ interacts with the* Enc *and* Dec *oracles and must output a bit b'. The game outputs* true *when $b = b'$. We require that the adversary not submit outputs from the* Enc *oracle to the* Dec *oracle.*

We define the advantage of $\mathcal{A}$ in the AE-ROR security game for π as:

$$\mathsf{Adv}_\pi^{\mathrm{ae}}(\mathcal{A}) = 2 \cdot \Pr\left[\text{AE-ROR}_\pi^{\mathcal{A}} \Rightarrow \mathsf{true}\right] - 1.$$

Unless otherwise stated, our AE schemes will be length-regular, so that the lengths of ciphertexts depend only on the lengths of plaintexts. This ensures that the above definition also implies a standard "left-or-right" security definition.

```
$\mathsf{Enc}(k, m)$
    $x \leftarrow_{\$} \mathcal{K}$
    $\tilde{C} \leftarrow_{\$} \mathcal{E}(k, x)$
    $\overline{C} \leftarrow_{\$} \mathcal{E}(x, m)$
    return $(\tilde{C}, \overline{C})$

$\mathsf{ReKeyGen}(k_1, k_2, \tilde{C})$
    $x = \mathcal{D}(k_1, \tilde{C})$
    if $x = \perp$ return $\perp$
    $\Delta_{1,2,\tilde{C}} \leftarrow_{\$} \mathcal{E}(k_2, x)$
    return $\Delta_{1,2,\tilde{C}}$

$\mathsf{Dec}(k, (\tilde{C}, \overline{C}))$
    $x = \mathcal{D}(k, \tilde{C})$
    if $x = \perp$ return $\perp$
    $m = \mathcal{D}(x, \overline{C})$
    return $m$

$\mathsf{KeyGen}$ : return $\mathcal{K}$
$\mathsf{ReEnc}(\Delta_{1,2,\tilde{C}}, (\tilde{C}, \overline{C}))$ : return $(\Delta_{1,2,\tilde{C}}, \overline{C})$
```

Fig. 3. Algorithms for the AE-hybrid updatable AE scheme.

4.2 (In-)Security of AE-hybrid Construction

Figure 3 defines an updatable AE scheme, AE-hybrid, for any AE scheme $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. This is a natural key-wrapping scheme that one might create in the absence of security definitions. It is preferred by practitioners because key rotation is straightforward and performant. Using this scheme means rekeying requires constant time and communication, independent of the length of the plaintext. In fact, we note that this scheme sees widening deployment for encrypted cloud storage services. Both Amazon Web Services [AWS] and Google Cloud Platform [Goo] use AE-hybrid to perform key rotations over encrypted customer data.

We demonstrate severe limits of AE-hybrid: when keys are compromised confidentiality and integrity cannot be recovered through re-encryption. Later we will demonstrate straightforward modifications to AE-hybrid that allow it to recover both confidentiality and integrity without impacting performance.

Theorem 2 (AE-hybrid insecurity in the UP-IND **sense).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and Π be the updatable AE scheme AE-hybrid built using π as defined in Figure 3.*

Then there exists an adversary $\mathcal{A}$ making 2 queries such that $\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) = 1$ for all $\kappa \geq 1$ and $t \geq 1$.

Proof. We construct a concrete adversary $\mathcal{A}$ satisfying the theorem statement.

$\mathcal{A}$ makes an initial query to $\mathrm{LR}(1, m_0, m_1)$ for distinct messages $m_0 \neq m_1$ and receives challenge ciphertext $C^* = (\mathcal{E}(k_1, x), \mathcal{E}(x, m_b))$. $\mathcal{A}$ subsequently calls $\mathrm{ReKeyGen}(1, t+1, C^*)$. k_{t+1} is corrupted and thus $\mathsf{Invalid}_{\mathsf{RK}}$ returns true, so the adversary receives the re-encrypted ciphertext header $\tilde{C'} = \mathcal{E}(k_{t+1}, x)$.

The adversary decrypts $x = \mathcal{D}(k_{t_1}, \tilde{C'})$, computes $m_b = \mathcal{D}(x, \overline{C}^*)$ and checks whether $m_b = m_0$ or m_1.

The best one can achieve with this scheme is to prove security when $\kappa = 0$, that is, security is not degraded beyond the underlying AE scheme when the adversary does not obtain any compromised keys. However, such a weak security notion is not particularly interesting, since the intention of key rotation is to provide enhanced security in the face of key compromises.

Theorem 3 (UP-IND **Security of AE-hybrid).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and Π be the updatable AE scheme AE-hybrid built using π as defined above. Then for any adversary $\mathcal{A}$ for the game* UP-IND*, there exists an adversary $\mathcal{B}$ for the AE security game where:*

$$\mathsf{Adv}^{\text{up-ind}}_{\Pi,0,t}(\mathcal{A}) \;\leq\; 2(t+1) \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B})$$

for all $t \geq 1$.

Proof. We argue using a series of games that the advantage of any adversary in the UP-IND game for KSS is bounded by the advantage in the AE security game for the underlying AE scheme π. The first game, G_0, is the UP-IND game for KSS using the underlying scheme π. For $1 \leq i \leq t$, in game G_i we replace all ciphertext headers encrypted under key k_i and instead return a random string of the same length used to answer the query. The value of the returned ciphertext header $\tilde{C}$ is stored along with the encrypted value $(\chi \parallel \tau)$ in order to simulate later calls to ReEnc and ReKeyGen.

Let S_i be the event that G_i outputs true. We claim that for $0 < i \leq t$, there exists $\mathcal{B}$ such that:

$$|\Pr[S_{i-1}] - \Pr[S_i]| \leq \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}).$$

We construct $\mathcal{B}$ by using the Enc oracle in the AE security game for π to encrypt ciphertext headers. When the hidden bit $\widehat{b}$ in the AE game is 0, then $\mathcal{B}$ perfectly simulates game G_{i-1} for key k_i and when $\widehat{b} = 1$, then $\mathcal{B}$ perfectly simulates game G_i. $\mathcal{B}$ simply returns as a guess $\widehat{b} = 1$ if the adversary is correct. Any difference between the success probabilities in G_{i-1} and G_i results in an advantage for the adversary.

Now, we observe that the adversary in game G_t cannot learn anything about the DEM key x used to encrypt a challenge. Even through a re-encryption to a corrupted key, the most the adversary can learn is the value $\chi' = x \oplus y'$, where y' is in the ciphertext body and unobtainable by the adversary.

Hence consider another set of hybrids G_j for $t < j \leq t{+}q$, where the adversary makes at most q queries to the left-or-right oracle LR. In the same spirit as the previous hybrids, we construct an AE adversary such that $|\Pr[S_{j-1}] - \Pr[S_j]| \leq \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B})$, this time replacing the encryption of the DEM during a challenge query with the output of the AE encryption oracle.

Finally, observe that in game G_{t+q}, all outputs from LR oracle are of the form (r, χ, z) where r, z are random strings. Since the outputs are unrelated to message inputs, and indeed the hidden bit b, the adversary can learn nothing from oracle queries and so $\Pr[S_{t+q}] = \frac{1}{2}$ and we conclude:

$$\begin{aligned}\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) &= 2 \cdot \Pr[S_0] - 1 \\ &= 2 \cdot (\Pr[S_0] - \Pr[S_1] + \Pr[S_1] + \cdots - \Pr[S_{t+q}] + \Pr[S_{t+q}]) - 1 \\ &\leq 2(t+q) \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}).\end{aligned}$$

Similarly, AE-hybrid is trivially insecure in the UP-INT sense when $\kappa \geq 1$.

Theorem 4 (AE-hybrid insecurity in the UP-INT **sense).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathit{\Pi}$ be the updatable AE scheme AE-hybrid built using π as defined in Figure 3.*

Then there exists an adversary $\mathcal{A}$ making 2 queries and one Try *query such that $\mathsf{Adv}^{\text{up-int}}_{\mathit{\Pi},\kappa,t}(\mathcal{A}) = 1$ for all $\kappa \geq 1$ and $t \geq 1$.*

Proof. We construct a concrete adversary $\mathcal{A}$ satisfying the theorem statement.

$\mathcal{A}$ first queries $\mathrm{Enc}(1, m)$ to obtain an encryption $C = (\mathcal{E}(k_1, x), \mathcal{E}(x, m))$, and subsequently queries $\mathrm{ReEnc}(1, t+1, C)$, receiving the re-encryption $C' = (\mathcal{E}(k_{t+1}, x), \mathcal{E}(x, m))$. Since $\mathcal{A}$ has key k_{t+1}, $\mathcal{A}$ recovers $x = \mathcal{D}(k_{t+1}, \tilde{C}')$ by performing the decryption locally.

Finally, $\mathcal{A}$ constructs the ciphertext $C^* = (\tilde{C}, \mathcal{E}(x, m'))$ for some $m' \neq m$ and queries $\mathrm{Try}(1, C^*)$. Since C^* is not derived from C and k_1 is not compromised, UP-INT outputs true.

Theorem 5 (UP-INT **Security of AE-hybrid).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathit{\Pi}$ be the AE-hybrid scheme built using π as defined above. Then for any adversary $\mathcal{A}$ for the game* UP-INT *there exists an adversary $\mathcal{B}$ for the AE security game where:*

$$\mathsf{Adv}^{\text{up-int}}_{\mathit{\Pi},0,t}(\mathcal{A}) \;\leq\; (t+1) \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B})$$

for all $t \geq 1$.

Proof. We use the same sequence of games as in the previous proof, leveraging the integrity properties of authenticated encryption.

In game G_i for $1 \leq i \leq t$, for any query (j, C) to the Try oracle where $j \leq i$, the decryption of the ciphertext header $\tilde{C}$ will always decrypt to $\perp$, unless $\tilde{C}$ was a previously generated random string output by the encryption oracle.

Hence in game G_t, the adversary can only win by re-using a previously seen header $\tilde{C}$.

Therefore, the adversary can only win by submitting a ciphertext of the form: $(\tilde{C}, \overline{C}')$, where $\overline{C}' \neq \overline{C}$, the ciphertext body returned in the same query as $\tilde{C}$.

Let $\overline{C} = (y, \overline{C}^1)$ and thus we require $\overline{C}' = (y', \mathcal{E}(\chi \oplus y', m'))$ for either $y' \neq y$ or $\mathcal{E}(\chi \oplus y', m') \neq \overline{C}^1$. Note that the adversary can learn the value of x, y used in $\overline{C}$ by querying $\mathrm{ReKeyGen}(i, t+1, \tilde{C})$ and receiving $\mathcal{E}(k_{t+1}, \chi \parallel h(\overline{C}^1)), y$. From this, they can simply decrypt and learn $x = \chi \oplus y$

To succeed, the adversary needs to find y', m' such that $h(m') = \mathcal{D}(\chi \oplus y', \mathcal{E}(x, h(m)))$. Modelling h as a random oracle, for any value of $y' \neq y$, the probability that the adversary finds an input z to h, such that $h(z) = \mathcal{D}(\chi \oplus y', \mathcal{E}(x, h(m)))$ is $\frac{1}{2^{\ell_h}}$. If such a value z is found, the adversary can compute $m' = \mathcal{D}(\chi \oplus y, z)$. However, the probability that the adversary, making at most q_h oracle queries to h finds such a value is given by $\frac{q_h}{2^{\ell_h}}$.

