# Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses

Jee Hea An*

**Abstract**

This paper addresses the security of authenticated encryption schemes in the public key setting. We present two new notions of authenticity that are stronger than the integrity notions given in the symmetric setting [5]. We also show that chosen-ciphertext attack security (IND-CCA) in the public key setting is not obtained in general from the combination of chosen-plaintext security (IND-CPA) and integrity of ciphertext (INT-CTXT), which is in contrast to the results shown in the symmetric setting [13, 5]. We provide security analyses of authenticated encryption schemes constructed by combining a given public key encryption scheme and a given digital signature scheme in a "generic" manner —namely, Encrypt-and-Sign, Sign-then-Encrypt, and Encrypt-then-Sign— and show that none of them, in general, provide security under all notions defined in this paper. We then present a scheme called *ESSR* that meets all security notions defined here. We also give security analyses on an efficient Diffie-Hellman based scheme called DHETM, which can be thought of as a transform of the encryption scheme "DHIES" [1] into an *authenticated* encryption scheme in the public key setting.

**Keywords:** Public key setting, Authenticated encryption, Privacy, Authenticity, Unforgeability.

*Dept. of Computer Science, & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-mail: jeehea@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/~jeehea. Supported in part by Mihir Bellare's NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

# Contents

# 1 Introduction

Background. Authenticity and Privacy have been the main goals of data communication in both private and public key settings. There are various schemes that are designed to meet these goals separately. In the public key setting, asymmetric encryption schemes are designed to provide privacy, while (digital) signature schemes are designed to provide authenticity. Well defined formal security notions for encryption schemes [9, 15, 7] and signature schemes [10] exist, and schemes have been analyzed according to those notions. Recently, there have been rising interests in combining these schemes in such a way that the goals of both privacy and authenticity are met at the same time [12, 17, 14, 11]. However, schemes that are designed to meet both goals have not received formal security treatments. No clear formal definitions of security or comprehensive security analyses on the schemes have been provided in the public key setting, although a recent work [5] gives these in the symmetric setting. Here we provide formal security definitions and security analyses for schemes whose goal is to provide both privacy and authenticity in the public key setting. Note that security definitions from the symmetric setting cannot just be "lifted up" for the public key setting because of the asymmetric nature of the latter. The asymmetry of keys makes a difference in the notions of both authenticity and privacy. In order to see this more clearly, we begin by describing the setting and security notions in more detail below.

The setting. We consider a public key setting where two parties (a sender and a receiver) want to communicate securely over an insecure channel. In order to provide both privacy and authenticity at the same time, both sender and receiver need to have their own public, secret key pairs. Note that this does *not* mean that the keys are unique for the specific pair of sender and receiver. Although we are considering just the two party case here, the setting can be extended to a multi-party case by assuming that each user has its own public, secret key pair.

We use the term *public key based authenticated encryption (abbreviated to PKAE) schemes* to refer to schemes whose goal is to provide both privacy and authenticity in the public key setting. Authenticated encryption schemes can also be viewed as encryption schemes with an added security goal of authenticity.

## 1.1 Security notions for public key based authenticated encryption schemes

Privacy. We consider privacy under both chosen-plaintext and chosen-ciphertext attacks. Notice that, unlike the usual public key encryption schemes, encryption in an authenticated encryption scheme is done based on a sender's secret key as well as a receiver's public key, and hence an adversary that does not have its own public, secret key pair cannot encrypt a message of its choice using the public key of the receiver only. Therefore, in order to model a chosen-plaintext attack, it is not sufficient to provide access to a receiver's public key. Adopting the left-or-right indistinguishability notion (IND-CPA) of the symmetric setting [3] to the public key setting, we model the chosen-plaintext attack by allowing an adversary access to a left-or-right encryption oracle. For the notion of privacy under chosen-ciphertext attack (i.e. IND-CCA), we allow access to a decryption oracle in addition to the left-or-right encryption oracle, following [3]. Because we allow a decryption oracle access to the adversary, we consider an adversary who has its own public, secret key pair. In this case, even though the adversary might not be able to generate a ciphertext that is valid with respect to the public key of a particular sender, it may be able to generate a ciphertext that is considered valid with respect to its own public key. This might make a difference in the adversary's ability to attack the given scheme in the IND-CCA sense.

Authenticity. Security regarding authenticity in the symmetric setting is normally measured by "unforgeability" by an adversarial third-person (meaning, excluding the sender and receiver) who is allowed a chosen message attack. Since the sender and receiver both share the same key, there is no distinction between the sender and receiver in their ability to create valid ciphertexts. However, in the public key setting, there is a distinction —only a sender has all the information (i.e. the sender's secret key and the receiver's public key) to create a valid (authentic) ciphertext. Hence, an adversary against a sender's authenticity may be a receiver as well as a third-person. Note, however, that a receiver and third-person may not have the same ability in forging a sender's ciphertext. This is because the receiver's key is also involved in creating valid ciphertexts and by creating and manipulating its own key pair, the receiver may be able to come up with a forgery which a third-person adversary cannot. Reflecting this difference into the unforgeability notion, we divide the notion into two parts: *receiver unforgeability (RUF)* and *third-person unforgeability (TUF)*. In the third-person unforgeability notion, which is the usual notion for authenticity, the goal of an adversary is to make the intended receiver believe that the forgery it received has indeed come from the original sender. In the receiver unforgeability notion, however, the goal of an adversary (i.e. receiver) is to convince a "third-party" (e.g. a judge) that a forgery it created has indeed come from the original sender. We note that this notion can model what is usually called "non-repudiation" for authenticated encryption schemes in the public key setting.

In the attack models for both notions, the adversary is allowed a chosen-message attack. Additionally, in the receiver unforgeability attack model, the adversary (receiver) is allowed to create its own key pair initially, and change its key pair later when it outputs a forgery so that its forgery is decrypted and verified using the new key pair. This models the versatility of the public key setting where it is possible for a party to register its own public key with a certification authority (CA) and later re-register a different public key without being detected by others. (Although associating timestamps with ciphertexts may help avoid this type of attack, it may not always be possible. Hence, we allow this type of attack for generality.) We remark that this kind of attack has appeared in the literature [2, 6]. The third-person unforgeability models unforgeability in the usual setting (similar to the symmetric setting). In both models, success of an adversary is measured by its ability to output a "new" and "valid" forgery. Depending on the definition of "newness", the forgery is further divided into two cases: *ciphertext* and *plaintext*. A forgery of a "new" ciphertext indicates that the forgery ciphertext output by the adversary is never output by the oracle given to the adversary while a forgery of a "new" plaintext indicates that the plaintext corresponding to the forgery ciphertext was never queried by the adversary to the oracle. Combining the receiver and third-person unforgeability (RUF, TUF) with ciphertext and plaintext forgery (CTXT, PTXT), we get the following four notions: RUF-CTXT, RUF-PTXT, TUF-CTXT, and TUF-PTXT. We note that the third-person unforgeability notions (TUF-CTXT and TUF-PTXT) in the public key setting are analogous to the integrity notions (INT-CTXT and INT-PTXT) in the symmetric setting shown in [5], while receiver unforgeability notions (RUF-CTXT and RUF-PTXT) are new and apply only to the public key setting.

### 1.2 Relations among the security notions

Figure 1 depicts the relations among the new notions (RUF-CTXT, RUF-PTXT, TUF-CTXT, TUF-PTXT) and the existing notions (IND-CCA, IND-CPA) presented in the style of [4, 5]. An implication $A \to B$ means that all schemes secure in the sense of $A$ are also secure in the sense of $B$. A separation $A \not\to B$ means that there exists a scheme that is secure in the sense of $A$ but not in the sense of $B$. Since TUF-CTXT and TUF-PTXT are analogous to INT-CTXT and INT-PTXT in the symmetric setting, the relations among them are not explicitly proved in this paper; instead we cite the papers that establish the relations. We combine IND-CPA with the unforgeability notions in Figure 1 in order to relate the combined security with privacy notions (i.e. IND-CPA, IND-CCA).

Note the horizontal relations that indicate that the receiver unforgeability properties are in general stronger than the third-person unforgeability —this is expected from the definitions of their attack models, where an adversarial receiver is given more "power" than an adversarial third-person. More

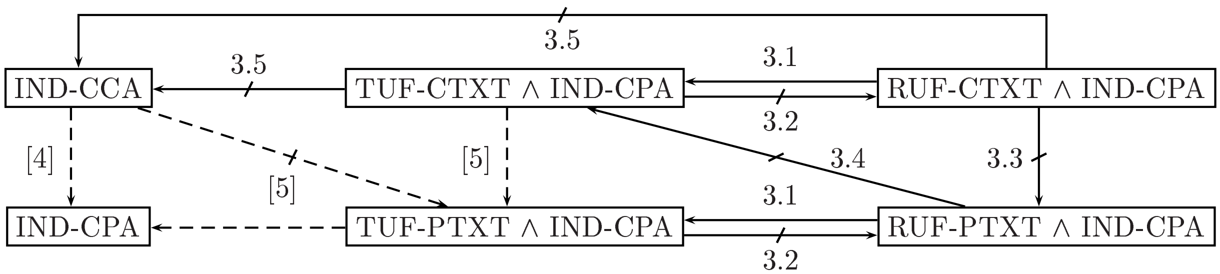


Figure 1: **Relations among security notions for public key based authenticated encryption schemes.** An arrow denotes an implication while a hatched arrow denotes a separation. The solid arrows indicate relations proved in this paper with the annotations corresponding to the propositions, and dotted arrows indicate existing relations (adapted to the public key setting described in this paper) annotated with citations.

---

interesting relations are shown between RUF-CTXT and RUF-PTXT notions —unlike the third-person case (where TUF-CTXT is a stronger notion than TUF-PTXT), the two notions RUF-CTXT and RUF-PTXT are not comparable, meaning the relative strengths between the two notions cannot be determined because a separation exists in both directions. (Note that, although not explicitly shown in Figure 1, RUF-PTXT does not imply RUF-CTXT because otherwise, by following arrows, we would get RUF-PTXT $\rightarrow$ TUF-CTXT, contradicting the stated separation.) This means that both RUF-CTXT and RUF-PTXT security (along with the IND-CCA security) need to be shown in order to prove that a scheme is secure for all notions.

In the symmetric setting, IND-CCA is implied by the combined notions of IND-CPA and INT-CTXT [13, 5]. However, in the public key setting, IND-CCA is *not* implied by IND-CPA and RUF-CTXT (which is a stronger notion than TUF-CTXT, the asymmetric counterpart of INT-CTXT). This is an important distinction that exists between the symmetric setting and the public key setting.

### 1.3 Generic composition of encryption and signature schemes

One of the most straightforward methods to design an authenticated encryption scheme in the public key setting is perhaps to "combine" the encryption and signature schemes in some "generic" way (which is called "generic composition", following [5]). We examine the security of three possible ways to combine the encryption and signature schemes: Encrypt-and-Sign plaintext, Sign-then-Encrypt, and Encrypt-then-Sign. The three methods are constructed based on a public key based encryption scheme and a signature scheme. The Encrypt-and-Sign method encrypts the plaintext and appends the signature of the plaintext. The Sign-then-Encrypt method appends a signature to the plaintext and then encrypts the plaintext and the signature together. The Encrypt-then-Sign method encrypts the plaintext to get a ciphertext $C$ and then appends the signature of $C$ to the ciphertext.

The summary of the results obtained from security analyses of the constructions is displayed in Figure 2 and Figure 3. The results shown in Figure 2 are obtained by assuming that the base signature scheme is "weakly unforgeable" (WUF-CMA), while those shown in Figure 3 are obtained by assuming that the base signature scheme is "strongly unforgeable" (SUF-CMA). The notions for signature schemes are adopted from the MAC security in [5]. In both figures, the security assumption on the base encryption scheme is chosen-plaintext security (IND-CPA). Strong unforgeability requires that it be computationally infeasible for an adversary to forge a "new" message, tag pair under a chosen message attack. What is different from the standard notion, weak unforgeability under chosen-message attack, is that the message does not need to be "new" as long as the tag is "new", meaning either the message or the tag needs to be new. In both figures, "secure" means that the scheme is shown to meet the security notion in question under the above-mentioned assumptions, while "insecure" means that there exist some IND-CPA secure encryption scheme and some signature scheme unforgeable under chosen-message attack such that the PKAE scheme that is constructed from them does not meet the security

| **Composition Method** | **Privacy** | | **Authenticity** | | | |
|---|---|---|---|---|---|---|
| | IND-CPA | IND-CCA | TUF-PTXT | TUF-CTXT | RUF-PTXT | RUF-CTXT |
| *Encrypt-and-Sign* | insecure | insecure | secure | insecure | secure | insecure |
| *Sign-then-Encrypt* | secure | insecure | secure | insecure | secure | insecure |
| *Encrypt-then-Sign* | secure | insecure | secure | insecure | insecure | insecure |

Figure 2: Summary of security results for the composed PKAE schemes under the assumptions that the base signature scheme is weakly unforgeable (WUF-CMA) and the base encryption scheme is IND-CPA secure. The shaded regions indicate the new or different results compared to those shown for the symmetric setting [5].

| **Composition Method** | **Privacy** | | **Authenticity** | | | |
|---|---|---|---|---|---|---|
| | IND-CPA | IND-CCA | TUF-PTXT | TUF-CTXT | RUF-PTXT | RUF-CTXT |
| *Encrypt-and-Sign* | insecure | insecure | secure | insecure | secure | insecure |
| *Sign-then-Encrypt* | secure | insecure | secure | insecure | secure | insecure |
| *Encrypt-then-Sign* | secure | insecure | secure | secure | insecure | secure |

Figure 3: Summary of security results for the composed PKAE schemes under the assumptions that the base signature scheme is strongly unforgeable (SUF-CMA) and the base encryption scheme is IND-CPA secure. The shaded regions indicate the new or different results compared to those shown for the symmetric setting [5].

notion in question.

Attacks on the protocols that encrypt before signing are presented in [2, 6], which includes an attack against RUF-PTXT of the schemes constructed via the Encrypt-then-Sign method, and hence, the Sign-then-Encrypt method has been considered to be a better method for constructing PKAE schemes. However, a drawback in the Sign-then-Encrypt method is that the ciphertext is not *publicly verifiable*. A ciphertext is publicly verifiable if the validity of the ciphertext can be verified using the public information only. Public verifiability of ciphertexts may be useful when a third-party needs to distinguish invalid ciphertexts from valid ones, and network filtering by firewalls shown in [8] is one such example. Hence, for some applications like firewall filters, the Sign-then-Encrypt method may not be appropriate. Also, the Sign-then-Encrypt method provides neither TUF-CTXT nor RUF-CTXT. Regarding privacy, a stronger property such as IND-CCA is not obtained in general from a weaker property like IND-CPA using either method. Furthermore, it turns out that the Encrypt-then-Sign method does not provide IND-CCA security even when the base encryption scheme has a stronger security property like IND-CCA. The result significantly differs from the symmetric key case where the *Encrypt-then-MAC* method provides IND-CCA security based only on the IND-CPA and SUF-CMA assumptions on the base primitives [5]. This implies that none of the three composition methods provides a PKAE scheme that is secure under all notions defined in this paper even when we make stronger assumptions on the base primitives. However, this does not mean that we cannot construct *any* scheme that is secure under all notions. In fact, we show below that the Encrypt-then-Sign method can be modified so as to provide security in all notions given in this paper under appropriate assumptions. We also show that an efficient scheme with "reasonable" security gurantees can be constructed when we consider a specific setting like a Diffie-Hellman (discrete-log) based key setting.

### 1.4 Generic and specific constructions achieving security and efficiency

The Encrypt-then-Sign method can be modified so as to give a generic construction that is secure for all notions including IND-CCA and RUF-PTXT, for which the original method is not secure. The modification is simple: for IND-CCA security, encrypt the sender's public key together with the plaintext, and for RUF-PTXT security, sign the receiver's public key together with the ciphertext. These

| Construction Method | Privacy | | Authenticity | | | |
|---|---|---|---|---|---|---|
| | IND-CPA | IND-CCA | TUF-PTXT | TUF-CTXT | RUF-PTXT | RUF-CTXT |
| *ESSR* | secure | secure | secure | secure | secure | secure |
| DHETM | secure | secure | secure | secure | insecure | insecure |

Figure 4: Summary of security results for the two example PKAE schemes. The assumptions for *ESSR* is that the base public key encryption scheme is IND-CCA secure, and the base signature scheme is strongly unforgeable. The assumptions for DHETM is that the base symmetric encryption scheme is IND-CPA secure, and the base MAC is strongly unforgeable, and a hash of a Diffie-Hellman based key looks random.

---

modifications are made from observing that changing the sender or receiver keys associated with the ciphertext is the main type of attacks against IND-CTXT and RUF-PTXT security. The reason that these modifications in the Encrypt-then-Sign method result in IND-CCA and RUF-PTXT security is that they effectively "bind" the public keys of the sender and receiver to the ciphertext so that the mentioned type of attacks does not work any more. We call the scheme constructed based on this modified generic method *ESSR (Encrypt Sender-key then Sign Receiver-key)*, and its security results are shown in Figure 4. Note that the security assumption on the base encryption scheme for *ESSR* is IND-CCA, which is stronger than that for the generic composition method. This might be considered as a weakness in the scheme. Recall, however, that the Encrypt-then-Sign method does not provide IND-CCA security even under the IND-CCA assumption on the base encryption scheme. Also, the other methods do not provide TUF-CTXT security regardless of the assumption on the base signature scheme. Hence, *ESSR* provides better security guarantees than the schemes constructed via the generic composition methods even if it comes at a cost of a strong assumption on the base encryption scheme.

The scheme *DHETM (Diffie-Hellman based Encrypt-Then-MAC)* uses symmetric encryption and MAC schemes whose keys are obtained from the computed common key $K$ by dividing it into two parts. (The common key $K = hash(g^{x_a x_b})$ is computed from the Diffie-Hellman based keys $(g^{x_a}, x_a)$ and $(g^{x_b}, x_b)$ of the sender and receiver, respectively.) After obtaining the keys for the symmetric encryption and MAC schemes, authenticated encryption is done by encrypting the plaintext to get a ciphertext $C$ and appending a MAC of $C$. This construction can be viewed as an adaptation of the asymmetric encryption scheme "DHIES" [1] to the setting mentioned above in order to achieve an additional goal of authenticity, as well as privacy. The summary of its security results are shown in Figure 4. Note that this scheme is more efficient than the generic schemes described earlier, and with regard to privacy, it achieves IND-CCA security based on the weaker IND-CPA assumption (as opposed to the IND-CCA assumption) on the base encryption scheme. With regard to authenticity, it achieves TUF-CTXT security although it does not achieve the receiver unforgeability. In case the stronger authenticity property (i.e. receiver unforgeability) is not needed, this scheme has pragmatic value due to its efficiency and reasonable security guarantees.

### 1.5 Related work

A comprehensive treatment of authenticated encryption (the goal of joint privacy and authenticity) in the symmetric setting is provided in [5]. However, the security notions cannot just be lifted up to the public key setting due to asymmetry of this setting.

Constructions of authenticated encryption schemes with low communication costs in the public key setting based on a discrete logarithm based signature scheme are given in [12], however, without any proofs of security.

In [17], a primitive called "signcryption" is introduced for the first time, and two clever and efficient constructions based on (shortened) variants of the ElGamal signature scheme are given as proposed signcryption schemes. Also, some security goals and security arguments are made for the proposed constructions. Nevertheless, they are not backed up by formal security notions and proofs of security. Subsequently, in [14], the mentioned signcryption schemes are cryptanalyzed and an improvement is suggested, which in turn is cryptanalyzed in [11]. However, most of these cryptanalysis claims are made without clear attack models or definitions of the properties that they are violating.

The security of a signed ElGamal encryption scheme is analyzed in the random-oracle and generic model in [16]; however, the goal of the scheme is to provide privacy, but not authenticity, and the signature scheme is used in order to provide privacy under chosen-ciphertext attack. Its setting assumes only the key pair of the receiver; hence, the signature scheme in this case does not involve the sender's secret key.

## 2 Definitions

This section provides formal definitions for the notions of security of a public key based authenticated encryption scheme discussed in Section 1, and also of digital signature schemes. Associated to each scheme, each notion of security and each adversary is an *experiment*, and based on that, an *advantage*. The latter is a function of the security parameter that measures the success probability of the adversary. Asymptotic notions of security result by asking this function to be negligible for adversaries of time complexity polynomial in the security parameter. Concrete security assessments are made by associating to the scheme another advantage function that for each value of the security parameter and given resources for an adversary returns the maximum, over all adversaries limited to the given resources, of the advantage of the adversary.

We begin by describing the *syntax* of a public key based authenticated encryption scheme, distinguishing syntax from the notions of security.

