Relaxing Chosen-Ciphertext Security

RAN CANETTI[*] HUGO KRAWCZYK[†] JESPER B. NIELSEN[‡]

August 19, 2003

Abstract

Security against adaptive chosen ciphertext attacks (or, CCA security) has been accepted as the standard requirement from encryption schemes that need to withstand active attacks. In particular, it is regarded as the appropriate security notion for encryption schemes used as components within general protocols and applications. Indeed, CCA security was shown to suffice in a large variety of contexts. However, CCA security often appears to be somewhat *too strong:* there exist encryption schemes (some of which come up naturally in practice) that are not CCA secure, but seem sufficiently secure "for most practical purposes."

We propose a relaxed variant of CCA security, called Replayable CCA (RCCA) security. RCCA security accepts as secure the non-CCA (yet arguably secure) schemes mentioned above; furthermore, it suffices for most existing applications of CCA security. We provide three formulations of RCCA security. The first one follows the spirit of semantic security and is formulated via an ideal functionality in the universally composable security framework. The other two are formulated following the indistinguishability and non-malleability approaches, respectively. We show that the three formulations are equivalent in most interesting cases.

[*] IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598. Email: `canetti@watson.ibm.com`.

[†] Department of Electrical Engineering, Technion, Haifa 32000, Israel, and IBM T.J. Watson Research Center, New York, USA. Email: `hugo@ee.technion.ac.il`.

[‡] BRICS, Centre of the Danish National Research Foundation. Department of Computer Science, University of Aarhus, DK-8000 Arhus C, Denmark. Email: `buus@brics.dk`. Work done while at IBM T.J. Watson Research Center.

Contents

1 Introduction

One of the main goals of cryptography is to develop mathematical notions of security that adequately capture our intuition for the security requirements from cryptographic tasks. Such notions are then used to assess the security of protocols and schemes. They also provide abstractions that, when formulated and used correctly, greatly facilitate the design and analysis of cryptographic applications.

With respect to encryption schemes, a first step was taken with the introduction of semantic security of (public key) encryption schemes in [15]. This first step is indeed a giant one, as it introduces the basic definitional approach and techniques that underlie practically all subsequent notions of security, for encryption as well as many other cryptographic primitives.

However, semantic security under chosen-plaintext attacks as defined in [15] captures only the very basic requirement from an encryption scheme, namely secrecy against "passive" eavesdroppers. In contrast, when an encryption scheme is used as a component within a larger protocol or system, a much wider array of attacks against the scheme are possible. Specifically, adversaries may have control over the messages that are being encrypted, but may also have control over the ciphertexts being delivered and decrypted. This opens new ways for the attacker to infer the outcome of the decryption of some ciphertexts by observing the system (see e.g. [6, 18]).

Several notions of "security against active adversaries" were proposed over the years in order to capture such often subtle security concerns. These notions include semantic security against Lunchtime Attacks (or, IND-CCA1 security), semantic security against Adaptive Chosen Ciphertext Attacks (or, IND-CCA2 security), Non-Malleability against the above attacks, and more [20, 22, 11, 3]. In particular, CCA2 security (where the semantic-security and the non-malleability formulations are equivalent) became the "golden standard" for security of encryption schemes in a general protocol setting. Indeed, CCA2 security (or simply CCA security) was demonstrated to suffice for a number of central applications, such as authentication and key exchange [11, 2, 9], encrypted password authentication [16], and non-interactive message transmission [7]. In addition, in [7] CCA is shown to suffice for realizing an "ideal public-key encryption functionality" within the universally composable security (UC) framework, thus demonstrating its general composability properties.

CCA security is indeed a very strong and useful notion. But is it *necessary* for an encryption scheme to be CCA-secure in order to be adequate for use within general protocol settings? Some evidence that this may not be the case has been known all along: Take any CCA-secure encryption scheme S, and change it into a scheme S' that is identical to S except that the encryption algorithm appends a 0 to the ciphertext, and the decryption algorithm discards the last bit of the ciphertext before decrypting. It is easy to see that S' is no longer CCA-secure, since by flipping the last bit of a ciphertext one obtains a different ciphertext that decrypts to the same value as the original one, and this "slackness" is prohibited by CCA security. But it seems that this added slackness of S' is of no "real consequence" in most situations. In other words, S' appears to be just as secure as S for most practical purposes. This example may seem contrived, but it in fact turns up, in thin disguises, in a number of very natural settings. (Consider for instance an implementation of some CCA-secure scheme, where for wider interoperability the decryption algorithm accepts ciphertexts represented both in big-endian and in little-endian encodings.) To give another example of a slightly different nature, consider the scheme S'' that is identical to S except that it re-encrypts the ciphertext generated by S using another encryption scheme which is only guaranteed to be secure against chosen plaintext attacks. As with S', we have that S'' is not necessarily CCA-secure, but it seems that the extra slackness is of little consequence in most situations. In fact, some relaxations of CCA-security were already proposed in the literature, e.g. [19, 18, 23, 1]. However, while being good first steps, these notions were not fully justified as either sufficient or necessary for applications. (In particular, these notions capture S', but do not capture S''.)

We propose a new relaxed version of CCA-security, called Replayable CCA (RCCA) security. In essence, RCCA is aimed at capturing encryption schemes that are CCA secure "except that they allow anyone to generate new ciphertexts that decrypt to the same value as a given ciphertext." RCCA is strictly weaker than CCA security. In fact, it is strictly weaker than the relaxations in [18, 23, 1]. The rationale behind RCCA is that as far as an attacker in a protocol setting is concerned, generating *different* ciphertexts that decrypt to the *same* plaintext as a given ciphertext has the same effect as copying (or, "replaying") the same ciphertext multiple times. Since replaying a ciphertext multiple times is unavoidable even for CCA secure encryptions, RCCA security would have "essentially the same effect" as CCA security.

To substantiate this intuition, we prove that RCCA security suffices for all of the above major applications of CCA secure encryption (authentication, key exchange, etc.). We also demonstrate that the *hybrid encryption* paradigm can be based on RCCA security rather than CCA security. (Hybrid encryption calls for encrypting a key k using an asymmetric encryption and then encrypting a long message using symmetric encryption with key k.)

It should be stressed that the above rationale holds only as long as the protocol that uses the scheme makes its decisions based only on the outputs of the decryption algorithm, and does not directly compare ciphertexts. Arguably, most applications of CCA secure encryption have this property. However, in some applications it is natural and helpful to directly compare ciphertexts. For instance, consider a voting scheme in which votes are encrypted, and illegal duplicate votes are detected via direct ciphertext comparison. In such cases, the full power of CCA security is indeed used.

We provide three formulations of RCCA security. The first two are formulated via "guessing games" along the lines of CCA security. The first of these, called IND-RCCA, has the flavor of "security by indistinguishability" (or, IND-CCA2 in the terminology of [3]) with a CCA-style game that allows for plaintext replay. The second notion, called NM-RCCA, has the flavor of non-malleability in a CCA-style game that allows for plaintext replay. The third notion, called UC-RCCA, is formulated via an ideal functionality in the UC framework [7]. This ideal functionality, called $\mathcal{F}_{\text{RPKE}}$, is obtained by modifying the ideal functionality $\mathcal{F}_{\text{PKE}}$ in [7] to explicitly allow the environment to generate ciphertexts that decrypt to the same value as a given ciphertext. Having been formulated in the UC framework, this notion provides strong and general composability guarantees. Furthermore, in the spirit of semantic security, it provides a clear and explicit formalization of the provided security guarantee. It also explicitly demonstrates the exact sense in which RCCA weakens CCA.[1]

We show that, when applied to encryption schemes where the message domain is "large" (i.e., super-polynomial in the security parameter), the three notions are equivalent.[2] When the message domain is polynomial in size, we have that UC-RCCA implies NM-RCCA, and NM-RCCA implies IND-RCCA. We also show, via a separating example, that in this case IND-RCCA does *not* imply

[1]Krohn in [19] studies various relaxations of CCA security and their respective strengths. One of these notions is essentially the same as IND-RCCA security. However, no concrete justification for this notion is provided.

[2]We say that an encryption scheme has message domain D if, for any message $m \in D$, the process of encrypting m and then decrypting the resulting ciphertext returns m. Thus the larger the domain, the stronger the requirement. (Encryption schemes with large message domain should not be confused with encryption schemes that guarantee security only if the message is taken uniformly from a large domain. The latter is a weak notion of security, whereas the former is only a correctness requirement, and can be used in conjunction with any security requirement.)

NM-RCCA. Whether NM-RCCA implies UC-RCCA for polynomial message domains remains open.

For schemes that handle large message domains, having the three equivalent formalizations allows us to enjoy the best of each one: We have the intuitive appeal and strong composability of the UC-RCCA, together with the relative simplicity of NM-RCCA and IND-RCCA. Indeed, the case of large message domains is arguably the most interesting one, since most existing encryption schemes are either directly constructed for large message domains, or can be extended to deal with large domains in a natural way. Also, most applications of public-key encryption, e.g. encrypting an identity or a key for symmetric encryption, require dealing with large message domains.

The three notions in a nutshell. Let us briefly sketch the three notions. See Section 3 for more detailed description and rationale. First recall the standard (indistinguishability based) formulation of CCA security for public-key cryptosystems. Let $S = (gen, enc, dec)$ be a public-key encryption scheme where gen is the key generation algorithm, enc is the encryption algorithm, and dec is the decryption algorithm. Informally, S is said to be CCA secure if any feasible attacker $\mathcal{A}$ succeeds in the following game with probability that is only negligibly more than one half . Algorithm gen is run to generate an encryption key e and a decryption key d. $\mathcal{A}$ is given e and access to a decryption oracle $dec(d, \cdot)$. When $\mathcal{A}$ generates a pair m_0, m_1 of messages, a bit $b \overset{\text{R}}{\leftarrow} \{0, 1\}$ is chosen and $\mathcal{A}$ is given $c = enc(e, m_b)$. From this point on, $\mathcal{A}$ may continue querying its decryption oracle, with the exception that if $\mathcal{A}$ asks to decrypt the "test ciphertext" c, then $\mathcal{A}$ receives a special symbol `test` instead of the decryption of c. $\mathcal{A}$ succeeds if it outputs b.

IND-RCCA is identical to CCA, with the exception that the decryption oracle answers `test` whenever it is asked to decrypt *any ciphertext that decrypts to either* m_0 *or* m_1, even if this ciphertext is different than the test ciphertext c. Indeed, in the IND-RCCA game the ability to generate new ciphertexts that decrypt to the test ciphertext does not help the adversary. (Yet, it is not immediately clear from this formulation that we did not weaken the security requirement by too much. The justification for this notion comes mainly from its equivalence with UC-RCCA, described below.)

NM-RCCA is identical to IND-RCCA, with the exception that $\mathcal{A}$ succeeds if $m_0 \neq m_1$ and it outputs *a ciphertext* c' *that decrypts to* m_{1-b}. Note that if we required $\mathcal{A}$ to output m_{1-b} explicitly, we would get a requirement that is only a reformulation of IND-RCCA. So the difference is in the fact that here $\mathcal{A}$ is only required to output an encryption of m_{1-b}, without necessarily being able to output m_{1-b} explicitly. This requirement has a flavor of non-malleability, thus the name. (Indeed, it can be regarded as a non-malleability requirement in which the attacker is considered successful as long as the "malleability relation" it uses is not the "equality relation.")

UC-RCCA is defined via an ideal functionality, $\mathcal{F}_{\text{RPKE}}$. To best understand $\mathcal{F}_{\text{RPKE}}$, let us first recall the "ideal public-key encryption" functionality, $\mathcal{F}_{\text{PKE}}$, from [7], that captures CCA security.[3] In fact, instead of getting into the actual mechanism of $\mathcal{F}_{\text{PKE}}$ (see Section 3), let us only sketch the security guarantee it provides. Functionality $\mathcal{F}_{\text{PKE}}$ captures the behavior of an "ideal encryption service." That is, $\mathcal{F}_{\text{PKE}}$ provides an encryption interface that is available to all parties, and a decryption interface that is available only to one privileged party, the decryptor. When querying the encryption interface with some message m, a ciphertext c is returned. The value of c is chosen by the adversary, without any knowledge of m. This guarantees "perfect secrecy" for encrypted messages. When the decryption interface is queried with a "legitimate encryption of m" (i.e., with

[3]In [7] it is mistakenly claimed that CCA security is a *strictly* stronger requirement than realizing $\mathcal{F}_{\text{PKE}}$ for non-adaptive adversaries. However, as shown in this work, the two requirements are actually *equivalent*. The mistake in [7] and the equivalence proof were independently discovered in [17].

a string c that was the outcome of a request to encrypt m), then the returned value is m. Since there is no requirement on how "illegitimate ciphertexts," i.e. strings that were not generated using the encryption interface, are being decrypted, $\mathcal{F}_{\text{PKE}}$ allows the adversary to choose the decryption values of these ciphertexts.

Functionality $\mathcal{F}_{\text{RPKE}}$ is identical to $\mathcal{F}_{\text{PKE}}$, except that it allows the adversary to request to decrypt "illegitimate ciphertexts" to the same value as some previously generated legitimate ciphertext. This directly captures the relaxation where the adversary is allowed to generate new ciphertexts which decrypt to the same (unknown) value as existing ciphertexts. It also demonstrates that RCCA does not weaken CCA-security beyond allowing for "plaintext replay" by the attacker.