Alternatively, for $y = y'$, the adversary needs to find a message $m' \neq m$ such that $h(m') = h(m)$. Again, modelling h as a random oracle, the probability that

```
$\mathsf{Enc}(k, m)$
$x, y \leftarrow_{\$} \mathcal{K}$
$\chi = x \oplus y$
$\overline{C}^1 \leftarrow_{\$} \mathcal{E}(x, m)$
$\tau = \mathcal{E}(x, h(m))$
$\tilde{C} \leftarrow_{\$} \mathcal{E}(k, \chi \parallel \tau)$
return $(\tilde{C}, (y, \overline{C}^1))$
```

```
$\mathsf{ReKeyGen}(k_1, k_2, \tilde{C})$
$(\chi \parallel \tau) = \mathcal{D}(k_1, \tilde{C})$
if $(\chi \parallel \tau) = \perp$ return $\perp$
$y' \leftarrow_{\$} \mathcal{K}$
return $(y', \mathcal{E}(k_2, (\chi \oplus y') \parallel \tau))$
```

```
$\mathsf{Dec}(k, (\tilde{C}, \overline{C}))$
$(\chi \parallel \tau) = \mathcal{D}(k, \tilde{C})$
if $(\chi \parallel \tau) = \perp$
    return $\perp$
$x = \chi \oplus \overline{C}^0$
$m = \mathcal{D}(x, \overline{C}^1)$
if $\mathcal{D}(x, \tau) \neq h(m)$
    return $\perp$
return $m$
```

```
$\mathsf{KeyGen}()$ : return $k \leftarrow \mathcal{K}$
$\mathsf{ReEnc}(\Delta_{1,2,\tilde{C}}, (\tilde{C}, \overline{C}))$ : return $(\Delta^1_{1,2,\tilde{C}}, (\overline{C}^0 \oplus \Delta^0_{1,2,\tilde{C}}, \overline{C}^1))$
```

Fig. 4. Algorithms for the KSS updatable AE scheme.

the adversary, making at most q_h oracle queries to h, finds a collision is given by $\frac{q_h^2}{2^{\ell_h}}$.

Since this probability is greater than the probability of finding a pre-image when $y' \neq y$, the probability that the adversary wins in G_{t+1} is bounded by this probability of finding a collision, and we conclude that:

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) \;\leq\; t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + \frac{q_h^2}{2^{\ell_h}} .$$

4.3 Improving AE-hybrid

We make small modifications to the AE-hybrid construction and show that the resulting construction has both UP-IND and UP-INT security. These modifications include masking the DEM key stored inside the ciphertext header (to gain UP-IND security), and including an encrypted hash of the message (for UP-INT). We note that these modifications are straightforward to implement on top of the AE-hybrid scheme and have only minimal impact on the scheme's performance in practice.

Let $(\mathcal{K}$,$\mathcal{E}$,$\mathcal{D})$ be an AE scheme and h a hash function with ℓ_h output bits. Then we define KSS (KEM/DEM with Secret Sharing) as in Figure 4.

Theorem 6 (UP-IND **Security of** KSS)**.** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and Π be the updatable AE scheme* KSS *built using π as defined in Figure 4. Then for any adversary $\mathcal{A}$ for the game* UP-IND*, making at most q queries to the* LR *oracle, there exists an adversary $\mathcal{B}$ for the AE security game where:*

$$\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) \;\leq\; 2(t+q) \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B})$$

for all $\kappa \geq 0, t \geq 1$.

Proof. We argue using a series of games that the advantage of any adversary in the UP-IND game for KSS is bounded by the advantage in the AE security game for the underlying AE scheme π. The first game, G_0, is the UP-IND game for KSS using the underlying scheme π. For $1 \leq i \leq t$, in game G_i we replace all ciphertext headers encrypted under key k_i and instead return a random string of the same length used to answer the query. The value of the returned ciphertext header $\tilde{C}$ is stored along with the encrypted value $(\chi \parallel \tau)$ in order to simulate later calls to ReEnc and ReKeyGen.

Let S_i be the event that G_i outputs true. We claim that for $0 < i \leq t$, there exists $\mathcal{B}$ such that:

$$|\Pr[S_{i-1}] - \Pr[S_i]| \leq \mathsf{Adv}^{\mathrm{ae}}_{\pi}(\mathcal{B}).$$

We construct $\mathcal{B}$ by using the Enc oracle in the AE security game for π to encrypt ciphertext headers. When the hidden bit $\widehat{b}$ in the AE game is 0, then $\mathcal{B}$ perfectly simulates game G_{i-1} for key k_i and when $\widehat{b} = 1$, then $\mathcal{B}$ perfectly simulates game G_i. $\mathcal{B}$ simply returns as a guess $\widehat{b} = 1$ if the adversary is correct. Any difference between the success probabilities in G_{i-1} and G_i results in an advantage for the adversary.

Now, we observe that the adversary in game G_t cannot learn anything about the DEM key x used to encrypt a challenge. Even through a re-encryption to a corrupted key, the most the adversary can learn is the value $\chi' = x \oplus y'$, where y' is in the ciphertext body and unobtainable by the adversary.

Hence consider another set of hybrids G_j for $t < j \leq t+q$, where the adversary makes at most q queries to the left-or-right oracle LR. In the same spirit as the previous hybrids, we construct an AE adversary such that $|\Pr[S_{j-1}] - \Pr[S_j]| \leq \mathsf{Adv}^{\mathrm{ae}}_{\pi}(\mathcal{B})$, this time replacing the encryption of the DEM during a challenge query with the output of the AE encryption oracle.

Finally, observe that in game G_{t+q}, all outputs from LR oracle are of the form (r, χ, z) where r, z are random strings. Since the outputs are unrelated to message inputs, and indeed the hidden bit b, the adversary can learn nothing from oracle queries and so $\Pr[S_{t+q}] = \frac{1}{2}$ and we conclude:

$$\begin{aligned}\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) &= 2 \cdot \Pr[S_0] - 1 \\ &= 2 \cdot (\Pr[S_0] - \Pr[S_1] + \Pr[S_1] + \cdots - \Pr[S_{t+q}] + \Pr[S_{t+q}]) - 1 \\ &\leq 2(t+q) \cdot \mathsf{Adv}^{\mathrm{ae}}_{\pi}(\mathcal{B}).\end{aligned}$$

Our modification to include an encrypted hash of the ciphertext is in order to provide a measure of integrity protection. As we will see in the following theorem, collision resistance of the hash function is sufficient to provide UP-INT security, since the hash itself is integrity-protected by the AE encryption of the KEM. The hash itself is encrypted in order to avoid compromise of the ciphertext header being sufficient to distinguish messages.

We achieve collision resistance by assuming h to be a random oracle. However, this assumption could be avoided by either re-using the DEM key x to additionally key the hash function.

We note that this combination of hash function and AE encryption is used to provide an additional integrity mechanism that works for any AE scheme. However, some schemes may be able to avoid this additional computation by re-using components of the AE encryption. For example, if an encrypt-then-MAC scheme is used such that the encryption and MAC keys are both uniquely derived from the DEM key x, then we conjecture that the MAC itself can be used in place of the encrypted hash.

Theorem 7 (UP-INT **Security of** KSS)**.** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme, h be a cryptographic hash function modelled as a random oracle with output length ℓ_h, and Π be the updatable AE scheme* KSS *built using π and h as defined in Figure 4. Then for any polynomial-time adversary $\mathcal{A}$, making at most q_h queries to the random oracle h, there exists an adversary $\mathcal{B}$ for the AE security game where:*

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) \;\leq\; t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + \frac{q_h^2}{2^{\ell_h}}$$

for all $\kappa \geq 0, t \geq 1$.

Proof. We use the same sequence of games as in the previous proof, leveraging the integrity properties of authenticated encryption.

In game G_i for $1 \leq i \leq t$, for any query (j, C) to the Try oracle where $j \leq i$, the decryption of the ciphertext header $\tilde{C}$ will always decrypt to $\perp$, unless $\tilde{C}$ was a previously generated random string output by the encryption oracle.

Hence in game G_t, the adversary can only win by re-using a previously seen header $\tilde{C}$.

Therefore, the adversary can only win by submitting a ciphertext of the form: $(\tilde{C}, \overline{C}')$, where $\overline{C}' \neq \overline{C}$, the ciphertext body returned in the same query as $\tilde{C}$.

Let $\overline{C} = (y, \overline{C}^1)$ and thus we require $\overline{C}' = (y', \mathcal{E}(\chi \oplus y', m'))$ for either $y' \neq y$ or $\mathcal{E}(\chi \oplus y', m') \neq \overline{C}^1$. Note that the adversary can learn the value of x, y used in $\overline{C}$ by querying ReKeyGen$(i, t+1, \tilde{C})$ and receiving $\mathcal{E}(k_{t+1}, \chi \parallel h(\overline{C}^1)), y$. From this, they can simply decrypt and learn $x = \chi \oplus y$

To succeed, the adversary needs to find y', m' such that $h(m') = \mathcal{D}(\chi \oplus y', \mathcal{E}(x, h(m)))$. Modelling h as a random oracle, for any value of $y' \neq y$, the probability that the adversary finds an input z to h, such that $h(z) = \mathcal{D}(\chi \oplus y', \mathcal{E}(x, h(m)))$ is $\frac{1}{2^{\ell_h}}$. If such a value z is found, the adversary can compute $m' = \mathcal{D}(\chi \oplus y, z)$. However, the probability that the adversary, making at most q_h oracle queries to h finds such a value is given by $\frac{q_h}{2^{\ell_h}}$.

Alternatively, for $y = y'$, the adversary needs to find a message $m' \neq m$ such that $h(m') = h(m)$. Again, modelling h as a random oracle, the probability that the adversary, making at most q_h oracle queries to h, finds a collision is given by $\frac{q_h^2}{2^{\ell_h}}$.

Since this probability is greater than the probability of finding a pre-image when $y' \neq y$, the probability that the adversary wins in G_{t+1} is bounded by this probability of finding a collision, and we conclude that:

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) \;\leq\; t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + \frac{q_h^2}{2^{\ell_h}}\,.$$

5 Indistinguishability of Re-encryptions

The KSS scheme in the previous section achieves message confidentiality and ciphertext integrity, even though the actual DEM key is not modified in the course of performing a rotation. Modifying the scheme to ensure the DEM key is also rotated is non-trivial, requiring either significant communication complexity (linear in the length of the encrypted message) between the key server and storage, or the introduction of more advanced primitives such as key- homomorphic PRFs. The question that arises is whether or not changing DEM keys leaves KSS vulnerable to attacks not captured by the definitions introduced thus far.