### 2.1 Syntax of public key based authenticated encryption schemes

The usual syntax of a public key (or asymmetric) encryption scheme is that the receiver has a public, secret key pair and encryption depends on the public key while decryption depends on the secret key. Here we wish to consider a setting where both sender and receiver have their own public and secret key pairs. We consider this setting in order to examine schemes whose goal is to achieve both privacy and authenticity in a public key setting. This requires a change in encryption scheme syntax. Accordingly, we define a *public key based authenticated encryption (PKAE)* scheme which extends the usual public key encryption scheme by addition of another key generation algorithm (i.e. a key generation algorithm for the sender). Specifically, we define a public key based authenticated encryption scheme $\mathcal{PKAE}$ as follows:

**Definition 2.1 [Public key based authenticated encryption (PKAE)]** A *public key based authenticated encryption scheme* $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ consists of five algorithms as follows:

- The *common key generation* algorithm $\mathcal{K}_c$ is randomized. It takes as input a security parameter $k$ and returns some global information $I$; we write $I \stackrel{R}{\leftarrow} \mathcal{K}_c(k)$.
- The *sender key generation* algorithm $\mathcal{K}_s$ is randomized. It takes as input some global information $I$ and returns a matching public and secret key pair $(pk_s, sk_s)$ for the sender; we write $(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{K}_s(I)$.
- The *receiver key generation* algorithm $\mathcal{K}_r$ is randomized. It takes as input some global information $I$ and returns a matching public and secret key pair $(pk_r, sk_r)$ for the receiver; we write $(pk_r, sk_r) \stackrel{R}{\leftarrow} \mathcal{K}_r(I)$.

- The *encryption* algorithm $\mathcal{E}$ is randomized. It takes as input a sender's secret key $sk_s$, a sender's public key $pk_s$, a receiver's public key $pk_r$ and a *plaintext* $M \in \mathcal{M}$, and it flips some coins internally, and then it returns a *ciphertext* $C \in \mathcal{C}$; we write $C \xleftarrow{R} \mathcal{E}_{\langle sk_s, pk_s, pk_r \rangle}(M)$
- The *decryption* algorithm $\mathcal{D}$ is deterministic. It takes as input a receiver's secret key $sk_r$, and a string $C$ to return either a pair $(pk, M)$ or the distinguished symbol $\perp$, where $pk$ is a sender's public key and $M$ is the corresponding plaintext; we write $x \leftarrow \mathcal{D}_{sk_r}(C)$, where $x$ is either $(pk, M)$ or $\perp$.

Above, $\mathcal{M}$ and $\mathcal{C}$ denote the message space and the ciphertext space associated to the scheme, respectively. We require that $\mathcal{D}_{sk_r}(\mathcal{E}_{\langle sk_s, pk_s, pk_r \rangle}(M)) = (pk_s, M)$ for all $M \in \mathcal{M}$. ▮

Discussion of Syntax. The common key generation algorithm produces the global information that is shared by everyone in the system. The global information $I$ includes a security parameter, and possibly some other information. For instance, in a Diffie-Hellman based scheme, $I$ might include a global prime number and generator of a group which all parties use to create their keys. The presence or absence of global information depends on each individual scheme. In case a scheme does not have any global information, the common key generation algorithm can be just the identity function that outputs the security parameter that is given as its input. Note that this algorithm should also be included in the usual public key based encryption scheme syntax when discussing its security in an asymptotic setting. Hence, this is not an added algorithm with respect to the usual public key based encryption scheme syntax. The sender key generation algorithm is what is added in order to consider the goal of authenticity in a public key setting. To generate an "authentic" ciphertext, a sender performs the encryption operation on a plaintext based on its own key pairs as well as the public key of the receiver. Hence, the encryption algorithm takes as input the sender's key pair as well as the receiver's public key. For the goal of authenticity, the receiver should be able to know if a ciphertext it received is "valid" (i.e. authentic) or not. Since there may be more than one sender who can send a ciphertext to a receiver using the receiver's public key, the receiver needs to be able to know who the sender is in order to check whether the ciphertext is authentic with respect to the purported sender. We let the decryption algorithm perform this functionality by requiring it to output the sender's public key (implicitly telling who is the sender), as well as the plaintext. In case the ciphertext is not valid (i.e. unauthentic with respect to the purported sender), the decryption algorithm outputs the distinguished symbol $\perp$. The key explicitly used in the decryption algorithm as input is just the secret key of the receiver, indicating that the decryption algorithm does not initially know which sender's public key to use, but it somehow extracts the needed information from the ciphertext and returns the sender's public key as part of the output.

Verifiable public key based authenticated encryption scheme. Similarly to digital signature schemes, which allow public verifiability, a public key based authenticated encryption scheme may allow public verifiability by providing a verification algorithm that checks the validity of the ciphertext depending only on the public information (such as the public key of a sender or receiver). This is an optional algorithm because not all schemes may have public verifiability —whether or not a scheme is publicly verifiable depends on how it is constructed— and we stress that public verifiability has to do with functionality rather than security. We define the verifiable public key based authenticated encryption as follows: A *verifiable public key based authenticated encryption scheme (VPKAE)* is a 6-tuple $(\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D}, \mathcal{V})$, where $(\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ is a PKAE and $\mathcal{V}$ is a deterministic, public verification algorithm. The latter takes a receiver's public key $pk_r$, and a string $C$ to return either $pk$ or $\perp$; we write $y \leftarrow \mathcal{V}_{pk_r}(C)$, where $y$ is either $pk$ or $\perp$. We require that $\mathcal{V}_{pk_r}(C)$ output $pk$ if $\mathcal{D}_{sk_r}(C) = (pk, M)$ for some $M \in \mathcal{M}$ and $\perp$ otherwise.

Public key (asymmetric) encryption scheme. A standard public key based (or asymmetric) encryption scheme, namely one where there is no sender key, can be recovered as the special case where the sender key generation algorithm $\mathcal{K}_s$ returns the empty string. Formally, we say that $\mathcal{PE} = (\mathcal{K}_c, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ is a public key (or asymmetric) encryption scheme if $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ is a PKAE scheme where $\mathcal{K}_s$ is the algorithm which, on any input, returns the empty string. When the sender key pair $(pk_s, sk_s)$ is the empty string, we may also omit it wherever applicable. For example, the decryption algorithm $\mathcal{D}_{sk_r}(\cdot)$ will return $M$ instead of a pair $(pk_s, M)$ in the standard asymmetric encryption scheme.

## 2.2 Privacy of public key based authenticated encryption schemes

PKAE schemes are different from the usual public key encryption schemes in that the encryption algorithm is computed based on a sender's secret key as well as a receiver's public key. This means that in order to send a message using an authenticated encryption scheme, a party must have its own public, secret key pair. It is not sufficient for it to have access to the receiver's public key. Because of this, we allow the adversary access to a left-or-right (LR) encryption oracle, modeling a chosen-plaintext attack as in the symmetric setting [3], instead of as in the public key setting [9, 4]. Note that this is quantitatively a stronger attack model than the usual one [4], where the adversary is given only the public key for the encryption scheme and no access to the left-or-right encryption oracle. For IND-CPA, a challenge bit $b$ is chosen, the adversary is given public information (including public keys), and can query, adaptively and as often as it likes, the left-or-right encryption oracle. The format of each query to the left-or-right encryption oracle is mandated to be a pair $(x_0, x_1)$ of equal length messages. The adversary wins if it can guess $b$. For IND-CCA the adversary gets, in addition, a decryption oracle with the restriction that it is not allowed to query it on a ciphertext previously returned by the left-or-right encryption oracle.

**Definition 2.2 [Privacy of public key based authenticated encryption schemes]** Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ be a PKAE scheme. Let $b \in \{0,1\}$ and let $k \in \mathsf{N}$ be a security parameter. Let $A_{\text{cpa}}$ and $A_{\text{cca}}$ be adversaries that output a bit $d$. The *left-or-right (LR)* encryption oracle $\mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(\mathcal{LR}(\cdot,\cdot,b))$, given to $A_{\text{cpa}}$ and $A_{\text{cca}}$, takes as input a pair $(x_0, x_1)$ of equal-length messages, computes a ciphertext $Y \leftarrow \mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(x_b)$, and returns $Y$ to the adversary. The decryption oracle $\mathcal{D}_{sk_r}(\cdot)$, given to $A_{\text{cca}}$, takes as input a ciphertext $C$, computes $(pk, M) \leftarrow \mathcal{D}_{sk_r}(C)$, and returns $(pk, M)$ to the adversary. Now, we consider the following experiments:

| Experiment $\mathbf{Exp}^{\text{ind-cpa-b}}_{\mathcal{PKAE},A_{\text{cpa}}}(k)$ | Experiment $\mathbf{Exp}^{\text{ind-cca-b}}_{\mathcal{PKAE},A_{\text{cca}}}(k)$ |
|---|---|
| $I \xleftarrow{R} \mathcal{K}_c(k)$ | $I \xleftarrow{R} \mathcal{K}_c(k)$ |
| $(pk_s, sk_s) \xleftarrow{R} \mathcal{K}_s(I)$ | $(pk_s, sk_s) \xleftarrow{R} \mathcal{K}_s(I)$ |
| $(pk_r, sk_r) \xleftarrow{R} \mathcal{K}_r(I)$ | $(pk_r, sk_r) \xleftarrow{R} \mathcal{K}_r(I)$ |
| $d \leftarrow A_{\text{cpa}}^{\mathcal{E}_{\langle sk_s,pk_s,pk_r\rangle}(\mathcal{LR}(\cdot,\cdot,b))}(I, pk_s, pk_r)$ | $d \leftarrow A_{\text{cca}}^{\mathcal{E}_{\langle sk_s,pk_s,pk_r\rangle}(\mathcal{LR}(\cdot,\cdot,b)),\mathcal{D}_{sk_r}(\cdot)}(I, pk_s, pk_r)$ |
| Return $d$ | Return $d$ |

Above it is mandated that $A_{\text{cca}}$ never queries the decryption oracle $\mathcal{D}_{sk_r}(\cdot)$ on a ciphertext previously output by the LR-encryption oracle $\mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(\mathcal{LR}(\cdot,\cdot,b))$, and that the queries made to the LR-encryption oracle always consist of messages of equal length.

We define the *advantages* of the adversaries via

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKAE},A_{\text{cpa}}}(k) &= \Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{PKAE},A_{\text{cpa}}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{PKAE},A_{\text{cpa}}}(k) = 1\right] \\
\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PKAE},A_{\text{cca}}}(k) &= \Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PKAE},A_{\text{cca}}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PKAE},A_{\text{cca}}}(k) = 1\right] .
\end{aligned}$$

We define the *advantage functions* of the scheme as follows. For any integers $t, q_e, q_d, \mu_e, \mu_d \geq 0$,

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKAE}}(k, t, q_e, \mu_e) &= \max_{A_{\text{cpa}}}\{\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKAE}, A_{\text{cpa}}}(k)\} \\ \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PKAE}}(k, t, q_e, q_d, \mu_e, \mu_d) &= \max_{A_{\text{cca}}}\{\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PKAE}, A_{\text{cca}}}(k)\}\end{aligned}$$

where the maximum is over all adversaries with time complexity $t$, each making at most $q_e$ queries to the $\mathcal{E}_{\langle sk_s, pk_r \rangle}(\mathcal{LR}(\cdot,\cdot,b))$ oracle, totaling at most $\mu_e$ bits, and, in the ind-cca case, also making at most $q_d$ queries to the $\mathcal{D}_{sk_r}(\cdot)$ oracle, totaling at most $\mu_d$ bits. The scheme $\mathcal{PKAE}$ is said to be *IND-CPA secure* (resp. *IND-CCA secure*) if the advantage function $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{PKAE},A}(\cdot)$ (resp. $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PKAE},A}(\cdot)$) is negligible for any adversary $A$ whose time complexity is polynomial in the security parameter $k$. ▮

## 2.3 Authenticity of public key based authenticated encryption schemes

Usually, in the symmetric setting, the adversary is a third-person that is watching over the communication link between the sender and receiver, who share the same key. The security regarding authenticity in this case is normally measured by "unforgeability" by an adversarial third-person (meaning, excluding the sender and receiver). However, when we consider a public key setting where the sender and receiver do not necessarily share the same key, the goal of authenticity can be divided further depending on the adversarial power. For the schemes where only the sender's secret key is involved in creating authentic data, (e.g. signature schemes), the adversary can be anyone (including the intended receiver as well as a third-person) who does not know the secret key of the sender. When a message is signed and sent to an intended receiver, there is no difference between a third-person (who intercepts the signed data) and the receiver (who actually receives the data) in their ability to forge a signature. This is because anyone can verify the signature, and only the sender who owns its secret key can sign the data. However, when we consider a PKAE scheme, where the ciphertext is generated based not only on the sender's key but also on the receiver's public key, there may be a distinction between a third-person and receiver in their ability to forge a ciphertext. What distinguishes a receiver from a third-person is that the receiver's public key is involved in generating the ciphertext, and only the receiver knows its own secret key. Furthermore, the receiver is the one who created its own public, secret key pair in the first place and can change it later. Taking this distinction into account, we divide the goal of authenticity into two parts: unforgeability by a third-person and unforgeability by a receiver. In a third-person attack model against unforgeability, the goal of the adversary is to create a "new forgery" so that the receiver may accept it as valid. This resembles the usual attack model for what is typically called "authenticity" or "data integrity" in the symmetric setting. In a receiver attack model, the goal of the adversary may be to create a forgery so that a third party, such as a judge, may accept it as valid. Proving to a third party that a message has been originated from the purported sender can be considered as the goal of so called "non-repudiation", which is a term commonly used in the literature as an additional security goal (other than "unforgeability") for digital signature schemes. Although there exist formal security notions for authenticity of digital signature schemes, no formal security notions for authenticated encryption schemes exist, regarding the goal of authenticity.

The usual goal of integrity against a third-person adversary for authenticated encryption schemes in the public key setting can be "lifted up" from that in the symmetric setting [5], due to their resemblance. However, the goal of unforgeability against an adversarial receiver (i.e. non-repudiation) has not been formally modeled before for authenticated encryption schemes and is an additional goal that is only applicable in the asymmetric (public key) setting. We call this goal "receiver unforgeability" and describe the attack model and the goal of an adversary in more detail below.

Receiver unforgeability. A receiver adversary publishes its public key by registering it with a CA (certification authority), and a sender encrypts a message using a PKAE scheme based on its own keys and the public key of the receiver. We allow a chosen message attack, modeling the case where the receiver is able to get the sender to encrypt any message that the receiver asks to encrypt. At some point, we allow the receiver to change its key pair, and publish a new public key by re-registering with the CA. The goal of the receiver adversary is to come up with a ciphertext so that a third-party (e.g. a judge) will accept it as "valid", meaning it came from the real sender it claims to have come from. The receiver may choose not to change its key pair, but we allow the change of keys for flexibility on the receiver part. Having accurate timestamps on messages (ciphertexts) and keys can make it possible to determine if a key was registered before or after the given ciphertext was generated, but it may impose additional burdens of clock synchronization and authentication, and it may not always be possible. Hence, we allow this type of key change attacks in order to achieve a stronger security guarantee in that aspect. Also, we note that this type of attacks has appeared in the literature [2, 6], which partially motivated our formalization. For an adversary to be considered successful in forging a ciphertext, in addition to the "validity" requirement, the ciphertext itself needs to be "fresh" —meaning, it was not legitimately generated (i.e. encrypted) by the sender. This is considered one type of successful forgery, and we call the security against this type of forgery "unforgeability of ciphertext". However, there is another type of forgery, which may deem to be more useful and meaningful in practice. That is a forgery of the plaintext corresponding to the ciphertext. The adversary is considered successful in this case if it can come up with a "valid" ciphertext whose corresponding plaintext has not been asked to be encrypted by the sender. We call security against this type of forgery "unforgeability of plaintext".

There remains an important issue of how the adversary will be able to convince a third-party (called "judge" hereafter) to accept the forgery as valid. One of the simplest methods for the adversary may be to give the judge its decryption key, so that it may freely decrypt and check the validity just like the adversary can. All the adversary needs to do is convince the judge that its secret key really matches its public key (i.e. their mathematical relationship holds), which must be verified in some way. Of course, if the secret key is given to the judge, the adversary would not be able to reuse it again (assuming the judge cannot be trusted not to reveal or use the secret key). This may be considered to be too much to ask from an adversary, but this is generally applicable to most PKAE schemes and facilitates security analysis in a uniform and simple way. We are aware that this makes the security notion weaker, but it enables us to provide a framework with which we can analyze security with regard to receiver unforgeability under one simple and uniform definition for all PKAE schemes, regardless of their construction methods. Other more robust and stronger definitions may be built upon this framework.

There may be a number of other ways to convince a judge about the validity of the ciphertext. For example, if the PKAE scheme in question is publicly verifiable, validity of the ciphertext can be proven without revealing the secret key of the receiver adversary. It may also be possible for an adversary to convince a judge that the plaintext corresponding to the forgery ciphertext is valid without revealing any information about its secret key —for example, performing a zero-knowledge protocol with the judge if it's applicable for the scheme. Note however that these methods may not be universally applicable to all PKAE schemes. Applicability of these methods are dependent on how each scheme is constructed. All these methods may be captured as "forgery verification procedures" and having a "forgery verification procedure" for each PKAE scheme may be one solution to this problem. However, it is not clear how this can be systematically analyzed and having a different verification method for each PKAE scheme may make it impossible to compare security accross the schemes. Note that having different methods for convincing a judge can make a difference in the security results for the same scheme. Since one of our goals is to provide comparative security analyses on different methods of constructing PKAE schemes, having a unified measure of security is important. Although there may be better uniform methods for convincing a judge about the validity of the ciphertext than simply giving him the secret decryption

| | |
|---|---|
| Experiment $\mathbf{Exp}^{\text{tuf-ptxt}}_{\mathcal{PKAE},F_p}(k)$<br>$I \leftarrow \mathcal{K}_c(k)$<br>$(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{K}_s(I)$ ; $(pk_r, sk_r) \stackrel{R}{\leftarrow} \mathcal{K}_r(I)$<br>$C \leftarrow F_p^{\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)}(I, pk_s, pk_r)$<br>$x \leftarrow \mathcal{D}_{sk_r}(C)$ ; If $x = \perp$ then return 0<br>Parse $x$ as $(pk, M)$<br>If $pk = pk_s$ and $M$ was never<br>a query to $\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)$<br>then return 1 else return 0 | Experiment $\mathbf{Exp}^{\text{tuf-ctxt}}_{\mathcal{PKAE},F_c}(k)$<br>$I \leftarrow \mathcal{K}_c(k)$<br>$(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{K}_s(I)$ ; $(pk_r, sk_r) \stackrel{R}{\leftarrow} \mathcal{K}_r(I)$<br>$C \leftarrow F_c^{\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)}(I, pk_s, pk_r)$<br>$x \leftarrow \mathcal{D}_{sk_r}(C)$ ; If $x = \perp$ then return 0<br>Parse $x$ as $(pk, M)$<br>If $pk = pk_s$ and $C$ was never<br>a response of $\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)$<br>then return 1 else return 0 |
| Experiment $\mathbf{Exp}^{\text{ruf-ptxt}}_{\mathcal{PKAE},B_p}(k)$<br>$I \leftarrow \mathcal{K}_c(k)$<br>$(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{K}_s(I)$<br>$(st, pk_r) \leftarrow B_{p_1}(I)$<br>$(C, pk_r{}', sk_r{}') \leftarrow B_{p2}^{\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)}(st, pk_s, pk_r)$<br>$x \leftarrow \mathcal{D}_{sk_r{}'}(C)$ ; If $x = \perp$ then return 0<br>Parse $x$ as $(pk, M)$<br>If $pk = pk_s$ and $M$ was never<br>a query to $\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)$<br>then return 1 else return 0 | Experiment $\mathbf{Exp}^{\text{ruf-ctxt}}_{\mathcal{PKAE},B_c}(k)$<br>$I \leftarrow \mathcal{K}_c(k)$<br>$(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{K}_s(I)$<br>$(st, pk_r) \leftarrow B_{c_1}(I)$<br>$(C, pk_r{}', sk_r{}') \leftarrow B_{c2}^{\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)}(st, pk_s, pk_r)$<br>$x \leftarrow \mathcal{D}_{sk_r{}'}(C)$ ; If $x = \perp$ then return 0<br>Parse $x$ as $(pk, M)$<br>If $pk = pk_s$ and $C$ was never<br>a response of $\mathcal{E}_{\langle sk_s,pk_s,pk_r \rangle}(\cdot)$<br>then return 1 else return 0 |

Figure 5: The experiments for Definition 2.3 that defines notions of authenticity of a PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$. The variable $st$ denotes internal state information.

---

key, it is not clear at this point, and pursuing this issue is left for future research.