Between RCCA security and CCA security. As sketched above, RCCA security allows anyone to modify a given ciphertext c into a different ciphertext c', as long as c and c' decrypts to the same message. One potential strengthening of RCCA security is to require that it will be possible to detect, given two ciphertexts c and c', whether one is a "modified version" of the other. (Indeed, the "endian changing" example given above has this additional property.) Here it is natural to distinguish between schemes where the detection algorithm uses the secret decryption key, and schemes where the detection can be done given only the public encryption key. We call such schemes secretly detectable RCCA (sd-RCCA) and publicly detectable RCCA (pd-RCCA), respectively.

We first observe that pd-RCCA security is essentially equivalent to the notions proposed by Krawczyk [18], Shoup [23], and An, Dodis and Rabin [1]. (The notions are called, respectively, loose ciphertext-unforgeability, benign malleability, and generalized CCA security, and are essentially the same.) Next we study the relations between these notions. It is easy to see that: CCA security $\Rightarrow$ pd-RCCA security $\Rightarrow$ sd-RCCA security $\Rightarrow$ RCCA security. We show that the two leftmost implications are strict. (The first is implied by the above "endian changing" example; the second is obtained via constructing a scheme that is based on the above "double encryption" example.) Whether sd-RCCA security is equivalent to RCCA security remains open.

Finally, we provide a generic construction that turns any RCCA secure scheme (with message domain $\{0,1\}^k$ where k is the security parameter) into a CCA scheme. The construction is quite efficient, and uses only shared-key primitives. This in essence demonstrates that the existence of an RCCA secure encryption scheme implies the existence of a CCA secure scheme without any additional computational assumptions. Also, this construction may provide an alternative way of obtaining CCA security.

Symmetric encryption. In this work we develop the RCCA notions mainly for public-key encryption. However, the notion can be adapted to the symmetric-key setting in a straightforward way. We outline this generalization in Section 5.4 where we use RCCA-secure symmetric encryption to build RCCA-secure public-key hybrid encryption schemes.

Organization. Section 2 recalls the formulation of CCA security, and establishes its equivalence with the universally composable notion of security for public-key encryption schemes (against non-adaptive adversaries) as defined in [7]. Section 3 presents the three variants of RCCA security and establishes the relationships among them. Section 4 studies the detectable variants of RCCA security. It also shows how to turn any RCCA secure scheme into a CCA secure one. Section 5 demonstrates several central applications where RCCA security can be used instead of CCA security.

2 Prologue: On CCA security

Before introducing our RCCA definitions, we recall the formulation of CCA security. We also demonstrate that the notion of secure public-key encryption in the universally composable framework [7] is *equivalent* to CCA security. (In particular, this corrects the erroneous claim from [7] that the UC characterization is *strictly weaker* than CCA security. The mistake in [7] and the equivalence proof were discovered independently in [17].) This equivalence sets the stage for the presentation of RCCA. In particular, by comparing the UC formalizations of CCA security and of RCCA security it is easier to see that the technical relaxation from CCA to RCCA coincides with the intuition behind the later notion as described above, namely, that "replayable CCA" is identical to CCA except for the added ability of the attacker to generate new ciphertexts that decrypt to the same plaintexts as previously seen ciphertexts. We start by establishing the basic formal setting for public-key encryption schemes.

2.1 CCA secure encryption schemes

Public-key encryption schemes. Throughout the paper we model (public key) encryption schemes as triples of probabilistic polynomial-time algorithms $S = (gen, enc, dec)$ together with an ensemble of finite domains (sets) $\mathcal{D} = \{D_k\}_{k\in\mathbf{N}}, D_k \subset \{0,1\}^*$ which can be recognized and sampled in PPT (in k). Algorithm *gen*, on input k (k is a security parameter), generates a pair of keys (e, d). The encryption and decryption algorithms, *enc* and *dec*, satisfy that if $(e, d) = gen(k)$, then for any message $m \in D_k$ we have $dec_d(enc_e(m)) = m$ except with negligible probability. The range of the decryption function may include a special symbol $\texttt{invalid} \notin \mathcal{D}_k, \forall k$.[4]

CCA security. We recall the definition of CCA security (or IND-CCA2) for public key encryption schemes. See [22, 11, 3]. Let $S = (gen, enc, dec)$ be an encryption scheme over domain $\mathcal{D} = \{D_k\}_{k\in\mathbf{N}}$. This definition, presented next, is based on the CCA game described in Figure 1.[5]

Definition 1 *An encryption scheme S is said to be* CCA-secure *if any polynomial-time adversary F wins the IND-CCA game of Figure 1 with probability that is at most negligibly more than one half.*

2.2 Equivalence of CCA and UC security of encryption schemes

A UC characterization of CCA security. Here we assume some familiarity of the reader with the UC framework. See [7] for an overview and full details. Within the UC framework, public-key encryption is defined via the public-key encryption functionality from [7], denoted $\mathcal{F}_{\text{PKE}}$ and presented in Figure 2. Functionality $\mathcal{F}_{\text{PKE}}$ is intended at capturing the functionality of a public-key encryption scheme as a *tool* to be used within other protocols. In particular, $\mathcal{F}_{\text{PKE}}$ is written in a way

[4]Jumping ahead, we remark that the `invalid` output is not necessary for obtaining CCA security. In fact, any CCA-secure encryption scheme that uses the `invalid` output can be modified into a CCA-secure scheme that does not use `invalid` outputs, by outputting a random message in the domain instead of `invalid`. See more details in [8]. Still, our formalization covers also schemes which do use the `invalid` output. Indeed, such output may be useful for providing other properties, in addition to CCA security.

[5]The explicit requirement in this game that $m_0 \neq m_1$ is immaterial for the definition of CCA and the later definition of IND-RCCA (in which choosing $m_0 = m_1$ is of no benefit to the attacker), but will be substantial in our definition of NM-RCCA. Thus, for the sake of uniformity we present all our definitions using the explicit requirement $m_0 \neq m_1$.

The CCA Game

The game proceeds as follows, given an encryption scheme $S = (gen, enc, dec)$, an adversary F, and value k for the security parameter.

Key generation: Run $(e, d) \leftarrow gen(k)$, and give e to F.

First decryption stage: When F queries `(ciphertext,`c`)`, compute $m = dec_d(c)$ and give m to F.

Encryption stage: When F queries `(test messages,`m_0, m_1`)`, $m_0, m_1 \in D_k$, and $m_0 \neq m_1$, compute $c^* = enc_e(m_b)$ where $b \overset{R}{\leftarrow} \{0,1\}$, and give c^* to F. (This step is performed only once.)

Second decryption stage: When F queries `(ciphertext,`c`)` after c^* is defined, proceed as follows. If $c = c^*$ then give `test` to F.[a] Otherwise, compute $m = dec_d(c)$ and give m to F.

Guessing stage: When F outputs `(guess,`b'`)`, the outcome of the game is determined as follows. If $b' = b$ then F wins the game. Otherwise, F loses the game.

Figure 1: The CCA game.

[a] The symbol `test` is a reserved symbol, which is different from all possible outputs of *dec*.

that allows realizations that consist of three non-interactive algorithms without any communication. (The three algorithms correspond to the key generation, encryption, and decryption algorithms in the traditional definitions.) All the communication is left to the higher-level protocols that use $\mathcal{F}_{\text{PKE}}$.

Referring to Figure 2, we note that *id* serves as a unique identifier for an instance of functionality $\mathcal{F}_{\text{PKE}}$ (this is needed in a general protocol setting when this functionality can be composed with other components, or even with other copies of $\mathcal{F}_{\text{PKE}}$). The "public key value" e has no particular meaning in the ideal scenario beyond serving as an identifier for the public key related to this instance of the functionality, and can be chosen arbitrarily by the attacker. Also, in the ideal setting ciphertexts serve as identifiers or tags with no particular relation to the encrypted messages (and as such are also chosen by the adversary without knowledge of the plaintext). Still, rule 1 of the decryption operation guarantees that "legitimate ciphertexts", i.e. those produced and recorded by the functionality under an `Encrypt` request, are decrypted correctly and the resultant plaintexts remain unknown to the adversary. In contrast, ciphertexts that were not legitimately generated can be decrypted in any way chosen by the ideal-process adversary. (Since the attacker obtains no information on legitimately encrypted messages, we are guaranteed that illegitimate ciphertexts will be decrypted to values that are independent from the legitimately encrypted messages.) Note that the same illegitimate ciphertext can be decrypted to different values in different activations. This provision allows the decryption algorithm to be non-deterministic with respect to ciphertexts that were not legitimately generated.

Another characteristics of $\mathcal{F}_{\text{PKE}}$ is that, when activated with a `KeyGen` request, it always responds with an (adversarially chosen) encryption key e'. Still, only the first key to be generated is recorded, and only messages that are encrypted with that key are guaranteed to remain secret. Messages encrypted with other keys are disclosed to the adversary in full. This modeling represents the fact that a single copy of the functionality captures the security requirements of only a single instance of a public-key encryption scheme (i.e., a single pair of encryption and decryption keys). All the other keys may provide correct encryption and decryption, but do not guarantee any security.[6]

$\mathcal{F}_{\text{PKE}}$ is parameterized by $\mathcal{D} = \{D_k\}_{k\in\mathbf{N}}$, the ensemble of domains of the messages to be encrypted. Given security parameter k, $\mathcal{F}_{\text{PKE}}$ encrypts messages in domain D_k.

Functionality $\mathcal{F}_{\text{PKE}}$

$\mathcal{F}_{\text{PKE}}$ proceeds as follows, when parameterized by message domain ensemble $\mathcal{D} = \{D_k\}_{k\in\mathbf{N}}$ and security parameter k, and interacting with an adversary $\mathcal{S}$, and parties $P_1, ..., P_n$.

Key Generation: Upon receiving a value $(\texttt{KeyGen}, id)$ from some party P_i, do:

1. Hand $(\texttt{KeyGen}, id)$ to the adversary.
2. Receive a value e from the adversary, and hand e to P_i.
3. If this is the first activation then record the value e.

Encryption: Upon receiving from some party P_j a value $(\texttt{Encrypt}, id, e', m)$ proceed as follows:

1. If $m \notin D_k$ then return an error message to P_j.
2. If $m \in D_k$ then hand $(\texttt{Encrypt}, id, e', P_j)$ to the adversary. (If $e' \neq e$ or e is not yet defined then hand also the entire value m to the adversary.)
3. Receive a "ciphertext" c from the adversary, record the pair (c, m), and hand c to P_j. (If $e' \neq e$ or e is not yet defined then do *not* record the pair (c, m). If the tag c already appears in a previously recorded pair then return an error message to P_j.)

Decryption: Upon receiving a value $(\texttt{Decrypt}, id, c)$ from P_i (and P_i only), proceed as follows:

1. If there is a recorded pair (c, m) then hand m to P_i.
2. Otherwise, hand the value $(\texttt{Decrypt}, id, c)$ to the adversary. When receiving a value m' from the adversary, hand m' to P_i.

Figure 2: The public-key encryption functionality, $\mathcal{F}_{\text{PKE}}$

$\mathcal{F}_{\text{PKE}}$ captures CCA security. We show the equivalence between the notion of security induced by functionality $\mathcal{F}_{\text{PKE}}$ and the notion of CCA security. First, recall the following natural transformation from an encryption scheme S to a protocol π_S that is geared toward realizing $\mathcal{F}_{\text{PKE}}$.

1. When activated, within some P_i and with input $(\texttt{KeyGen}, id)$, run algorithm *gen*, output the encryption key e and record the decryption key d.

[6]An alternative formulation would instruct $\mathcal{F}_{\text{PKE}}$ to ignore all KeyGen requests except for the first one. However, such formulation cannot be realized by key generation algorithms that are run locally within a party without interaction. This is so since, without interaction, a key generation algorithm cannot tell whether other requests have occurred in the network. Yet another alternative would instruct $\mathcal{F}_{\text{PKE}}$ to ignore a KeyGen request unless the session identifier contains the identity of the requesting party. However, such a formulation would assume that the key generation algorithm necessarily knows the identity of the party it runs on; we would like to avoid such assumptions in the basic definition of encryption schemes.

2. When activated, within some party P_j and with input (`Encrypt`$, id, e', m$), return $enc_{e'}(m, r)$ for a randomly chosen r. (Note that it does not necessarily hold that $e' = e$.)

3. When activated, within P_i and with input (`Decrypt`$, id, c$), return $dec_d(c)$.

We show:

Theorem 2 *Let $S = (gen, enc, dec)$ be an encryption scheme over domain $\mathcal{D}$. Then S is CCA-secure if and only if π_S securely realizes $\mathcal{F}_{\text{PKE}}$ with respect to domain $\mathcal{D}$ and non-adaptive adversaries.*

Remark: We stress that protocol π_S is only required to realize $\mathcal{F}_{\text{PKE}}$ with respect to *non-adaptive* adversaries. Realizing $\mathcal{F}_{\text{PKE}}$ with respect to *adaptive* adversaries is a considerably stronger requirement. Indeed, using the techniques of [21], it can be shown that no protocol of the above form can realize $\mathcal{F}_{\text{PKE}}$ with respect to adaptive adversaries, for any scheme S. Realizing $\mathcal{F}_{\text{PKE}}$ with respect to adaptive adversaries requires additional mechanisms, such as forward secure encryption or highly interactive solutions based on non-committing encryption.

Proof (sketch): We first show that if π_S securely realizes $\mathcal{F}_{\text{PKE}}$ in the presence of non-adaptive adversaries then S is CCA-secure. Assume that there exists an adversary F that predicts the bit b correctly with probability $1/2 + \epsilon$, in a CCA interaction with scheme S. We construct an environment $\mathcal{Z}$ that distinguishes with probability ϵ between an interaction in the ideal process for $\mathcal{F}_{\text{PKE}}$ with any ideal-process adversary $\mathcal{S}$, and a real-life interaction with the dummy adversary $\tilde{\mathcal{A}}$ and π_S. Environment $\mathcal{Z}$ invokes a copy of $\mathcal{F}$, and proceeds as follows, in a network of two uncorrupted parties P_1, P_2.