BLMR's brief treatment of updatable encryption attempts to speak to this issue by requiring that all randomness be refreshed during a rotation. Intuitively this would seem to improve security, but the goal they formalize for this, detailed below, is effectively a correctness condition (i.e., it does not seem to account for adversarial behaviors). It doesn't help clarify what attacks would be ruled out by changing DEM keys.

Exfiltration attacks. We identify an issue with our KSS scheme (and the other schemes in the preceding section) in the form of an attack that is not captured by the confidentiality definitions introduced so far. Consider our simple KSS scheme in the context of our motivating key server and storage service application (described in Section 2). Suppose an attacker compromises for some limited time both the key server and the storage service. Then for each ciphertext $(\tilde{C}, \overline{C})$ encrypted under a key k_1, the attacker can compute the DEM key $y \oplus \chi = x$ and exfiltrate it.

Suppose the compromise is cleaned up, and the service immediately generates new keys and rotates all ciphertexts to new secret keys. For the KSS scheme, the resulting ciphertexts will still be later decryptable using the previously exfiltrated DEM keys.

Although a confidentiality issue — the attacker later obtains access to plaintext data they should not have — our UP-IND security notion (and, by implication, the weaker BLMR confidentiality notion) do not capture these attacks. Technically this is because the security game does not allow a challenge ciphertext to be encrypted to a compromised key (or rotated to one). Intuitively, the UP-IND notion gives up on protecting the plaintexts underlying such ciphertexts, as the attacker in the above scenario already had access to the plaintext in the first phase of the attack.

One might therefore argue that this attack is not very important. All of the plaintext data eventually at risk of later decryption was already exposed to the adversary in the first time period because she had access to both the key and ciphertexts. But quantitatively there is a difference: for a given ciphertext an adversary in the first time period can exfiltrate just $|x|$ bits per ciphertext

```
UP-REENC
$b \leftarrow_{\$} \{0,1\}$
$k_1, \ldots, k_{t+\kappa} \leftarrow_{\$} \mathsf{KeyGen}()$
$b' \leftarrow_{\$} \mathcal{A}^{O}(k_{t+1}, \ldots, k_{t+\kappa})$
return $(b' = b)$
```

```
Enc$(i, m)$
return $\mathsf{Enc}(k_i, m)$
```

```
ReKeyGen$(i, j, \tilde{C})$
if $\mathsf{Invalid}_{\mathsf{RK}}(i, j, \tilde{C})$ then
  return $\perp$
$\Delta_{i,j,\tilde{C}} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
return $\Delta_{i,j,\tilde{C}}$
```

```
ReEnc$(i, j, (\tilde{C}, \overline{C}))$
$\Delta_{i,j,\tilde{C}} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
$C' = (\tilde{C}', \overline{C}') \leftarrow \mathsf{ReEnc}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$
if $\mathsf{Invalid}_{\mathsf{RE}}(i, j, \tilde{C})$ then
  return $\tilde{C}'$
else
  return $C'$
```

```
ReLR$(i, j, C_0, C_1)$
if $j > t$ or $|C_0| \neq |C_1|$ then
  return $\perp$
for $\beta \in \{0,1\}$ do
    $\Delta_{i,j,\tilde{C}_\beta} \leftarrow_{\$} \mathsf{ReKeyGen}(k_i, k_j, \tilde{C}_\beta)$
    $C'_\beta \leftarrow \mathsf{ReEnc}(\Delta_{i,j,\tilde{C}_\beta}, C_\beta)$
    if $C'_\beta = \perp$ then return $\perp$
return $C'_b$
```

Fig. 5. The game used to define re-encryption indistinguishability.

to later recover as much plaintext as she likes, whereas the trivial attack may require exfiltrating the entire plaintext.

The chosen-message attack game of UP-IND does not capture different time periods in which the adversary knows plaintexts in the first time period but "forgets them" in the next. One could explicitly model this, perhaps via a two-stage game with distinct adversaries in each stage, but such games are complex and often difficult to reason about (cf., [RSS11]). We instead develop what we believe is a more intuitive route that asks that the re-encryption of a ciphertext should leak nothing about the *ciphertext* that was re-encrypted. We use an indistinguishability-style definition to model this. The interpretation of our definition is that any information derivable from a ciphertext (and its secret key) before a re-encryption isn't helpful in attacking the re-encrypted version.

Re-encryption indistinguishability. We formalize this idea via the game shown in Figure 5. The adversary is provided with a left-or-right *re-encryption* oracle, ReLR, instead of the usual left-or-right encryption oracle, in addition to the usual collection of compromised keys, a re-encryption oracle, encryption oracle, and rekey token generation oracle. We assume that the adversary always submits ciphertext pairs such that $|C_0| = |C_1|$.

To avoid trivial wins, the game must disallow the adversary from simply re-encrypting the challenge to a corrupted key. Hence we define a $\mathsf{Derived}_{\mathsf{ReLR}}$ predicate, which is identical to the $\mathsf{Derived}_{\mathsf{LR}}$ predicated defined in Section 3 for

UP-IND security, except that it uses the ReLR challenge oracle. We give it in full detail in the next definition.

Definition 6 (ReLR-derived headers). *We recursively define the function* $\mathsf{Derived}_{\mathsf{ReLR}}(i, \tilde{C})$ *to output* true *iff* $\tilde{C} \neq \varepsilon$ *and any of the following conditions hold:*

- $\tilde{C}$ *was the ciphertext header output in response to a query* $\mathrm{ReLR}(i, C_0, C_1)$*.*
- $\tilde{C}$ *was the ciphertext header output in response to a query* $\mathrm{ReEnc}(j, i, C')$ *and* $\mathsf{Derived}_{\mathsf{ReLR}}(j, \tilde{C}') = \mathsf{true}$*.*
- $\tilde{C}$ *is the ciphertext header output by running* $\mathsf{ReEnc}(\Delta_{j,i,\tilde{C}'}, C')$ *where* $\Delta_{j,i,\tilde{C}'}$ *is the result of a query* $\mathrm{ReKeyGen}(j, i, C')$ *for which* $\mathsf{Derived}_{\mathsf{ReLR}}(j, \tilde{C}') = \mathsf{true}$*.*

Then the subroutines $\mathsf{Invalid}_{\mathsf{RK}}, \mathsf{Invalid}_{\mathsf{RE}}$ used in the game output true if $\mathsf{Derived}_{\mathsf{ReLR}}(i, \tilde{C})$ outputs true and $j > t$. We associate to an updatable encryption scheme Π, UP-REENC adversary $\mathcal{A}$, and parameters κ, t the advantage measure:

$$\mathsf{Adv}^{\text{up-reenc}}_{\Pi,\kappa,t}(\mathcal{A}) = 2 \cdot \Pr\left[\text{UP-REENC}^{\mathcal{A}}_{\Pi,\kappa,t} \Rightarrow \mathsf{true}\right] - 1 \; .$$

Informally, an updatable encryption scheme is UP-REENC secure if no adversary can achieve advantage far from zero given reasonable resources (run time, queries, and number of target keys).

Notice that exfiltration attacks as discussed informally above would not apply to a scheme that meets UP-REENC security. Suppose otherwise, that the exfiltration still worked. Then one could build an UP-REENC adversary that worked as follows. It obtains two encryptions of different messages under a compromised key, calculates the DEM key (or whatever other information is useful for later decryption) and then submits the ciphertexts to the ReLR oracle, choosing as target a non-compromised key ($j \leq t$). Upon retrieving the ciphertext, it uses the DEM key to decrypt, and checks which message was encrypted. Of course our notion covers many other kinds of attacks, ruling out even re-encryption that allows a single bit of information to leak.

BLMR re-encryption security. BLMR introduced a security goal that we will call basic re-encryption indistinguishability.[1] In words, it asks that the distribution of a ciphertext and its re-encryption should be identical to the distribution of a ciphertext and a re-encryption of a distinct ciphertext of the same message. More formally we have the following pair of experiments, each parameterized by a message m.

[1] BLMR called this ciphertext independence, but we reserve that terminology for schemes that do not require ciphertexts during token generation as per Section 2.

```
$\text{UP-REENC01}_m$
$k_1, k_2 \leftarrow_\$ \mathsf{KeyGen}()$
$C_0 \leftarrow_\$ \mathsf{Enc}(k_1, m), C_1 \leftarrow_\$ \mathsf{Enc}(k_1, m)$
$\Delta_{1,2,\tilde{C}_1} \leftarrow_\$ \mathsf{ReKeyGen}(k_1, k_2, \tilde{C}_1)$
$C_1' \leftarrow_\$ \mathsf{ReEnc}(\Delta_{1,2,\tilde{C}_1}, C_1)$
return $(C_0, C_1')$
```

```
$\text{UP-REENC00}_m$
$k_1, k_2 \leftarrow_\$ \mathsf{KeyGen}()$
$C_0 \leftarrow_\$ \mathsf{Enc}(k_1, m), C_1 \leftarrow_\$ \mathsf{Enc}(k_1, m)$
$\Delta_{1,2,\tilde{C}_0} \leftarrow_\$ \mathsf{ReKeyGen}(k_1, k_2, \tilde{C}_0)$
$C_0' \leftarrow_\$ \mathsf{ReEnc}(\Delta_{1,2,\tilde{C}_0}, C_0)$
return $(C_0, C_0')$
```

Then BLMR require that for all m and all ciphertext pairs (C, C')

$$\left|\Pr[\text{UP-REENC00}_m \Rightarrow (C, C')] - \Pr[\text{UP-REENC01}_m \Rightarrow (C, C')]\right| = 0$$

where the probabilities are over the coins used in the experiments.

This goal misses a number of subtleties which are captured by our definition. Our definition permits the adversary, for example, to submit *any* pair of ciphertexts to the ReLR oracle. This includes ciphertexts which are encryptions of distinct messages, and even maliciously formed ciphertexts which may not even decrypt correctly. It is simple to exhibit a scheme that meets the BLMR notion but trivially is insecure under ours.[2]

On the other hand, suppose a distinguisher exists that can with some probability ϵ distinguish between the outputs UP-REENC00_m and UP-REENC01_m for some m. Then there exists an adversary against our UP-REENC notion which achieves advantage ϵ. This can be seen by the following simple argument. The adversary gets $C \leftarrow_\$ \mathsf{Enc}(1, m), C' \leftarrow_\$ \mathsf{Enc}(1, m)$ and submits the tuple $(1, 2, C, C')$ to its ReLR oracle and receives a re-encryption of one of the ciphertexts, C^*. The adversary then runs the distinguisher on (C, C^*) and outputs whatever the distinguisher guesses. If the distinguisher is computationally efficient, then so too is the UP-REENC adversary. Thus our UP-REENC notion would be stronger than a computational version of the BLMR notion.