Summary. The difference between a third-person and a receiver is in its ability to create and change its own key pair. The receiver (to whom a sender sends a ciphertext) can create its own public, secret key pair and later change it. The third-person cannot do so. In both cases, the adversary is allowed a chosen message attack modeled by giving it access to an encryption oracle. Success is measured by its ability to output a "new" forgery that makes the decryption algorithm output a plaintext rather than reject by outputting $\perp$. Depending on the definition of "newness" of the forgery, we divide the forgery into two types: *ciphertext forgery (CTXT)* and *plaintext forgery (PTXT)*. A "ciphertext forgery" means that the ciphertext output by the adversary is different from the ciphertexts obtained from the encryption oracle, and a "plaintext forgery" means that the plaintext corresponding to the ciphertext output by the adversary is different from the plaintext queries made by the adversary to the encryption oracle. Combining the receiver and third-person unforgeability (abbreviated to RUF, TUF) with ciphertext and plaintext forgery (abbreviated to CTXT, PTXT), we get the following four notions: RUF-CTXT, RUF-PTXT, TUF-CTXT, and TUF-PTXT. Note that the symmetric setting counterparts of TUF-CTXT and TUF-PTXT are INT-CTXT and INT-PTXT, defined in [5]. The formal definitions of authenticity (i.e. the four unforgeability notions) are given below.

**Definition 2.3 [Authenticity of a PKAE scheme]** Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ be a PKAE scheme, and let $k \in \mathsf{N}$ be a security parameter. Let $F_p, F_c, B_p, B_c$ be adversaries. Consider the experiments shown in Figure 5.

The experiments $\mathbf{Exp}^{\text{tuf-ptxt}}_{\mathcal{PKAE},F_p}(k)$ and $\mathbf{Exp}^{\text{tuf-ctxt}}_{\mathcal{PKAE},F_c}(k)$ model the forgery attacks by a third-person who is given the public information only, and a valid forgery is measured by the "newness" of either the plaintext or the ciphertext, respectively. The experiments $\mathbf{Exp}^{\text{ruf-ptxt}}_{\mathcal{PKAE},B_p}(k)$ and $\mathbf{Exp}^{\text{ruf-ctxt}}_{\mathcal{PKAE},B_c}(k)$ model the forgery attacks (of plaintext and ciphertext, respectively) by a receiver, where the adversary creates its own key pair in the first stage and is allowed to change its key pair in the second stage. It is mandated that a "valid" public, secret key pair relationship hold for every key pair created by an adversary if such a relationship should exist for a given scheme.

We define the *advantages* of the adversaries via,

$$\mathbf{Adv}^{\text{xxx-yyyy}}_{\mathcal{PKAE},F}(k) = \Pr\Big[\, \mathbf{Exp}^{\text{xxx-yyyy}}_{\mathcal{PKAE},F}(k) = 1 \,\Big]$$

where xxx $\in \{\text{tuf}, \text{ruf}\}$ and yyyy $\in \{\text{ptxt}, \text{ctxt}\}$.

We define the *advantage functions* of the scheme for *third-person/receiver unforgeability of plaintext/ciphertext (TUF-PTXT, TUF-CTXT, RUF-PTXT, RUF-CTXT resp.)* as follows. For $t, q, \mu \geq 0$, and xxx $\in \{\text{tuf}, \text{ruf}\}$ and yyyy $\in \{\text{ptxt}, \text{ctxt}\}$ let

$$\mathbf{Adv}^{\text{xxx-yyyy}}_{\mathcal{PKAE}}(k, t, q, \mu) = \max_{F}\{\mathbf{Adv}^{\text{xxx-yyyy}}_{\mathcal{PKAE},F}(k)\}$$

where the maximum is taken over all $F$ with time complexity $t$, making at most $q$ queries to the oracle $\mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)$, such that the sum of the lengths of all oracle queries is at most $\mu$ bits. The scheme $\mathcal{PKAE}$ is said to be *XXX-YYYY secure*, where $XXX \in \{TUF, RUF\}$ and $YYYY \in \{PTXT, CTXT\}$, if the function $\mathbf{Adv}^{\text{xxx-yyyy}}_{\mathcal{PKAE},F}(\cdot)$, where xxx $\in \{\text{tuf}, \text{ruf}\}$ and yyyy $\in \{\text{ptxt}, \text{ctxt}\}$, is negligible for any adversary $F$ whose time complexity is polynomial in the security parameter $k$. ▮

### 2.4 Signature schemes

Syntax of Signature schemes. A *digital signature scheme* $\mathcal{DS} = (\mathcal{K}_c, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ consists of four algorithms. The randomized *common key generation algorithm* $\mathcal{K}_c$ takes as input the security parameter $k$ and returns some global information $I$; we write $I \stackrel{R}{\leftarrow} \mathcal{K}_c(k)$. The randomized *signature key generation algorithm* $\mathcal{KS}$ takes as input some global information $I$ and returns a *public and secret key pair* $(pk, sk)$; we write $(pk, sk) \stackrel{R}{\leftarrow} \mathcal{KS}(I)$. The randomized or deterministic *signing* algorithm $\mathcal{S}$ takes as input $sk$ and a message $M$ to be signed and returns a signature $\sigma$ ; we write $\sigma \stackrel{R}{\leftarrow} \mathcal{S}_{sk}(M)$. The deterministic *signature verification* algorithm $\mathcal{VS}$ takes as input $pk$, a message $M$ and a candidate signature $\sigma$ for $M$ and returns a bit $b \in \{0, 1\}$; we write $b \leftarrow \mathcal{VS}_{pk}(M, \sigma)$. We require that for all $(pk, sk)$ and $M$, $\mathcal{VS}_{pk}(M, \mathcal{S}_{sk}(M)) = 1$.

Security Notions of signature schemes. We recall the standard definition of security ("unforgeability") of a signature scheme under chosen-message attack (cf. [10]), and adapt a stronger notion of security ("strong unforgeability") defined for message authentication schemes [5] to signature schemes. Security for signature schemes considers an adversary $F$ who is allowed a chosen-message attack, modeled by allowing it access to an oracle for $\mathcal{S}_{sk}(\cdot)$. $F$ is "successful" if it can make the verifying algorithm $\mathcal{VS}_{pk}(\cdot, \cdot)$ accept a pair $(M, \sigma)$ that was not "legitimately produced." There are two possible conventions depending on the meaning of "legitimately produced", leading to two measures of advantage. The "standard" measure is that the message $M$ is "new," meaning $F$ never made query $M$ to its signing oracle. We call this security measure *weak unforgeability under chosen-message attack (WUF-CMA).* A more stringent measure, called *strong unforgeability under chosen-message attack (SUF-CMA)*, considers the adversary successful even if the message is not new, as long as the signature is new. This means that the adversary wins as long as $\sigma$ was never returned by the signing oracle in response to query $M$ (i.e. $(M, \sigma)$ as a pair is new). The formal definitions of these notions are given below.

**Definition 2.4 [Signature Scheme Security]** Let $\mathcal{DS} = (\mathcal{K}_c, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ be a digital signature scheme, and let $k \in \mathsf{N}$ be a security parameter. Let $F$ be an adversary (forger) that has access to a signing oracle $\mathcal{S}_{sk}(\cdot)$. Consider the following experiments:

| Experiment $\mathbf{Exp}^{\text{wuf-cma}}_{\mathcal{DS},F}(k)$ | Experiment $\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k)$ |
|---|---|
| $I \xleftarrow{R} \mathcal{K}_c(k)$ | $I \xleftarrow{R} \mathcal{K}_c(k)$ |
| $(pk, sk) \xleftarrow{R} \mathcal{KS}(I)$ | $(pk, sk) \xleftarrow{R} \mathcal{KS}(I)$ |
| $(M, \sigma) \leftarrow F^{\mathcal{S}_{sk}(\cdot)}(I, pk)$ | $(M, \sigma) \leftarrow F^{\mathcal{S}_{sk}(\cdot)}(I, pk)$ |
| If $\mathcal{VS}_{pk}(M, \sigma) = 1$ and $M$ was never a query to the oracle $\mathcal{S}_{sk}(\cdot)$ | If $\mathcal{VS}_{pk}(M, \sigma) = 1$ and $\mathcal{S}_{sk}(\cdot)$ never returned $\sigma$ on input $M$ |
| `then return` 1 `else return` 0 | `then return` 1 `else return` 0 |

We define the *advantages* of adversaries via,

$$\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(k) = \Pr\left[\, \mathbf{Exp}^{\text{wuf-cma}}_{\mathcal{DS},F}(k) = 1 \,\right]$$

$$\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k) = \Pr\left[\, \mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k) = 1 \,\right]$$

We define the advantage functions of the scheme $\mathcal{DS}$ as follows. For any $t, q, \mu \geq 0$,

$$\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS}}(k, t, q, \mu) = \max_F\{\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(k)\}$$

$$\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS}}(k, t, q, \mu) = \max_F\{\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k)\}$$

where the maximum is over all $F$ with time complexity $t$, making at most $q$ oracle queries to $\mathcal{S}_{sk}(\cdot)$, such that the sum of the lengths of all queries is at most $\mu$ bits. The scheme $\mathcal{DS}$ is said to be *WUF-CMA secure* (resp. *SUF-CMA secure*) if the advantage function $\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(\cdot)$ (resp. $\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(\cdot)$) is negligible for any adversary $F$ whose time complexity is polynomial in the security parameter $k$. ∎

# 3 Relations among notions of unforgeability

In this section, we provide formal statements of the results summarized in Figure 1. We start with implications first and then present separations.

The third-person unforgeability notions (TUF-CTXT and TUF-PTXT) in the public key setting are analogous to the integrity notions (INT-CTXT and INT-PTXT) in the symmetric setting shown in [5]. Since the proofs for the separation and implication relations between INT-CTXT and INT-PTXT for the symmetric setting can be easily "lifted up" to the public key setting, we omit the proofs for the relations between TUF-PTXT and TUF-CTXT here, and move onto the other relations.

The implication relations between RUF-CTXT and TUF-CTXT, and between RUF-PTXT and TUF-PTXT shown in Proposition 3.1 can be directly obtained from their definitions. We prove the implication relations using the standard reduction arguments. A separation relation is shown by presenting a scheme that is secure under the assumed security notion, and yet not secure under the other security notion.

There is an important distinction between the symmetric setting and the public key setting regarding the relationship between unforgeability of ciphertext and chosen-ciphertext attack security. Recall that, in the symmetric setting, the chosen-ciphertext attack security (IND-CCA) is implied by the combined security of IND-CPA and INT-CTXT [13, 5]. Note, however, that in the public key setting, IND-CCA is *not* implied by the combination of IND-CPA and even a stronger notion of unforgeability, RUF-CTXT. This is mainly because there is a distinction between the notions of IND-CCA and RUF-CTXT with regard to the definitions of "valid ciphertexts", which, in turn, is caused by the orthogonality of the goals of privacy and authenticity and the asymmetry of the key structure in the public key setting. In the chosen-ciphertext attack model, a ciphertext queried to a decryption oracle is considered "valid" if it is valid with respect to the public key of *any* sender, allowing the adversary to use its own public, secret key pair to generate a "valid" ciphertext query with respect to its own key. This is because the goal of the adversary is not to "forge" a ciphertext, but to obtain information about the challenge ciphertexts output by its left-or-right encryption oracle. However, to be considered successful in an attack against "authenticity" of a particular sender, the adversary must generate a ciphertext that is considered valid with respect to the public key of the particular sender, but not its own.

In the following propositions, we omit the concrete statements for simplicity; but, they can be easily derived from the asymptotic ones. The proofs of the following propositions are given in Section A.

**Proposition 3.1** [RUF-CTXT $\rightarrow$ TUF-CTXT (resp. RUF-PTXT $\rightarrow$ TUF-PTXT)] Let $\mathcal{PKAE}$ be a PKAE scheme. If $\mathcal{PKAE}$ is RUF-CTXT secure (resp. RUF-PTXT secure), then it is TUF-CTXT secure (resp. TUF-PTXT secure). ▮

**Proposition 3.2** [TUF-CTXT $\not\rightarrow$ RUF-CTXT (resp. TUF-PTXT $\not\rightarrow$ RUF-PTXT)] Given a group generator algorithm $\mathcal{GG}$ and a hash function $H$, where $H$ is hardcore on group $\mathcal{GG}$ (the HDH assumption) as per Definition 5.5, and a SUF-CMA secure MAC scheme $\mathsf{MA}$, we can construct a PKAE scheme that is TUF-CTXT secure (resp. TUF-PTXT secure) but is *not* RUF-CTXT secure (resp. RUF-PTXT secure). ▮

**Proposition 3.3** [RUF-CTXT $\not\rightarrow$ RUF-PTXT] Given an IND-CPA secure public key encryption scheme $\mathcal{PE}$ and a SUF-CMA secure signature scheme $\mathcal{DS}$, we can construct a PKAE scheme $\mathcal{PKAE}$ that is RUF-CTXT secure, but is *not* RUF-PTXT secure. ▮

**Proposition 3.4** [RUF-PTXT $\not\rightarrow$ TUF-CTXT] Given a PKAE scheme $\mathcal{PKAE}$ that is RUF-PTXT secure, we can construct another PKAE scheme $\mathcal{PKAE}'$ that is also RUF-PTXT secure but is *not* TUF-CTXT secure. ▮

**Proposition 3.5** [IND-CPA $\wedge$ RUF-CTXT $\not\rightarrow$ IND-CCA (resp. IND-CPA $\wedge$ TUF-CTXT $\not\rightarrow$ IND-CCA)] Given an IND-CPA secure public key based encryption scheme $\mathcal{PE}$ and a SUF-CMA secure signature scheme $\mathcal{DS}$, we can construct a PKAE scheme $\mathcal{PKAE}$ that is IND-CPA secure and RUF-CTXT secure (resp. IND-CPA secure and TUF-CTXT secure), but is *not* IND-CCA secure. ▮

# 4 Generic compositions of signature and encryption

We now present PKAE schemes constructed based on generic compositions of a signature and an encryption scheme, and their formal security results.

## 4.1 Constructions

Here we show constructions of a PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ based on $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ using the following three composition methods: *Encrypt-and-Sign*, *Sign-then-Encrypt* and *Encrypt-then-Sign*. The key generation algorithms $\mathcal{K}_c$, $\mathcal{K}_s$ and $\mathcal{K}_r$ remain the same across the three methods. Hence, we do not present the key generation algorithms for each method separately, but present them below once for all three composition methods:

| Algorithm $\mathcal{K}_c(k)$ | Algorithm $\mathcal{K}_s(I)$ | Algorithm $\mathcal{K}_r(I)$ |
|---|---|---|
| $I_e \xleftarrow{R} \mathcal{K}_{ce}(k)$ | Parse $I$ as $(I_e, I_s)$ | Parse $I$ as $(I_e, I_s)$ |
| $I_s \xleftarrow{R} \mathcal{K}_{cs}(k)$ | $(pk_s, sk_s) \xleftarrow{R} \mathcal{KS}(I_s)$ | $(pk_r, sk_r) \xleftarrow{R} \mathcal{K}_e(I_e)$ |
| $I \leftarrow (I_e, I_s)$ | return $(pk_s, sk_s)$ | return $(pk_r, sk_r)$ |
| return $I$ | | |

We now present the encryption and decryption algorithms for each composition method separately, starting from the Encrypt-and-Sign method.

**Construction 4.1** [**Encrypt-and-Sign**] The Encrypt-and-Sign method encrypts the plaintext using the base public key encryption scheme and signs the plaintext using the signature scheme, and then concatenates the encryption output and the signature. The details of the encryption and decryption algorithms are shown below:

$$
\begin{array}{l|l}
\text{Algorithm } \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(M) & \text{Algorithm } \overline{\mathcal{D}}_{sk_r}(C) \\
\quad C' \leftarrow \mathcal{E}_{pk_r}(M) & \quad \text{Parse } C \text{ as } pk_s \| C' \| \sigma \text{ ; } M \leftarrow \mathcal{D}_{sk_r}(C') \\
\quad \sigma \leftarrow \mathcal{S}_{sk_s}(M) & \quad \text{If } \mathcal{VS}_{pk_s}(M, \sigma) = 1 \\
\quad \text{return } pk_s \| C' \| \sigma & \qquad \text{then return } (pk_s, M) \text{ else return } \perp
\end{array}
$$

Above, the encryption algorithm outputs the public key of the sender as part of the output ciphertext so that the decryption algorithm can output the public key of the sender upon receiving the ciphertext. ▌

Note that the encryption algorithms for the remaining two methods also output the public key of the sender as part of the ciphertext. Intuitively, it is used for an identification purpose in the decryption algorithm so as to tell who is the sender or which key to use to verify the ciphertext.

**Construction 4.2** [**Sign-then-Encrypt**] The Sign-then-Encrypt method first signs the plaintext using the base signature scheme, and then encrypts the plaintext along with the signature (i.e. the signature is appended to the plaintext and then the resulting string is encrypted). The details of the algorithms are shown below:

$$
\begin{array}{l|l}
\text{Algorithm } \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(M) & \text{Algorithm } \overline{\mathcal{D}}_{sk_r}(C) \\
\quad \sigma \leftarrow \mathcal{S}_{sk_s}(M) & \quad \text{Parse } C \text{ as } pk_s \| C' \text{ ; } M \| \sigma \leftarrow \mathcal{D}_{sk_r}(C') \\
\quad C' \leftarrow \mathcal{E}_{pk_r}(M \| \sigma) & \quad \text{If } \mathcal{VS}_{pk_s}(M, \sigma) = 1 \\
\quad \text{return } pk_s \| C' & \qquad \text{then return } (pk_s, M) \text{ else return } \perp
\end{array}
$$

Above, $M \| \sigma \leftarrow \mathcal{D}_{sk_r}(C')$ denotes the combined opertations of decrypting and parsing (i.e. a string is obtained from the decryption algorithm on input $C'$ and then divided into two parts $M$ and $\sigma$). ▌

**Construction 4.3** [**Encrypt-then-Sign**] The Encrypt-then-Sign method first encrypts the plaintext using the base public key encryption scheme to obtain a ciphertext, and then signs the obtained ciphertext. A scheme constructed based on the Encrypt-then-Sign method is not only a PKAE scheme, but also a verifiable PKAE (VPKAE) scheme since the ciphertext is publicly verifiable. Hence, for the Encrypt-then-Sign method, we present the verification algorithm as well as the encryption and decryption algorithms. The details of the algorithms are shown below:

$$
\begin{array}{l|l|l}
\text{Algorithm } \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(M) & \text{Algorithm } \overline{\mathcal{D}}_{sk_r}(C) & \text{Algorithm } \overline{\mathcal{V}}_{pk_r}(C) \\
\quad C' \leftarrow \mathcal{E}_{pk_r}(M) & \quad \text{Parse } C \text{ as } pk_s \| C' \| \sigma & \quad \text{Parse } C \text{ as } pk_s \| C' \| \sigma \\
\quad \sigma \leftarrow \mathcal{S}_{sk_s}(C') & \quad M \leftarrow \mathcal{D}_{sk_r}(C') & \quad \text{If } \mathcal{VS}_{pk_s}(C', \sigma) = 1 \\
\quad \text{return } pk_s \| C' \| \sigma & \quad \text{If } \mathcal{VS}_{pk_s}(C', \sigma) = 1 & \quad \text{then return } pk_s \\
 & \quad \text{then return } (pk_s, M) \text{ else return } \perp & \quad \text{else return } \perp
\end{array}
$$

Above, the verification algorithm can be thought of as a decryption algorithm that does not perform decryption but just checks the validity of the ciphertext using the public information (i.e. public keys) only. ▌

## 4.2 Security Analyses

We now show the formal security results for the above schemes as summerized in Figure 3. Security of schemes based on generic compositions of symmetric encryption and MAC has been shown in the symmetric setting (with respect to symmetric encryption and MAC security) in [5]. With regard to the privacy properties, their analyses on the IND-CPA and IND-CCA security can be easily carried over to the public key setting for the Encrypt-and-Sign and Sign-then-Encrypt methods because the signature is computed based on the plaintext in both methods. However, for the Encrypt-then-Sign method, there is a distinction, especially with regard to the IND-CCA security. Hence, regarding the privacy properties, we will give explicit analysis only on the IND-CCA security of the Encrypt-then-Sign method here, and omit the analyses for other methods. With regard to the authenticity properties, recall that the integrity notions (INT-CTXT, INT-PTXT) in [5] are analogous to the third-person unforgeability notions (TUF-CTXT, TUF-PTXT) defined in this paper. However, the notions related to the receiver unforgeability (RUF-PTXT and RUF-CTXT) are new and specific to the public key setting. Here we focus on security analyses with respect to these new notions of unforgeability and refer to the analyses of [5] for the rest of security results.