1. Initially, $\mathcal{Z}$ activates P_1 with input (`KeyGen`$, id$) for some value of id (say, $id = 0$), obtains the public key e and hands e to F.

2. When F generates the two test plaintexts (m_0, m_1), $\mathcal{Z}$ chooses $b \overset{\text{R}}{\leftarrow} \{0, 1\}$, activates P_2 with input (`Encrypt`$, id, e, m_b$), obtains a ciphertext c^*, and hands c^* to F as the test ciphertext.

3. When F asks its decryption oracle to decrypt a ciphertext $c \neq c^*$, $\mathcal{Z}$ activates P_1 with input (`Decrypt`$, id, c$), obtains a plaintext m, and hands m to F.

4. When F outputs a bit b', $\mathcal{Z}$ outputs $b \oplus b'$ and halts.

Analyzing $\mathcal{Z}$, notice that if $\mathcal{Z}$ operates in the real-life model with the dummy adversary $\tilde{\mathcal{A}}$ and parties running π_S, then the copy of F within $\mathcal{Z}$ sees in fact a CCA interaction with scheme S. Thus, in this case $b' = b$ with probability at least $1/2 + \epsilon$.

In contrast, we claim that when $\mathcal{Z}$ operates in the ideal process for $\mathcal{F}_{\text{PKE}}$ and *any* ideal-process adversary, the view of the copy of F within $\mathcal{Z}$ is statistically independent of b, thus in this case $b' = b$ with probability exactly one half. To see why F's view is independent of b recall that the view of $\mathcal{F}$ consists of the text ciphertext c^*, plus the decryptions of all the ciphertexts generated by F (except for the decryption of c^*). Notice that all of these values are generated by the ideal-process adversary $\mathcal{S}$. Furthermore, throughout the interaction, the view of $\mathcal{S}$ is independent of b.

It remains to show that if S is CCA-secure then π_S securely realizes $\mathcal{F}_{\text{RPKE}}$. This part was already proven in Claim 15 in [7]. For self containment we repeat this proof here. Let $S = (gen, enc, dec)$ be a CCA-secure encryption scheme, and let π_S be the protocol constructed from S as described above. We show that π_S securely realizes $\mathcal{F}_{\text{PKE}}$. Using the alternative definition of security (Definition 4 on page 22 in [7]), we construct an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell with non-negligible probability whether it interacts with $\mathcal{F}_{\text{PKE}}$ and $\mathcal{S}$ in the ideal process or with parties running π_S and the dummy adversary $\tilde{\mathcal{A}}$ in the real-life model.

Recall that $\tilde{\mathcal{A}}$ takes three types of messages from $\mathcal{Z}$: either to corrupt parties, or to report on messages sent in the protocol, or to deliver some message. However, since we are dealing with non-adaptive adversaries, there are no party corruption instructions. Furthermore, since protocol π_S involves no sending of messages, there are no requests to report on or deliver messages. In fact, there is no communication between $\mathcal{Z}$ and $\tilde{\mathcal{A}}$ at all. Thus, the only way in which $\mathcal{S}$ can affect the view of $\mathcal{Z}$ is by communicating with the ideal functionality $\mathcal{F}_{\text{PKE}}$. Adversary $\mathcal{S}$ proceeds as follows.[7]

1. When $\mathcal{S}$ receives a message $(\texttt{KeyGen}, id)$ from $\mathcal{F}_{\text{PKE}}$, it runs the key generation algorithm gen, obtains an encryption key e and an decryption key d, and returns e to $\mathcal{F}_{\text{PKE}}$.

2. When $\mathcal{S}$ receives a message $(\texttt{Encrypt}, id, e', P_j)$ from $\mathcal{F}_{\text{PKE}}$, it first verifies that $e' = e$. If so, then it computes $c = enc_e(m_0, r)$ for a random r and some fixed $m_0 \in D_k$ and returns c to $\mathcal{F}_{\text{PKE}}$. If $e' \neq e$ then $\mathcal{S}$ receives also the full value m from $\mathcal{F}_{\text{PKE}}$. In this case it returns $enc_{e'}(m, r)$ to $\mathcal{F}_{\text{PKE}}$.

3. When $\mathcal{S}$ receives a message $(\texttt{Decrypt}, id, c)$ from $\mathcal{F}_{\text{PKE}}$, it computes $m = dec_d(c)$ and returns m to $\mathcal{F}_{\text{PKE}}$.

Analyzing $\mathcal{S}$, assume for contradiction that there is an environment $\mathcal{Z}$ that distinguishes between the real and ideal interactions. We use $\mathcal{Z}$ to construct an adversary $\mathcal{F}$ that breaks the CCA security of the encryption scheme S. More precisely, assume that for some value of the security parameter k we have $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k) - \text{REAL}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}}(k) > \epsilon$. We show that $\mathcal{F}$ guesses the bit b correctly in the CCA game with probability $1/2 + e/2p$, where p is the total number of messages that were encrypted throughout the run of the system. (Without loss of generality, we assume that in each execution of the protocol $\mathcal{Z}$ asks to encrypt exactly p messages.)

Adversary $\mathcal{F}$ proceeds as follows, given a public key e, and having access to a decryption oracle D and an encryption oracle E. $\mathcal{F}$ first randomly chooses a number $h \overset{\text{R}}{\leftarrow} \{1, ..., p\}$. Next, $\mathcal{F}$ runs $\mathcal{Z}$ on the following simulated interaction with a system running π_S (and the dummy adversary $\tilde{\mathcal{A}}$). Let m_i denote the ith message that $\mathcal{Z}$ asks to encrypt in an execution.[8]

1. When $\mathcal{Z}$ activates some party P_i with input $(\texttt{KeyGen}, id)$, $\mathcal{F}$ lets P_i output the value e from $\mathcal{F}$'s input.

2. For the first $h - 1$ times that $\mathcal{Z}$ asks to encrypt some message, m_i, $\mathcal{F}$ lets the encrypting party return $c_i = enc_e(m_i)$.

3. At the hth time that $\mathcal{Z}$ asks to encrypt a message, m_h, $\mathcal{F}$ queries its encryption oracle with the pair of messages (m_h, m_0), where $m_0 \in D_k$ is the fixed message used above, and obtains the test ciphertext c_h. It then hands c_h to $\mathcal{Z}$ as the encryption of m_h.

4. For the remaining $p - h$ times that $\mathcal{Z}$ asks to encrypt some message, m_i, $\mathcal{F}$ lets the encrypting party return $c_i = enc_e(m_0)$.

[7]We concentrate on the case where the decrypting party is uncorrupted. The case where the decryptor is corrupted is handled is a straightforward way (details omitted).

[8]Without loss of generality we assume that $\mathcal{Z}$ only asks to encrypt messages with the public key e that was generated by the decrypting party. Indeed, when $\mathcal{Z}$ asks to encrypt a message m with a public key $e' \neq e$, it receives a value $c = enc_{e'}(m, r)$ that it can compute by itself.

5. Whenever the decryptor P_i is activated with input $(\texttt{Decrypt}, id, c)$ where $c = c_i$ for some i, $\mathcal{F}$ lets P_i return the corresponding plaintext m_i. (This holds for the case $i = h$ as well as $i \neq h$.) If c is different from all the c_i's then $\mathcal{F}$ queries its decryption oracle on c, obtains a value v, and lets P_i return v to $\mathcal{Z}$.
6. When $\mathcal{Z}$ halts, $\mathcal{F}$ outputs whatever $\mathcal{Z}$ outputs and halts.

Analyzing the success probability of $\mathcal{F}$ is done via a standard hybrids argument. Let the random variable H_i denote the output of $\mathcal{Z}$ from an interaction that is identical to an interaction with $\mathcal{S}$ in the ideal process, with the exception that the first i ciphertexts are computed as an encryption of the real plaintexts, rather than encryptions of m_0.

It is easy to see that H_0 is indistinguishable from the output of $\mathcal{Z}$ in the ideal process, and H_p is identical the output of $\mathcal{Z}$ in the real-life model. (This follows from the fact that the scheme S guarantees that $dec_d(enc_e(m)) = m$ except with negligible probability.) Furthermore, in a run of $\mathcal{F}$, if the value c_h that $\mathcal{F}$ obtains from its encryption oracle is an encryption of m_h then the output of the simulated $\mathcal{Z}$ has the distribution of H_{h-1}. If c_h is an encryption of m_0 then the output of the simulated $\mathcal{Z}$ has the distribution of H_h. The theorem follows. □

3 Replayable CCA (RCCA) Security

This section presents the three notions of security for public-key encryption sketched in the Introduction, all aimed at capturing the intuition that "the adversary should gain nothing from seeing a legitimately generated ciphertext, except for the ability to generate new ciphertexts that decrypt to the same value as the given ciphertext". Following the presentation of the three notions, we demonstrate their equivalence for encryption schemes with super-polynomial message domains, and present separating examples for polynomial message domains.

3.1 UC-RCCA: Functionality $\mathcal{F}_{\text{RPKE}}$

The UC-based formulation of RCCA security is obtained by modifying the $\mathcal{F}_{\text{PKE}}$ functionality from Figure 2, as to explicitly allow the adversary, together with the environment, to generate ciphertexts that decrypt to the same values as legitimately generated ciphertexts. Specifically, the new functionality, $\mathcal{F}_{\text{RPKE}}$, modifies step 2 of the **Decryption** stage in Figure 2 in which the decrypting party asks to decrypt a ciphertext that was not legally generated by the functionality. In this case, $\mathcal{F}_{\text{RPKE}}$ allows the adversary to fix the decrypted value to be the same as a previously encrypted value (without letting the adversary know what this value is). Thus, functionality $\mathcal{F}_{\text{RPKE}}$ is defined identically to $\mathcal{F}_{\text{PKE}}$ from Figure 2 except that step 2 of the **Decryption** stage is re-defined as follows:

Decryption: Upon receiving a value $(\texttt{Decrypt}, id, c)$ from P_i (and P_i only), proceed as follows:

1. If there is a recorded pair (c, m) then hand m to P_i.
2. Otherwise, hand the value $(\texttt{Decrypt}, id, c)$ to the adversary, and receive a value (α, v) from the adversary. If $\alpha =$'plaintext' then hand v to P_i. If α='ciphertext' then find a stored pair (c', m) such that $c' = v$, and hand m to P_i. (If no such c' is found then halt.)

In the following definition we use the transformation from an encryption scheme S into a protocol π_S as described in the previous section.

Definition 3 *Let S be an encryption scheme. We say that S is* UC-RCCA secure *if protocol π_S securely realizes $\mathcal{F}_{\text{RPKE}}$ with respect to non-adaptive adversaries.*

3.2 NM-RCCA and IND-RCCA: The RCCA games

This section presents two notions of security for encryption schemes, that are formulated via relaxed versions of the CCA game (See Figure 1 in Section 2), and demonstrates the equivalence of these notions to the UC-RCCA formulation for encryption schemes with super-polynomial domains. The two notions are called IND-RCCA and NM-RCCA, and are defined via the IND-RCCA game and the NM-RCCA game, respectively.

The IND-RCCA game (see Figure 3) differs from the CCA game in one point: When the adversary generates a `(decrypt,` c`)` request, the answer is `test` whenever c decrypts to either m_0 or m_1. Roughly speaking, this captures the intuition that "the ability to generate different ciphertexts that decrypt to the same values as a given ciphertext should not help the adversary to win the game." (As we demonstrate below, in the case of large message spaces, this intuition is supported by the equivalence between IND-RCCA and UC-RCCA.)

The IND-RCCA game

The game proceeds as follows, given an encryption scheme $S = (gen, enc, dec)$ over a domain ensemble $\mathcal{D}$, an adversary F, and value k for the security parameter.

Key generation: Run $(e, d) \leftarrow gen(k)$, and give e to F.

First decryption stage: When F queries `(ciphertext,`c`)`, compute $m = dec_d(c)$ and give m to F.

Encryption stage: When F queries `(test messages,`m_0, m_1`)`, with $m_0, m_1 \in D_k$, and $m_0 \neq m_1$, compute $c^* = enc_e(m_b)$ where $b \overset{\text{R}}{\leftarrow} \{0,1\}$, and give c^* to F. (This step is performed only once.)

Second decryption stage: When F queries `(ciphertext,`c`)` after c^* is defined, compute $m = dec_d(c)$. If $m \in \{m_0, m_1\}$ then give `test` to F. Otherwise, give m to F.

Guessing stage: When F outputs `(guess,`b'`)`, the outcome of the game is determined as follows. If $b' = b$ then F wins the game. Otherwise, F loses the game.

Figure 3: The IND-RCCA game

The NM-RCCA game is identical to the IND-RCCA game (Figure 3), with the exception that the guessing stage is defined as follows:

Guessing stage for NM-RCCA: When F outputs `(guess,`c`)`, the outcome of the game is determined as follows. Compute $m = dec_d(c)$; if $m = m_{1-b}$ then F wins the game. Otherwise, F loses the game.

In order to consider F successful we require it to output an encryption of m_{1-b}. Changing this requirement to explicitly output m_{1-b} would result in a reformulation of IND-RCCA. Thus the difference from IND-RCCA is in the fact that F is only required to output an encryption of m_{1-b}, without necessarily being able to explicitly output m_{1-b} (or, equivalently, b). As we demonstrate below, this added strength relative to IND-CCA is significant for small message domains. The formulation of the attacker's goal in the NM-RCCA definition follows the non-malleability approach (and hence the name); see the discussion below.