6 Revisiting the BLMR Scheme

The fact that the simple KEM/DEM schemes of Section 4 fail to meet re-encryption security begs the question of finding new schemes that achieve it, as well as UP-IND and UP-INT security. Our starting point is the BLMR construction of an updatable encryption from key-homomorphic PRFs. Their scheme does not (nor did it attempt to) provide integrity guarantees, and so trivially does not meet UP-INT. But before seeing how to adapt it to become suitable as an updatable AE scheme, including whether it meets our stronger notions

[2] Such a scheme can be constructed by starting with a scheme that satisfies both security notions and adding a "counter" component to ciphertexts that records how many re-encryptions have been performed to obtain that ciphertext; one now exploits the property that any pair of ciphertexts can be input to the ReLR oracle in our UP-REENC game, while only fresh ciphertexts C_0, C_1 are rotated in the BLMR notion.

of UP-IND and UP-REENC security, we first revisit the claims of UP-IND-BI security from [BLMR15].

As mentioned in the introduction, BLMR claim that the scheme can be shown secure, and sketch a proof of UP-IND-BI security. Unfortunately the proof sketch contains a bug, as we explain below. Interestingly revelation of this bug does not lead to a direct attack on the scheme, and at the same time we could not determine if the proof could be easily repaired. Instead we are able to show that a proof is unlikely to exist.

Our main result of this section is the following: giving a proof showing the BLMR UP-IND-BI security would imply the existence of a reduction showing that (standard) IND-CPA security implies circular security [BRS03,CL01] for a simple KEM/DEM style symmetric encryption scheme. The latter seems quite unlikely given the known negative results about circular security [ABBC10,CGH12], suggesting that the BLMR scheme is not likely to be provably secure.

First we recall some basic tools that BLMR use to build their scheme.

Definition 7 (Key-homomorphic PRF [BLMR15]). *Consider an efficiently computable function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ such that $(\mathcal{K}, \oplus)$ and $(\mathcal{Y}, \otimes)$ are both groups. We say that the tuple $(F, \oplus, \otimes)$ is a* key-homomorphic PRF *if the following properties hold:*

1. *F is a secure pseudorandom function.*
2. *For every $k_1, k_2 \in \mathcal{K}$ and every $x \in \mathcal{X}$, $F(k_1, x) \otimes F(k_2, x) = F(k_1 \oplus k_2, x)$.*

A simple example in the ROM is the function $F(k, x) = H(x)^k$ where $\mathcal{Y} = \mathbb{G}$ is a group in which the decisional Diffie–Hellman assumption holds.

As an application of key-homomorphic PRFs, BLMR proposed the following construction. The construction follows a similar approach to the AE-hybrid scheme, but by using a key-homomorphic PRF in place of regular encryption the data encryption key can also be rotated.

Definition 8 (BLMR scheme). *Let π be a symmetric-key IND-CPA encryption scheme $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$. Furthermore, let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a key-homomorphic PRF where $(\mathcal{K}, +)$ and $(\mathcal{Y}, +)$ are groups. The BLMR scheme is the tuple of algorithms (KeyGen, Enc, ReKeyGen, ReEnc, Dec) depicted in Figure 6.*

Note that encryption in the BMLR scheme is a key wrap followed by CTR mode encryption using the wrapped key x and PRF F.

6.1 Negative Result about Provable UP-IND Security of BLMR

BLMR sketch a proof for the security of this construction in the UP-IND-BI model (as we refer to it). However, the proof misses a subtle point: the interaction with the ReKeyGen oracle behaves similarly to a decryption oracle and the informal argument given that the IND-CPA security of the KEM is sufficient to argue security is wrong. In fact, the BLMR scheme seems unlikely to be

```
$\mathsf{KeyGen}()$
    $k \leftarrow_\$ \mathcal{KG}()$
    return $k$
```

```
$\mathsf{Enc}(k, m)$
    $x \leftarrow_\$ \mathcal{K}$
    $\tilde{C} \leftarrow_\$ \mathcal{E}(k, x)$
    $\overline{C} = (m_1 + F(x, 1), \ldots, m_\ell + F(x, \ell))$
    return $C = (\tilde{C}, \overline{C})$
```

```
$\mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
    $x = \mathcal{D}(k_i, \tilde{C})$
    $x' \leftarrow_\$ \mathcal{K}$
    $\tilde{C}' = \mathcal{E}(k_j, x')$
    $\Delta_{i,j,\tilde{C}} = (\tilde{C}', x' - x)$
    return $\Delta_{i,j,\tilde{C}}$
```

```
$\mathsf{ReEnc}(\Delta_{i,j,\tilde{C}}, C)$
    $(\tilde{C}, \overline{C}) = C$
    $(\tilde{C}', y) = \Delta_{i,j,\tilde{C}}$
    $\overline{C}' = (\overline{C}_1 + F(y, 1), \ldots, \overline{C}_\ell + F(y, \ell))$
    return $C = (\tilde{C}', \overline{C}')$
```

```
$\mathsf{Dec}(k, C)$
    $(\tilde{C}, \overline{C}) = C$
    $x = \mathcal{D}(k, \tilde{C})$
    $m = (\overline{C}_1 - F(x, 1), \ldots, \overline{C}_\ell - F(x, \ell))$
    return $m$
```

Fig. 6. The BLMR scheme.

provably secure even in our basic security model. To argue this, we show that proving security of the BLMR scheme implies the 1-circular security of a specific KEM/DEM construction. Figure 7 depicts the security game capturing a simple form of 1-circular security for an encryption scheme $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$.

While our main result here (Theorem 8), can be stated for the BLMR scheme as described earlier, for the sake of simplicity we instead give the result for the special case of using a simple one-time pad DEM instead of the key-homomorphic PRF. This is a trivial example of what BLMR call a key-homomorphic PRG, and their theorem statement covers this construction as well. We will show that proving security for this special case is already problematic, and this therefore suffices to call into question their (more general) theorem. Thus encryption becomes $\mathsf{Enc}(k, m) = \mathcal{E}(k, r), r \oplus m$ where $\mathcal{E}$ is an IND-CPA secure KEM. We assume $|m| = n$. We then have $\mathsf{ReKeyGen}(k_1, k_2, \tilde{C}) = (\mathcal{E}(k_2, r'), r' \oplus \mathcal{D}(k_1, \tilde{C}))$. We have the following theorem:

Theorem 8. *If one can reduce the* BLMR UP-IND-BI-*security to the IND-CPA security of $\mathcal{E}$, then one can show a reduction that* Enc *is 1-circular secure assuming $\mathcal{E}$ is IND-CPA.*

Proof. We start by introducing a slight variant of $\mathcal{E}$, denoted $\overline{\mathcal{E}}$, shown in Figure 7. It adds a bit to the ciphertext[3] that is read during decryption: if the bit is 1 then decryption outputs the secret key xor'd with the plaintext. Let EncBad be the same as Enc above but using $\overline{\mathcal{E}}$, i.e., $\mathsf{EncBad}(k, m) = \overline{\mathcal{E}}(k, r), r \oplus m$ and $\mathsf{ReKeyGenBad}(k_1, k_2, C) = \overline{\mathcal{E}}(k_2, r'), r' \oplus \overline{\mathcal{D}}(k_1, C)$.

[3] Notice that this scheme is not tidy in the sense of [NRS14]. While that doesn't affect the implications of our analysis — BLMR make no assumptions about tidiness — finding a tidy counter-example is an interesting open question.

```
Game 1-circular
$b \leftarrow_{\$} \{0,1\}$
$k \leftarrow_{\$} \mathcal{KG}()$
if $b = 1$ then
    $C \leftarrow_{\$} \mathsf{Enc}(k,k)$
else
    $U \leftarrow_{\$} \{0,1\}^n$
    $C \leftarrow_{\$} \mathsf{Enc}(k,U)$
$b' \leftarrow_{\$} \mathcal{A}(C)$
return $(b = b')$
```

```
$\overline{\mathcal{E}}(k,m)$
return $\mathcal{E}(k,m)\|\ 0$

$\overline{\mathcal{D}}(k,C\|\ b)$
if $(b = 0)$ then
    return $\mathcal{D}(k,C)$
else
    return $k \oplus \mathcal{D}(k,C)$
```

Fig. 7. Left: The 1-circular security game. Right: Definition of $\overline{\mathcal{E}}, \overline{\mathcal{D}}$ used in the proof of Theorem 8.

```
$\mathcal{B}^{\mathrm{LR,ReKeyGen}}$
$U \leftarrow_{\$} \{0,1\}^n$
$(\tilde{C}\|0, \overline{C}) \leftarrow_{\$} \mathrm{LR}(1,U,0^n)$
$(\tilde{C}'\|0, \overline{C}') \leftarrow_{\$} \mathsf{ReKeyGen}(1,1,\tilde{C}\|\ 1)$
$b' \leftarrow_{\$} \mathcal{A}(\tilde{C}'\|0, \overline{C} + \overline{C}')$
return $b'$
```

Fig. 8. Adversary $\mathcal{B}$ for UP-IND using as a subroutine the adversary $\mathcal{A}$ attacking 1-circular security of EncBad.

If $\mathcal{E}$ is IND-CPA then $\overline{\mathcal{E}}$ is as well. Thus if $\overline{\mathcal{E}}$ is IND-CPA, then the security claim of BLMR implies that EncBad is UP-IND-BI. We will now show that UP-IND-BI security of EncBad implies the 1-circular security of EncBad. In turn it's easy to see that if EncBad is 1-circular secure then so too is Enc, and, putting it all together, the claim of BLMR implies a proof that IND-CPA of $\mathcal{E}$ gives 1-circular security of Enc.

It remains to show that UP-IND-BI security implies EncBad 1-circular security. Let $\mathcal{A}$ be a 1-circular adversary against EncBad. Then we build an adversary $\mathcal{B}$ against the UP-IND security of EncBad. It is shown in Figure 8. The adversary makes an LR query on a uniform message and the message 0^n. If the UP-IND-BI challenge bit is 1 then it gets back a ciphertext $C_1 = (\overline{\mathcal{E}}(k_1, r)\|0, r \oplus U)$ and if it is 0 then $C_0 = (\overline{\mathcal{E}}(k_1, r)\|0, r)$. Next it queries ReKeyGen oracle on the first component of the returned ciphertext but with the trailing bit switched to 1. It asks for a rekey token for rotating from k_1 back to k_1. The value returned by this query is equal to $\overline{\mathcal{E}}(k_1, r')\|0, r' \oplus k_1 \oplus r$. By XOR'ing the second component with the second component returned from the LR query the adversary gets finally a ciphertext that is, in the left world, the encryption of k_1 under itself and, in the right world, the encryption of a uniform point under k_1. Adversary $\mathcal{B}$ runs a 1-circular adversary $\mathcal{A}$ on the final ciphertext and outputs whatever $\mathcal{A}$ outputs.

The above result uses 1-circular security for simplicity of presentation, but one can generalize the result to longer cycles by making more queries.

The result is relative, only showing that a proof of BLMR's claim implies another reduction between circular security and IND-CPA security for the particular KEM/DEM scheme Enc above. It is possible that this reduction exists, however it seems unlikely. Existing counter-examples show IND-CPA schemes that are not circular-secure [KRW15]. While these counter-examples do not have the same form as the specific scheme under consideration, it may be that one can build a suitable counter-example with additional effort.