For security analyses, we assume that the base encryption scheme is IND-CPA secure and the base digital signature scheme is either WUF-CMA or SUF-CMA secure, whose notions are defined in Section 2. We note that SUF-CMA security implies WUF-CMA security, which is shown in [5]. For the Encrypt-and-Sign and Sign-then-Encrypt methods, both WUF-CMA and SUF-CMA security assumptions on the base signature scheme give rise to the same results for all security notions. For the Encrypt-then-Sign method, however, the SUF-CMA assumption on the base signature scheme makes a difference in the TUF-CTXT and RUF-CTXT security results; that is, the WUF-CMA assumption is not strong enough to make the PKAE scheme secure even in the TUF-CTXT sense (which in turn makes it not strong enough for the RUF-CTXT security), while the SUF-CMA assumption on the base signature scheme suffices to make the associated PKAE scheme secure in the RUF-CTXT sense. Note that regardless of the assumptions on the base encryption or signature scheme, *any* scheme constructed via the Encrypt-then-Sign method is neither RUF-PTXT secure nor IND-CCA secure. These negative results are quite strong since they mean that the method is inherently insecure in these senses. The fact that the Encrypt-then-Sign method does not provide IND-CCA security regardless of the security assumption on the base primitives is one of the main differences between the results in the public key setting and those in the symmetric setting. This means that the PKAE scheme constructed via the Encrypt-then-Sign method cannot be IND-CCA secure even when the base encryption scheme is IND-CCA secure. The main reason for this is that the ciphertext part and the signature part in the output of the encryption algorithm are generated independently. Since the keys for the base encryption scheme and the signature scheme are independent of each other, anyone who can sign with its own secret key can just take the ciphertext part of encrypted output and replace the signature part with its own signature on the ciphertext.

The Encrypt-and-Sign and Sign-then-Encrypt methods are not TUF-CTXT secure and since RUF-CTXT security implies TUF-CTXT security (cf. Proposition 3.1), by the contrapositive argument, they are not RUF-CTXT secure. The proofs of the following propositions and theorems are presented in Section A.

**Proposition 4.4 [Encrypt-and-Sign and Sign-then-Encrypt methods are RUF-PTXT secure]** Let $\mathcal{PE}$ be a public key encryption scheme, and let $\mathcal{DS}$ be a digital signature scheme. Then, the PKAE scheme constructed from $\mathcal{PE}$ and $\mathcal{DS}$ via the Encrypt-and-Sign method or Sign-then-Encrypt method is RUF-PTXT secure if $\mathcal{DS}$ is WUF-CMA or SUF-CMA secure. ▮

**Proposition 4.5 [Encrypt-and-Sign and Sign-then-Encrypt methods are not TUF-CTXT secure]** Given an IND-CPA secure public key encryption scheme $\mathcal{PE}$, and a WUF-CMA or SUF-CMA secure digital signature scheme $\mathcal{DS}$, we can construct a public key encryption scheme $\mathcal{PE}'$ such that $\mathcal{PE}'$ is IND-CPA secure, but the PKAE scheme constructed from $\mathcal{PE}'$ and $\mathcal{DS}$ via the Encrypt-and-Sign method or Sign-then-Encrypt method is not TUF-CTXT secure. ▮

**Theorem 4.6 [Encrypt-then-Sign method is not RUF-PTXT secure for any encryption and signature scheme]** Let $\mathcal{PE}$ be a public key encryption scheme, and let $\mathcal{DS}$ be a digital signature scheme. Then, the PKAE scheme constructed from $\mathcal{PE}$ and $\mathcal{DS}$ via the Encrypt-then-Sign method is not RUF-PTXT secure. ▮

**Theorem 4.7 [Encrypt-then-Sign method with WUF-CMA signature scheme is not TUF-CTXT secure]** Given an IND-CPA secure public key encryption scheme $\mathcal{PE}$, and a WUF-CMA secure digital signature scheme $\mathcal{DS}$, we can construct a digital signature scheme $\mathcal{DS}'$ such that $\mathcal{DS}'$ is WUF-CMA secure, but the PKAE scheme $\mathcal{PKAE}$ constructed from $\mathcal{PE}$ and $\mathcal{DS}'$ via the Encrypt-then-Sign method is not TUF-CTXT secure. ▮

**Theorem 4.8 [Encrypt-then-Sign method with SUF-CMA secure signature scheme is RUF-CTXT secure]** Let $\mathcal{PE}$ be a public key encryption scheme, and let $\mathcal{DS}$ be a digital signature scheme. Then, the PKAE scheme constructed from $\mathcal{PE}$ and $\mathcal{DS}$ via the Encrypt-then-Sign method is RUF-CTXT secure if $\mathcal{DS}$ is SUF-CMA secure. ▮

**Theorem 4.9 [Encrypt-then-Sign method is not IND-CCA secure for any encryption and signature scheme]** Let $\mathcal{PE}$ be a public key encryption scheme, and let $\mathcal{DS}$ be a signature scheme. Then, the PKAE scheme constructed from $\mathcal{PE}$ and $\mathcal{DS}$ via the Encrypt-then-Sign method is not IND-CCA secure. ▮

# 5 Example constructions for achieving security and efficiency

In this section, we give analyses of two example constructions, called *ESSR* and *DHETM*, where the former achieves security under *all* notions and the latter achieves efficiency with "reasonable" security guarantees.

## 5.1 A modified generic composition method that meets all security notions

Here we give a generic construction of a public key based authenticated encryption scheme that is secure under all notions of privacy and authenticity defined in this paper. The scheme is constructed based on the Encrypt-then-Sign method with slight modifications in order to provide security in the RUF-PTXT and IND-CCA sense, for which the (unmodified) Encrypt-then-Sign method is not secure. Basically, the modifications come in two places: in the content that is being encrypted, and in the content that is being signed. In the modified Encrypt-then-Sign method, the sender's public key is encrypted together with the plaintext, and also the receiver's public key is signed together with the ciphertext.

We now present the construction of $ESSR = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$. The key generation algorithms ($\mathcal{K}_c$, $\mathcal{K}_s$, $\mathcal{K}_r$) are the same as those for the generic composition methods shown in Section 4.1; hence, we omit their descriptions here. The construction is shown in more detail below:

**Construction 5.1** [*ESSR*] Let $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. Let $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ be a signature scheme. From these primitives, we define the public key based authenticated encryption scheme $ESSR = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ as follows (where the key generation algorithms $\mathcal{K}_c$, $\mathcal{K}_s$, $\mathcal{K}_r$ are the same as the ones shown in Section 4.1):

| Algorithm $\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(M)$ | Algorithm $\overline{\mathcal{D}}_{sk_r}(C)$ |
|---|---|
| $C' \leftarrow \mathcal{E}_{pk_r}(M \| pk_s)$ | Parse $C$ as $pk_s \| C' \| \sigma$ |
| $\sigma \leftarrow \mathcal{S}_{sk_s}(C' \| pk_r)$ | Obtain $pk_r$ from $sk_r$ |
| $C \leftarrow pk_s \| C' \| \sigma$ | If $\mathcal{VS}_{pk_s}(C' \| pk_r, \sigma) = 0$ then return $\perp$ |
| return $C$ | $M \| pk \leftarrow \mathcal{D}_{sk_r}(C')$ |
| | If $pk = pk_s$ then return $M$ else return $\perp$ |

Above, the statement "Obtain $pk_r$ from $sk_r$" in the decryption algorithm is based on the assumption that the public key is *included* in the secret key. Note that the assumption does not necessarily impose a restriction in real implemtations of the scheme, because the receiver's public key is usually included as part of the secret key anyway. ▮

Note that *ESSR* is not a VPKAE scheme, unlike the original Encrypt-then-Sign method. This is because the verification requires decryption (for the comparison of the decrypted sender key against the sender key used in the signature verification), hence not allowing ciphertext verification based only on public keys.

SECURITY ANALYSIS. In order to show that the scheme is secure under all notions, we just need to show it is secure under IND-CCA, RUF-PTXT, and RUF-CTXT since the rest follows from the implications of notions summarized in Figure 1. Since the construction is based on the Encrypt-then-Sign method, its security is preserved and with the modifications, additional security properties (i.e. IND-CCA and RUF-PTXT) that are missing in the original Encrypt-then-Sign method are obtained. Intuitively, encrypting the sender's public key together with the plaintext and signing the receiver's public key together with the ciphertext help provide security in the IND-CCA and RUF-PTXT sense, since the former binds the plaintext with the sender's public key (for the IND-CCA security), while the latter binds the ciphertext with the receiver's public key (for the RUF-PTXT security). Note that in order to obtain the IND-CCA security for *ESSR*, we assume that the underlying public key encryption scheme is IND-CCA secure, which is different from the assumption we make for security of the generic composition methods in Section 4. The proofs of the following theorems are given in Section A.

**Theorem 5.2** [*ESSR* **is IND-CCA secure**] Let $\mathcal{PE}$ be a public key encryption scheme, and let $\mathcal{DS}$ be a digital signature scheme. Then, if $\mathcal{PE}$ is IND-CCA secure, and $\mathcal{DS}$ is SUF-CMA secure then *ESSR* constructed based on $\mathcal{PE}$ and $\mathcal{DS}$ as per Construction 5.1 is IND-CCA secure. ▮

**Theorem 5.3** [*ESSR* **is both RUF-PTXT and RUF-CTXT secure**] Let $\mathcal{PE}$ be a public key encryption scheme and let $\mathcal{DS}$ be a digital signature scheme. Then, if $\mathcal{DS}$ is WUF-CMA secure (resp. SUF-CMA secure), then *ESSR* constructed based on $\mathcal{PE}$ and $\mathcal{DS}$ as per Construction 5.1 is RUF-PTXT secure (resp. RUF-CTXT secure). ▮

## 5.2 An Efficient scheme based on Diffie-Hellman keys

Here we give security analyses of an efficient public key based authenticated encryption scheme called DHETM, which can be viewed as an adaptation of the DHIES (encryption) scheme [1] to the public key setting where a sender as well as a receiver has its public, secret key pair for enabling constructions of authenticated encryption schemes. Note that the difference between the two is in their security goals: the security goal of the DHIES encryption scheme is to provide privacy only and that of DHETM is to provide the joint goals of privacy *and* authenticity. The PKAE scheme DHETM is constructed based on the following four primitives: a group of a prime order which is generated by a group generator algorithm, a hash function, a symmetric encryption scheme, and a MAC scheme. We describe below the syntactic definitions of the primitives and then move onto the actual construction of DHETM.

Group generator algorithm. A *group generator algorithm* $\mathcal{GG}$ is a randomized algorithm. It takes as input a security parameter $k$ and returns a pair $(q, g)$, where $q$ is a prime number indicating the order of a group $G$, and $g$ is a generator of the group. We write $(q, g) \xleftarrow{R} \mathcal{GG}(k)$.

Hash function. A *hash function* $H\colon \{0,1\}^* \to \{0,1\}^{L_h}$ is a function that takes a string of an arbitray length and returns a string of a certain fixed length $L_h \in \mathsf{N}$.

Symmetric encryption scheme. A symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The randomized *key generation* algorithm $\mathcal{K}$ takes as input a security parameter $k$ and returns a string called *key* $K$; we let $\mathsf{Keys}(\mathcal{SE})$ denote the set of all strings that have non-zero probability of being output by $\mathcal{K}(k)$; we write $K \xleftarrow{R} \mathcal{K}(k)$. The randomized or stateful *encryption* algorithm $\mathcal{E}$ takes as input the key $K \in \mathsf{Keys}(\mathcal{SE})$ and a *plaintext* $M \in \{0,1\}^*$ and returns a *ciphertext* $C \in \{0,1\}^*$; we write $C \xleftarrow{R} \mathcal{E}_K(M)$. The deterministic decryption algorithm $\mathcal{D}$ takes as input a key $K \in \mathsf{Keys}(\mathcal{SE})$ and a ciphertext $C \in \{0,1\}^*$ and returns a plaintext $M \in \{0,1\}^*$; we write $M \leftarrow \mathcal{D}_K(C)$. We require that for any key $K \in \mathsf{Keys}(\mathcal{SE})$ and any message $M \in \{0,1\}^*$, $\mathcal{D}_K(\mathcal{E}_K(M)) = M$.

MAC scheme. A MAC scheme $\mathsf{MA} = (\mathcal{K}, \mathsf{T}, \mathsf{VT})$ consists of three algorithms. The randomized *key generation* algorithm $\mathcal{K}$ takes as input a security parameter $k$ and returns a key $K$; we let $\mathsf{Keys}(\mathsf{MA})$ denote the set of all strings that have non-zero probability of being output by $\mathcal{K}(k)$; we write $K \xleftarrow{R} \mathcal{K}(k)$. The randomized or deterministic *tagging* algorithm $\mathsf{T}$ takes as input the key $K \in \mathsf{Keys}(\mathsf{MA})$ and a message $M \in \{0,1\}^*$ and returns a *tag* $\tau \in \{0,1\}^{L_t}$, where $L_t \in \mathsf{N}$ is a certain fixed length called a *tag length*; we write $\tau \xleftarrow{R} \mathsf{T}_K(M)$. The deterministic *verification* algorithm $\mathsf{VT}$ takes as input the key $K \in \mathsf{Keys}(\mathsf{MA})$, a message $M$, and a candidate tag $\tau \in \{0,1\}^{L_t}$ for $M$ and returns a bit $b$; we write $b \leftarrow \mathsf{VT}_K(M, \tau)$. We require that $\mathsf{VT}_K(M, \mathsf{T}_K(M)) = 1$ for all $M \in \{0,1\}^*$ and $K \in \mathsf{Keys}(\mathsf{MA})$.

DHETM. The PKAE scheme DHETM is constructed based on the Encrypt-then-MAC method using the computed Diffie-Hellman key as the common secret key. The details of the construction are described below.

**Construction 5.4** [DHETM] Let $\mathsf{SE} = (\mathsf{K_e}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme where the key generation algorithm simply returns a random $L_e$-bit string, so that the key space is $\mathsf{Keys}(\mathsf{SE}) = \{0,1\}^{L_e}$. Let $\mathsf{MA} = (\mathsf{K_m}, \mathsf{T}, \mathsf{VT})$ be a MAC scheme where the key generation algorithm returns a random $L_m$-bit string, so that the key space is $\mathsf{Keys}(\mathsf{MA}) = \{0,1\}^{L_m}$. Let $H\colon \{0,1\}^* \to \{0,1\}^{L_h}$ be a hash function, where $L_h = L_e + L_m$, and let $\mathcal{GG}$ be a group generator algorithm. Based on these primitives, the PKAE scheme $\mathsf{DHETM} = (\overline{\mathcal{K}_c}, \overline{\mathcal{K}_s}, \overline{\mathcal{K}_r}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ is constructed as follows:

| Algorithm $\overline{\mathcal{K}_c}(k)$ | Algorithm $\overline{\mathcal{K}_s}(q,g)$ | Algorithm $\overline{\mathcal{K}_r}(q,g)$ |
|---|---|---|
| $(q,g) \xleftarrow{R} \mathcal{GG}(k)$ | $x_a \xleftarrow{R} Z_q$ | $x_b \xleftarrow{R} Z_q$ |
| return $(q,g)$ | $y_a \leftarrow g^{x_a}$ | $y_b \leftarrow g^{x_b}$ |
| | return $(y_a, x_a)$ | return $(y_b, x_b)$ |

| Algorithm $\overline{\mathcal{E}}_{\langle x_a, y_a, y_b \rangle}(M)$ | Algorithm $\overline{\mathcal{D}}_{x_b}(C)$ |
|---|---|
| $K \leftarrow H(y_b^{x_a})$ ; Parse $K$ as $K_e \Vert K_m$ | Parse $C$ as $y_a \Vert C' \Vert \tau$ |
| $C' \leftarrow \mathsf{E}_{K_e}(M)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(C')$ | $K \leftarrow H(y_a^{x_b})$ ; Parse $K$ as $K_e \Vert K_m$ |
| $C \leftarrow y_a \Vert C' \Vert \tau$ | If $\mathsf{VT}_{K_m}(C', \tau) = 0$ then return $\perp$ |
| return $C$ | else $M \leftarrow \mathsf{D}_{K_e}(C')$ ; return $(y_a, M)$ |

Here the common key generation algorithm $\overline{\mathcal{K}_c}$ is the same as the group generation algorithm $\mathcal{GG}$, effectively returning $(q, g)$ as the global information $I$. The sender and receiver key generation algorithms return the Diffie-Hellman based keys, which in turn are the basis for computing the common key $K$.

Parsing $K$ as $K_e \| K_m$ means that it is divided into two strings of appropriate lengths (i.e. $L_e$ and $L_m$, resp.) and assigned to $K_e$ and $K_m$, respectively. ▮

Security Analysis. We now show the formal security results for DHETM as summerized in Figure 3. Security of DHETM depends on the security of base primitives, namely a hash function $H$ operating on a group $G$ generated by a group generator algorithm $\mathcal{GG}$, a symmetric encryption scheme SE and a MAC scheme MA. We will first define the security notions of the base primitives starting from the following security assumption on $H$ operating on a group.

Diffie-Hellman Assumptions. Since the computed common key $K$ is generated from a hash of the Diffie-Hellman based key $(H(g^{x_a x_b}))$ in the Diffie-Hellan based scheme DHETM, we make an assumption called "Oracle Diffie-Hellman" (ODH) following [1] for their IND-CCA security analyses. For the TUF-CTXT security of DHETM, a weaker assumption called "Hash Diffie-Hellman" (HDH) is needed. The assumptions are composite ones in the sense that they are about the interaction between the hash function and the Diffie-Hellman problem. We recall their formal definitions below.

**Definition 5.5** [1]**[ODH and HDH]** Let $\mathcal{GG}$ be a group generator algorithm, and let $k \in \mathsf{N}$ be a security parameter. Let $H\colon \{0,1\}^* \to \{0,1\}^{L_h}$ be a hash function, where $L_h \in \mathsf{N}$. Let $A_{\mathrm{odh}}$ and $A_{\mathrm{hdh}}$ be adversaries. Consider the following experiments.

| Experiment $\mathbf{Exp}^{\text{odh-b}}_{H,\mathcal{GG},A_{\mathrm{odh}}}(k)$ | Experiment $\mathbf{Exp}^{\text{hdh-b}}_{H,\mathcal{GG},A_{\mathrm{hdh}}}(k)$ |
|---|---|
| $(q,g) \stackrel{R}{\leftarrow} \mathcal{GG}(k)$ | $(q,g) \stackrel{R}{\leftarrow} \mathcal{GG}(k)$ |
| $u,v \stackrel{R}{\leftarrow} Z_q$ ; $U \leftarrow g^u$ ; $V \leftarrow g^v$ | $u,v \stackrel{R}{\leftarrow} Z_q$ ; $U \leftarrow g^u$ ; $V \leftarrow g^v$ |
| If $b=1$ then $W \leftarrow H(g^{uv})$ else $W \stackrel{R}{\leftarrow} \{0,1\}^{L_h}$ | If $b=1$ then $W \leftarrow H(g^{uv})$ else $W \stackrel{R}{\leftarrow} \{0,1\}^{L_h}$ |
| $d \leftarrow A^{\mathcal{H}_v(\cdot)}_{\mathrm{odh}}((q,g),U,V,W)$ ; Return $d$ | $d \leftarrow A_{\mathrm{hdh}}((q,g),U,V,W)$ ; Return $d$ |

Above, $\mathcal{H}_v(X) \stackrel{\mathrm{def}}{=} H(X^v)$, and $A_{\mathrm{odh}}$ is *not* allowed to query its oracle on $g^u$. We define the advantages of the adversary via,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{odh}}_{H,\mathcal{GG},A_{\mathrm{odh}}}(k) &= \Pr\left[\mathbf{Exp}^{\text{odh-1}}_{H,\mathcal{GG},A_{\mathrm{odh}}}(k)=1\right] - \Pr\left[\mathbf{Exp}^{\text{odh-0}}_{H,\mathcal{GG},A_{\mathrm{odh}}}(k)=1\right] \\
\mathbf{Adv}^{\mathrm{hdh}}_{H,\mathcal{GG},A_{\mathrm{hdh}}}(k) &= \Pr\left[\mathbf{Exp}^{\text{hdh-1}}_{H,\mathcal{GG},A_{\mathrm{hdh}}}(k)=1\right] - \Pr\left[\mathbf{Exp}^{\text{hdh-0}}_{H,\mathcal{GG},A_{\mathrm{hdh}}}(k)=1\right]
\end{aligned}$$

We define the advantage functions of $(H,\mathcal{GG})$ as follows. For any integers $t,q \geq 0$,

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{odh}}_{H,\mathcal{GG}}(k,t,q) &= \max_{A_{\mathrm{odh}}}\{\mathbf{Adv}^{\mathrm{odh}}_{H,\mathcal{GG},A_{\mathrm{odh}}}(k)\} \\
\mathbf{Adv}^{\mathrm{hdh}}_{H,\mathcal{GG}}(k,t) &= \max_{A_{\mathrm{hdh}}}\{\mathbf{Adv}^{\mathrm{hdh}}_{H,\mathcal{GG},A_{\mathrm{hdh}}}(k)\}\ ,
\end{aligned}$$

where the maximum is over all adversaries $A_{\mathrm{odh}}, A_{\mathrm{hdh}}$ with time-complexity $t$, and in the case of $A_{\mathrm{odh}}$, also making at most $q$ queries to $\mathcal{H}_v(\cdot)$. The hash function $H$ is said to be *hardcore on* $\mathcal{GG}$ *under adaptive DH attack* (resp. *hardcore on* $\mathcal{GG}$) if the function $\mathbf{Adv}^{\mathrm{odh}}_{H,\mathcal{GG},A}(\cdot)$ (resp. $\mathbf{Adv}^{\mathrm{hdh}}_{H,\mathcal{GG},A}(\cdot)$) is negligible for any adversary $A$ whose time-complexity is polynomial in the security parameter $k$. ▮

Security notion of symmetric encryption schemes. The privacy of symmetric encryption schemes is measured in a similar manner as per Definition 2.2 except for the following differences: The main differences in the security measure between the public key and symmetric settings are first, the key structures —in the symmetric setting, the sender and the receiver share the same (symmetric) secret key (for both encryption and decryption algorithms), whereas in the public key setting, they have their own public and secret key pair and the keys for the encryption and decryption algorithms are different (asymmetric)— and second, the information that is publicly available —in the symmetric setting, no information about the key is given to the public (including the adversary) unlike the public key setting. We will use essentially the same security notions for the privacy of symmetric encryption schemes with slight modifications reflecting the differences. In particular, we use the left-or-right indistinguishability notions shown in [3] and omit the details of their descriptions here.