Definition 4 *An encryption scheme S is said to be* IND-RCCA secure *(resp.,* NM-RCCA secure*) if any polynomial-time adversary F wins the IND-RCCA game of Figure 3 (resp., the NM-RCCA game) with probability that is at most negligibly more than one half.*

Discussion. The formulation of NM-RCCA is syntactically different than the usual formulation of definitions of non-malleability. We thus provide an intuitive explanation as to why NM-RCCA indeed captures the non-malleability requirement, in spite of its different formalization. Roughly speaking, an encryption scheme is called *non-malleable* in [11, 3] if it is infeasible for an adversary to output ciphertexts which decrypt to plaintexts that satisfy some (non-trivial) relation with the plaintext encrypted under a given "challenge ciphertext" c^*. (A bit more precisely, the attacker is not given the plaintext encrypted under c^* but she may choose the probability distribution under which this plaintext is taken. Thus, "trivial relations" are those that hold for randomly chosen elements from this distribution.) In the case of non-malleability under chosen-ciphertext attacks (NM-CCA) the only restriction on the attacker is that it is not allowed to include the ciphertext c^* as one of its output ciphertexts (otherwise, the attacker could always output c^* and satisfy the "equality" relation.)
In our formulation of NM-RCCA we use the above non-malleability approach to capture our intuition behind the "replayable CCA" notion. The idea is to relax NM-CCA so that there is only one form of malleability *allowed* to the attacker: outputting a ciphertext that decrypts to *the same* plaintext as c^*. In other words, the attacker is considered successful as long as it uses any relation *other than the "equality" relation.* Now, if we carry this idea to the case where the probability distribution $\mathcal{P}$, from which the plaintext to be encrypted as c^* is selected, is of the special form $\mathcal{P} = [\{m_0, m_1\}, Prob(m_0) = Prob(m_1) = 1/2]$, with m_0, m_1 chosen by the attacker, then we obtain our "non-malleability game" NM-RCCA. Beyond this intuition and relationship to general non-malleability, the main source of confidence for this definition comes from its equivalence (at least over super-polynomial domains) with the UC-RCCA notion which captures in a more explicit way the "intuitive semantics" of RCCA.

3.3 Equivalence for large message domains

Theorem 5 *Let S be an encryption scheme whose domain ensemble $\mathcal{D}$ is super-polynomial in size. Then the following three conditions are equivalent: (I) S is UC-RCCA secure; (II) S is NM-RCCA secure; (III) S is IND-RCCA secure.*

Proof: Before proving the claim, it may be instructive to note where the proof of Theorem 2 (Section 2) fails when we move from CCA security to RCCA security. Assume first that we try to use the proof of Theorem 2 to demonstrate that if π_S securely realizes $\mathcal{F}_{\text{RPKE}}$ then S is CCA secure. Recall that the proof assumes existence of a forger F against S, and constructs an environment $\mathcal{Z}$ that distinguishes an interaction in the ideal process from an interaction with S. A central argument in the analysis of $\mathcal{Z}$ is that, in the ideal process, the view of the simulated F is independent from the bit b chosen by $\mathcal{Z}$. However, in the case of $\mathcal{F}_{\text{RPKE}}$ this is no longer the case, since the ideal-process adversary $\mathcal{S}$ can instruct $\mathcal{F}_{\text{RPKE}}$ to decrypt any unregistered ciphertext c to the same value m_b that is recoded by $\mathcal{F}_{\text{RPKE}}$. In this case, $\mathcal{Z}$ will hand m_b to F, and F will be able to determine b.

Next, assume that we try to use the proof of Theorem 2 to demonstrate that if S is either IND-RCCA secure or NM-RCCA secure then π_S securely realizes $\mathcal{F}_{\text{PKE}}$. Then the ideal-process adversary $\mathcal{S}$ constructed there would no longer be valid: Assume that the scheme S has the property that it is possible, given only the public key e, to "maul" a ciphertext c into another ciphertext c' of the same message. (As we have seen, RCCA security allows this property.) Now, consider an environment $\mathcal{Z}$ that obtains a legitimate ciphertext c of, say, the value 1. It then asks the decryptor to decrypt c' (which was generated from c as described above). Then, when interacting with S in the real-life model, c' would be decrypted to 1. However, when interacting with $\mathcal{F}_{\text{PKE}}$ and $\mathcal{S}$ in the ideal process, c' would be decrypted to 0.

We proceed with the proof. This is done in three steps, showing (I)$\Rightarrow$(II)$\Rightarrow$(III)$\Rightarrow$(I).

UC-RCCA security implies NM-RCCA security. We modify the proof for the case of CCA (Theorem 2), showing that if π_S securely realizes $\mathcal{F}_{\text{RPKE}}$ in the presence of non-adaptive adversaries then S is NM-RCCA secure. (The proof will hold for any domain ensemble $\mathcal{D}$.) Assume that there exists an adversary F that predicts the bit b correctly with probability $1/2+\epsilon$, in a RCCA interaction with scheme S. As in the case of CCA, we construct an environment $\mathcal{Z}$ that distinguishes with probability ϵ between an interaction in the ideal process for $\mathcal{F}_{\text{RPKE}}$ with any ideal-process adversary $\mathcal{S}$, and a real-life interaction with the dummy adversary $\tilde{\mathcal{A}}$ and π_S. Environment $\mathcal{Z}$ invokes a copy of $\mathcal{F}$, and proceeds as follows, in a network of three parties P_1, P_2, P_3 where only P_3 is corrupted. (The only differences from the construction of $\mathcal{Z}$ in the CCA case is that here $\mathcal{Z}$ does not hand F the decrypted message in case this message is either m_0 or m_1, and that here $\mathcal{Z}$ decrypts F's guess.)

1. Initially, $\mathcal{Z}$ activates P_1 with input $(\texttt{KeyGen}, id)$ for some value of id (say, $id = 0$), obtains the public key e and hands e to F.

2. When F generates the two test plaintexts (m_0, m_1), $\mathcal{Z}$ chooses $b \overset{\text{R}}{\leftarrow} \{0,1\}$, activates P_2 with input $(\texttt{Encrypt}, id, e, m_b)$, obtains a ciphertext c^*, and hands c^* to F as the test ciphertext.

3. When F asks its decryption oracle to decrypt a ciphertext c, $\mathcal{Z}$ activates P_1 with input $(\texttt{Decrypt}, id, c)$, and obtains a plaintext m. If $m \in \{m_0, m_1\}$ then hand $\texttt{test}$ to F. Otherwise, hand m to F.

4. When F outputs a "guess ciphertext" c, $\mathcal{Z}$ activates P_1 with input $(\texttt{Decrypt}, id, c)$, and obtains a plaintext m. If $m = m_{b'}$ for $b' \in \{0,1\}$ then $\mathcal{Z}$ outputs $b \oplus b'$. Otherwise, $\mathcal{Z}$ outputs a random coin toss.

Analogously to the CCA case, it is easy to see that if $\mathcal{Z}$ operates in the real-life model with the dummy adversary $\tilde{\mathcal{A}}$ and parties running $\pi_{\mathcal{S}}$, then the copy of F within $\mathcal{Z}$ sees in fact a NM-RCCA interaction with scheme S. Thus, in this case outputs 1 with probability at least $1/2 + \epsilon$.

As in the CCA case we claim that when $\mathcal{Z}$ operates in the ideal process for $\mathcal{F}_{\text{RPKE}}$ and *any* ideal-process adversary, the view of the copy of F within $\mathcal{Z}$ is statistically independent of b, thus $b' = b$ with probability exactly one half. Here however the independence comes from the fact $\mathcal{Z}$ answers the decryption queries of F in a way that is independent from the bit b, regardless of the values received by $\mathcal{Z}$ from the decryptor P_i. (This is so since, in contrast to the CCA game, in the NM-RCCA game the attacker never receives a decryption of a ciphertext that decrypts to any one of the two test messages.)

NM-RCCA security implies IND-RCCA security. This part of the proof is immediate (and does not depend on the size of the domain ensemble). Assume we have an adversary $\mathcal{A}$ that succeeds in an IND-RCCA game against S with probability $1/2+\epsilon$. Then construct the following adversary $\mathcal{A}'$ that succeeds in an NM-RCCA game against S with probability $1/2+\epsilon$. Adversary $\mathcal{A}'$ runs $\mathcal{A}$, with the exception that when $\mathcal{A}$ outputs a "guess message" m, $\mathcal{A}'$ encrypts m to obtain $c = enc_e(m)$, and outputs a "guess ciphertext" c.

IND-RCCA security implies UC-RCCA security. We modify the proof for the case of CCA (Theorem 2), showing that if S is IND-RCCA secure then π_S securely realizes $\mathcal{F}_{\text{RPKE}}$. This part of the proof works only if the domain ensemble $\mathcal{D}$ is of super-polynomial size. As in the case of CCA (Theorem 2), we construct an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell with non-negligible probability whether it interacts with $\mathcal{F}_{\text{RPKE}}$ and $\mathcal{S}$ in the ideal process or with parties running π_S and the dummy adversary $\tilde{\mathcal{A}}$ in the real-life model. However, here the way $\mathcal{S}$ deals with encryption and decryption requests (made by $\mathcal{F}_{\text{RPKE}}$) is somewhat different:

1. When $\mathcal{S}$ receives a message $(\texttt{KeyGen}, id)$ from $\mathcal{F}_{\text{RPKE}}$, it runs the key generation algorithm gen, obtains an encryption key e and a decryption key d, and returns e to $\mathcal{F}_{\text{RPKE}}$.

2. When $\mathcal{S}$ receives a message $(\texttt{Encrypt}, id, e', P_j)$ from $\mathcal{F}_{\text{RPKE}}$, it first verifies that $e' = e$. If so, it computes $c = enc_e(r, r')$ for $r \overset{\text{R}}{\leftarrow} D_k$ and random r' of the appropriate length, records the pair (r, c), and returns c to $\mathcal{F}_{\text{RPKE}}$. If $e' \neq e$ then $\mathcal{S}$ receives also the full value m from $\mathcal{F}_{\text{RPKE}}$. In this case it returns $enc_{e'}(m, r)$ to $\mathcal{F}_{\text{RPKE}}$.

3. When $\mathcal{S}$ receives a message $(\texttt{Decrypt}, id, c)$ from $\mathcal{F}_{\text{RPKE}}$, it first computes $m = dec_d(c)$. If $m = r$ for one of the recorded r's then $\mathcal{S}$ returns the value $(\text{'ciphertext'}, c')$ to $\mathcal{F}_{\text{RPKE}}$, where c' is the ciphertext recorded with r. Otherwise, $\mathcal{S}$ returns m to $\mathcal{F}_{\text{RPKE}}$. (The rationale here is that if $dec_d(c) = r$ and r was generated by $\mathcal{S}$ in association with some encryption operation, then the plaintext that the environment expects to see associated with c is the plaintext that was given to $\mathcal{F}_{\text{RPKE}}$ in that encryption operation. Here we use the fact that the size of the message space is large, thus the probability of collisions is small.)

Analyzing $\mathcal{S}$, assume for contradiction that there is an environment $\mathcal{Z}$ that distinguishes between the real and ideal interactions. We use $\mathcal{Z}$ to construct an adversary F that breaks the IND-RCCA security of the encryption scheme S. More precisely, assume that for some value of the security parameter k we have $\text{IDEAL}_{\mathcal{F}_{\text{RPKE}},\mathcal{S},\mathcal{Z}}(k) - \text{REAL}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}}(k) > \epsilon$. We show that F guesses the bit b correctly in the IND-RCCA game with probability $1/2 + e/4p$, where p is the total number of messages that were encrypted throughout the run of the system. (Without loss of generality, we assume that in each execution of the protocol $\mathcal{Z}$ asks to encrypt exactly p messages.)

We first establish the following. Consider an interaction of $\mathcal{Z}$ with simulator $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\text{RPKE}}$, and let B be the event where one of the values that $\mathcal{Z}$ asks to encrypt equals one of the random values r that $\mathcal{S}$ chose in its Step 2. We claim that event B happens with negligible probability. More precisely, if event B occurs with probability greater than $\epsilon/2$ then it is possible the construct an adversary F that wins in an IND-RCCA game with scheme S with probability $1/2 + 1/2p \cdot (\epsilon/2 - 1/s)$, where s is the size of the message space. (F runs $\mathcal{Z}$ and plays the role of both $\mathcal{S}$ and $\mathcal{F}_{\text{RPKE}}$ for $\mathcal{Z}$. It then chooses two random challenge plaintexts r_0 and r_1, obtains a test ciphertext c^*, and hands c^* to $\mathcal{Z}$ as the encryption of the hth plaintext to be encrypted, where $h \overset{\text{R}}{\leftarrow} [p]$. If $\mathcal{Z}$ later asks to encrypt either r_0 (resp., r_1) then F outputs 0 (resp., 1) and halts. If $\mathcal{Z}$ does not ask to decrypt either r_0 or r_1 then F outputs a coin-toss. Indeed, if c^* is an encryption of r_0 then F outputs 1 with probability $1/s$. If c^* is an encryption of r_1 then F outputs 1 with probability at least $\epsilon/2$. We omit further details. Note that F does not use its decryption oracle at all.)