7 An Updatable AE Scheme with Re-encryption Indistinguishability

We first point out that one can avoid the issues raised in Section 6 by replacing the IND-CPA KEM with a proper AE scheme. This does not yet, however, address integrity of the full encryption scheme. To provide integrity overall, we can include a hash of the message in the ciphertext header. However, to prevent this from compromising confidentiality during re-keying, we further mask the hash by an extra PRF output.

This amended construction — which we refer to as ReCrypt — is detailed in Figure 9. It uses an AE scheme $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$, a key-homomorphic PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, and a hash function $h : \{0,1\}^* \to \mathcal{Y}$.

In the remainder of this section we show that the new scheme meets our strongest security notions for updatable encryption. We then assess the viability of using this scheme in practice, discussing how to instantiate F for high performance and reporting on performance of the full scheme.

7.1 Security of ReCrypt

We state three security theorems for ReCrypt: UP-IND, UP-INT, and UP-REENC notions . The proof of UP-INT relies on the collision resistance of the hash h, while the other two proofs do not. For simplicity, and because we will later instantiate the PRF F in the Random Oracle Model (ROM), we model h as a random oracle throughout our analysis. This modelling of h could be avoided using the approach of Rogaway [Rog06], since concrete collision-producing adversaries can be be extracted from our proofs. Note also that the *almost* key-homomorphic PRF construction in the standard model presented by BLMR would not achieve UP-REENC since the number of re-encryptions is leaked by the ciphertext, allowing an adversary to distinguish two re-encryptions.

Theorem 9 (UP-IND **security of** ReCrypt**).** *Let $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ be an AE scheme, $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a key-homomorphic PRF, and let Π be the* ReCrypt *scheme as depicted in Figure 9.*

Then for any adversary $\mathcal{A}$ against Π, there exist adversaries $\mathcal{B}$, $\mathcal{C}$ such that

$$\mathsf{Adv}^{\text{up-ind}}_{\Pi,\kappa,t}(\mathcal{A}) \leq 2t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + 2 \cdot \mathsf{Adv}^{\text{prf}}_{F}(\mathcal{C})$$

```
$\mathsf{KeyGen}()$
    $k \leftarrow_{\$} \mathcal{KG}()$
    return $k$
```

```
$\mathsf{Enc}(k, m)$
    $x, y \leftarrow_{\$} \mathcal{K}$
    $\chi = x + y$
    $\tau = h(m) + F(x, 0)$
    $\tilde{C} = \mathcal{E}(k, (\chi, \tau))$
    for $1 \leq l \leq \ell$
        $\overline{C}_l = m_l + F(x, l)$
    return $(\tilde{C}, \overline{C} = (y, \overline{C}_1, \ldots, \overline{C}_\ell))$
```

```
$\mathsf{ReKeyGen}(k_i, k_j, \tilde{C})$
    $(\chi, \tau) = \mathcal{D}(k_i, \tilde{C})$
    if $(\chi, \tau) = \perp$ return $\perp$
    $x', y' \leftarrow_{\$} \mathcal{K}$
    $\chi' = \chi + x' + y'$
    $\tau' = \tau + F(x', 0)$
    $\tilde{C}' \leftarrow_{\$} \mathcal{E}(k_j, (\chi', \tau'))$
    return $\Delta_{i,j,\tilde{C}} = (\tilde{C}', x', y')$
```

```
$\mathsf{ReEnc}(\Delta_{i,j,\tilde{C}}, (\tilde{C}, \overline{C}))$
    $(\tilde{C}', x', y') = \Delta_{i,j,\tilde{C}}$
    $y = \overline{C}_0$
    for $1 \leq l \leq \ell$
        $\overline{C}'_l = \overline{C}_l + F(x', l)$
    return $(\tilde{C}', \overline{C}' = (y + y', \overline{C}'_1, \ldots, \overline{C}'_\ell))$
```

```
$\mathsf{Dec}(k, (\tilde{C}, \overline{C}))$
    $(\chi, \tau) \leftarrow_{\$} \mathcal{D}(k, \tilde{C})$
    if $(\chi, \tau) = \perp$ return $\perp$
    $y = \overline{C}_0$
    for $1 \leq l \leq \ell$
        $m_l = \overline{C}_l - F(\chi - y, l)$
    if $h(m) + F(\chi - y, 0) = \tau$ then
        return $m = (m_1, \ldots, m_\ell)$
    else
        return $\perp$
```

Fig. 9. The ReCrypt scheme.

for all $\kappa \geq 0, t \geq 1$.

Proof. We split the proof into two parts. The first part uses the AE security of π to show that the value of x used to key the key-homomorphic PRF is hidden from the adversary. The second part uses the fact that F is a PRF to show indistinguishability holds, given that the adversary cannot learn x.

Let G_0 be the original UP-IND game. For $1 \leq i \leq t$, Game G_i is the same as G_{i-1}, except we replace the encryption $\mathcal{E}(k_i, (\chi, h(m) + F(x, 0)))$ with a random string r and store the tuple (x, y, m) as a lookup in the table $\mathbb{C}$ indexed by the random string r. For queries to $\mathsf{ReKeyGen}(i, j, \tilde{C})$, if $\tilde{C} = r$ for some previously returned r, then compute x', y' as usual and either:

- if $j \leq i$, return (r', x', y') for some random r', storing the value $(x + x', y + y', m)$, or
- if $j > i$, return $(\mathcal{E}(k_j, (\chi', h(m) + F(x + x', 0))), x', y')$.

If $\tilde{C}$ has not been previously seen, return $\perp$. These modifications are shown in Figure 10.

Let S_i be the event that $\mathcal{A}$ outputs the correct bit in game G_i. Then we can construct an adversary $\mathcal{B}$ such that $|\Pr[S_i] - \Pr[S_{i-1}]| \leq \mathsf{Adv}^{\mathrm{ae}}_{\pi}(\mathcal{B})$. To see this, notice that by replacing the $\mathcal{E}(k_i, \cdot)$ and $\mathcal{D}(k_i, \cdot)$ functions with calls to an instance of the AE game for Π, in the real world, we get G_{i-1}, and the random world corresponds to G_i.

In game G_t, suppose the attacker queries the LR oracle to receive a challenge ciphertext C. Then the header $\tilde{C}$ will be equal to some random string r, while the body $\overline{C}$ consists the key mask y and the message encrypted by the PRF F keyed by some value x. Note that the use of AE forces the adversary to submit only previously seen ciphertext headers to oracles with $i \leq t$.

From re-keygen queries, the attacker can learn the fresh values of x', y', and r'. However, these are insufficient to learn anything about the value of x used to encrypt the message.

Similarly, re-encryption queries where the target key j is uncompromised, i.e., $j \leq t$, the adversary receives a fresh ciphertext encrypted under x' with no way of learning x'.

On the other hand, if $j > t$, the invalidity condition $\mathsf{Invalid}_{\mathsf{RE}}$ will be true, so the adversary learns the ciphertext header. This is of the form $\mathcal{E}(k_j, (x + y + x' + y', h(m) + F(x + x', 0)))$. Since the adversary possesses k_j, they may decrypt the ciphertext header. However, x is still masked by y.

From here, we proceed similarly to the original proof outline given by Boneh et al. We construct a game G_{t+1} such that $|\Pr[S_{t+1}] - \Pr[S_t]| \leq \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{C})$.

When $i \leq t$, we replace the usage of $F(x, l)$ in the LR oracle by using the PRF challenge oracle, denoted $f(l)$. Suppose K is the key used by the PRF challenger. When the PRF challenge is from the real world, the computed encryption is of the form $m_l + F(x, l) + f(l) = m_l + F(K + x, l)$. While the ciphertext header includes the value $\tau = h(m) + F(K + x, 0)$.

Since we showed that the value of x is unknown to the adversary for challenge ciphertexts, this change perfectly models the game G_t.

Furthermore, the computation of re-keying tokens and re-encryptions does not remove the PRF mask f from the ciphertext: if j is uncorrupted, then we compute $\overline{C}_l + F(x', l) = m_l + F(x + x' + K, l)$; otherwise, either $\mathsf{Invalid}_{\mathsf{RE}}$ or $\mathsf{Invalid}_{\mathsf{RK}}$ will be false, and the oracle returns $\perp$ for the ciphertext body. The adversary can also obtain the ciphertext header containing an encryption of $(x + x', \tau = h(m) + F(K + x + x', 0))$.

Now, let G_{t+1} correspond to the event that the PRF challenge returns a random value. Then the message is always perfectly masked by the output of $f(l)$ – we showed earlier that this value is present even after a re-encryption is computed. Therefore both the message and the hash are perfectly masked by the PRF values, and the adversary cannot win with a probability greater than $\frac{1}{2}$. Hence, we conclude that:

$$\begin{aligned}\mathsf{Adv}_{\Pi,\kappa,t}^{\text{up-ind}}(\mathcal{A}) &= |2 \cdot \Pr[S_0] - 1| \\ &\leq 2t \cdot \mathsf{Adv}_{\pi}^{\mathrm{ae}}(\mathcal{B}) + 2 \cdot \mathsf{Adv}_F^{\mathrm{prf}}(\mathcal{C})\end{aligned}$$

for all $\kappa \geq 0, t \geq 1$.

```
$\mathsf{Enc}(i, m)$
$x, y \leftarrow_\$ \mathcal{K}$
$r \leftarrow_\$ \{0,1\}^{|\tilde{C}|}$
$\mathbb{C}_i[r] = (x, y, m)$
$\overline{C}_0 = y$
for $1 \leq l \leq \ell$
    $\overline{C}_l = m_l + F(x, l)$
return $(\tilde{C} = r, \overline{C})$
```

```
$\mathsf{ReKeyGen}(i, j, \tilde{C})$
$(x, y, m) \leftarrow \mathbb{C}[\tilde{C}]$
if $(x, y, m) =\perp$ return $\perp$
$x', y' \leftarrow_\$ \mathcal{K}$
$\chi' = x + y + x' + y'$
if $j \leq i$ then
    $r \leftarrow_\$ \{0,1\}^{|\tilde{C}|}$
    $\mathbb{C}_j[r] = (x + x', y + y', m)$
    $\tilde{C}' = r$
else
    $\tau = h(m) + F(x + x', 0)$
    $\tilde{C}' \leftarrow_\$ \mathcal{E}(k_j, (\chi', \tau))$
return $\Delta_{i,j,\tilde{C}} = (\tilde{C}', x', y')$
```

```
$\mathsf{Dec}(i, (\tilde{C}, \overline{C}))$
$(x, y, m) \leftarrow \mathbb{C}_i[\tilde{C}]$
if $(x, y, m) =\perp$ return $\perp$
$y' = \overline{C}_0$
$x' = x + y - y'$
for $1 \leq l \leq \ell$
    $m'_l = \overline{C}_l - F(x', l)$
$\tau = h(m) + F(x, 0)$
if $\tau - F(x', 0) = h(m')$ then
    return $m = (m_1, \ldots, m_\ell)$
else
    return $\perp$
```

Fig. 10. The replacement algorithms used in game G_i for the proofs of security for the ReCrypt construction. For the i-th key, encryption/decryption by Π is replaced by random strings r of the same length and a lookup.