Security notion of MAC schemes. Here we recall the notion of *strong unforgeability under chosen-message attack (SUF-CMA)* for a MAC scheme following [5]. Note that the same security notion has been defined for a digital signature scheme in Definition 2.4. We overload the experiment $\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k)$ used in Definition 2.4 for the definition for the security of a MAC scheme here. The definition is similar to that of a digital signature scheme, but the difference is that the adversary is allowed access to a verification oracle as well as a tag oracle in the attack model for a MAC scheme, whereas in the attack model for a signature schem, the adversary is given access to a signing oracle only and instead of a verification oracle, it is given a public key for the digital signature scheme. Note that by giving the public key for the signature scheme, the adversary can perform verification by itself, hence there is no need for allowing oracle access to the verification algorithm in the attack model for the signature scheme. Similarly to the attack model for a digital signature scheme, the adversary against the security of a MAC scheme is allowed access to a tagging oracle modeling a chosen message attack. Since the adversary against the security of a MAC scheme is allowed access to a verification oracle, we let the adversary win if it makes the verification oracle accept by querying it on a valid message and tag pair $(M, \mathsf{tag})$ that is "new" as a pair. What is meant by "new" here is that the forgery tag $\mathsf{tag}$ was never output by the tagging oracle in response to query $M$.

**Definition 5.6 [Strong unforgeability under chosen message attack (SUF-CMA) of a MAC]** Let $\mathsf{MA} = (\mathcal{K}, \mathsf{T}, \mathsf{VT})$ be a a MAC scheme. Let $k \in \mathsf{N}$. Let $F$ be an adversary. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MA},F}(k)$
  $K \stackrel{R}{\leftarrow} \mathcal{K}(k)$
  If $F^{\mathsf{T}_K(\cdot),\mathsf{VT}_K(\cdot,\cdot)}(k)$ makes a query $(M, \tau)$ to the oracle $\mathsf{VT}_K(\cdot,\cdot)$ such that
    – $\mathsf{VT}_K(M, \tau)$ returns 1, and
    – $\tau$ was never returned by the oracle $\mathsf{T}_K(\cdot)$ in response to query $M$
      then return 1 else return 0

We define the *advantage* of the adversary via,

$$\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},F}(k) = \Pr\left[\, \mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MA},F}(k) = 1 \,\right]$$

We define the *advantage function* of the scheme as follows. For any integers $t, q_t, q_v, \mu \geq 0$,

$$\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA}}(k, t, q_t, q_v, \mu) = \max_F\{\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},F}(k)\}$$

where the maximum is over all $F$ with time-complexity $t$, making at most $q_t$ oracle queries to $\mathsf{T}_K(\cdot)$ and at most $q_v$ oracle queries to $\mathsf{VT}_K(\cdot,\cdot)$ such that the sum of the lengths of all oracle queries is at most $\mu$ bits. The MAC scheme $\mathsf{MA}$ is said to be *SUF-CMA secure* if the function $\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},F}(\cdot)$ is negligible for any adversary $F$ whose time-complexity is polynomial in $k$. ▌

Security of $\mathsf{DHETM}$. As usual, the security of $\mathsf{DHETM}$ is based on its base primitives. In particular, the IND-CCA security of $\mathsf{DHETM}$ is based on IND-CPA, SUF-CMA, ODH assumptions on $\mathsf{SE}$, $\mathsf{MA}$, and $(H, \mathcal{GG})$, respectively. Its TUF-CTXT security, however, is based only on SUF-CMA and HDH assumptions on $\mathsf{MA}$ and $(H, \mathcal{GG})$, respectively. As can be seen in Definition 5.5, the attack model against the HDH security is very similar to that against the ODH security except that the adversary does *not* get access to the hash oracle $\mathcal{H}_v(\cdot)$ in the HDH case; hence, HDH is a weaker assumption than the ODH assumption. As with the analyses of *ESSR*, other security results (i.e. IND-CPA, TUF-PTXT) of $\mathsf{DHETM}$ can be inferred from the relations among notions shown Figure 1, and their explicit statements are omitted. Note, however, that $\mathsf{DHETM}$ is *not* secure under the receiver unforgeability notions (RUF-PTXT, RUF-CTXT) because the key used for generating valid ciphertexts can be computed by the receiver as well as the sender. The following theorems state the security results. Concrete security assessments are made in the following theorems by taking the maximum advantage over all adversaries limited to the given resources.

**Theorem 5.7** [$\mathsf{DHETM}$ **is IND-CCA secure**] Let $H$ be a hash function, and let $\mathcal{GG}$ be a group generator algorithm. Let $\mathsf{MA}$ be a MAC scheme, and let $\mathsf{SE}$ be a symmetric encryption scheme. Let $\mathsf{DHETM}$ be a PKAE scheme constructed based on $\mathsf{SE}$, $\mathsf{MA}$, $H$, and $\mathcal{GG}$ as per Construction 5.4. Then, if $H$ is hardcore on $\mathcal{GG}$ under adaptive DH attack (i.e. the ODH assumption), $\mathsf{SE}$ is IND-CPA secure, and $\mathsf{MA}$ is SUF-CMA secure then $\mathsf{DHETM}$ is IND-CCA secure. Concretely,

$$\begin{aligned} &\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{DHETM}}(k,t,q_e,\mu_e,q_d,\mu_d) \\ &\le\ 2\cdot\mathbf{Adv}^{\text{odh}}_{H,\mathcal{GG}}(k,t,q_d)+\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE}}(k,t,q_e,\mu_e)+2\cdot\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA}}(k,t,q_e,q_d,\mu_e+\mu_d) \quad \blacksquare \end{aligned}$$

**Theorem 5.8** [$\mathsf{DHETM}$ **is TUF-CTXT secure**] Let $H$ be a hash function, and let $\mathcal{GG}$ be a group generator algorithm. Let $\mathsf{MA}$ be a MAC scheme, and let $\mathsf{SE}$ be a symmetric encryption scheme. Let $\mathsf{DHETM}$ be a PKAE scheme constructed based on $\mathsf{SE}$, $\mathsf{MA}$, $H$, and $\mathcal{GG}$ as per Construction 5.4. Then, if $H$ is hardcore on $\mathcal{GG}$ (i.e. the HDH assumption) and $\mathsf{MA}$ is SUF-CMA secure, then $\mathsf{DHETM}$ is TUF-CTXT secure. Concretely,

$$\mathbf{Adv}^{\text{tuf-ctxt}}_{\mathsf{DHETM}}(k,t,q,\mu)\le\mathbf{Adv}^{\text{odh}}_{H,\mathcal{GG}}(k,t)+\mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA}}(k,t,q,1,\mu) \quad \blacksquare$$

**Theorem 5.9** [$\mathsf{DHETM}$ **is neither RUF-PTXT secure nor RUF-CTXT secure**] Let $H$ be a hash function, and let $\mathcal{GG}$ be a group generator algorithm. Let $\mathsf{MA}$ be a MAC scheme, and let $\mathsf{SE}$ be a symmetric encryption scheme. Let $\mathsf{DHETM}$ be a PKAE scheme constructed based on $\mathsf{SE}$, $\mathsf{MA}$, $H$, and $\mathcal{GG}$ as per Construction 5.4. Then, $\mathsf{DHETM}$ is neither RUF-PTXT secure nor RUF-CTXT secure. Concretely, there is an adversary $B$ making zero oracle queries, and achieving

$$\mathbf{Adv}^{\text{ruf-ctxt}}_{\mathsf{DHETM},B}(k)=\mathbf{Adv}^{\text{ruf-ctxt}}_{\mathsf{DHETM},B}(k)=1 \quad \blacksquare$$

Comparison to DHIES. Since $\mathsf{DHETM}$ can also be thought of as an adaptation of the IND-CCA secure asymmetric encryption scheme, DHIES of [1], into the PKAE scheme's setting, the IND-CCA security of $\mathsf{DHETM}$ can be shown in a similar manner as that of DHIES. However, it turns out that there is no direct reduction from the IND-CCA security of $\mathsf{DHETM}$ to the IND-CCA security of DHIES, because of the difference in the way encryption is done. In particular, in $\mathsf{DHETM}$, the computed common key remains the same throughout multiple invocations of the encryption algorithm (oracle) because the key is computed based on the fixed keys of the sender and receiver, while in DHIES, the common key may change each time the encryption algorithm is invoked because the key is computed depending not only on the fixed public key of the receiver, but also on a "fresh", random choice of an element (in $Z_q$) by the encryption algorithm. This is mainly because of the fact that DHIES is not an authenticated encryption scheme and the encryption key does not depend on a fixed, secret key of a sender. (For sending an encrypted message using DHIES, a sender's key is not needed.) This is reflected in the attack model of the adversaries against IND-CCA security: the inputs given to the adversaries are different in the following sense: the adversary against the IND-CCA security of $\mathsf{DHETM}$ gets the public keys of both sender and receiver, while that of DHIES gets just the receiver's public key because DHIES is an encryption scheme, but not an authenticated encryption scheme.

EFFICIENCY OF DHETM. Compared to the schemes constructed from the generic composition method based on public key encryption schemes and signature schemes, the main cost savings come from the fact that the Diffie-Hellman based common secret (symmetric encryption and MAC keys) can be computed off-line using the public key of the other party and can also be stored once computed, saving the cost of computing the common secret each time. Also the use of symmetric key based schemes (i.e. a symmetric encryption scheme and a MAC) as its base primitives is another contributing factor for efficiency.

## Acknowledgements

I am grateful to Mihir Bellare for providing invaluable support and guidance with this work. Many of the ideas found here are due to him. I would also like to thank Michel Abdalla and Adriana Palacio for helpful discussions and comments.

## References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer-Verlag, Berlin Germany, Apr. 2001.

[2] R. Anderson and R. Needham. Robustness principles for public key protocols. In D. Coppersmith, editor, *Crypto'95*, volume 963 of *LNCS*, pages 236–247, Berlin, Germany, Aug. 1995. Springer-Verlag, Berlin Germany.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Crypto'98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, Berlin Germany, Aug. 1998.

[5] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Asiacrypt 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, Berlin Germany, Dec. 2000.

[6] M. Chen and E. Hughes. Protocol failures related to order of encryption and signature: Computation of discrete logarithms in rsa groups. In C. Boyd and E. Dawson, editors, *ACISP'98*, volume 1438 of *LNCS*, pages 238–249, Berlin, Germany, 1998. Springer-Verlag.

[7] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd ACM STOC*, pages 542–552, New Orleans, Louisiana, May 6–8 1991. ACM Press.

[8] C. Gamage, J. Leiwo, and Y. Zheng. Encrypted Message Authentication by Firewalls. In H. Imai and Y. Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 69–81. Springer-Verlag, Berlin Germany, Mar. 1999.

[9] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[10] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.

[11] W.-H. He and T.-C. Wu. Cryptanalysis and improvement of Petersen-Michels signcryption schems. *IEE Proc. – Computers and Digital Techniques*, 146(2):123–124, Mar. 1999.

[12] P. Horster, M. Michels, and H. Petersen. Authenticated encryption schemes with low communication costs. Technical Report TR-94-2-R, University of Technology, Chemnitz-Zwickau, May 1994. appeared in Electronics Letters, Vol. 30, No. 15, 1994, pp. 1231-1231.

[13] J. Katz and M. Yung. Unforgeable encryption and adaptively secure modes of operation. In B. Schneier, editor, *FSE 2000*, volume ?? of *LNCS*. Springer-Verlag, Berlin Germany, 2000.

[14] H. Petersen and M. Michels. Cryptanalysis and improvement of signcryption schemes. *IEE Proc. – Computers and Digital Techniques*, 145(2):149–151, Mar. 1998.

[15] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Crypto'91*, volume 576 of *LNCS*. Springer-Verlag, Berlin Germany, Aug. 1991.

[16] C. Schnorr and M. Jakobsson. Security of signed elgamal encryption. In T. Okamoto, editor, *Asiacrypt 2000*, volume 1976 of *LNCS*, pages 73–89. Springer-Verlag, Berlin Germany, Dec. 2000.

[17] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In B. Kaliski, editor, *Crypto'97*, volume 1294 of *LNCS*, pages 165–179. Springer-Verlag, Berlin Germany, Aug. 1997.

# A Proofs

**Proof of Proposition 3.1:**

Intuitively, the horizontal implications among the unforgeability notions hold because the adversary gets more power as it goes from the third-person unforgeability to the receiver unforgeability attack models. We show the proof for the unforgeability of ciphertext case (RUF-CTXT $\rightarrow$ TUF-CTXT) only here. The unforgeability of plaintext case (RUF-PTXT $\rightarrow$ TUF-PTXT) can be shown in a similar manner.

Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ be a PKAE scheme. Let $F$ be any poly($k$)-time adversary attacking TUF-CTXT of $\mathcal{PKAE}$. Using the adversary $F$, we can construct an adversary $B = (B_1, B_2)$ attacking RUF-CTXT of $\mathcal{PKAE}$, having time-complexity also polynomial in the security parameter. Since $B$ has more flexibility in the attack model than $F$, $B$ can run $F$ answering $F$'s queries using its own encryption oracle and outputting the same ciphertext as $F$ without changing the key pair it originally chose in its first stage. The algorithm for $B$ is shown below.

Algorithm $B_1(I)$
  $(pk_r, sk_r) \stackrel{R}{\leftarrow} \mathcal{K}_r(I)$
  $st \leftarrow (I, sk_r)$
  Return $(st, pk_r)$

Algorithm $B_2^{\mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)}(st, pk_s, pk_r)$
  Parse $st$ as $(I, sk_r)$
  Run $F(I, pk_s, pk_r)$ answering $F$'s queries as follows:
    When $F$ makes a query $x$ to its encryption oracle do
      $c \leftarrow \mathcal{E}_{\langle sk_s, pk_s, pk_r\rangle}(x)$ ; return $c$ to $F$
  Until $F$ outputs a ciphertext forgery $C$
  Return $(C, pk_r, sk_r)$

Since the encryption oracle $B$ accesses is essentially the same as the oracle $F$ accesses in the definition of the attack models, the adversary $B$ uses its own encryption oracle to answer $F$'s queries. It is easy to see that $B$ succeeds as long as $F$ succeeds in forging the ciphertext because $B$ did not change its public, secret key pair, and hence the definitions of a successful forgery in both attack models become effectively the same. Hence, the following equation holds:

$$\mathbf{Adv}^{\text{ruf-ctxt}}_{\mathcal{PKAE},B}(k) \geq \mathbf{Adv}^{\text{tuf-ctxt}}_{\mathcal{PKAE},F}(k) \ .$$

The assumption that $\mathcal{PKAE}$ is RUF-CTXT secure implies that $\mathbf{Adv}^{\text{ruf-ctxt}}_{\mathcal{PKAE},B}(\cdot)$ is negligible, and hence it follows that $\mathbf{Adv}^{\text{tuf-ctxt}}_{\mathcal{PKAE},F}(\cdot)$ is also negligible, which results in the conclusion of Proposition 3.1. ▮

**Proof of Proposition 3.2:**

Instead of explicitly presenting the construction of the PKAE scheme here, we point to a construction presented in another section (i.e. Section 5.2) which also provides the security results regarding the constructed scheme that support the statements in Proposition 3.2. It turns out that the PKAE scheme $\mathsf{DHETM}$ constructed as per Construction 5.4 is such a scheme. Based on the HDH assumption on the base hash function and group $(H, \mathcal{GG})$ and SUF-CMA security assumption on the base MAC scheme, the proof of Theorem 5.8 shows that $\mathsf{DHETM}$ is TUF-CTXT secure (which in turn implies TUF-PTXT security [5]), and the proof of Theorem 5.9 shows that $\mathsf{DHETM}$ is neither RUF-CTXT secure nor RUF-PTXT secure. Hence, from the two mentioned theorems, we obtain the conclusion of the Proposition 3.2. ▮

**Proof of Proposition 3.3:**

For the proof of separation, we show a PKAE scheme that is RUF-CTXT secure, and yet *not* RUF-PTXT secure. It turns out that one of the generic composition methods shown in Section 4 gives the desired result in general. Given a public key encryption scheme and a digital signature scheme, generic composition methods are methods that combine the encryption and signature schemes, treating the underlying schemes as black-boxes, in order to obtain a PKAE scheme. Generic composition methods are described in more detail in Section 4. Among the methods, the "Encrypt-then-Sign" method composed as per Construction 4.3 gives a PKAE scheme that is RUF-CTXT secure, but not RUF-PTXT secure under the assumptions that the underlying encryption scheme is IND-CPA secure and the signature scheme is SUF-CMA secure. The proofs that the method is RUF-CTXT secure and not RUF-PTXT secure under the assumptions that the underlying encryption scheme is IND-CPA secure and the signature is SUF-CMA secure are shown in the proofs of Theorem 4.8 and Theorem 4.6, respectively. ▮

**Proof of Proposition 3.4:**

The idea is similar to that of the proof of the third-person unforgeability case (i.e. TUF-PTXT $\not\rightarrow$ TUF-CTXT). The ability of an adversary to create and manipulate the receiver's key pair does not affect the proof in this case. Given any RUF-PTXT secure scheme $\mathcal{PKAE}$, we can transform it to a scheme $\mathcal{PKAE}'$ that is still RUF-PTXT secure, but not even TUF-CTXT secure. Basic modification is in the encryption algorithm which adds a redundant bit to a ciphertext and the decryption algorithm that ignores the redundant bit. Because the bit is ignored in the decryption algorithm, an adversary against TUF-CTXT can flip the redundant bit in the ciphertext obtained from the encryption oracle and output it as a "new" forgery ciphertext, which in turn would be considered successful with respect to the TUF-CTXT sense. It is easy to see that $\mathcal{PKAE}'$ is RUF-PTXT secure if $\mathcal{PKAE}$ is RUF-PTXT secure, because the modification of adding a redundancy bit to the ciphertext, which in turn is ignored by the decryption algorithm does not affect its RUF-PTXT security. Intuitively, it is RUF-PTXT secure but not RUF-CTXT secure because of the distinction in the definition of the "newness" of ciphertext (i.e. for RUF-PTXT, the *plaintext* corresponding to the forgery ciphertext has to be "new" while for RUF-CTXT, the forgery *ciphertext* itself has to be "new"). The details are shown below.

Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}, \mathcal{D})$ be the given PKAE scheme. We define a new scheme $\mathcal{PKAE}' = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \mathcal{E}', \mathcal{D}')$ based on the original scheme $\mathcal{PKAE}$, where the key generation algorithms are the same as the original ones and the encryption and decryption algorithms are modified as follows:

| | |
|---|---|
| Algorithm $\mathcal{E}'_{\langle sk_s, pk_s, pk_r \rangle}(M)$ | Algorithm $\mathcal{D}'_{sk_r}(C)$ |
| $C \leftarrow \mathcal{E}_{\langle sk_s, pk_s, pk_r \rangle}(M)$ | Parse $C$ as $b \| C'$ where $b$ is a bit |
| Return $0 \| C$ | $X \leftarrow \mathcal{D}_{sk_r}(C')$ ; Return $X$ |

The new encryption algorithm prepends a redundant bit '0' to the ciphertext output by the original encryption algorithm and the new decryption algorithm ignores the first bit and outputs whatever the original decryption algorithm outputs (i.e. $X$ can be either $\perp$ or $(pk_s, M)$, where $M \in \mathcal{M}$) on the rest of the ciphertext input $C'$ (which excludes the first bit $b$).

We first show that $\mathcal{PKAE}'$ is not TUF-CTXT secure and then show that it is yet RUF-PTXT secure as long as the original $\mathcal{PKAE}$ scheme is RUF-PTXT secure. Both proofs are pretty straightforward.