For the rest of the analysis we assume that event B does not occur, and construct an adversary F such that if $\text{IDEAL}_{\mathcal{F}_{\text{RPKE}},\mathcal{S},\mathcal{Z}}(k) - \text{REAL}_{\pi,\tilde{\mathcal{A}},\mathcal{Z}}(k) > \epsilon/2$ then F wins the IND-RCCA game with probability $1/2 + e/2p$. Adversary F proceeds as follows, given a public key e, and having access to a decryption oracle D and an encryption oracle E. F first randomly chooses a number $h \overset{\text{R}}{\leftarrow} \{1, ..., p\}$. Next, F runs $\mathcal{Z}$ on the following simulated interaction with a system running π_S (and the dummy adversary $\tilde{\mathcal{A}}$). Let m_i denote the ith message that $\mathcal{Z}$ asks to encrypt in an execution.[9]

1. When $\mathcal{Z}$ activates some party P_i with input $(\texttt{KeyGen}, id)$, F lets P_i output the value e from F's input.

2. For the first $h-1$ times that $\mathcal{Z}$ asks to encrypt some message, m_i, F lets the encrypting party return $c_i = enc_e(m_i)$.

3. At the hth time that $\mathcal{Z}$ asks to encrypt a message, m_h, F queries its encryption oracle with the pair of messages (m_h, r) where $r \overset{\text{R}}{\leftarrow} D_k$, and obtains the test ciphertext c_h. It then hands c_h to $\mathcal{Z}$ as the encryption of m_h.

4. For the remaining $p-h$ times that $\mathcal{Z}$ asks to encrypt some message, m_i, F lets the encrypting party return $c_i = enc_e(r, r')$ where $r \overset{\text{R}}{\leftarrow} D_k$ and r' is random.

5. Whenever the decryptor P_i is activated with input $(\texttt{Decrypt}, id, c)$ where $c = c_i$ for some i, F lets P_i return the corresponding plaintext m_i. (This holds for the case $i = h$ as well as $i \neq h$.) If c is different from all the c_i's then F queries its decryption oracle on c, and obtains a value v. If $v \neq \texttt{test}$ then F lets P_i return v to $\mathcal{Z}$. If $v = \texttt{test}$ then F lets P_i return m_h to $\mathcal{Z}$. (The rationale here is as follows: If $v = \texttt{test}$ then c decrypts either to m_h or to r. However, as we assumed that $\mathcal{Z}$ never asks to encrypt r, then in the ideal process c would always decrypt to m_h.)

6. When $\mathcal{Z}$ halts, F outputs whatever $\mathcal{Z}$ outputs and halts.

Analyzing the success probability of F is done via a hybrids argument, similar to the proof of Theorem 2. Let the random variable H_i denote the output of $\mathcal{Z}$ from an interaction that is identical to an interaction with $\mathcal{S}$ in the ideal process, with the exception that the first i ciphertexts are computed as an encryption of the real plaintexts, rather than encryptions of random values.

It can be seen that H_0 is distributed indistinguishable from the output of $\mathcal{Z}$ in the ideal process (conditioned on the event that B never occurs). Also, H_p is distributed identically to the output of $\mathcal{Z}$ in the real-life model (this follows from the fact that the scheme S guarantees that $dec_d(enc_e(m)) = m$). Furthermore, in a run of F, if the value c_h that F obtains from its encryption oracle is an encryption of m_h then the output of the simulated $\mathcal{Z}$ has the distribution of H_{h-1}. If c_h is an encryption of r_h then the output of the simulated $\mathcal{Z}$ has the distribution of H_h. The claim follows. $\square$

[9]Without loss of generality we assume that $\mathcal{Z}$ only asks to encrypt messages with the public key e that was generated by the decrypting party. Indeed, when $\mathcal{Z}$ asks to encrypt a message m with a public key $e' \neq e$, it receives a value $c = enc_{e'}(m, r)$ that it can compute (or, sample) by itself.

3.4 Polynomial message domains

In Theorem 5 it was proved that for super-polynomial domain ensembles, the notions UC-RCCA, NM-RCCA and IND-RCCA are equivalent. Here we show that this premise is necessary. As mentioned in the proof of Theorem 5 it holds for all domain ensembles that UC-RCCA implies NM-RCCA and that NM-RCCA implies IND-RCCA. A minimal assumption for separating is therefore that there exist IND-RCCA secure encryption schemes in the first place. We do not know whether UC-RCCA and NM-RCCA are equivalent for polynomial domain ensembles.

Theorem 6 *For all polynomial-size domain ensembles $\mathcal{D}$, if there exists an IND-RCCA secure encryption scheme with domain ensemble $\mathcal{D}$, then there exists an IND-RCCA secure encryption scheme with domain ensemble $\mathcal{D}$ which is not NM-RCCA secure.*

Proof: Let $S = (gen, enc, dec)$ be an IND-RCCA secure encryption scheme with polynomial-size domain ensemble $\mathcal{D} = \{D_k\}$. Consider the encryption scheme $S' = (gen', enc', dec')$, where $gen' = gen$, $enc'_e(m) = enc(m)_e enc_e(n)$ for $n \in_R D_k \setminus \{m\}$, and $dec'_d(c_1, c_2) = dec_d(c_1)$.

We first argue that S' is not NM-RCCA secure. Consider the following adversary $\mathcal{A}$. $\mathcal{A}$ outputs $(\texttt{test messages}, m_0, m_1)$ for any pair of messages with $m_0 \neq m_1$. It then receives $(c, d) = (enc_e(m_b), enc_e(n))$, where $n \neq m_b$. $\mathcal{A}$ then outputs $(\texttt{ciphertext}, (d, c))$. If the answer is $\texttt{test}$, then $n \in \{m_0, m_1\}$ and since $n \neq m_b$ it must be the case that $n = m_{1-b}$, so $\mathcal{A}$ outputs $(\texttt{guess}, (d, c))$ and wins the game. If the answer is not $\texttt{test}$, then $\mathcal{A}$ outputs $(\texttt{guess}, enc'_e(m_0))$ and wins the game with probability $\frac{1}{2}$. As long as $\mathcal{D}$ is polynomial, then $n = m_{1-b}$ with non-negligible probability, and so $\mathcal{A}$ has an advantage non-negligibly over $\frac{1}{2}$.

Next we argue that S' is IND-RCCA secure. Intuitively, the reason is the only difference between S and S' is that in S' the adversary obtains also the ciphertext c_2, and c_2 provides no help in decrypting c_1. For a more rigorous proof, consider any IND-RCCA adversary A attacking S', and define the following probabilities. For $b, b' \in \{0, 1\}$ let p_b be the probability that A returns $(\texttt{guess}, 1)$ when given the challenge ciphertext $(enc_e(m_b), enc_e(n))$ where $n \in_R D_k \setminus \{m_b\}$, let $p_{b,*}$ be the probability that A returns $(\texttt{guess}, 1)$ when given the challenge ciphertext $(enc_e(m_b), enc_e(n))$ where $n \in_R D_k \setminus \{m_0, m_1\}$ and let $p_{b,b'}$ be the probability that A returns $(\texttt{guess}, 1)$ when given the challenge ciphertext $(enc_e(m_b), enc_e(m_{b'}))$.

We wish to prove that $|p_0 - p_1|$ is negligible. Let $d = \frac{1}{|D_k|-1}$. Observe that $p_b = dp_{b,1-b} + (1 - d)p_{b,*}$, so $p_0 - p_1 = d(p_{0,1} - p_{1,0}) + (1 - d)(p_{0,*} - p_{1,*})$. We prove the claim by demonstrating that both $p_{0,1} - p_{1,0}$ and $p_{0,*} - p_{1,*}$ are negligible.

Consider first the following IND-RCCA adversary B_1 that interacts with the underlying scheme S. B_1 receives e and hands it to A. If A outputs $(\texttt{test messages}, m_0, m_1)$ then B_1 outputs $(\texttt{test messages}, m_0, m_1)$ and receives an encryption $c = enc_e(m_b)$. Then B_1 computes $d = enc_e(n)$ for $n \in_R D_k \setminus \{m_0, m_1\}$ and hands (c, d) to A. If after this A outputs $(\texttt{ciphertext}, (c_1, c_2))$ then B_1 outputs $(\texttt{ciphertext}, c_1)$ and returns the answer to A. If A outputs $(\texttt{guess}, e)$, then B_1 outputs $(\texttt{guess}, e)$. Notice that the probability that B_1 outputs 1 given that $c = enc_e(m_b)$ is exactly $p_{b,*}$. This implies that $p_{0,*} - p_{1,*}$ is negligible.

Next consider the following IND-RCCA adversary B_2. B_2 interacts with S and is identical to B_1 with the following exceptions. First, it computes d as $d = enc_e(m_a)$ for $a \in_R \{0, 1\}$. Next, it chooses a bit $f \in_R \{0, 1\}$. If $f = 0$ then it hands (c, d) to A. If $f = 1$ then it hands (d, c) to A. Finally, when A outputs $(\texttt{guess}, e)$, then B_2 outputs $(\texttt{guess}, e + f)$. The probability that B_2 outputs $(\texttt{guess}, 1)$ given that $c = enc_e(m_b)$ is $\tilde{p}_b = \frac{1}{4}(p_{b,0} + p_{b,1} + (1 - p_{0,b}) + (1 - p_{1,b}))$, and notice that $\tilde{p}_0 - \tilde{p}_1 = \frac{1}{2}(p_{0,1} - p_{1,0})$. Since $\tilde{p}_0 - \tilde{p}_1$ is negligible, then also $p_{0,1} - p_{1,0}$ is negligible. $\square$

Important remark about terminology. Due to the equivalence between the notions of IND-RCCA, NM-RCCA and UC-RCCA security for super-polynomial domain ensembles (Theorem 5), we will usually refer to these notions under the generic term of RCCA security, and assume, for simplicity, super-polynomial domains (except if otherwise stated).

4 Between RCCA and CCA security: Detectable RCCA

Here we investigate the relations between RCCA security and CCA security, and introduce the notion(s) of "detectable RCCA". In particular, we establish the relationship between RCCA security and the relaxation of CCA security presented in [18, 23, 1]. These results include a (strict) separation between these notions, and consequently between RCCA and CCA security. In particular, this demonstrates that there exist encryption schemes that are not secure in the sense of the definitions from [18, 23, 1] and yet are sufficiently secure for most practical applications of CCA secure encryption. We complement these findings by showing (Section 4.3) how to construct a CCA-secure scheme (with any domain size) from any RCCA-secure scheme whose domain size is exponential in the security parameter. This transformation uses symmetric encryption and message authentication only, thus demonstrating that once RCCA security is obtained for large enough message spaces, CCA security can be obtained with moderate overhead (and without additional assumptions).

4.1 Detectable RCCA

The first and obvious fact to observe regarding the relation between RCCA and CCA security is that the former is strictly weaker than the latter. Indeed, a simple inspection of the definitions of CCA and RCCA security shows that any scheme that is CCA-secure is also RCCA-secure (under any of the definitions of RCCA security from Section 3). On the other hand, there are simple examples of encryption schemes that are RCCA but not CCA secure. One such example was mentioned in the introduction in which a CCA-secure scheme is modified by instructing the (modified) encryption to append a '0' bit to each ciphertext, and defining the (modified) decryption algorithm to ignore this bit. It is easy to see that the obtained scheme is not CCA but it does satisfy our definition(s) of RCCA security. Other examples exist. Specifically, consider the usual practice of allowing encryption schemes to add arbitrary padding to ciphertexts and later discard this padding before performing decryption. (This padding is usually required in order to align the length of ciphertexts to a prescribed length-boundary – e.g., to a multiple of 4 bytes.) Other examples include encryption schemes that naturally allow for more than one representation of ciphertexts, such as the endianess example in the introduction, or the example in [23] related to dual point representations in elliptic-curve cryptosystems.

All these examples have the property that given a certain ciphertext c, anyone can easily produce a different ciphertext c' that decrypts to the same plaintext (e.g., by changing the endianess representation or modifying the padding). Also common to these examples is the fact that if someone (say, the attacker) indeed modifies a ciphertext c into a ciphertext c' in one of the above ways then c and c' satisfy a relation that is easy to test with the sole knowledge of the public key. This fact (and the realization that these "syntactic deficiencies" do not seem to effect the actual security of these encryption schemes when used in many applications) has motivated the introduction of the relaxations of CCA security presented in [18, 23, 1]. Essentially, all these notions allow for "replay" of plaintexts by modifying the ciphertext, but restrict the allowed modifications to be efficiently detectable given the public key. RCCA further relaxes this requirement by allowing *any form* of ciphertext modification that does not change the plaintext, without insisting on the ability to detect such a replay. (Indeed, as argued in the introduction and demonstrated in Section 5, RCCA security is sufficient for many applications that use CCA secure encryption.)

A natural question that arises from this discussion is whether RCCA security is truly more relaxed than the notions considered in [18, 23, 1], namely, is there an RCCA-secure scheme for which the modification of ciphertexts is not "publicly detectable"? Here we provide a positive answer to this question. Moreover, we show a separation between the notions of "publicly detectable" and "secretly detectable" RCCA. In the later notion, the intentional (malicious) mauling of one ciphertext into another that decrypts to the same plaintext may not be publicly detectable, i.e. detected with the sole knowledge of the public key, but can be detected by the decryptor using the secret decryption key. We start by formalizing the notions discussed above.