Theorem 10 (UP-INT **security of** ReCrypt)**.** *Let $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ be an AE scheme, $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a key-homomorphic PRF, h be a cryptographic hash function modelled as a random oracle with outputs in $\mathcal{Y}$, and let Π be the* ReCrypt *scheme as depicted in Figure 9.*

Then for any adversary $\mathcal{A}$ against Π, there exists an adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) \leq 2t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + \frac{q^2 + q_h^2}{|\mathcal{Y}|} + \frac{q^2}{|\mathcal{X}| \cdot |\mathcal{Y}|}$$

for all $\kappa \geq 0, t \geq 1$, where the adversary makes q_h queries to h, and q oracle queries.

Proof. Apply the same t transformations shown in Figure 10, which substitutes encryption by π with random strings for uncorrupted keys. Decryption is then simulated by using a lookup of previously returned values. Let S_i be the event that $\mathcal{A}$ outputs the correct bit in game G_i.

In game G_{t+1}, we replace the Dec verification check $\tau - F(x', 0) = h(m')$ and instead directly verify that $m' = m$. These two games are identical, unless the adversary submits for verification a ciphertext $C = (\tilde{C}, \overline{C})$ such that $\tilde{C}$ has previously been generated for the tuple (x, y, m) and $\overline{C}$, decrypts using $\chi - y$ to a message m' such that $m' \neq m$ but $h(m) + F(x, 0) - F(x + y - y') = h(m')$. I.e. $h(m) + F(y - y', 0) = h(m')$. For all adversarially chosen y', the adversary needs to construct a ciphertext $\overline{C}'$ such that $h(\overline{C}'_1 - F(x+y-y', 1), \ldots, \overline{C}'_\ell - F(x+y-y', \ell))$ is equal to some fixed value $h(m) - F(y - y', 0)$. This is equivalent to the preimage resistance of h. Alternatively, if the adversary does not modify y, then this reduces to finding a value of m' such that $h(m) = h(m')$, which is just the collision resistance of h.

By modelling h as a random oracle, we can bound the probability that the adversary making q_h queries to the random oracle, succeeds in finding suitable values of y', m, and m' by $q_h^2/|\mathcal{Y}|$. Therefore$|\Pr[S_{t+1}] - \Pr[S_t]| \leq q_h^2/|\mathcal{Y}|$.

Let (i, C) be a tuple submitted by the adversary to the Try oracle. We show that with high probability, either $\mathsf{InvalidCTXT}(i, C) = 1$ or $\mathsf{Dec}(k_i, C) = \perp$. By the first t game hops, we know that $\mathcal{D}(k_i, \tilde{C}) \perp$ unless $\tilde{C}$ was previously output by an oracle query (whether Enc or ReKeyGen). Furthermore, for all $\tilde{C}$, with high probability, there exists precisely one $\overline{C}$. For $1 \leq l \leq \ell$, decryption is computed as $\overline{C}_l + F(x, l)$. Therefore, any modification to a ciphertext block will result in a distinct message, which will result in a failed decryption due to the equality check. The exception to this is when the same random value $\tilde{C}$ was generated for distinct messages. If the adversary makes a total of q queries, this happens with probability $q^2/2^{|\tilde{C}|} \leq q^2/2^{|h(x)|+|x|} \leq q^2/2^{\log|\mathcal{X}|+\log|\mathcal{Y}|} \leq q^2/(|\mathcal{X}| \cdot |\mathcal{Y}|)$.

This shows that for each $\tilde{C}$, there can only be one $\overline{C}$. But what about the converse? For a given ciphertext body $\overline{C}$, suppose there exists (x', m') such that $\overline{C}_l = m_l + F(x, l) = m'_l + F(x', l)$. Therefore $F(x' - x, l) = m_l - m'_l$. However, m' was submitted to the (re-)encryption oracle before x' was generated. Therefore, for any pair $(x, m), (x', m')$ the probability that $F(x' - x, l) = m_l - m'_l$ is $1/|\mathcal{Y}|$, and thus if the adversary makes q queries, the probability is bounded by $q^2/|\mathcal{Y}|$.

Ultimately, together with the probability that two $\tilde{C}$ are equal, this becomes the probability the adversary wins in game G_{t+1}, otherwise we have established that all valid ciphertext headers have precisely one matching ciphertext body, and therefore $\mathsf{InvalidCTXT}(i, C)$ will be true.

Therefore, the advantage of the adversary is given by

$$\mathsf{Adv}^{\text{up-int}}_{\Pi,\kappa,t}(\mathcal{A}) \leq t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + \frac{q^2 + q_h^2}{|\mathcal{Y}|} + \frac{q^2}{|\mathcal{X}| \cdot |\mathcal{Y}|}.$$

Finally, we prove that ReCrypt meets our re-encryption indistinguishability notion.

Theorem 11 (UP-REENC **security of** ReCrypt)**.** *Let $\pi = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ be an AE scheme, $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$ be a key-homomorphic PRF, and let Π be the* ReCrypt *scheme as depicted in Figure 9.*

Then for any adversary $\mathcal{A}$ against Π, there exist adversaries $\mathcal{B}$, $\mathcal{C}$ such that

$$\mathsf{Adv}^{\text{up-reenc}}_{\Pi,\kappa,t}(\mathcal{A}) \leq 2t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + 2 \cdot \mathsf{Adv}^{\text{prf}}_{F}(\mathcal{C})$$

for all $\kappa \geq 0, t \geq 1$.

Proof. Apply the same t steps as used in the previous proof, shown in Figure 10. Then we have the ciphertext headers replaced with random strings, and the re-keygen process is simulated by recording inputs previously seen.

Hence, in game G_t, the adversary cannot learn anything about the PRF key x used in the challenge, nor does the ciphertext header reveal anything for compromised keys, since all values are masked by values stored in the ciphertext body.

As before, we construct a PRF adversary $\mathcal{C}$ to finish the game. Suppose in G_t we replace the re-encryption routine from challenges to return: $(r, \overline{C}_b + f(l) + F(x' + y', l))$. I.e. the ciphertext header $\tilde{C}$ is a random string, by the previous hybrids, and we re-encrypt the ciphertext body $\overline{C}_b$ to use (implicitly) key $\chi + K + x'$, where K is the key by in the PRF challenger. The re-keying token would be $(r, K + x', y')$, however, since we do not return this to the adversary, we do not need to compute it, nor does the adversary learn $K + x'$. Any further re-encryption or re-keygen queries by the adversary preserves the usage of $f(l)$ as a mask.

Now we let game G_{t+1} be the scenario in which the PRF challenger returns random values, and we conclude that the adversary cannot learn any information about the bit b.

The same computation as before results in

$$\mathsf{Adv}^{\text{up-reenc}}_{\Pi,\kappa,t}(\mathcal{A}) \leq 2t \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B}) + 2 \cdot \mathsf{Adv}^{\text{prf}}_{F}(\mathcal{C}).$$

7.2 Instantiating the Key-homomorphic PRF

We dedicate the remainder of this section to analysis of ReCrypt for use in practical scenarios. We delve into the implementation details of the key-homomorphic PRF in order to further explore some of the subtle security issues that arise when instantiating our scheme in practice.

While BLMR construct key-homomorphic PRFs in the standard model, a more efficient route is to use the classic ROM construction due originally to Naor, Pinkas, and Reingold [NPR99] in which $F(k, x) = k \cdot H(x)$ where H is modelled as a random oracle $H : \mathcal{X} \to \mathbb{G}$ and $\mathbb{G}$ is a group (now written **additively**, since we shall shortly move to the elliptic curve setting) in which the decisional Diffie–Hellman (DDH) assumption holds.

Instantiation details. We will use (a subgroup of) $\mathbb{G} = E(\mathbb{F}_p)$, an elliptic curve over a prime order finite field. However, recall that encryption is done block-wise as $\overline{C}_l = m_l + F(x, l)$. Implicitly, it is assumed that messages m are already in the group $\mathbb{G}$. To make a practical scheme for encrypting data represented as bitstrings, we additionally require an encoding function $\sigma : \{0,1\}^n \to \mathbb{G}$.

Additionally, the existence of such a function proves useful in the construction of the PRF: we show how to instantiate the random oracle H using a regular cryptographic hash function $h : \{0,1\}^* \to \{0,1\}^n$, modelled as a random oracle, together with the encoding function. We also use this definition of H for the instantiation of the random oracle used in the computation of the ciphertext header, which was needed to provide integrity. However, we add a unique prefix to inputs to either computation of H to provide separation.

For a suitable message encoding function, we of course require the function and its inverse to be efficiently computable. However, in addition we also require the inverse to be uniquely defined. Suppose σ^{-1} is defined for all $P \in \mathbb{G}$; then there is the possibility of creating a conflict with the integrity requirements. For example, suppose we have two points P, P' such that $\sigma^{-1}(P) = \sigma^{-1}(P') = m$. Then an adversary can potentially exploit this collision in σ^{-1} to construct a forged ciphertext. While this might not threaten the integrity of the plaintext, it would become problematic for ciphertext integrity.

One solution to this is to add a check to σ to verify that $\sigma(\sigma^{-1}(P)) = P$, and return $\perp$ if not.

In the following theorem, we prove the security of the ReCrypt scheme when instantiated with a carefully chosen encoding function, and the Naor-Pinkas-Reingold PRF.

Theorem 12. *Let $\mathbb{G} = E(\mathbb{F}_p)$ be an elliptic curve of prime order in which the DDH assumption holds. For $n \in \mathcal{O}(\log \#\mathbb{G})$ let the encoding function $\sigma : \{0,1\}^n \rightarrow \mathbb{G}$ be an injective mapping such that for any point P outside of the range, i.e. $P \notin \{\sigma(x) : x \in \{0,1\}^n\}$, then $\sigma^{-1}(P) =\perp$.*

Let $H : \{0,1\}^n \rightarrow \mathbb{G}$ be defined as $H(x) = \sigma(h(x))$ for a cryptographic hash function h modelled as a random oracle.

Then $F(k, x) = k \cdot H(x)$ is a key-homomorphic PRF.

By rejecting encoded messages outside of the range of σ, we effectively restrict σ to be a bijection from $\{0,1\}^n$ to a subset of $\mathbb{G}$. Given this, it is easy to see instantiating ReCrypt with this message encoding and key-homomorphic PRF results in a secure updatable AE scheme as proven above.

We identified two candidates for the message encoding function. The first uses rejection sampling, in which a bitstring is first treated as an element of $\mathbb{F}_p$, with some redundancy, and subsequently mapped to the elliptic curve. If a matching point cannot be found on the curve, the value is incremented (using the redundancy) and another attempt is made. Repeating this process results in a probabilistic method.