In order to show that $\mathcal{PKAE}'$ is not TUF-CTXT secure, we show an attack against $\mathcal{PKAE}'$ in the TUF-CTXT sense. The attack exploits the fact that the first ciphertext bit is ignored in the decryption algorithm. The adversary queries the encryption oracle with a message ("0" in this case) and replaces the first bit 0 by 1 in the ciphertext obtained as a response and outputs the new ciphertext. The adversary $F$ against TUF-CTXT of $\mathcal{PKAE}'$ is shown in more detail below.

Algorithm $F^{\mathcal{E}'_{\langle sk_s, pk_s, pk_r \rangle}(\cdot)}(I, pk_s, pk_r)$
    $C \leftarrow \mathcal{E}'_{\langle sk_s, pk_s, pk_r \rangle}(0)$ ; Parse $C$ as $0 \| C'$
    Return $(1 \| C')$

Note that $F$'s output ciphertext $1 \| C'$ is "new", meaning that it was never output by the encryption oracle, whose ciphertext output always begins with the bit 0. It is easy to see that the ciphertext $1 \| C'$ is "valid" because the un-ignored part $C'$ is valid. Hence, $F$ succeeds in attacking TUF-CTXT security of the scheme $\mathcal{PKAE}'$ with the probability 1.

We now move to the proof that $\mathcal{PKAE}'$ is RUF-PTXT secure if $\mathcal{PKAE}$ is RUF-PTXT secure. Using the standard reduction argument, we show that if given any adversary $A = (A_1, A_2)$ attacking $\mathcal{PKAE}'$ in the RUF-PTXT sense, we can construct an adversary $B = (B_1, B_2)$ attacking $\mathcal{PKAE}$ in the RUF-PTXT sense using the adversary $A$. The adversary $B$ runs $A$ in a straightforward manner answering $A$'s oracle queries using its own oracle and uses $A$'s output forgery as part of its own output forgery. The details of the algorithms are shown below.

| | |
|---|---|
| Algorithm $B_1(I)$ | Algorithm $B_2^{\mathcal{E}_{\langle sk_s, pk_s, pk_r \rangle}(\cdot)}(st, pk_s, pk_r)$ |
| $(st, pk_r) \stackrel{R}{\leftarrow} A_1(I)$ | Run $A_2(st, pk_s, pk_r)$ answering its oracle queries as follows: |
| Return $(st, pk_r)$ | When $A_2$ makes a query $m$ to its encryption oracle do |
| | $c \leftarrow 0 \| \mathcal{E}_{\langle sk_s, pk_s, pk_r \rangle}(m)$ ; return $c$ to $A_2$ |
| | Until $A_2$ outputs $(C, pk_r{}', sk_r{}')$ |
| | Parse $C$ as $b \| C'$ where $b$ is a bit |
| | Return $(C', pk_r{}', sk_r{}')$ |

It is easy to see that $B$ succeeds in attacking RUF-PTXT security of $\mathcal{PKAE}$ if $A$ succeeds in attacking RUF-PTXT security of $\mathcal{PKAE}'$ since the definitions for a successful forgery are the same in both cases.

Hence, this implies that $\mathcal{PKAE}'$ is RUF-PTXT secure if $\mathcal{PKAE}$ is RUF-PTXT secure, which in turn is implied by the assumption. ▮

**Proof of Proposition 3.5:**

The separation relation that the combined notions of IND-CPA and RUF-CTXT do not imply IND-CCA also means that the combined notions of IND-CPA and TUF-CTXT do not imply IND-CCA either, since TUF-CTXT is a weaker notion than RUF-CTXT, as shown in Proposition 3.1. Hence, here we just prove the separation for the stronger notion, RUF-CTXT. For the proof of separation, we present a PKAE scheme that is IND-CPA and RUF-CTXT secure, and yet *not* IND-CCA secure. Assuming that the base encryption scheme is IND-CPA secure and the base signature scheme is SUF-CMA secure, a PKAE scheme composed via the Encrypt-then-Sign method as per Construction 4.3 is IND-CPA and RUF-CTXT secure, but is not IND-CCA secure. The proof that the composed PKAE scheme is IND-CPA secure is straightforward and is omitted. The proof that the scheme is RUF-CTXT secure is given in the proof of Theorem 4.8, and the proof that the scheme is *not* IND-CCA secure is given in the proof of Theorem 4.9. This completes the proof. ▮

**Proof of Proposition 4.4:**

The RUF-PTXT security of the PKAE scheme that is constructed via the Encrypt-and-Sign or Sign-then-Encrypt method depends only on the security (WUF-CMA) of the underlying signature scheme. In both cases, the security is shown by the standard reduction argument. Given an adversary $B = (B_1, B_2)$ attacking RUF-PTXT of the PKAE scheme, we can easily construct an adversary $F$ attacking WUF-CMA of the underlying signature scheme. We first prove the security of the Encrypt-and-Sign method by showing the reduction algorithm below and then move on to the security of the Sign-then-Encrypt method.

Let $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme and let $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ be a digital signature scheme. Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be a PKAE scheme constructed based on $\mathcal{PE}$ and $\mathcal{DS}$ via the Encrypt-and-Sign method. We first show the adversary $F$ attacking WUF-CMA of the base signature scheme $\mathcal{DS}$ using the adversary $B$ attacking RUF-PTXT of the PKAE scheme $\mathcal{PKAE}$ as follows:

Algorithm $F^{\mathcal{S}_{sk}(\cdot)}(I_s, pk)$
  $I_e \stackrel{R}{\leftarrow} \mathcal{K}_{ce}(k)$ ; $I \leftarrow (I_e, I_s)$
  $(st, pk_r) \leftarrow B_1(I)$
  Run $B_2(st, pk, pk_r)$ answering $B$'s oracle queries as follows:
    When $B_2$ makes a query $x$ to its encryption oracle do
      $c' \leftarrow \mathcal{E}_{pk_r}(x)$ ; $\sigma \leftarrow \mathcal{S}_{sk}(x)$ ; $c \leftarrow pk \| c' \| \sigma$ ; return $c$ to $B_2$
  Until $B_2$ outputs $(C, pk_r{}', sk_r{}')$
  Parse $C$ as $pk \| C' \| \sigma'$ ; $M' \leftarrow \mathcal{D}_{sk_r{}'}(C')$
  Return $(M', \sigma')$

As can be seen, $F$ simulates the encryption oracle for the PKAE scheme constructed via the Encrypt-and-Sign method using its sign-oracle and the encryption key obtained from $B_1$ in the above algorithm. Because the definitions of "newness" and "validity" of $B$'s forgery ciphertext match those of $F$'s signature forgery output, $F$ succeeds if $B$ succeeds. Hence, the following equation holes: $\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(k) \geq \mathbf{Adv}^{\text{ruf-ptxt}}_{\mathcal{PKAE},B}(k)$. The assumption that $\mathcal{DS}$ is secure in the WUF-CMA sense implies that $\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(\cdot)$ is negligible, and hence it follows that $\mathbf{Adv}^{\text{ruf-ptxt}}_{\mathcal{PKAE},B}(\cdot)$ is negligible, which results in the conclusion of the proposition. Note that the forgery of the adversary $F$ shown above is not only an attack in the WUF-CMA sense but also in the SUF-CMA sense since by definition a successful forgery in the WUF-CMA sense is also considered a successful forgery in the SUF-CMA sense.

The security of the Sign-then-Encrypt method can be proven in a similar manner. For the proof of the Sign-then-Encrypt case, the difference in the reduction algorithm will be in the way $F$ generates the encryption oracle for $B_2$ (i.e. instead of using the Encrypt-and-Sign method, $F$ will generate the ciphertext based on the Sign-then-Encrypt method) and how $F$ gets its forgery signature from $B$'s forgery ciphertext. For completeness, we show the algorithm below:

Algorithm $F^{\mathcal{S}_{sk}(\cdot)}(I_s, pk)$
    $I_e \stackrel{R}{\leftarrow} \mathcal{K}_{ce}(k)$ ; $I \leftarrow (I_e, I_s)$
    $(st, pk_r) \leftarrow B_1(I)$
    Run $B_2(st, pk, pk_r)$ answering $B$'s oracle queries as follows:
        When $B_2$ makes a query $x$ to its encryption oracle do
            $\sigma \leftarrow \mathcal{S}_{sk}(x)$ ; $c' \leftarrow \mathcal{E}_{pk_r}(x \| \sigma)$ ; $c \leftarrow pk \| c'$ ; return $c$ to $B_2$
    Until $B_2$ outputs $(C, {pk_r}', {sk_r}')$
    Parse $C$ as $pk \| C'$ ; $X' \leftarrow \mathcal{D}_{{sk_r}'}(C')$ ; Parse $X'$ as $M' \| \sigma'$
    Return $(M', \sigma')$

It is easy to see that by a similar reason as the proof for the Encrypt-and-Sign case, $\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(k) \geq \mathbf{Adv}^{\text{ruf-ptxt}}_{\mathcal{PKAE},B}(k)$. ▮

**Proof of Proposition 4.5:**

Let $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be the given IND-CPA secure public key based encryption scheme. Based on $\mathcal{PE}$, we define the scheme $\mathcal{PE}' = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}', \mathcal{D}')$ such that $\mathcal{PE}'$ is IND-CPA secure, but its associated PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ constructed via the Encrypt-and-Sign or Sign-then-Encrypt method is not TUF-CTXT secure. By showing it is not TUF-CTXT secure, we can infer also that it is not RUF-CTXT secure because RUF-CTXT security implies TUF-CTXT security (which is shown in Proposition 3.1).

Basically, the modification in $\mathcal{PE}$ comes in the encryption and decryption algorithms: the encryption algorithm $\mathcal{E}'_{pk}(\cdot)$ adds a redundant bit to the ciphertext output by the original encryption algorithm $\mathcal{E}_{pk}(\cdot)$ and the decryption algorithm $\mathcal{D}'_{sk}(\cdot)$ ignores the redundant bit and decrypts the rest of the ciphertext using the original decryption algorithm $\mathcal{D}_{sk}(\cdot)$. The new scheme $\mathcal{PE}' = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}', \mathcal{D}')$ has the same key generation algorithms as the original scheme $\mathcal{PE}$, and the encryption and decryption algorithms are shown below:

| Algorithm $\mathcal{E}'_{pk}(M)$ | Algorithm $\mathcal{D}'_{sk}(C)$ |
| --- | --- |
| $C \leftarrow \mathcal{E}_{pk}(M)$ | Parse $C$ as $b \| C'$ where $b$ is a bit |
| Return $0 \| C$ | $M \leftarrow \mathcal{D}_{sk}(C')$ ; Return $M$ |

We show that the scheme $\mathcal{PKAE}$ constructed based on the above encryption scheme $\mathcal{PE}'$ and a signature scheme $\mathcal{DS}$ via the Encrypt-and-Sign or Sign-then-Encrypt method is not TUF-CTXT secure by presenting an adversary $F$ attacking $\mathcal{PKAE}$ in the TUF-CTXT sense.

Algorithm $F^{\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(\cdot)}(I, pk_s, pk_r)$
    $C \leftarrow \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(0)$ ; Parse $C$ as $pk_s \| 0 \| C'$
    Return $(pk_s \| 1 \| C')$

In both Encrypt-and-Sign and Sign-then-Encrypt methods, the same adversary shown above will succeed in attacking the TUF-CTXT security where the ciphertext $1\|C'$ in the above algorithm is interpreted differently in each case. For the Encrypt-and-Sign method, $1\|C'$ will be further divided into two parts: the encryption output part $(1\|C'')$, where $C''$ is the output of the original encryption algorithm $\mathcal{E}_{pk_r}(\cdot)$ of $\mathcal{PE}$ and the signature part $(\sigma)$. For the Sign-then-Encrypt method, $1\|C'$ will be the encryption output, where $C'$ is the output of the original encryption algorithm. In both cases, it is easy to see that the ciphertext is "valid" because the first bit in $1\|C'$ is ignored by the decryption algorithm $\mathcal{D}'_{sk_r}(\cdot)$ and hence is decrypted to be the same plaintext as $F$'s original encryption query, whose signature remains the same in the forgery ciphertext. Because the bit 0 in front of $C'$ is flipped to 1, it is considered "new". Hence, $F$ succeeds in attacking TUF-CTXT of $\mathcal{PKAE}$ in both methods.

Intuitively, the reason why the Encrypt-and-Sign and Sign-then-Encrypt methods are vulnerable to this kind of attack is that the signature is based on the plaintext, but not on the ciphertext. Hence, the change in the ciphertext goes undetected if its corresponding plaintext remains the same as before.

There is a more direct attack when we consider the RUF-CTXT notion. Consider the following adversary $B = (B_1, B_2)$ attacking RUF-CTXT security of a PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ constructed based on any public key encryption scheme $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and any signature scheme $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ via the Encrypt-and-Sign method:

| | |
|---|---|
| Algorithm $B_1(I)$<br>$(pk_r, sk_r) \xleftarrow{R} \mathcal{K}_r(I)$ ; $st \leftarrow (I, sk_r)$<br>Return $(st, pk_r)$ | Algorithm $B_2^{\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)}(st, pk_s, pk_r)$<br>$C \leftarrow \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(0)$ ; Parse $C$ as $pk_s\|C'\|\sigma$<br>$(pk_r{}', sk_r{}') \xleftarrow{R} \mathcal{K}_r(I)$ ; $C'' \leftarrow \mathcal{E}_{pk_r{}'}(0)$<br>Return $(pk_s\|C''\|\sigma, pk_r{}', sk_r{}')$ |

Since an adversary attacking against the RUF-CTXT security can create its own encryption key and is allowed to change it later, with just one query to the (authenticated) encryption oracle, it can get the signature corresponding the query message (plaintext) and then re-encrypt the same message with a different key it generated, and then output the "new" ciphertext as a forgery ciphertext. (We are assuming here that the probability that the ciphertext resulting from re-encrypting the same plaintext under a new public key differs from the original ciphertext is non-negligible when its key is randomly generated by the receiver key generation algorithm. If this is not the case, $B_2$ can repeat the process of choosing its key pair until this is true). Basically, in both Encrypt-and-Sign and Sign-then-Encrypt methods, the ability to generate and change the encryption key will enable the adversary to be able to change the ciphertext part by re-encrypting it with a defferent key. Since the signature is based on the plaintext only in both methods, the signature obtained from the oracle (by the chosen message attack) can be reused with the new ciphertext because the ciphertext is just an encryption of the same plaintext (under a different key). Above, the adversary $B_2$ does not need to decrypt the ciphertext using its secret key because it already knows the plaintext and the signature part is not encrypted. Note, however, that for the Sign-then-Encrypt method, the adversary needs to decrypt the ciphertext in order to get the signature part because the signature part is encrypted. This is a stronger result than the TUF-CTXT case, because the attack works on *any* public key encryption scheme $\mathcal{PE}$, not just on a particular scheme $\mathcal{PE}'$, and this is where the ability to create and manipulate the receiver's key makes a difference.

Note also that we did not use the security of signature schemes explicitly in our attack (the attack does not involve the security of signature schemes). This means that regardless of the security (strength) of the given signature scheme is (whether it be SUF-CMA or WUF-CMA), the scheme $\mathcal{PKAE}$ composed by the Encrypt-and-Sign method is not RUF-CTXT secure.

It is easy to see that the new scheme $\mathcal{PE}'$ is also IND-CPA secure if the original scheme $\mathcal{PE}$ is IND-CPA secure. Intuitively, prepending a fixed bit to a ciphertext does not reveal any additional information about its plaintext, hence does not help an adversary to distinguish a ciphertext output by the new left-or-right encryption oracle any more than that output by the original left-or-right encryption oracle. Hence, an adversary against IND-CPA of $\mathcal{PE}$ can do as well as that against IND-CPA of $\mathcal{PE}'$ in distinguishing the left-or-right oracle. ▮

**Proof of Theorem 4.6:**

Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{V}})$ be a public key based authenticated encryption scheme constructed via the Encrypt-then-Sign method from the encryption scheme $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and the signature scheme $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$. We show that $\mathcal{PKAE}$ is RUF-PTXT insecure by presenting a poly($k$)-time adversary $B = (B_1, B_2)$ attacking $\mathcal{PKAE}$ in the RUF-PTXT sense with success probability of 1. The algorithm $B$ in the first stage chooses its key pair and outputs the pair. In its second stage $B_2$, it makes one plaintext query $m$ to the encryption oracle and obtains a ciphertext $C$ as a response. From $C$, it then extracts the first part $c'$ that corresponds to the encryption of plaintext $m$ (i.e. $c' = \mathcal{E}_{pk_r}(m)$) and picks a public, secret key pair $(pk_r{}', sk_r{}')$ such that $\mathcal{D}_{sk_r{}'}(c') \neq m$, and then it returns the ciphertext $C$ obtained from the oracle response and the newly picked public secret key pair $(pk_r{}', sk_r{}')$ as its output forgery and key pair, respectively. The algorithm is described in more detail below.

| Algorithm $B_1(I)$ | Algorithm $B_2^{\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)}(st, pk_s, pk_r)$ |
|---|---|
| $(pk_r, sk_r) \overset{R}{\leftarrow} \mathcal{K}_r(I)$ | Parse $st$ as $(I, sk_r)$ ; $m \leftarrow 0$ |
| $st \leftarrow (I, sk_r)$ | $C \leftarrow \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(m)$ ; Parse $C$ as $pk_s \| c' \| \sigma$ |
| Return $(st, pk_r)$ | Pick a key pair $(pk_r{}', sk_r{}')$ from $\mathcal{K}_r(I)$ |
| | such that $\mathcal{D}_{sk_r{}'}(c') \neq m$ |
| | Return $(C, pk_r{}', sk_r{}')$ |

Recall that $C$ is considered a successful forgery if $\overline{\mathcal{D}}_{sk_r{}'}(C) \neq \perp$ and its corresponding plaintext $M$ was never a query to $\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)$. Since $C$ was originally output by the encryption oracle, the signature verification algorithm $\mathcal{VS}_{pk_s}(\cdot)$ will return 1 on input $(c', \sigma)$, where $C$ is parsed as $pk_s \| c' \| \sigma$. The plaintext $M$ corresponding to $c'$ obtained from applying the underlying decryption algorithm $\mathcal{D}_{sk_r{}'}(\cdot)$ with a new secret key $sk_r{}'$ on input $c'$ is different from $m$ because in the algorithm $B_2$ shown above, the secret key is picked so that the decrypted message obtained from using the new key differ from the original message. Hence, the plaintext $M$ would be considered "new" (i.e. was never a query to the encryption oracle), which makes the forgery valid in the RUF-PTXT sense. Note that since we did not assume anything specific for the underlying primitives, the attack works for any scheme that is constructed from an encryption and signature scheme via the Encrypt-then-Sign method. ▮

**Proof of Theorem 4.7:**

Let $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme, and let $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ be the given WUF-CMA secure digital signature scheme. Based on $\mathcal{DS}$, we define the scheme $\mathcal{DS}' = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}', \mathcal{VS}')$ such that $\mathcal{DS}'$ is WUF-CMA secure, but its associated PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ constructed via the Encrypt-then-Sign method is not TUF-CTXT secure. By showing it is not TUF-CTXT secure, we can infer also that it is not RUF-CTXT secure because RUF-CTXT security implies TUF-CTXT security (which is shown in Proposition 3.1).

The key generation algorithms $\mathcal{K}_{cs}(\cdot)$, $\mathcal{KS}(\cdot)$ in $\mathcal{DS}'$ remain the same as $\mathcal{DS}$ and the signing and verifying algorithm $\mathcal{S}'_{sk}(\cdot)$, $\mathcal{VS}'_{pk}(\cdot)$ are modified as follows:

| Algorithm $\mathcal{S}'_{sk}(M)$ | Algorithm $\mathcal{VS}'_{pk}(M,\sigma)$ |
|---|---|
| $\sigma' \leftarrow \mathcal{S}_{sk}(M)$ | Parse $\sigma$ as $\sigma' \| b$ where $b$ is a bit |
| Return $\sigma' \| 0$ | If $\mathcal{VS}_{pk}(M,\sigma') = 1$ then return 1 else return 0 |

As shown above in the modified signing algorithm, a redundant bit is appended to the output of the original signing algorithm $\mathcal{S}_{sk}(\cdot)$ and is ignored in the modified verification algorithm. This makes the PKAE scheme $\mathcal{PKAE}$ constructed based on $\mathcal{PE}$ and $\mathcal{DS}'$ via the Encrypt-then-Sign method to be TUF-CTXT insecure. We show this by presenting an adversary attacking against TUF-CTXT of $\mathcal{PKAE}$ as follows:

Algorithm $F^{\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(\cdot)}(I, pk_s, pk_r)$
    $C \leftarrow \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r\rangle}(0)$
    Parse $C$ as $pk_s \| C' \| 0$
    $C \leftarrow pk_s \| C' \| 1$
    Return $C$

It is easy to see that the forgery output $C$ is a "new" ciphertext (i.e. it was never a response from the encryption oracle) since the last bit of the ciphertext is changed with respect to the encryption oracle output. The ciphertext part $C' \| 1$ can be further divided into $C'' \| \sigma' \| 1$ where $C''$ corresponds to the base encryption output part and and $\sigma' \| 1$ is the signature part. Since the last bit in the signature part is ignored by the signature verification algorithm, the pair $(C'', \sigma' \| 1)$ will be considered valid by the signature verification algorithm $\mathcal{VS}'_{pk_s}(\cdot,\cdot)$. Hence, the adversary succeeds in attacking RUF-CTXT of $\mathcal{PKAE}$.