Definition 7 *Let $S = (gen, enc, dec)$ be an encryption scheme.*

1. *We say that a family of binary relations $\equiv_e$ (indexed by the public keys of S) on ciphertext pairs is a* compatible *relation for S if for all key-pairs (e, d) of S we have:*
   (a) *For any two ciphertexts c, c', if $c \equiv_e c'$ then $dec_d(c) = dec_d(c')$, except with negligible probability over the random choices of algorithm dec.*
   (b) *For any plaintext m in the domain of S, if c and c' are two ciphertexts obtained as independent encryptions of m (i.e., two applications of algorithm enc on m using independent random bits), then $c \equiv_e c'$ only with negligible probability.*
2. *We say that a relation family as above is* publicly computable *(resp.* secretly computable*) if for all key pairs (e, d) and ciphertext pairs (c, c') it can be determined whether $c \equiv_e c'$ using a probabilistic polynomial time algorithm taking inputs (e, c, c') (resp. (e, d, c, c')).*
3. *We say that S is* publicly-detectable replayable-CCA (pd-RCCA) *if there exists a compatible and publicly computable relation family $\equiv_e$ such that S is secure according to the standard definition of CCA with the following modification to the CCA game from Figure 1: if, after receiving the challenge ciphertext c^*, the adversary queries the decryption oracle on a ciphertext c such that $c \equiv_e c^*$ then the decryption oracle returns* `test`.

   Similarly, we say that S is secretly-detectable replayable-CCA (sd-RCCA) *if the above holds for a secretly computable relation family $\equiv_e$.*

The reader can verify that the notion of pd-RCCA is essentially equivalent to the notions of loose ciphertext-unforgeability, benign malleability, and generalized CCA security, presented, respectively, in [18, 23, 1]. The term "compatible relation" is adapted from [23] where it is defined without item 1(b). Indeed, in the case of "public detectability" requirement 1(b) is redundant as it follows from the semantic security of the encryption scheme. In contrast, this requirement is significant in the case of "secret detectability"; without it this notion is trivially equivalent to RCCA. More significantly, requirement 1(b) captures the need of some applications in which "detectability" is useful provided that the decryptor can tell apart legitimately generated ciphertexts from those that are created by mauling a given (legitimate) ciphertext.

4.2 Relations among notions of detectable RCCA

We investigate the relations among the different flavors of detectable RCCA security, and between these and CCA security. We show:

Theorem 8 *CCA $\Rightarrow$ pd-RCCA $\Rightarrow$ sd-RCCA $\Rightarrow$ RCCA. Furthermore, pd-RCCA secure encryption schemes exist then the first two implications are strict.*

It is easy to see that any encryption scheme that is CCA secure is also pd-RCCA (with the equality relation as the compatible relation). Also, immediate from the definition we get that pd-RCCA implies sd-RCCA. On the other hand, as discussed above, any pd-RCCA scheme can be transformed into a pd-RCCA scheme that is not CCA by appending to the ciphertext a "dummy bit" that is ignored by the decryption operation. Therefore, we get a strict separation between CCA and pd-RCCA security. In the rest of this subsection we show that sd-RCCA security implies RCCA security (Claim 9), and establish a separation between the class of pd-RCCA and sd-RCCA secure encryption schemes (Claims 10 and 11). Whether RCCA and sd-RCCA are equivalent remains open – see Remark 12.

Claim 9 *For any encryption scheme S, if S is sd-RCCA secure, then S is IND-RCCA secure.*

Proof: Let $\mathcal{A}$ be an IND-RCCA adversary that interacts with scheme S. We construct an sd-RCCA adversary $\mathcal{A}'$ that is successful with essentially the same probability as $\mathcal{A}$. Adversary $\mathcal{A}'$ runs a copy of $\mathcal{A}$ and follows the instructions of $\mathcal{A}$ with the following exception. When $\mathcal{A}$ asks to decrypt a ciphertext c and the response is one of the two test messages, then $\mathcal{A}'$ provides $\mathcal{A}$ with the response `test`.

The view of $\mathcal{A}$ when run by $\mathcal{A}'$ differs only negligibly from its view in a true IND-RCCA game. This is so since in the sd-RCCA game $\mathcal{A}'$ gets a `test` response for a ciphertext c that does not decrypt to one of the test messages only when the relation $\equiv_e$ violates the compatibility requirement with respect to c and the test ciphertext, and this occurs only with negligible probability. $\Box$

Separation between pd-RCCA and sd-RCCA We provide two separating examples between pd-RCCA and sd-RCCA. The first one (Claim 10) only assumes that there exist sd-RCCA schemes. The second one (Claim 11) is essentially the "double encryption" example from the introduction, where the outer encryption is taken to be the El-Gamal encryption scheme. This example relies on a number-theoretic assumption (the Decisional Diffie-Hellman assumption), but can be regarded as somewhat more natural. In addition, it points to another interesting issue, randomizability of RCCA-secure encryption, which is presented below.

Claim 10 *If there exists sd-RCCA secure encryption schemes, then there exists an encryption scheme which is sd-RCCA secure and which is not pd-RCCA secure.*

Proof: Let $S = (gen, enc, dec)$ be an sd-RCCA secure encryption scheme and consider the following encryption scheme $S' = (gen', enc', dec')$. The key generation algorithm runs gen twice to obtain key-pairs (e_0, d_0) and (e_1, d_1). The public key is (e_0, e_1). The encryption algorithm is given by $enc'_{e_0,e_1}(m) = (enc_{e_0}(m), enc_{e_1}(0))$. The decryption algorithm on input (c_0, c_1) outputs `invalid` if $dec_{d_1}(c_1) = 1$ and otherwise outputs $dec_{d_0}(c_0)$. (We remark that the instance (e_1, d_1) can be replaced by any scheme that is semantically secure against chosen plaintext attacks. It is chosen to be an instance of S for simplicity only.)

We first show that scheme S' is sd-RCCA secure. Define a compatible relation for S' as follows. Let $\equiv_{e,d}$ be a compatible relation for S given by Def. 7. Consider the following relation $\equiv_{e_0,e_1,d_0,d_1}$ for S': $(c_0, c_1) \equiv_{e_0,e_1,d_0,d_1} (c'_0, c'_1)$ if either $dec_{d_1}(c_1) = dec_{d_1}(c'_1) = 1$, or $dec_{d_1}(c_1) \neq 1$ and $dec_{d_1}(c'_1) \neq 1$ and $c_0 \equiv_{e_0,d_0} c'_0$. It is straightforward to verify that $\equiv_{e_0,e_1,d_0,d_1}$ is compatible with S' whenever $\equiv_{e,d}$ is compatible with S. Notice in particular that if e.g. $dec_{d_1}(c_1) = 1$ and $dec_{d_1}(c'_1) \neq 1$, then $(c_0, c_1) \not\equiv_{e_0,e_1,d_0,d_1} (c'_0, c'_1)$ as required by the definition of compatibility. (This is required since $dec'_{d_0,d_1}(c_0, c_1) = \texttt{invalid}$ and $dec'_{d_0,d_1}(c'_0, c'_1) \in D_k$ and consequentially $dec_{d_0,d_1}(c_0, c_1) \neq dec'_{d_0,d_1}(c'_0, c'_1)$).

Now, let $\mathcal{A}'$ be an adversary that interacts with S' with relation $\equiv_{e_0,e_1,d_0,d_1}$. We construct an adversary $\mathcal{A}$ that interacts with scheme S and relation $\equiv_{e,d}$, and succeeds with essentially the same probability as S'. Given public key e_0 of scheme S, Adversary $\mathcal{A}$ generates a new pair of keys (e_1, d_1) of S, and invokes a copy of $\mathcal{A}'$ with public key $(e = e_0, e_1)$. When $\mathcal{A}'$ provides the test messages m_0, m_1, $\mathcal{A}$ forwards m_0, m_1 to its oracle, obtains a ciphertext c, and hands $\mathcal{A}'$ the ciphertext $(c, E_{e_1}(0))$. When $\mathcal{A}'$ generates a decryption query (c_0, c_1), $\mathcal{A}$ decrypts c_1. If c_1 decrypts to 1 then it answers `invalid` to $\mathcal{A}'$; otherwise, $\mathcal{A}$ asks its own decryption oracle on c_0 and forwards the answer to $\mathcal{A}'$. Finally $\mathcal{A}$ outputs whatever$\mathcal{A}'$ outputs. The validity of $\mathcal{A}$ is straightforward. (Here we only use the fact that the instance (e_1, d_1) decrypts messages correctly.)

It remains to show that scheme S' is not pd-RCCA secure. Assume for the sake of contradiction that S' is pd-RCCA secure and let $\equiv_{e_0,e_1}$ be the compatible publicly computable relation guaranteed by Def. 7. We first assert the following two facts:

1. Let m be a message in the domain of S, let $c_0 = E_{e_0}(m)$, and let c_1, c'_1 be two independently generated encryptions of 0 with public key e_1. Then $(c_0, c'_1) \equiv_{(e_0,e_1)} (c_0, c_1)$ except with negligible probability. To see that this holds, consider the following adversary $\mathcal{A}$ for the pd-RCCA game with S'. $\mathcal{A}$ receives the public key (e_0, e_1) and picks arbitrary test messages m_0 and m_1 where $m_0 \neq m_1$. When it receives the challenge ciphertext $c^* = (c_0, c_1)$ it computes $c'_1 = enc_{e_1}(0)$ and computes whether $(c_0, c'_1) \equiv_{(e_0,e_1)} (c_0, c_1)$. If $(c_0, c'_1) \equiv_{(e_0,e_1)} (c_0, c_1)$, then it outputs a random guess. Otherwise it queries the decryption oracle on (c_0, c'_1), obtains m_b and outputs b. In the second case $\mathcal{A}$ wins the game, so by the assumption that S' is pd-RCCA secure it follows that $(c_0, c'_1) \equiv_{(e_0,e_1)} (c_0, c_1)$ except with negligible probability.

2. Let m be a message in the domain of S, let $c_0 = E_{e_0}(m)$, and let $c_1 = E_{e_1}(0)$ and let $c'_1 = E_{e_1}(1)$. Then $(c_0, c'_1) \equiv_{(e_0,e_1)} (c_0, c_1)$ only with negligible probability. This follows from the compatibility of $\equiv_{(e_0,e_1)}$ and the facts that (c_0, c'_1) decrypts to `invalid` and (c_0, c_1) decrypts to m.

It is now easy to use the relation $\equiv_{(e_0,e_1)}$ to construct an adversary $\mathcal{A}$ that breaks the semantic security of S against chosen plaintext attacks. Specifically, given a public key e_1 and a ciphertext c which is either an encryption of 0 or an encryption of 1, $\mathcal{A}$ generates keys (e_0, d_0) of S and ciphertexts $c' = E_{e_1}(0)$ and $c_0 = E_{e_0}(m)$ for some m, and outputs '0' iff $(c_0, c) \equiv_{(e_0,e_1)} (c_0, c')$. □

Publicly randomizable encryption. An encryption scheme is publicly randomizable if it is possible, given a public key e and a ciphertext c, to generate another ciphertext c' such that c' decrypts to the same value v as c, and in addition c' is indistinguishable to an external observer from $E_e(v)$ (i.e., from an independently generated encryption of v). Of course, publicly randomizable schemes can never be pd-RCCA, since the existence of a compatible relation contradicts the randomizability. Claim 11 below demonstrates that, under the Decisional Diffie-Hellman assumption, there exist sd-RCCA schemes that are publicly randomizable.

Claim 11 *If there exists an sd-RCCA secure encryption scheme and there exists an instantiation of the ElGamal encryption scheme[12] that is semantically secure against chosen-plaintext attacks (CPA), then there exists an encryption scheme which is sd-RCCA secure and is not pd-RCCA.*

Proof: Let $S = (gen, enc, dec)$ be an sd-RCCA secure encryption scheme and consider the following encryption scheme $S' = (gen', enc', dec')$. The key generation algorithm runs gen to obtain a key-pair (e, d) and then generates a key-pair $((g, h = g^x), x)$ for a semantically secure instantiation of the ElGamal encryption scheme. The encryption algorithm is given by $enc'_{e,g,h}(m) = (g^r, enc_e(m)h^r)$ for $r \in_R \mathbf{Z}_{\mathrm{ord}(g)}$ and decryption is given by $dec'_{d,x}(\alpha, \beta) = dec_d(\beta\alpha^{-x})$.

Scheme S' publicly randomizable. Specifically, given a public key (e, g, h) and a ciphertext $c = (\alpha, \beta)$, choose r at random from the underlying group and compute $c' = (\alpha \cdot g^r, \beta \cdot h^r)$. It is easy to see that c' decrypts to the same value as c, and that under the DDH assumption c' is indistinguishable from a random pair of elements in G.

We next prove that S' is sd-RCCA secure. In fact, we prove a more general claim: Given any sd-RCCA secure encryption scheme $S = (G, E, D)$, and a scheme $S'' = (G'', E'', D'')$ that is semantically secure against chosen plaintext attacks, the composed scheme $S' = (G', E', D')$ where G' runs $(e, d) = G()$, $e'', d'') = G''()$ and outputs $((e, e''), (d, d''))$, $E'_{e,e''}(m) = E''_{e''}(E_e(m))$, and $D''_{d,d''}(c) = D_d(D''_{d''}(c))$, is sd-RCCA secure. Let $\equiv_{(e,d)}$ be the compatible relation for S given by Def. 7. Then define a compatible relation $\equiv_{(e,d,e'',d'')}$ for S' by $c_0 \equiv_{(e,d,e'',d'')} c_1$ iff $D''_{d''}(c_0) \equiv_{e,d} D''_{d''}(c_1)$. It is easy to verify that $\equiv_{(e,d,e'',d'')}$ is compatible (with S') when $\equiv_{(e,d)}$ is compatible (with S). So, to prove that S' is sd-RCCA secure consider any adversary F' for the sd-RCCA game with S' and $\equiv_{(e,d,e'',d'')}$ and consider the following adversary F for the sd-RCCA game with S and $\equiv_{e,d}$. First F receives the key e and generates a random key-pair (e'', d'') for S'', hands (e, e'') to F' and starts running F'. If F' requests a decryption of c, then F requests a decryption of $D''_{d''}(c)$ and hands the result to F'. When F' specifies test messages m_0 and m_1, then F specifies the same test messages, receives a challenge ciphertext c^* and hands $E''_{e''}(c^*)$ to F'. When F' makes a guess, F makes the same guess. It is straight forward to verify that the probability that F wins the sd-RCCA game with S and $\equiv_{e,d}$ is exactly the probability that F' wins the RCCA game with S' and $\equiv_{e,d,e'',d''}$, which proves the claim. $\square$

Remark 12 (RCCA vs. sd-RCCA) The above results leave open the question of whether one can separate between sd-RCCA and RCCA (i.e., if there exists an RCCA encryption scheme which is not sd-RCCA). A related "meta question" is whether there are natural applications where sd-RCCA suffices but plain RCCA does not. (These would be applications where the decryptor, and only the decryptor, wishes to know whether a given ciphertext was "honestly generated.")