Corollary 1. *Define the encoding function $\sigma : x \mapsto E(\mathbb{F}_p)$ mapping bitstrings of length n to group elements by first equating the bitstring as an element $x \in \mathbb{F}_p$. Let $\bar{x}$ be the minimum value of the set $\{x + i \cdot 2^n \ : \ 0 \leq i < \lfloor \frac{p}{2^n} \rfloor\}$ such that $(\bar{x}, y) \in E(\mathbb{F}_p)$ for some y. Then define $\sigma(x)$ to be the point $(\bar{x}, \bar{y})$ where $\bar{y} = \min\{y, p - y\}$.*

The inverse mapping $\sigma^{-1}(P)$ is computed by taking the x-coordinate and reducing mod 2^n. I.e. set $x' = x(P) \mod 2^n$ and verify $P \neq \sigma(x')$, otherwise return $\perp$.

Then σ satisfies the requirements of Theorem 12.

Proof. It is easy to see that for any x such that $\sigma(x)$ returns an encoding P, then σ is an injective function from $\{0,1\}^n$ to $\mathbb{G}$ satisfying the conditions outlined in the theorem statement.

On the other hand, correctness of this construction is only probabilistic. For an field element $x \in \mathbb{F}_p$, the probability that x corresponds to a point on the curve is approximately $\frac{\#E(\mathbb{F}_p)}{2p} \approx \frac{1}{2}$, where the factor of 2 is due to the fact that for each (x, y), the point $(x, -y)$ is also on the curve.

For each message m, there are $\left\lfloor \frac{p}{2^n} \right\rfloor$ possible field elements to test for points on the curve. Therefore the probability that $\sigma(m)$ has no valid encoding is given by $2^{-\lfloor p/2^n \rfloor}$.

As an alternative to rejection sampling, an injective mapping can be used directly, again first treating the bitstring as an element of $\mathbb{F}_p$. Some examples include the SWU algorithm [SvdW06], Icart's function [Ica09], and the Elligator encoding [BHKL13].

Corollary 2. *For a compatible elliptic curve $E(\mathbb{F}_p)$, the Elligator function as defined in [BHKL13] satisfies the requirement of Theorem 12 for all $m \in \{0,1\}^{\lfloor \log(p-1) \rfloor - 1}$*

Proof. The Elligator function maps injectively from $\{1, \ldots, \frac{p-1}{2}\}$ to $E(\mathbb{F}_p)$. For the inverse map, if the returned value is greater than $\frac{p-1}{2}$, we return $\perp$.

7.3 Implementation and Performance

We now provide a concrete instantiation of the ReCrypt scheme using the method described in Section 7.2 and report on the performance of our prototype implementation. Our goal is assess the performance gap between in-use schemes that do not meet UP-REENC security, and ReCrypt, which does.

Implementation. We built our reference implementation using the Rust [MKI14] programming language. This implementation uses Relic [AG], a cryptographic library written in C, and the GNU multi-precision arithmetic library (GMP). Our implementation is single-threaded and we measured performance on an Intel CPU (Haswell), running at 3.8GHz in turbo mode.

We use `secp256k1` [Cer00] for the curve and SHA256 as the hash function h. The plaintext block length is 31 bytes. We use AES128-GCM for the AE scheme π.

The Relic toolkit provided a number of different curve options, as well as access to the low level elliptic curve operations which was essential in our early prototyping and testing. However, Relic does not at the time of writing support curves in Montgomery form, and therefore has an inefficient implementation of scalar multiplication on Curve25519. Therefore, we choose `secp256k1` because it was the most performant among all curve implementations at our disposal with (approximately) 128 bits of security. We project that Curve25519 would offer comparable efficiency, whereas a hand-tuned, optimised variant of a specific curve would result in a significant speedup.

When encoding plaintext bit strings into elliptic curve points, we must choose a block length that is strictly smaller than the log of the order of the curve. We must also reserve a few bits to perform rejection sampling when encoding plaintext bit strings into curve points. Our chosen encoding method is only probabilistically correct, since for each block, the probability there exists an image on the group is approximately $1 - 2^{-\lfloor p/2^n \rfloor}$ where p is the prime of the underlying

ReCrypt	**Time per CPU**				
Operation	1 block	1 KB	1 MB	1 GB	cycles/byte
Encrypt	663 μs	10.0 ms	9.2 s	2.6 hours	32.4 K
ReEnc	302 μs	8.8 ms	8.7 s	2.4 hours	30.7 K
Decrypt	611 μs	9.1 ms	8.6 s	2.4 hours	30.6 K
ReKeyGen (total)	450 μs				1.96 M

Fig. 11. Processing times for ReCrypt operations measured on a 3.8GHz CPU. 1 block represents any plaintext $\leq$ 31 bytes. Number of iterations: 1000 (for 1 block, 1 KB), 100 (for 1 MB) and 1 (for 1 GB). Cycles per byte given for 1MB ciphertexts.

finite field, and n is the block length. However, we are able to chose parameters to make the scheme overwhelming correct (see below). We also experimented with the injective encodings such as the Elligator encoding [BHKL13]. The mapping did not appear to improve performance, and moreover is incompatible with `secp256k1`. Additionally, we do not require ciphertexts to be indistinguishable from random, one of the key benefits offered by the Elligator encoding.

Splitting bytes would have a deleterious impact on performance, and so we choose a 31-byte block size. With this choice, each plaintext block has a valid encoding with probability $1 - 2^{-256}$.

When a curve point is serialized, only the x coordinate and the sign of the y coordinate (1-bit) needs to be recorded (using point compression). Since the x coordinate requires strictly less than the full 32 bytes, we can serialize points as 32 byte values. Each 32 byte serialized value represents 31 bytes of plaintext giving a ciphertext expansion of 3%. Upon deserialization, the y coordinate must be recomputed. This requires computing a square root, taking approximately 20 μs. Of course this cost can be avoided by instead serializing both x and y coordinates. This creates a 64 byte ciphertext for each 31 bytes of plaintext which is an expansion of 106%. We consider that to be unacceptable.

Microbenchmarks. Figure 11 shows wall clock times for ReCrypt operations over various plaintext sizes. As might be expected given the nature of the cryptographic operations involved, performance is far from competitive with conventional AE schemes. For comparison, AES-GCM on the same hardware platform encrypts 1 block, 1 KB, 1 MB and 1 GB of plaintext in 15 μs, 24 μs, 9 ms, and 11 s, respectively. KSS has performance determined by that of AES-GCM, while the performance of the ReCrypt scheme is largely determined by the scalar multiplications required to evaluate the PRF. Across all block sizes there is a 1000x performance cost to achieve our strongest notion of security.

Discussion. Given this large performance difference, ReCrypt is best suited to very small or very valuable plaintexts (ideally, both). Compact and high-value plaintexts such as payment card information, personally identifiable information, and sensitive financial information are likely targets for ReCrypt. If the plaintext corpus is moderately or very large, cost and performance may prohibit practitioners from using ReCrypt over more performant schemes like KSS that give strictly weaker security.

8 Conclusion and Open Problems

We have given a systematic study of updatable AE, providing a hierarchy of security notions meeting different real-world security requirements and schemes that satisfy them efficiently. Along the way, we showed the limitations of currently deployed approach, as represented by AE-hybrid, improved it at low cost to obtain the KSS scheme meeting our UP-IND and UP-INT notions, identified a flaw in the BLMR scheme, repaired it, and showed how to instantiate the repaired scheme in the ROM. Through this, we arrived at ReCrypt, a scheme that is secure in our strongest security models (UP-IND, UP-INT and UP-REENC). We implemented ReCrypt and presented basic speed benchmarks for our prototype. The scheme is slow compared to the hybrid approaches but offers true key rotation.

Our work puts updatable AE on a firm theoretical foundation and brings schemes with improved security closer to industrial application. While there is a rich array of different security models for practitioners to chose from, it is clear that achieving strong security (currently) comes at a substantial price. Meanwhile weaker but still useful security notions can be achieved at almost zero cost over conventional AE. It is an important challenge to find constructions which lower the cost compared to ReCrypt without reducing security. But it seems that fundamentally new ideas will be needed here, since what are essentially public key operations are intrinsic to our construction.

From a more theoretical perspective, it would also be of interest to study the exact relations between our security notions, in particular whether UP-REENC is strong enough to imply UP-IND and UP-INT. There is also the question of whether a scheme that is UP-REENC is necessarily ciphertext-dependent. Finally, we reiterate the possibility of formulating updatable AE in the nonce-based setting.

Acknowledgements

We thank the anonymous Crypto reviewers for their feedback and discussion relating to a bug in our initial KSS construction, and for their additional comments and suggestions. This work was supported in part by NSF grant CNS-1330308, EPSRC grants EP/K035584/1 and EP/M013472/1, by a research programme funded by Huawei Technologies and delivered through the Institute for Cyber Security Innovation at Royal Holloway, University of London, and a gift from Microsoft.

References

ABBC10. Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422. Springer, Heidelberg, May 2010.

ABL$^+$14. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125. Springer, Heidelberg, December 2014.

AG. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

AWS. AWS. Protecting data using client-side encryption. `http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingClientSideEncryption.html`.

BDPS14. Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 367–390. Springer, Heidelberg, March 2014.

BHKL13. Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 967–980. ACM Press, November 2013.

BK03. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 491–506. Springer, Heidelberg, May 2003.

BLMR13. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 410–428. Springer, Heidelberg, August 2013.

BLMR15. Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. Cryptology ePrint Archive, Report 2015/220, 2015. `http://eprint.iacr.org/2015/220`.

BN00. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000.

BRS03. John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

Cer00. SEC Certicom. Sec 2: Recommended elliptic curve domain parameters. *Proceeding of Standards for Efficient Cryptography, Version*, 1, 2000.

CGH12. David Cash, Matthew Green, and Susan Hohenberger. New definitions and separations for circular security. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 540–557. Springer, Heidelberg, May 2012.

CH07. Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07*, pages 185–194. ACM Press, October 2007.

CK05. DL Cool and Angelos D Keromytis. Conversion and proxy functions for symmetric key ciphers. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 1, pages 662–667. IEEE, 2005.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

Goo. Google. Managing data encryption. `https://cloud.google.com/storage/docs/encryption`.

Ica09. Thomas Icart. How to hash into elliptic curves. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 303–316. Springer, Heidelberg, August 2009.

ID03. Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS 2003*. The Internet Society, February 2003.

KRW15. Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 378–400. Springer, Heidelberg, March 2015.

MKI14. Nicholas D Matsakis and Felix S Klock II. The rust language. *ACM SIGAda Ada Letters*, 34(3):103–104, 2014.

NPR99. Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 327–346. Springer, Heidelberg, May 1999.

NRS14. Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.

PCI16. PCI Security Standards Council. Requirements and security assessment procedures. In *PCI DSS v3.2*, 2016.

PRS11. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 372–389. Springer, Heidelberg, December 2011.

Rog06. Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progressin Cryptology - VIETCRYPT 2006, First International Conferenceon Cryptology in Vietnam, Hanoi, Vietnam, September 25-28, 2006, Revised Selected Papers*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.

RS06. Phillip Rogaway and Thomas Shrimpton. Deterministic authenticated-encryption: A provable-security treatment of the key-wrap problem. Cryptology ePrint Archive, Report 2006/221, 2006. `http://eprint.iacr.org/2006/221`.