It remains to show that $\mathcal{DS}'$ is WUF-CMA secure if $\mathcal{DS}$ is WUF-CMA secure. To do so, we can build an adversary $A$ against WUF-CMA of $\mathcal{DS}$ using an assumed adversary $A'$ against WUF-CMA of $\mathcal{DS}'$ as follows:

Algorithm $A^{\mathcal{S}_{sk}(\cdot)}(I, pk)$
    Run $A'(I, pk)$ answering its sign-oracle queries as follows:
        When $A'$ makes a query $x$ to the sign-oracle do
            $y \leftarrow \mathcal{S}_{sk}(x) \| 0$ ; return $y$ to $A'$
    Until $A'$ outputs a forgery $(M, \sigma)$
    Parse $\sigma$ as $\sigma' \| b$ where $b$ is a bit
    Return $(M, \sigma')$

It is easy to see that if the forgery output $(M,\sigma)$ of $A'$ is "valid" (i.e. $\mathcal{VS}'_{pk}(M,\sigma) = 1$) and $M$ is "new" (i.e. $A'$ never made an oracle query $M$), then the forgery output $(M,\sigma')$ of $A$ will be also "valid" and $M$ will be also considered "new" with respect to its sign-oracle. This implies $\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},A}(k) \geq \mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS}',A'}(k)$, as desired. ∎

**Proof of Theorem 4.8:**

Let $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{V}})$ be a public key based authenticated encryption scheme constructed via the Encrypt-then-Sign method from the encryption scheme $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and the signature scheme $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$. We show that $\mathcal{PKAE}$ is RUF-CTXT secure if $\mathcal{DS}$ is SUF-CMA secure using the standard reduction method. We construct an adversary $F$ attacking SUF-CMA of $\mathcal{DS}$ using an adversary $B = (B_1, B_2)$ attacking RUF-CTXT of $\mathcal{PKAE}$ as follows:

Algorithm $F^{\mathcal{S}_{sk}(\cdot)}(I_s, pk)$
$I_e \xleftarrow{R} \mathcal{K}_{ce}(k)$ ; $I \leftarrow (I_e, I_s)$
$(st, pk_r) \leftarrow B_1(I)$
Run $B_2(st, pk, pk_r)$ answering its encryption oracle queries as follows:
When $B_2$ makes a query $x$ to the encryption oracle do
$c' \leftarrow \mathcal{E}_{pk_r}(x)$ ; $\sigma \leftarrow \mathcal{S}_{sk}(c')$ ; $c \leftarrow pk \| c' \| \sigma$ ; return $c$ to $B_2$
Until $B_2$ outputs $(C, pk_r{}', sk_r{}')$
Parse $C$ as $pk' \| C' \| \sigma'$
Return $(C', \sigma')$

Using its signing oracle, $F$ can easily simulate $B$'s encryption oracle and the adversary $F$ succeeds as long as $B$ succeeds because their definitions of valid forgery (signature and ciphertext) match. Hence, $\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k) \geq \mathbf{Adv}^{\text{ruf-ctxt}}_{\mathcal{PKAE},B}(k)$, which results in the conclusion of the theorem. ▮

**Proof of Theorem 4.9:**

In order to prove that the Encrypt-then-Sign method does not provide IND-CCA security in general, we present an adversary $A$ attacking the IND-CCA security of a PKAE scheme $\mathcal{PKAE} = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$, constructed based on any public key encryption scheme $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and any digital signature scheme $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ via the Encrypt-then-Sign method. The algorithm for the adversary $A$ is shown below:

Algorithm $A^{\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(\mathcal{LR}(\cdot,\cdot,b)), \overline{\mathcal{D}}_{sk_r}(\cdot)}(I, pk_s, pk_r)$
$M_0 \leftarrow 0$ ; $M_1 \leftarrow 1$
$C \leftarrow \overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(\mathcal{LR}(M_0, M_1, b))$
Parse $C$ as $pk_s \| C' \| \sigma$
$(pk', sk') \xleftarrow{R} \mathcal{K}_s(I)$
$\sigma' \leftarrow \mathcal{S}_{sk'}(C')$
$M \leftarrow \overline{\mathcal{D}}_{sk_r}(pk' \| C' \| \sigma')$
If $M = M_1$ then return 1 else return 0

Above, the decryption oracle query $(pk' \| C' \| \sigma')$ will be considered valid by the decryption oracle because it was never output by the LR-encryption oracle (because $pk' \| C' \| \sigma' \neq pk_s C' \| \sigma$) and the signature $\sigma'$ is valid with respect to the adversary's own public key $pk'$ (meaning, $\mathcal{VS}_{pk'}(C', \sigma') = 1$). Because $A$ used $C'$ (which is the ciphertext part of the LR-encryption oracle corresponding to the encryption of $M_b$) as the ciphertext part of its decryption oracle query, the plaintext $M$ output by the decryption oracle will the same as the original plaintext $M_b$, which was encrypted by the LR-encryption oracle. By comparing $M_b$ with $M_0$ and $M_1$, the adversary can determine the bit $b$. Note that this attack works regardless of the security properties of the base primitives of $\mathcal{PKAE}$ (meaning, even if the base public key encryption scheme is IND-CCA secure, the associated PKAE scheme is still INC-CCA insecure). This implies that the Encrypt-then-Sign method is inherently insecure in the IND-CCA sense. ▮

**Proof of Theorem 5.2:**

Let $ESSR = (\mathcal{K}_c, \mathcal{K}_s, \mathcal{K}_r, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the PKAE scheme as per Construction 5.1, and let $\mathcal{PE} = (\mathcal{K}_{ce}, \mathcal{K}_e, \mathcal{E}, \mathcal{D})$ and $\mathcal{DS} = (\mathcal{K}_{cs}, \mathcal{KS}, \mathcal{S}, \mathcal{VS})$ be its base (asymmetric) encryption scheme and a digital signature scheme, respectively.

Let $B$ be an adversary attacking IND-CCA of $ESSR$. We will construct adversaries $A$ and $F$ attacking

IND-CCA of $\mathcal{PE}$, and SUF-CMA of $\mathcal{DS}$, respectively, running $B$ as a subroutine, and show

$$\mathbf{Adv}^{\text{ind-cca}}_{ESSR,B}(k) \leq \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PE},A}(k) + \mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k) \ .$$

The adversary $A$ is given as input the public information $I_e$ and the public key $pk_r$ for the encryption scheme $\mathcal{PE}$ and has access to a left-or-right encryption oracle $\mathcal{E}_{pk_r}(\mathcal{LR}(\cdot,\cdot,b))$. It generates a signature key, and runs $B$ answering its encryption query using the generated signature key and its own encryption oracle. When $B$ outputs a bit $d$, $A$ outputs the same bit. The adversary $F$ is given as input the public informations $(I_s, pk_s)$ for the signature scheme $\mathcal{DS}$ and has access to the sign-oracle $\mathcal{S}_{pk_s}(\cdot)$. It generates an encryption key, and runs $B$ answering its encryption oracle using the generated encryption key and its own sign-oracle. The details of the algorithms $A$ and $F$ are shown in Figure 6.

In Figure 6, the algorithm for $A$ simulates the encryption oracle in the same way as the real experiment for $B$. It also simulates the decryption oracle in the same way as the real experiment except when $B$ makes a query that causes $A$ to query its decryption oracle on one of $A$'s LR-encryption oracle outputs —i.e. when $B$'s decryption oracle query is of the form $pk_s\|E'\|\sigma'$, where $E'$ was output by $A$'s LR-encryption oracle — and when this happens, $A$ just sets $\mathsf{SBad}$ to true and halts. Let $\mathsf{AskedSig}$ denote this event. Note that concatenating the sender key $pk_s$ to the plaintext and encrypting it as the ciphertext part $E$ for the encryption algorithm and then checking whether the decrypted sender key matches the purported sender key in the decryption algorithm prevent $B$ from using the same encrypted ciphertext part $E$ and replacing the signature part with its own signature under its own key. This is why $A$ can give $\perp$ to $B$ as a response without invoking its decryption oracle when $B$'s query is of the form $pk\|E'\|\sigma'$ where $pk \neq pk_s$ and $E'$ is one of the responses of $A$'s LR-encryption oracle. In other cases, $A$ can invoke the decryption oracle and responds to $B$'s decryption query in a straightforward manner.

Hence, under the absence of the event $\mathsf{AskedSig}$, $A$ can use $B$ to attack against IND-CCA of $\mathcal{PE}$ as shown in the algorithm for $A$ above. Since $A$ outputs the same bit as what $B$ outputs, we have

$$\begin{aligned}\Pr\Big[\mathbf{Exp}^{\text{ind-cca-1}}_{ESSR,B}(k) = 1 \wedge \overline{\mathsf{AskedSig}}\Big] &- \Pr\Big[\mathbf{Exp}^{\text{ind-cca-0}}_{ESSR,B}(k) = 1 \wedge \overline{\mathsf{AskedSig}}\Big] \\ &= \Pr\Big[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PE},A}(k) = 1\Big] - \Pr\Big[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PE},A}(k) = 1\Big]\end{aligned}$$

When the event $\mathsf{AskedSig}$ occurs, $F$ can use $B$ to attack SUF-CMA of $\mathcal{DS}$ as shown in the algorithm for $F$ above. The reason is that when $B$'s query is of the form $pk_s\|E'\|\sigma'$, and when it is "new" as a ciphertext, it implies that $(E',\sigma')$ as a pair is "new" as a pair, and if $\sigma'$ is a valid signature of $E'$, then it is exactly what is considered a successful forgery against SUF-CMA. Note that the algorithm for the adversary $F$ does not complete the execution of $B$ when the event $\mathsf{AskedSig}$ occurs, but it just outputs $(E',\sigma)$ as the forgery. This means that regardless of which bit $B$ outputs as its guess or what the value of the bit $b$ was chosen for the LR-encryption oracle, as long as the event $\mathsf{AskedSig}$ occurs, $F$ can succeed in attacking SUF-CMA of $\mathcal{DS}$, which means the advantage of $B$ in this case is upper bounded by the advantage of $F$. Hence, the following equation holds:

$$\Pr\Big[\mathbf{Exp}^{\text{ind-cca-1}}_{ESSR,B}(k) = 1 \wedge \mathsf{AskedSig}\Big] - \Pr\Big[\mathbf{Exp}^{\text{ind-cca-0}}_{ESSR,B}(k) = 1 \wedge \mathsf{AskedSig}\Big] \ \leq \ \Pr\Big[\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k) = 1\Big]$$

Combining the above two equations, we have

$$\begin{aligned}&\mathbf{Adv}^{\text{ind-cca}}_{ESSR,B}(k) \\ &\quad = \quad \Pr\Big[\mathbf{Exp}^{\text{ind-cca-1}}_{ESSR,B}(k) = 1\Big] - \Pr\Big[\mathbf{Exp}^{\text{ind-cca-0}}_{ESSR,B}(k) = 1\Big]\end{aligned}$$

Algorithm $A^{\mathcal{E}_{pk_r}(\mathcal{LR}(\cdot,\cdot,b)),\mathcal{D}_{sk_r}(\cdot)}(I_e, pk_r)$

$I_s \stackrel{R}{\leftarrow} \mathcal{K}_{cs}(k)$ ; $(pk_s, sk_s) \stackrel{R}{\leftarrow} \mathcal{KS}(I_s)$ ; $I \leftarrow (I_e, I_s)$

Run $B(I, pk_s, pk_r)$ answering $B$'s queries as follows:

When $B$ makes a LR-encryption oracle query $(m_0, m_1)$ do

$E \leftarrow \mathcal{E}_{pk_r}(\mathcal{LR}(m_0 \| pk_s, m_1 \| pk_s, b))$ ; $\sigma \leftarrow \mathcal{S}_{sk_s}(E \| pk_r)$

$C \leftarrow pk_s \| E \| \sigma$ ; return $C$ to $B$

When $B$ makes a decryption oracle query $C'$ do

Parse $C'$ as $pk \| E' \| \sigma'$

If $\mathcal{VS}_{pk}(E' \| pk_r, \sigma') = 0$ then return $\perp$ to $B$

If $pk = pk_s$ then

If $E'$ was output by $A$'s LR-encryption oracle

then $\mathsf{SBad} \leftarrow$ `true` ; `halt`

else $X \leftarrow \mathcal{D}_{sk_r}(E')$

Parse $X$ as $M \| pk'$

If $pk' = pk$ then return $(pk, M)$ to $B$ else return $\perp$ to $B$

If $pk \neq pk_s$ then

If $E'$ was output by $A$'s LR-encryption oracle

then return $\perp$ to $B$

else $X \leftarrow \mathcal{D}_{sk_r}(E')$

Parse $X$ as $M' \| pk'$

If $pk' = pk$ then return $(pk, M)$ to $B$ else return $\perp$ to $B$

Until $B$ outputs a bit $d$

Return $d$

---

Algorithm $F^{\mathcal{S}_{sk_s}(\cdot)}(I_s, pk_s)$

$I_e \stackrel{R}{\leftarrow} \mathcal{K}_{ce}(k)$ ; $I \leftarrow (I_e, I_s)$ ; $(pk_r, sk_r) \stackrel{R}{\leftarrow} \mathcal{K}_e(I_e)$ ; $b \stackrel{R}{\leftarrow} \{0,1\}$

Run $B(I, pk_s, pk_r)$ answering $B$'s queries as follows:

When $B$ makes a LR-encryption oracle query $(m_0, m_1)$ do

$E \leftarrow \mathcal{E}_{pk_r}(m_b)$ ; $\sigma \leftarrow \mathcal{S}_{sk_s}(E \| pk_r)$

$C \leftarrow pk_s \| E \| \sigma$ ; return $C$ to $B$

When $B$ makes a decryption oracle query $C'$ do

Parse $C'$ as $pk \| E' \| \sigma'$

If $\mathcal{VS}_{pk}(E' \| pk_r, \sigma') = 0$ then return $\perp$ to $B$

If $pk = pk_s$ then

Return $(E', \sigma')$ as the forgery output

If $pk \neq pk_s$ then

$X \leftarrow \mathcal{D}_{sk_r}(E')$

Parse $X$ as $M \| pk'$

If $pk' = pk$ then return $(pk, M)$ to $B$ else return $\perp$ to $B$

Until $B$ outputs a bit $d$

Figure 6: Algorithms for the proof of Theorem 5.2

$$
\begin{aligned}
= \quad & \Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{ESSR,B}(k) = 1 \wedge \overline{\mathsf{AskedSig}}\right] + \Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{ESSR,B}(k) = 1 \wedge \mathsf{AskedSig}\right] \\
& - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{ESSR,B}(k) = 1 \wedge \overline{\mathsf{AskedSig}}\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{ESSR,B}(k) = 1 \wedge \mathsf{AskedSig}\right] \\
\leq \quad & \Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathcal{PE},A}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathcal{PE},A}(k) = 1\right] + \Pr\left[\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k) = 1\right]
\end{aligned}
$$

$$\leq \quad \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PE},A}(k) + \mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k)$$

as desired.

The assumption that $\mathcal{PE}$ is IND-CCA secure and $\mathcal{DS}$ is SUF-CMA secure implies that $\mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PE},A}(\cdot)$ and $\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(\cdot)$ are negligible, and hence it follows that $\mathbf{Adv}^{\text{ind-cca}}_{ESSR,B}(\cdot)$ is also negligible, which results in the conclusion of the theorem. ▮

**Proof of Theorem 5.3:**

We will first show that *ESSR* is RUF-PTXT secure if $\mathcal{DS}$ is WUF-CMA secure. Let $B = (B_1, B_2)$ be any adversary attacking RUF-PTXT of *ESSR*. We construct an adversary $F$ attacking WUF-CMA of $\mathcal{DS}$ running the adversary $B$ and show that $F$ succeeds in attacking WUF-CMA of $\mathcal{DS}$ if $B$ succeeds in attacking RUF-PTXT of *ESSR*. The adversary $F$ has access to a sign oracle $\mathcal{S}_{sk_s}(\cdot)$ and is given as input the public information including the public key $(pk_s)$ of the signature scheme, and runs $B$ as follows:

Algorithm $F^{\mathcal{S}_{sk_s}(\cdot)}(I_s, pk_s)$
  $I_e \xleftarrow{R} \mathcal{K}_{ce}(k)$ ; $I \leftarrow (I_e, I_s)$
  $(st, pk_r) \leftarrow B_1(I)$
  Run $B_2(st, pk_s, pk_r)$ answering its oracle queries as follows:
    When $B_2$ makes a query $m$ to its encryption oracle do
      $E \leftarrow \mathcal{E}_{pk_r}(m \| pk_s)$ ; $\sigma \leftarrow \mathcal{S}_{sk_s}(E \| pk_r)$
      return $pk_s \| E \| \sigma$ to $B_2$
  Until $B_2$ outputs $(C, {pk_r}', {sk_r}')$
  Parse $C$ as $(pk \| C' \| \sigma')$
  Return $(C' \| {pk_r}', \sigma')$

Let $x$ denote the output of the decryption algorithm computed on $B$'s ciphertext forgery $C = pk \| C' \| \sigma'$ (i.e. $x \leftarrow \overline{\mathcal{D}}_{{sk_r}'}(C)$). The success of $B$ in attacking RUF-PTXT of *ESSR* requires that $x \neq \perp$, which means $x$ can be parsed as $(pk, M)$. Furthermore, it requires that $pk = pk_s$ and $M$ be "new" (i.e. was never queried to the encryption oracle $\overline{\mathcal{E}}_{\langle sk_s, pk_s, pk_r \rangle}(\cdot)$). Since $pk$ has to equal $pk_s$, $pk_s$ needs to be used for verification throughtout the proof. For the decryption $x$ to be "valid" with respect to $pk_s$, $\mathcal{VS}_{pk_s}(C' \| {pk_r}', \sigma)$ needs to return 1. For $M$ to be "new", either $C'$ or ${pk_r}'$ has to be "new" because otherwise $\mathcal{D}_{{sk_r}'}(C')$ will not return a "new" message $M$. If either $C'$ or ${pk_r}'$ is "new", the message part $C' \| {pk_r}'$ of the adversary $F$'s forgery will be "new" with respect to previous sign-oracle queries. Because their requirements for "validness" and "newness" match in both attack models, the success of $B$ attacking RUF-PTXT of *ESSR* implies the success of $F$ attacking WUF-CMA of $\mathcal{DS}$ in the above algorithm, which results in the following equation:

$$\mathbf{Adv}^{\text{wuf-cma}}_{\mathcal{DS},F}(k) \;\geq\; \mathbf{Adv}^{\text{ruf-ptxt}}_{ESSR,B}(k)$$

From the above equation, the theorem statement regarding the RUF-PTXT security of *ESSR* is obtained.

The RUF-CTXT security of *ESSR* can be shown using the same algorithm $F$ shown above. We show that $F$ succeeds in attacking SUF-CMA of $\mathcal{DS}$ if $B$ succeeds in attacking RUF-CTXT of *ESSR*. From the definition of SUF-CMA of $\mathcal{DS}$, the adversary $F$ wins (i.e. $\mathbf{Exp}^{\text{suf-cma}}_{\mathcal{DS},F}(k)$ returns 1) if $\mathcal{VS}_{pk_s}(C' \| {pk_r}', \sigma) = 1$ and $(C' \| {pk_r}', \sigma)$ as a pair is "new". For $B$'s forger ciphertext $C = pk \| C' \| \sigma$ to be a successful forgery in the RUF-CTXT sense, $pk = pk_s$ and the ciphertext part $C' \| \sigma$ needs to be "new" and $\overline{\mathcal{D}}_{\langle pk_s, sk_r \rangle}(C)$ does not return $\perp$ (i.e. $\mathcal{VS}_{pk_s}(C' \| pk_r', \sigma) = 1$). Since meeting these two conditions are sufficient for meeting the two conditions for the success of $F$, $F$ will succeed in attacking SUF-CMA of $\mathcal{DS}$ if $B$ succeeds in attacking RUF-CTXT of *ESSR*. Hence, we have the following equation:

$$\mathbf{Adv}^{\text{suf-cma}}_{\mathcal{DS},F}(k) \;\geq\; \mathbf{Adv}^{\text{ruf-ctxt}}_{ESSR,B}(k)$$

From the above equation, the theorem statement regarding the RUF-CTXT security of *ESSR* is obtained. ▮

**Proof of Theorem 5.7:**

Let $H$ be a hash function, and let $\mathcal{GG}$ be a group generator algorithm. Let $\mathsf{MA} = (\mathsf{K_m}, \mathsf{T}, \mathsf{VT})$ and $\mathsf{SE} = (\mathsf{K_e}, \mathsf{E}, \mathsf{D})$ be the given MAC and symmetric encryption scheme, respectively. Let $\mathsf{DHETM} = (\overline{\mathcal{K}_c}, \overline{\mathcal{K}_s}, \overline{\mathcal{K}_r}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the given PKAE scheme. We want to show IND-CCA of $\mathsf{DHETM}$ based on IND-CPA of $\mathsf{SE}$ and SUF-CMA of $\mathsf{MA}$ under the ODH assumption on $(H, \mathcal{GG})$.