Another interesting related question is whether there exist RCCA-secure encryption schemes which are secretly randomizable. A secretly randomizable scheme is a publicly randomizable one, with the additional requirement that the generated ciphertext c' remain indistinguishable from a random encryption of the same value, *even when the decryption key is known.*

4.3 From RCCA security to CCA security

This section demonstrates that the existence of an RCCA secure public-key encryption scheme with large message domain implies the existence of a CCA secure public-key encryption scheme. To be precise, by large we will mean that the encryption scheme can encrypt messages of length k, where k is the security parameter.

The construction consists of two steps. First we recall that the existence of any secure encryption scheme implies the existence of a CCA secure *symmetric* encryption scheme. We then show how to combine an RCCA secure public-key encryption scheme with large domain and a CCA secure symmetric encryption scheme to obtain a CCA secure public-key encryption scheme. The first step is very inefficient, whereas the second step results in an efficient encryption scheme if the RCCA secure public-key encryption scheme and the CCA secure symmetric encryption scheme are both efficient.

For the first step, if an RCCA secure public-key encryption scheme (gen, enc, dec) exists, then a one-way function exists, e.g. the function $(e, d) = gen(r)$, where r is the random bits used in generating (e, d). Now, the existence of a one-way function through a series of well-known reductions implies the existence of a CCA secure symmetric encryption scheme (E, D) encrypting unbounded length messages.[10] To prepare for the second step, let $l(k)$ denote the key-length of (E, D) as a function of the security parameter k and consider the public-key encryption scheme $(\overline{gen}, enc, dec)$ given by

$$\overline{gen}(k) = gen(1^{\max(k,l(k))}) \ .$$

Clearly $(\overline{gen}, enc, dec)$ is RCCA secure if (gen, enc, dec) is RCCA secure. Furthermore, $(\overline{gen}, enc, dec)$ can encrypt messages of length $l(k)$. In the following we will therefore assume that we have access to a CCA secure symmetric encryption scheme (E, D) with key-length l and an RCCA secure public-key encryption scheme (gen, enc, dec) capable of encrypting messages of length l.

Consider then the public-key encryption scheme $(gen, \overline{enc}, \overline{dec})$ given by

$$\overline{enc}_e(m) = [K \overset{\text{R}}{\leftarrow} \{0,1\}^{l(k)}; c_1 = enc_e(K); c_2 = E_K(c_1 \| m) : (c_1, c_2)] \ ,$$

$$\overline{dec}_d(c_1, c_2) = [K \leftarrow dec_d(c_1); c_1' \| m = D_K(c_2); \text{if } c_1' \neq c_1 \text{ then } m \leftarrow \texttt{invalid} : m] \ .$$

This construction of $(gen, \overline{enc}, \overline{dec})$ resembles the usual extension of a CCA secure public-key encryption scheme with a CCA secure symmetric encryption scheme for doing hybrid encryption (as described in Section 5.4). The only difference is the encryption of c_1 under the symmetric key. This encryption of c_1 functions as a MAC which protects against 'mauling' of c_1. Indeed, if one is not interested in hybrid encryption but only in obtaining CCA security, the encryption of m could be done as $c_1 = enc_e(K \| m); c_2 = mac_K(c_1)$, where mac is a strong message authentication code. The proof would follow the same lines as the proof of the following theorem.

Theorem 13 *If (E, D) is a CCA secure symmetric encryption scheme with key-length l and (gen, enc, dec) is an RCCA secure public-key encryption scheme capable of encrypting messages of length l, then the public-key encryption scheme $(gen, \overline{enc}, \overline{dec})$ is CCA secure.*

Proof: For all adversaries F let $\texttt{success}_{(gen,\overline{enc},\overline{dec})}(F)$ denote the probability that F wins the CCA game against $(gen, \overline{enc}, \overline{dec})$. We are going to prove that $\texttt{success}_{(gen,\overline{enc},\overline{dec})}(F)$ is negligibly close to $\frac{1}{2}$.

Consider first the following slightly modified CCA game:

1. Run $(e, d) \leftarrow gen(k)$ and give e to F.

   Then generate uniformly random $K' \in \{0,1\}^{l(k)}$.

[10] I.e., the encryption scheme itself does not contain any bound on the message-length. However, under attack by a given adversary F the length of the messages encrypted is of course bounded by a polynomial as F is required to be PPT.

2. When F outputs $(\texttt{test messages}, m_0, m_1)$, where $m_0 \neq m_1$, compute $c_1^* = enc_e(K^*)$ for uniformly random $K^* \in \{0,1\}^{l(k)}$. Then compute $c_2^* = E_{K'}(c_1^* \| m_b)$ for uniformly random $b \in \{0,1\}$ and give $c^* = (c_1^*, c_2^*)$ to F.

3. When F outputs $(\texttt{ciphertext}, (c_1, c_2))$, if $(c_1, c_2) = (c_1^*, c_2^*)$ then output $\texttt{test}$. Otherwise, compute $K = dec_d(c_1)$. If $K = K^*$, then let $K = K'$. Then compute $c_1' \| m = D_K(c_2)$. If $c_1' = c_1$, then output m, otherwise output $\texttt{invalid}$.

4. When F outputs $(\texttt{guess}, b')$, the output of the game is determined as follows. If $b' = b$ then output 1. Otherwise, output 0.

Let $\texttt{success}'_{(gen,\overline{enc},\overline{dec})}(F)$ denote the expected output of the game. We are going to prove that for all adversaries F we have that $|\texttt{success}'_{(gen,\overline{enc},\overline{dec})}(F) - \frac{1}{2}|$ is negligible. We do this by reducing to the CCA security of (E, D). Given any adversary F for the slightly modified CCA game we construct a CCA adversary F' attacking (E, D) as follows:

1. Run $(e, d) \leftarrow gen(k)$ and give e to F.

   We have access to a CCA game for the symmetric encryption scheme (E, D).

   Let K' denote the uniformly random key $K' \in \{0,1\}^{l(k)}$ generated by that CCA game.

2. When F outputs $(\texttt{test messages}, m_0, m_1)$, compute $c_1^* = enc_e(K^*)$ for uniformly random $K^* \in \{0,1\}^{l(k)}$. Then let $M_0 = c_1^* \| m_0$, let $M_1 = c_1^* \| m_1$ and output $(\texttt{test messages}, M_0, M_1)$ to receive $c_2^* = E_{K'}(c_1^* \| m_b)$ for a uniformly random $b \in \{0,1\}$. Then give $c^* = (c_1^*, c_2^*)$ to F.

3. When F outputs $(\texttt{ciphertext}, (c_1, c_2))$, if $(c_1, c_2) = (c_1^*, c_2^*)$ then input $\texttt{test}$ to F. Otherwise, compute $K = dec_d(c_1)$.

   If $K \neq K^*$, then compute $c_1' \| m = D_K(c_2)$. If $c_1' = c_1$, then output m, otherwise, output $\texttt{invalid}$.

   If $K = K^*$ we would like to decrypt c_2 under the key K' defined in Step 1. We know that $c_1 \neq c_1^*$ or $c_2 \neq c_2^*$. If $c_2 \neq c_2^*$, then output c_2 to the CCA game to receive $c_1' \| m = D_{K'}(c_2)$. If $c_1' = c_1$, then output m, otherwise, output $\texttt{invalid}$. If $c_2 = c_2^*$, then $c_1 \neq c_1^*$. Since $c_2 = c_2^* = E_{K'}(c_1^* \| m_b)$ it therefore follows that (c_1, c_2) is an invalid ciphertext. Therefore input $\texttt{invalid}$ to F.[11]

4. When F outputs $(\texttt{guess}, b')$, output $(\texttt{guess}, b')$.

It follows by inspection that the expected value of $\texttt{success}'_{(gen,\overline{enc},\overline{dec})}(F)$ is exactly the probability that F' wins the CCA game against (E, D),[12] which by assumption is negligibly close to $\frac{1}{2}$.

It is therefore enough now to prove that $|\texttt{success}_{(gen,\overline{enc},\overline{dec})}(F) - \texttt{success}'_{(gen,\overline{enc},\overline{dec})}(F)|$ is negligible for all adversaries F. We are going to do this by a reduction to the RCCA security of (gen, enc, dec). For any CCA adversary F attacking $(gen, \overline{enc}, \overline{dec})$ we construct a RCCA adversary F' attacking (gen, enc, dec) as follows:

[11]This is the point in the proof where we use that c_1 is encrypted under the symmetric key, as we would not be able to return the plaintext m_b to F.

[12]Actually, negligibly close to, as we allow the encryption schemes to decrypt incorrectly with a negligible probability. This opens the possibility that the an incorrect $(c_1 \neq c_1^*, c_2^*)$ is decrypted by the RCCA game, whereas in the above game F' always return $\texttt{invalid}$ to F.

1. We assume that F' has access to a RCCA game for (gen, enc, dec).

   We start be receiving e from the RCCA game.

   We then generate uniformly random $K' \in \{0,1\}^{l(k)}$.

2. When F outputs $(\texttt{test messages}, m_0, m_1)$, where $m_0 \neq m_1$ then output $(\texttt{test messages}, K', K^*)$ for uniformly random $K^* \in \{0,1\}^{l(k)}$ to obtain some ciphertext c_1^*. Then compute $c_2^* = E_{K'}(c_1^* \| m_{b'})$ for uniformly random $b' \in \{0,1\}$ and give $c^* = (c_1^*, c_2^*)$ to F.

3. When F outputs $(\texttt{ciphertext}, (c_1, c_2))$, if $(c_1, c_2) = (c_1^*, c_2^*)$ then input $\texttt{test}$ to F. Otherwise, output $(\texttt{ciphertext}, c_1)$ to the RCCA game to get some value K. If $K = \texttt{test}$, then let $K = K'$. Then compute $c_1' \| m = D_K(c_2)$. If $c_1' = c_1$, then input m to F, otherwise input $\texttt{invalid}$ to F.

4. When F outputs $(\texttt{guess}, b')$, F' outputs 1 iff $b' = b$. Otherwise, output 0.

If $c_1^* = enc_e(K^*)$, then the expected value of $\texttt{success}_{(gen,enc,dec)}(F')$ is exactly $\texttt{success}'_{(gen,\overline{enc},\overline{dec})}(F)$.

If on the other hand $c_1^* = enc_e(K')$, then the expected value of $\texttt{success}_{(gen,enc,dec)}(F')$ is exactly $\texttt{success}_{(gen,\overline{enc},\overline{dec})}(F)$, unless it happens that F outputs $(\texttt{ciphertext}, (c_1, c_2))$ where $dec_d(c_1) = K^*$. However, when $c_1^* = enc_e(K')$ the view of F is stochastically independent of K^* until F outputs $(\texttt{ciphertext}, (c_1, c_2))$ where $dec_d(c_1) = K^*$. Since K^* is chosen uniformly at random from $\{0,1\}^{l(k)}$ and F is PPT in k it follows that the probability that F outputs $(\texttt{ciphertext}, (c_1, c_2))$ where $dec_d(c_1) = K^*$ is negligible.

This immediately implies that it is enough to prove that the expected value of $\texttt{success}_{(gen,enc,dec)}(F')$ when $b = 0$ is negligibly close to the expected value of $\texttt{success}_{(gen,enc,dec)}(F')$ when $b = 1$. This follows immediately from the RCCA security of (gen, enc, dec). □

5 Using RCCA security

This section demonstrates the power of RCCA security by proving its sufficiency for several core applications of public key encryption. Prior proofs for the security of these applications relied on the CCA security (or, in some cases, on the pd-RCCA security) of the underlying encryption schemes.

Remark. While RCCA security is adequate for most typical encryption applications, one cannot consider it as a "drop-in" replacement for *all* applications of CCA. As pointed out in the introduction, if an application makes decisions based on the ciphertext strings themselves (e..g compares them), rather than just using the ciphertexts as inputs to the decryption algorithm, then replacing CCA with RCCA may not be secure. It is indeed unusual that such examination of the ciphertext strings is performed by applications, yet this cannot be discounted. An example is a voting scheme in which (malicious) duplicate votes are detected via ciphertext comparison.

5.1 RCCA suffices for non-interactive secure communication

We demonstrate that RCCA security (with sufficiently large message domain) suffices for realizing non-interactive secure communication. This is done using the UC characterization of RCCA security. More specifically, we recall the protocol, presented in [7], for realizing the secure communication functionality $\mathcal{F}_{\text{M-SMT}}$ given access to $\mathcal{F}_{\text{PKE}}$ and authenticated communication. (In other words, this protocol, denoted σ, operates in the $(\mathcal{F}_{\text{PKE}}, \mathcal{F}_{\text{AUTH}})$-hybrid model.) Protocol σ is non-interactive in the sense that transmitting an encrypted message does not require the recipient to send any messages. This stands in contrast with secure communication protocols that are based on key exchange followed by symmetric encryption.