RSS11. Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

SvdW06. Andrew Shallue and Christiaan E van de Woestijne. Construction of rational points on elliptic curves over finite fields. In *Algorithmic number theory*, pages 510–524. Springer, 2006.

A Bidirectional Updatable AE

A.1 XOR-KEM: A Bidirectional Updatable AE Scheme

The AE-hybrid and KSS schemes are unidirectional and ciphertext-dependent. This means that in practice the client must fetch from storage ciphertext headers in order to compute the rekey tokens needed to update individual ciphertexts. It could be simpler to utilize a ciphertext-independent scheme that has rekey tokens that work for any ciphertext encrypted with a particular key. This would make the re-encryption process "non-interactive", requiring that the key holder only push a single rekey token to the place where ciphertexts are stored. Given the obvious performance benefits that such a scheme would have, we also provide such a scheme, called XOR-KEM. This scheme is exceptionally fast, and is built from a (non-updatable) AE scheme that is assumed to be secure against a restricted form of related-key attack (RKA). This latter notion adapts the Bellare-Kohno RKA-security notions for block ciphers [BK03] to the setting of AE schemes. To the best of our knowledge, this definition is novel, and RKA secure AE may itself be of independent interest as a primitive. However, the XOR-KEM scheme cannot meet our integrity notions against an attacker in possession of compromised keys. (And because of its bidirectionality, XOR-KEM also provides the counter-example that we used to separate UP-IND-BI and UP-IND security in Section 3.1.)

Let ($\mathcal{K}$,$\mathcal{E}$,$\mathcal{D}$) be an AE scheme. Then we define the ciphertext-independent scheme, XOR-KEM, as follows:

- $\mathsf{KeyGen}()$: **return** $k \leftarrow \mathcal{K}$
- $\mathsf{Enc}(k, m)$: $x \leftarrow \mathcal{K}$; $C \leftarrow (x \oplus k, \mathcal{E}(x, M))$; **return** C
- $\mathsf{ReKeyGen}(k_1, k_2)$: **return** $\Delta_{1,2} = k_1 \oplus k_2$
- $\mathsf{ReEnc}(\Delta_{1,2}, C = (C_0, C_1))$: $C' \leftarrow (\Delta_{1,2} \oplus C_0, C_1)$; **return** C'
- $\mathsf{Dec}(k, C = (C_0, C_1))$: **return** $\mathcal{D}(C_0 \oplus k, C_1)$

The XOR-KEM scheme has a similar format to the AE-hybrid scheme above. However, instead of protecting the DEM key x by encrypting it, we instead XOR it with the secret key k. The resulting scheme becomes a bidirectional, ciphertext-independent scheme, and one that has extremely high performance and deployability.

Note that although the value $x \oplus k$ fulfils a similar purpose as the ciphertext header in AE-hybrid, since this value is not needed in re-keying, it resides in the ciphertext body.

Theorem 13 (UP-IND-BI **Security of XOR-KEM).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let Π be the updatable AE scheme XOR-KEM built using π as above. Then for any* UP-IND-BI *adversary $\mathcal{A}$ against Π, there exists an AE adversary $\mathcal{B}$ against π such that:*

$$\mathsf{Adv}^{\text{up-ind-bi}}_{\Pi,\kappa,t}(\mathcal{A}) \;\leq\; 2 \cdot \mathsf{Adv}^{\text{ae}}_{\pi}(\mathcal{B})$$

for all $\kappa \geq 0, t \geq 1$.

Proof. We consider a sequence of games $G_0, \dots, G_{q+1}$, where q is the number of queries made by $\mathcal{A}$ to its LR oracle.

Let game G_0 correspond to the regular UP-IND-BI game. Let S_i correspond to the event that game G_i outputs true.

In game G_1, a key $x^* \leftarrow_{\$} \mathcal{K}$ is generated at the start of the game. This is used to encrypt the message in the first query made to the LR oracle. That is, on input (i, m_0, m_1), the LR oracle computes:

$$x \leftarrow_{\$} \mathcal{K};\ \overline{C} \leftarrow_{\$} (x \oplus k_i, \mathcal{E}(x^*, m_b))\,,$$

and returns $\overline{C}$.

All other queries are answered as in G_0. The adversary has the same view in both games, unless the adversary recovers k_i (in which case the adversary which can win with one extra query). Either way, we have $\Pr[S_0] = \Pr[S_1]$.

For $1 < \tau \leq q$, game G_τ is identical to $G_{\tau-1}$ except for the adversary's τ-th LR query. There, encryption is computed as:

$$x_\tau \leftarrow_{\$} \mathcal{K};\ \overline{C} \leftarrow_{\$} (x_\tau \oplus k_i, \mathcal{E}(x^*, m_b))$$

where x^* is the same key generated at the start of G_1. As before, these games are identical to the adversary, and therefore $\Pr[S_\tau] = \Pr[S_{\tau-1}]$.

In game G_{q+1}, we replace encryption by x^* with randomly sampled values. It is straightforward to construct an adversary $\mathcal{B}$ in the AE-ROR game such that $|\Pr[S_{q+1}] - \Pr[S_q]| \leq \mathsf{Adv}_{\pi}^{\mathrm{ae}}(\mathcal{B})$.

Finally, in G_{q+1} the adversary can learn nothing about which m_b is encrypted, therefore $\Pr[S_{q+1}] = \frac{1}{2}$. Combining the above, we get the stated result.

XOR-KEM does not provide integrity guarantees in the face of compromised keys: an attacker who learns both $C = \mathsf{Enc}(k_1, m)$ and $C' = \mathsf{ReEnc}(\mathit{\Delta}_{1,t+1,\tilde{C}}, C)$ can derive k_1.

For ciphertext integrity of bidirectional schemes, we modify the $\mathsf{InvalidCTXT}$ conditions to check whether the ciphertext is a re-encryption using a bidirectional rekey token. We call this new predicate $\mathsf{InvalidCTXT}_{\mathsf{BI}}$ which returns true whenever $\mathsf{InvalidCTXT}$ returns true, and additionally returns true if:

– $(\tilde{C}, \overline{C})$ is the ciphertext output by running $\mathsf{ReEnc}(\mathsf{Invert}(\mathit{\Delta}_{j,i}), C')$ for $C' = (\tilde{C}', \overline{C}')$ where $\mathit{\Delta}_{j,i}$ was the result of a query $\mathrm{ReKeyGen}(j, i)$ and $\mathsf{InvalidCTXT}_{\mathsf{BI}}(i, C')) = \mathsf{true}$.

We refer to the security game using $\mathsf{InvalidCTXT}_{\mathsf{BI}}$ as UP-INT-BI.

Unfortunately, proving that XOR-KEM even achieves UP-INT-BI security for no compromised keys ($\kappa = 0$) is not straightforward. It is clear that the adversary can produce trivial manipulations of the ciphertext header: $\tilde{C} \oplus z = (x \oplus k) \oplus z = (x \oplus z) \oplus k$. Thus, to prove UP-INT-BI security additionally requires us to assume that the AE scheme π used in the construction is secure against related-key attacks, in which the adversary can access the encryption and decryption functions of π under XOR-offsets of the unknown key.

Definition 9 (Related-Key Secure AE). *Let $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme with keyspace $\mathcal{S}_{\mathcal{K}}$ and let Φ be a set of functions $\Phi = \{\phi : \mathcal{S}_{\mathcal{K}} \to \mathcal{S}_{\mathcal{K}}\ \}$. Let the Φ-restricted RKA security game be the game in which the adversary has access to a pair of oracles $\mathsf{Enc}, \mathsf{Dec}$ which, on input (ϕ, x), $\phi \in \Phi$, return for $b = 0$:*

$$\mathsf{Enc}(\phi, x) = \mathcal{E}(\phi(k), x),\ \mathsf{Dec}(\phi, x) = \mathcal{D}(\phi(k), x)$$

and for $b = 1$:

$$\mathsf{Enc}(\phi, x) = \$(\cdot),\ \mathsf{Dec}(\phi, x) = \perp$$

where $k \leftarrow_{\$} \mathcal{K}$ and $b \leftarrow_{\$} \{0, 1\}$ are sampled at the start of the game.

The $RKA^{\mathcal{A}}_{\pi,\Phi}$ game for encryption scheme π, family of functions Φ, and adversary $\mathcal{A}$ outputs true *if the adversary outputs the correct bit b at the end of the game.*

We define the advantage of an adversary $\mathcal{A}$ by:

$$\mathsf{Adv}^{\text{rka-ae}}_{\pi,\Phi}(\mathcal{A}) = 2 \cdot \Pr\left[RKA\text{-}AE^{\mathcal{A}}_{\pi,\Phi} \Rightarrow \mathsf{true}\right] - 1.$$

Theorem 14 (UP-INT-BI **Security of XOR-KEM).** *Let $\pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with keyspace $\{0, 1\}^n$ and let Π be the updatable AE scheme XOR-KEM built using π as defined above. Furthermore, let Φ be the set of permutations $\Phi := \{XOR_{\Delta} : \Delta \in \{0, 1\}^n\}$ where $XOR_{\Delta} : \{0, 1\}^n \to \{0, 1\}^n$ denotes the function $XOR_{\Delta}(k) = k \oplus \Delta$.*

Then for any adversary $\mathcal{A}$ for the game UP-INT, *there exists an adversary $\mathcal{B}$ for the Φ-restricted RKA security game where:*

$$\mathsf{Adv}^{\text{up-int-bi}}_{\Pi,0,t}(\mathcal{A}) \ \leq\ \mathsf{Adv}^{\text{rka-ae}}_{\pi,\Phi}(\mathcal{B})$$

for all $t \geq 1$.

Proof. Let game G_0 be the original UP-INT-BI game.

Let game G_1 be identical to G_0, except we replace all encryption queries with random strings, and condition on the event that every query (i, C) the adversary makes to the Try oracle does not result in setting the win flag to true.

We show that $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\text{rka-ae}}_{\pi,\Phi}(\mathcal{B})$. To see this, we construct an adversary $\mathcal{B}$ which simulates the UP-INT-BI game.

For every encryption query to the underlying encryption scheme $\mathcal{E}$, the adversary instead makes a call to the RKA oracle $\mathsf{Enc}(\text{XOR}_x, m)$, where x is the usual randomly sampled DEM key.

In effect, the key used for encrypting the data is $K + x$ where K is the key randomly generated in the RKA game.

When $b = 0$ in the RKA game, encryption queries are returned as $\mathcal{E}(K + x, m)$. This perfectly simulates G_0.

When $b = 1$, decryption queries are all replaced by $\perp$. Therefore, every (i, C) submitted to the Try oracle was either seen before, and hence $\mathsf{InvalidCTXT}_{\mathsf{BI}}(i, C)$ returns true, or results in a call to $\mathsf{Dec}(\text{XOR}_x(i), C)$ which returns $\perp$.

Therefore this perfectly simulates game G_1.

Finally, $\Pr[S_1] = 0$, and thus we get the stated result.

We conjecture that an authenticated encryption scheme achieving this new notion could be obtained by using a RKA-secure PRF to build a CTR mode cipher, with the PRF also serving as the MAC in encrypt-then-mac mode.