The proof is based on the standard reduction argument. Given an adversary $B$ attacking IND-CCA of $\mathsf{DHETM}$, we can construct an adversary $A$ attacking the "hardcoreness" of $H$ on $\mathcal{GG}$ under adaptive DH attack (the ODH assumption), and an adversary $D$ attacking IND-CPA of $\mathsf{SE}$ and an adversary $F$ attacking SUF-CMA of $\mathcal{SDS}$. The success of each adversary $A$, $D$, and $F$ depends on the success of $B$ in different settings. Let $(X, x)$, and $(Y, y)$ be the public, secret key pairs of the sender and receiver, respectively, where $X = g^x$ and $Y = g^y$. There are two cases for the output of $H$: it can look random or not random (i.e. $H(g^{xy})$). There are two types for $B$'s query to the decryption oracle $\overline{\mathcal{D}}_y(\cdot)$, which is of the form $X' \| C$, where $X'$ is a public key of a sender, $C$ is a ciphertext: *critical* query and *non-critical* query. A critical query is a query $X \| C$, such that $X' = X$ (i.e. the public key $X'$ is the same as the real sender's public key) and $\overline{\mathcal{D}}_y(X' \| C) \neq \perp$ (the ciphertext is valid with respect to $X'$). A non-critical query is the rest of the query type (e.g. $X' \| C$ where $X' \neq X$).

There are three cases to consider depending on the output of $H$ and the types of $B$'s decryption oracle query. First case is that the output of $H$ does not look random, and in this case, we can construct $A$ that uses $B$ to violate the ODH assumption. The second case is that the output of $H$ looks random, and $B$ does not make a critical query to the decryption oracle. In this case, we can construct $D$ that violates IND-CPA of $\mathsf{SE}$ using $B$. The third case is that the output of $H$ looks random, and $B$ makes a critical query, and in this case, we can construct $F$ that violates SUF-CMA of $\mathcal{SDS}$ using $B$. From the three cases, we can conclude that if the base primitives are secure in their own senses, $\mathsf{DHETM}$ is also secure in the IND-CCA sense.

The algorithms for the adversaries $A$, $D$, and $F$ are shown in Figure 7.

From Definition 5.5, we have

$$\mathbf{Adv}^{\text{odh}}_{H,\mathcal{GG},A}(k) = \Pr\left[\mathbf{Exp}^{\text{odh-1}}_{H,\mathcal{GG},A}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{odh-0}}_{H,\mathcal{GG},A}(k) = 1\right]$$

*Claim 1.* When $Z$ given as input to the adversary $A$ shown in Figure 7 is of the form $H(g^{xy})$, the following equation holds:

$$\Pr\left[\mathbf{Exp}^{\text{odh-1}}_{H,\mathcal{GG},A}(k) = 1\right] \geq \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cca}}_{ECSR,B}(k)$$

*Proof:* When $Z$ is of the form of $H(g^{xy})$ given $X = g^x$ and $Y = g^y$, where $x, y \overset{R}{\leftarrow} Z_q$, the encryption and decryption oracles are simulated by $A$ in the same way as the real encryption and decryption oracles

Algorithm $A^{\mathcal{H}_y(\cdot)}((q,g),X,Y,Z)$
  Parse $Z$ as $K_e \| K_m$ ; $b \xleftarrow{R} \{0,1\}$
  Run $B$ on input $((q,g),X,Y)$
    For each LR-encryption query $(m_0,m_1)$ do
      $E \leftarrow \mathsf{E}_{K_e}(m_b)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(E)$
      return $X\|E\|\tau$ to $B$
    For each decryption query $C$ do
      Parse $C$ as $X'\|C'\|\tau'$
      If $X' \neq X$ then $K'_e\|K'_m \leftarrow \mathcal{H}_y(X')$
      else $K'_e \leftarrow K_e$ ; $K'_m \leftarrow K_m$
      If $\mathsf{VT}_{K'_m}(C',\tau') = 1$ then
        $M' \leftarrow \mathsf{D}_{K'_e}(C')$
        return $(X',M')$ to $B$
      else return $\perp$ to $B$
  Until $B$ outputs a bit $d$
  If $b = d$ then return 1 else return 0

Algorithm $D^{\mathsf{E}_{K_e}(\mathcal{LR}(\cdot,\cdot,b))}(k)$
  $(q,g) \xleftarrow{R} \overline{\mathcal{K}_c}(k)$ ; $(X,x) \xleftarrow{R} \overline{\mathcal{K}_s}(q,g)$
  $(Y,y) \xleftarrow{R} \overline{\mathcal{K}_r}(q,g)$ ; $K_m \xleftarrow{R} \{0,1\}^{L_m}$
  Run $B$ on input $((q,g),X,Y)$
    For each LR-encryption query $(m_0,m_1)$ do
      $E \leftarrow \mathsf{E}_{K_e}(\mathcal{LR}(m_0,m_1,b))$ ; $\tau \leftarrow \mathsf{T}_{K_m}(E)$
      return $X\|E\|\tau$ to $B$
    For each decryption query $C$ do
      Parse $C$ as $X'\|C'\|\tau'$
      If $X' \neq X$ then $K'_e\|K'_m \leftarrow H((X')^y)$
      else $K'_m \leftarrow K_m$
      If $\mathsf{VT}_{K'_m}(C',\tau') = 1$ then
        If $X' = X$ then $\mathsf{Asked} \leftarrow$ `true` ; `halt`
        else $M' \leftarrow \mathsf{D}_{K'_e}(C')$ ; return $(X',M')$ to $B$
      else return $\perp$ to $B$
  Until $B$ outputs a bit $d$
  return $d$

Algorithm $F^{\mathsf{T}_{K_m}(\cdot),\mathsf{VT}_{K_m}(\cdot)}(k)$
  $(q,g) \xleftarrow{R} \overline{\mathcal{K}_c}(k)$ ; $(X,x) \xleftarrow{R} \overline{\mathcal{K}_s}(q,g)$ ; $(Y,y) \xleftarrow{R} \overline{\mathcal{K}_r}(q,g)$ ; $K_e \xleftarrow{R} \{0,1\}^{L_e}$ ; $b \xleftarrow{R} \{0,1\}$
  Run $B$ on input $((q,g),X,Y)$
    For each LR-encryption query $(m_0,m_1)$ do
      $E \leftarrow \mathsf{E}_{K_e}(m_b)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(E)$ ; return $X\|E\|\tau$ to $B$
    For each decryption query $C$ do
      Parse $C$ as $X'\|C'\|\tau'$
      If $X' \neq X$ then
        $K'_e\|K'_m \leftarrow H((X')^y)$
        If $\mathsf{VT}_{K'_m}(C',\tau) = 1$ then $M' \leftarrow \mathsf{D}_{K'_e}(C')$ ; return $(X',M')$ to $B$ else return $\perp$ to $B$
      If $X' = X$ then
        If $\mathsf{VT}_{K_m}(C',\tau') = 1$
          then Return // F succeeded in forging
          else return $\perp$ to $B$
  Until $B$ outputs a bit $d$

Figure 7: Algorithms for the proof of Theorem 5.7

given to $B$ in the experiment $\mathbf{Exp}^{\text{ind-cca-b}}_{\mathsf{DHETM},B}(k)$. Hence, we have

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{odh-1}}_{H,\mathcal{GG},A}(k)=1\right] &= \frac{1}{2}\cdot\Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathsf{DHETM},B}(k)=1\right]+\frac{1}{2}\cdot\Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathsf{DHETM},B}(k)=0\right]\\
&= \frac{1}{2}\cdot\Pr\left[\mathbf{Exp}^{\text{ind-cca-1}}_{\mathsf{DHETM},B}(k)=1\right]+\frac{1}{2}\cdot\left(1-\Pr\left[\mathbf{Exp}^{\text{ind-cca-0}}_{\mathsf{DHETM},B}(k)=1\right]\right)\\
&= \frac{1}{2}+\frac{1}{2}\cdot\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{DHETM},B}(k)
\end{aligned}$$

as desired. □

When $Z$ is a random string (i.e. $Z \xleftarrow{R} \{0,1\}^{L_h}$), depending on whether a critical query is made by $B$

or not, it can be further divided into two cases. Note that when a critical query is made by $B$, the variable $\mathsf{Asked}$ set true in the algorithm $D$ shown in Figure 7. Let $\mathsf{AskedValid}$ be the event where $\mathsf{Asked}$ is set true (i.e. a critical query is made by $B$). In case the event $\mathsf{AskedValid}$ did not occur, we have the following claim.

*Claim 2.* When $Z$ is a random string ($Z \xleftarrow{R} \{0,1\}^{L_h}$), and the event $\mathsf{AskedValid}$ occurs, the following equation holds:

$$\Pr\left[\mathbf{Exp}^{\text{odh-0}}_{H,\mathcal{GG},A}(k) = 1 \wedge \overline{\mathsf{AskedValid}}\right] \leq \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},D}(k)$$

*Proof:* If $Z$ given to $A$ as input is randomly chosen (i.e. $Z \xleftarrow{R} \{0,1\}^{L_h}$), the simulation of the LR-encryption oracle provided by the adversary $A$ to the adversary $B$ is essentially the same as that provided by the adversary $D$ to $B$ because in both simulations, the keys for the base symmetric encryption scheme and MAC are effectively chosen at random and independent of the values of $X$ and $Y$. The simulation of decryption oracle is also correct except when the variable $\mathsf{Asked}$ is set. Since $A$ outputs 1 whenever $B$ guesses the bit $b$ right, while $D$ outputs whatever $B$ outputs, the probability that $A$ outputs 1 in this case is at most the probability that $D$ guesses the bit $b$ right ($b = d$). Hence, we have

$$\begin{aligned}
&\Pr\left[\mathbf{Exp}^{\text{odh-0}}_{H,\mathcal{GG},A}(k) = 1 \wedge \overline{\mathsf{AskedValid}}\right] \\
\leq\;& \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{SE},D}(k) = 1\right] + \frac{1}{2} \cdot \Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{SE},D}(k) = 0\right] \\
=\;& \frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},D}(k)
\end{aligned}$$

as desired. □

If the event $\mathsf{AskedValid}$ occurs when $Z$ is a random string, the forger $F$ can break the SUF-CMA security of the MAC scheme $\mathsf{MA} = (\mathsf{T}, \mathsf{VT})$ running the adversary $B$.

*Claim 3.* When $Z$ is a random string ($Z \xleftarrow{R} \{0,1\}^{L_h}$), and the event $\mathsf{AskedValid}$ occurs, the following equation holds:

$$\Pr\left[\mathbf{Exp}^{\text{odh-0}}_{H,\mathcal{GG},A}(k) = 1 \wedge \mathsf{AskedValid}\right] \leq \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},F}(k)$$

*Proof:* If $Z$ given to $A$ as input is randomly chosen (i.e. $Z \xleftarrow{R} \{0,1\}^{L_h}$), the simulation of the LR-encryption oracle provided by the adversary $A$ to the adversary $B$ is essentially the same as that provided by the forger $F$ to $B$ because in both simulations, the keys for the base symmetric encryption scheme and MAC are effectively chosen at random and independent of the values of $X$ and $Y$. If $B$ makes a critical query $X'\|C'\|\tau'$ to the decryption oracle (which defines the event $\mathsf{AskedValid}$), it implies, by definition, $X' = X$ and $C'\|\tau'$ is valid with respect to $X$, meaning $\mathsf{VT}_{K_m}(C', \tau') = 1$. Since the decryption oracle query $X'\|C'\|\tau$ needs to be "new" with respect the encryption oracle responses, it implies that $C'\|\tau'$ is "new", which means $\tau$ was never output by $\mathsf{VT}_{K_m}$ on input $C'$. These two conditions implied by a critical query of $B$ are the conditions required for a successful forgery against SUF-CMA of $\mathsf{MA}$. Hence, the forger $F$ running $B$ succeeds in forging when $B$ makes a critical query, and this justifies the above claim. □

Combining the results from the above three claims, we have

$$\mathbf{Adv}^{\text{odh}}_{H,\mathcal{GG},A}(k) \;\geq\; \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cca}}_{ECSR,B}(k) - \frac{1}{2} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{SE},D}(k) - \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},F}(k)$$

Transposing terms and taking the maximum of the advantages over all adversaries limited to the given resources, we obtain the conclusion of Theorem 5.7. ▮

**Proof of Theorem 5.8:**

Let $H$ be a hash function, and let $\mathcal{GG}$ be a group generator algorithm. Let $\mathsf{SE} = (\mathsf{K_e}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme, and let $\mathsf{MA} = (\mathsf{K_m}, \mathsf{T}, \mathsf{VT})$ be a MAC scheme. Let $\mathsf{DHETM} = (\overline{\mathcal{K}_c}, \overline{\mathcal{K}_s}, \overline{\mathcal{K}_r}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be a $\mathcal{PKAE}$ scheme constructed based on $H$, $\mathsf{SE}$ and $\mathsf{MA}$ as per Construction 5.4. We prove the security of $\mathsf{DHETM}$ using the standard reduction argument. Given an adversary $F$ attacking TUF-CTXT of $\mathsf{DHETM}$, we construct an adversary $A$ attacking the "hardcoreness" of $H$ on $\mathcal{GG}$ (the HDH assumption) and an adversary $B$ attacking SUF-CMA of $\mathsf{MA}$.

The adversary $A$ and $B$ run $F$ as a subroutine answering its oracle queries, and then using the output forgery of $F$, $A$ determines whether $Z$ is random or not and $B$ outputs a forgery in the SUF-CMA sense. The details of the algorithms for $A$ and $B$ are shown below.

| | |
|---|---|
| Algorithm $A((q,g), X, Y, Z)$<br>  Parse $Z$ as $K_e \| K_m$<br>  Run $F$ on input $((q,g), X, Y)$<br>    When $F$ queries the $\overline{\mathcal{E}}$ oracle on $m$ do<br>      $E \leftarrow \mathsf{E}_{K_e}(m)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(E)$<br>      return $X \| E \| \tau$ to $F$<br>  Until $F$ outputs a forgery ciphertext $C$<br>  Parse $C$ as $X' \| C' \| \tau'$<br>  If $X' = X$ and $C$ was never a response<br>  given to $F$ and $\mathsf{VT}_{K_m}(C', \tau') = 1$<br>    then return 1 else return 0 | Algorithm $B^{\mathsf{T}_{K_m}(\cdot), \mathsf{VT}_{K_m}(\cdot)}(k)$<br>  $(q,g) \xleftarrow{R} \overline{\mathcal{K}_c}(k)$ ; $(X,x) \xleftarrow{R} \overline{\mathcal{K}_s}(q,g)$<br>  $(Y,y) \xleftarrow{R} \overline{\mathcal{K}_r}(q,g)$ ; $K_e \xleftarrow{R} \{0,1\}^{L_e}$ ; $b \xleftarrow{R} \{0,1\}$<br>  Run $F$ on input $((q,g), X, Y)$<br>    When $F$ queries the $\overline{\mathcal{E}}$ oracle on $m$ do<br>      $E \leftarrow \mathsf{E}_{K_e}(m)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(E)$<br>      return $X \| E \| \tau$ to $F$<br>  Until $F$ outputs a forgery ciphertext $C$<br>  Parse $C$ as $X' \| C' \| \tau'$<br>  $b \leftarrow \mathsf{VT}_{K_m}(C', \tau')$ |

As can be seen from the above algorithms, the algorithms $A$ and $F$ use $B$'s outputs to achieve their own goals depending on whether $Z$ looks random or not. If $Z$ is of the form $H(g^{xy})$ (i.e. the case where the hash output does not look random), the encryption oracle $A$ simulates for $B$ is the same as the real encryption algorithm given to $B$ in the experiment $\mathbf{Exp}^{\text{tuf-ctxt}}_{\mathsf{DHETM},F}(k)$. Since $A$ outputs 1 if $B$ succeeds in forging a ciphertext against TUF-CTXT of $\mathsf{DHETM}$, the following equation holds:

$$\Pr\left[\mathbf{Exp}^{\text{hdh-1}}_{H,\mathsf{Group},A}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{tuf-ctxt}}_{\mathsf{DHETM},F}(k) = 1\right]$$

If $Z \xleftarrow{R} \{0,1\}^{L_h}$ (i.e. the case where the hash output does look random), the distribution of the encryption oracle responses $A$ simulates for $B$ is essencially the same as what $F$ simulates for $B$, and $F$ can succeed in attacking SUF-CMA of $\mathsf{MA}$ using $B$'s successful forgery ciphertext in this case. Hence, the following equation holds:

$$\Pr\left[\mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MA},B}(k) = 1\right] \;\geq\; \Pr\left[\mathbf{Exp}^{\text{hdh-0}}_{H,\mathcal{GG},A}(k) = 1\right]$$

Combining the above results, we obtain

$$\begin{aligned}\mathbf{Adv}^{\text{hdh}}_{H,\mathcal{GG},A}(k) \;&\geq\; \Pr\left[\mathbf{Exp}^{\text{tuf-ctxt}}_{\mathsf{DHETM},F}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{suf-cma}}_{\mathsf{MA},B}(k) = 1\right] \\ &=\; \mathbf{Adv}^{\text{tuf-ctxt}}_{\mathsf{DHETM},F}(k) - \mathbf{Adv}^{\text{suf-cma}}_{\mathsf{MA},B}(k)\ .\end{aligned}$$

Transposing the terms, and taking into account the resources used by the adversaries, we get the conclusion of the theorem. ▮

**Proof of Theorem 5.9:** Let $\mathsf{DHETM} = (\overline{\mathcal{K}_c}, \overline{\mathcal{K}_s}, \overline{\mathcal{K}_r}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the PKAE scheme constructed as per Construction 5.4. The adversary $B = (B_1, B_2)$ against RUF-CTXT of $\mathsf{DHETM}$ is given an oracle $\overline{\mathcal{E}}_{\langle x_a, y_a, x_b \rangle}(\cdot)$ in its second stage and works as follows:

| | |
|---|---|
| Algorithm $B_1(q, g)$ | Algorithm $B_2^{\overline{\mathcal{E}}_{\langle x_a, y_a, x_b \rangle}(\cdot)}(st, y_a, y_b)$ |
| $\quad x_b \xleftarrow{R} Z_q$ | $\quad$ Parse $st$ as $(x_b, q, g)$ |
| $\quad y_b \leftarrow g^{x_b}$ | $\quad K \xleftarrow{R} H(y_a^{x_b})$ ; Parse $K$ as $K_e \Vert K_m$ |
| $\quad st \leftarrow (x_b, q, g)$ | $\quad c' \leftarrow \mathsf{E}_{K_e}(0)$ ; $\tau \leftarrow \mathsf{T}_{K_m}(c')$ |
| $\quad$ return $(st, y_b)$ | $\quad C \leftarrow y_a \Vert c' \Vert \tau$ |
| | $\quad$ return $(C, y_b, x_b)$ |

It is easy to see that the ciphertext $C$ output by $B$ in the second stage is "valid", meaning $\overline{\mathcal{D}}_{x_b}(C) \neq \perp$. Because the encryption is based on the common key computed from the keys of both sender and receiver, $B$ as an adversarial receiver who knows its own key can easily compute the common key as well as the sender. The adversary $B$ does not have to invoke the encryption oracle and it can encrypt any messages of its choice. Since $B$ did not query the encryption oracle, both the output forgery ciphertext $C$ and its corresponding plaintext "0" will be considered "new". Hence, the same adversary $B$ can be used for attacking both RUF-PTXT and RUF-CTXT security of DHETM. Combining these results, we obtain the following equation

$$\mathbf{Adv}^{\text{ruf-ptxt}}_{\mathsf{DHETM},B}(k) = \mathbf{Adv}^{\text{ruf-ctxt}}_{\mathsf{DHETM},B}(k) = 1$$

as desired.

Note that although $B$ did not change its own key pair in the second stage, it could have easily changed its key and still succeed in generating a valid ciphertext forgery of its choice with respect to the changed key. Here, $B$'s ability to change the key does not affect its success probability. ▮