We demonstrate that protocol σ continues to realize $\mathcal{F}_{\text{M-SMT}}$ even when the underlying encryption scheme is only guaranteed to realize $\mathcal{F}_{\text{RPKE}}$. (In other words, we show that protocol σ realizes $\mathcal{F}_{\text{M-SMT}}$ even in the $(\mathcal{F}_{\text{RPKE}}, \mathcal{F}_{\text{AUTH}})$-hybrid model.)

We first recall functionality $\mathcal{F}_{\text{M-SMT}}$; See Figure 4.[13] $\mathcal{F}_{\text{M-SMT}}$ allows multiple parties to send multiple messages to a single recipient, while guaranteeing the secrecy and integrity of each encrypted message independently of all others.

Functionality $\mathcal{F}_{\text{M-SMT}}$

$\mathcal{F}_{\text{M-SMT}}$ proceeds as follows, running on security parameter k, with parties $P_1, ..., P_n$ and an adversary $\mathcal{S}$.

1. In the first activation, expect to receive a value $(\texttt{receiver}, id)$ from some party P_i. Then, send $(\texttt{receiver}, id, P_i)$ to the adversary. From now on, ignore all $(\texttt{receiver}, id)$ values.
2. Upon receiving a value $(\texttt{send}, id, m)$, with $m \in \{0,1\}^k$, from some party P_j, send (id, P_j, m) to P_i and (id, P_j) to the adversary.

Figure 4: The multi-message, multi-sender secure transmission functionality, $\mathcal{F}_{\text{M-SMT}}$

Recall the protocol σ from [7] for realizing $\mathcal{F}_{\text{M-SMT}}$ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{PKE}})$-hybrid model. (Having access to $\mathcal{F}_{\text{AUTH}}$ means that we assume ideally authenticated communication. As seen below, this is essential for this solution.) When invoked with input $(\texttt{receiver}, id)$ within party P_i, protocol σ invokes $\mathcal{F}_{\text{PKE}}$ with input $(\texttt{KeyGen}, id)$, obtains the public key e and sends (id, e) to all parties. When activated within party P_j with incoming message $(P_i, P_j, (id, e))$ from $\mathcal{F}_{\text{AUTH}}$, party P_j records the triple (P_i, id, e). From now on, when activated within P_j with input $(\texttt{send}, id, m)$, the protocol invokes $\mathcal{F}_{\text{PKE}}$ with input $(\texttt{Encrypt}, id, e, (P_j, m))$, obtains a ciphertext c, and invokes $\mathcal{F}_{\text{AUTH}}$ to send (id, c) from P_j to P_i.[14] Finally, when activated within P_i with incoming message $(\texttt{ciphertext}, id, c)$ from P_j, the protocol invokes $\mathcal{F}_{\text{PKE}}$ with input $(\texttt{Decrypt}, id, c)$, obtains the decryption m', and verifies that m' is of the form $m' = (P_j, m)$. If the verification succeeds, then the protocol outputs m. Otherwise it outputs nothing.

We stress that the sender incorporates its own identity in the encrypted message, and that the receiver verifies that the identity in the decrypted message agrees with the identity of the sender. This precaution is used to prevent copying of messages: Without it, a corrupted party could copy a ciphertext sent by an uncorrupted party, and re-send it as coming from itself. Such copying of messages is not allowed by $\mathcal{F}_{\text{M-SMT}}$.

[13] For convenience, use a formulation of $\mathcal{F}_{\text{M-SMT}}$ which is slightly different from the one in [7] in that we only allow fixed-size messages.

[14] For convenience we will assume that the party identifiers, P_j, are elements from $\{0,1\}^k$ and that the domain ensemble of $\mathcal{F}_{\text{PKE}}$ is $\{0,1\}^{2k}$, to allow encrypting the message (P_j, m).

We now show that running protocol σ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{RPKE}})$-hybrid model has the same effect as running this protocol in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{PKE}})$-hybrid model. (There are no syntactic problems here since the interface of $\mathcal{F}_{\text{RPKE}}$ with the uncorrupted parties is syntactically the same as the interface of $\mathcal{F}_{\text{PKE}}$.)

Claim 14 *Protocol σ securely realizes $\mathcal{F}_{\text{M-SMT}}$ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{RPKE}})$-hybrid model.*

Proof (sketch): Intuitively, we claim that the relaxation of $\mathcal{F}_{\text{PKE}}$ to $\mathcal{F}_{\text{RPKE}}$ (i.e., allowing the adversary to generate new ciphertexts that decrypt to the same values as existing ciphertexts) has no effect in the context of protocol σ. This is so since each party includes its own identity in each ciphertext. Thus, assume that some uncorrupted party P_i generates a ciphertext c that decrypts to (P_i, m), and that some corrupted party manages to generate another ciphertext c' that decrypts to the same plaintext. Then, when P_j sends c', the decryptor will reject c' since it contains the wrong sender identity.

We repeat the proof in [7] with the necessary corrections. Let $\mathcal{A}$ be an adversary that operates against parties running σ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{RPKE}})$-hybrid model. We construct an ideal process adversary $\mathcal{S}$ such that no environment can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and σ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{RPKE}})$-hybrid model or with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\text{M-SMT}}$. Adversary $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$, and proceeds as follows.

1. Whenever $\mathcal{S}$ is activated with an input value v from $\mathcal{Z}$, it activates the simulated $\mathcal{A}$ with input v. Whenever $\mathcal{A}$ generates an output value v, $\mathcal{S}$ writes v on its output tape.

2. *Key generation:* Initialize a flag $g = 0$. Then:

   (a) *Uncorrupted decryptor:* When $\mathcal{S}$ receives a value $(\texttt{receiver}, id, P_i)$ from $\mathcal{F}_{\text{M-SMT}}$, it checks the value of g. If $g = 1$ then $\mathcal{S}$ does nothing. If $g = 0$ then $\mathcal{S}$ sets $g = 1$ and activates $\mathcal{A}$ with input $(\texttt{KeyGen}, id)$ coming from $\mathcal{F}_{\text{RPKE}}$. When $\mathcal{A}$ returns a value e, $\mathcal{S}$ starts a series of activations of $\mathcal{A}$: for each party P_j, $\mathcal{S}$ activates $\mathcal{A}$ with a value $(P_i, P_j, (id, e))$ coming from $\mathcal{F}_{\text{AUTH}}$. (Recall that this value means that P_i sent a message (id, e) to P_j.

   (b) *Corrupted decryptor:* When the simulated $\mathcal{A}$ activates $\mathcal{F}_{\text{RPKE}}$ with input $(\texttt{KeyGen}, id)$ in the name of a corrupted party P_i, $\mathcal{S}$ checks the value of g. If $g = 1$ then $\mathcal{S}$ does nothing. If $g = 0$ then $\mathcal{S}$ sets $g = 1$, returns $(\texttt{KeyGen}, id)$ to $\mathcal{A}$, and obtains a value e from $\mathcal{A}$. Next, $\mathcal{S}$ has P_i send a value $(\texttt{receiver}, id, P_i)$ to $\mathcal{F}_{\text{M-SMT}}$. Finally, when the simulated $\mathcal{A}$ activates $\mathcal{F}_{\text{AUTH}}$ with a value $(P_i, P_j, (id, e))$, $\mathcal{S}$ activates $\mathcal{A}$ with a value $(P_i, P_j, (id, e))$ coming from $\mathcal{F}_{\text{AUTH}}$.

3. *Delivery of the public key:* When the simulated $\mathcal{A}$ activates some uncorrupted party P_j to receive a value $(P_i, P_j, (id, e'))$ coming from $\mathcal{F}_{\text{AUTH}}$, $\mathcal{S}$ activates P_j to receive the message $(\texttt{receiver}, id, P_i)$ from $\mathcal{F}_{\text{M-SMT}}$.

4. *Encryption by an uncorrupted party:* When $\mathcal{S}$ receives a value (id, P_j) from $\mathcal{F}_{\text{M-SMT}}$, it checks whether the value e' that P_j received in Step 3 equals the value e obtained in Step 2. If $e' \neq e$ then $\mathcal{S}$ does nothing. (The rationale is that in this case P_i is bound to reject the message and output nothing.) If $e' = e$ then $\mathcal{S}$ activates $\mathcal{A}$ with input $(\texttt{Encrypt}, id, e)$ coming from $\mathcal{F}_{\text{RPKE}}$. When $\mathcal{A}$ returns a value c, $\mathcal{S}$ records the pair (P_j, c) and activates $\mathcal{A}$ with a value $(P_j, P_i, (id, c))$ coming from $\mathcal{F}_{\text{AUTH}}$. (If the value c is already recorded then send an error message to $\mathcal{A}$, just like $\mathcal{F}_{\text{RPKE}}$ would.)

5. *Encryption by a corrupted party:* When the simulated $\mathcal{A}$ activates $\mathcal{F}_{\text{RPKE}}$ with input $(\texttt{Encrypt}, id, e', m')$ in the name of some corrupted party P_i, and $e' = e$, then $\mathcal{S}$ returns $(\texttt{Encrypt}, id, e')$ to $\mathcal{A}$ and obtains c from $\mathcal{A}$. It then records (P_j, c, m'). If $m' = (P_j, m)$ then $\mathcal{S}$ activates $\mathcal{F}_{\text{M-SMT}}$ with a value $(\texttt{send}, id, m)$ from P_j. (If $e' \neq e$ then $\mathcal{S}$ hands $\mathcal{A}$ also the value m', and does not record (P_j, c, m').) When the simulated $\mathcal{A}$ activates $\mathcal{F}_{\text{AUTH}}$ with a value $(P_j, P_i, (id, c))$, $\mathcal{S}$ activates $\mathcal{A}$ with the value $(P_j, P_i, (id, c))$ sent from $\mathcal{F}_{\text{AUTH}}$.

6. *Decryption by an uncorrupted decryptor:* When $\mathcal{A}$ invokes party P_i to receive a message $(P_j, P_i, (id, c))$ from P_j, $\mathcal{S}$ proceeds as follows.

   (a) If the pair (P_j, c) has been recorded (i.e., if P_j has sent a message that was associated with the ciphertext c) then activate P_i to receive the corresponding message from $\mathcal{F}_{\text{M-SMT}}$.

   (b) If $(P_{j'}, c)$ has been recorded for $j' \neq j$ then do nothing. (The rationale is that in this case P_i will reject c and output nothing.)

   (c) If c was never recorded then hand c to $\mathcal{A}$ (in the name of $\mathcal{F}_{\text{RPKE}}$) for decryption. If $\mathcal{A}$ returns a value ('ciphertext', c') and the pair (P_j, c') is recorded, then activate P_i to receive the corresponding message from $\mathcal{F}_{\text{M-SMT}}$. Otherwise, do nothing.

7. *Decryption by a corrupted decryptor:* When the simulated $\mathcal{A}$ activates $\mathcal{F}_{\text{RPKE}}$ with input $(\texttt{Decrypt}, id, c)$ in the name of a corrupted decryptor P_i, $\mathcal{S}$ behaves as $\mathcal{F}_{\text{RPKE}}$ would. That is, if the pair (P_j, c) was recorded for some P_j then $\mathcal{S}$ hands the plaintext (P_j, m) associated with c to $\mathcal{A}$. ($\mathcal{S}$ knows m since $\mathcal{F}_{\text{M-SMT}}$ has sent (id, P_j, m) to the corrupted P_i in the ideal process.) Otherwise, $\mathcal{S}$ hands $(\texttt{Decrypt}, id, c)$ to $\mathcal{A}$. If $\mathcal{A}$ returns ('plaintext', m) then hand m to $\mathcal{A}$. If $\mathcal{A}$ returns ('ciphertext', c') where (P_j, c') is recorded, then $\mathcal{S}$ hands the plaintext m associated with c' to $\mathcal{A}$.

It can be verified that the view of $\mathcal{Z}$ in an interaction with $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\text{M-SMT}}$ is distributed *identically* to the view of $\mathcal{Z}$ in an interaction with $\mathcal{A}$ and σ in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{RPKE}})$-hybrid model. This holds even for computationally unbounded $\mathcal{Z}$. $\Box$

5.2 RCCA suffices for key-exchange and authenticators

Here we illustrate the power of RCCA security by showing that it suffices to implement secure key exchange protocols based on public key encryption. Specifically, we show that the encryption-based key-exchange protocol, ENC, from [9] can be proved secure on the assumption that the underlying encryption scheme is RCCA-secure. This same protocol is shown in [9] to be secure on the basis of the stronger assumption that the encryption scheme is CCA secure.

We recall the encryption based protocol ENC from [9]. Let (gen, enc, dec) be a key-generation, encryption and decryption algorithm, respectively, of a public-key encryption scheme. Let k be the security parameter. Assume that each party P_i has invoked $gen(k)$ to get a pair (e_i, d_i) of encryption and decryption keys, and all parties have the public encryption key e_i of the other parties. The encryption algorithm enc_e is assumed to work on the message space $\{0,1\}^k$ where k is the security parameter. In addition, let $\{f_\kappa\}_{\kappa \in \{0,1\}^k}$ be a pseudorandom function family (as in [14]). The protocol, denoted ENC, is described in Figure 5.

The following Theorem is a strengthening of Theorem 9 from [9] where the security of ENC is claimed on the basis of the CCA security of the encryption scheme (gen, enc, dec).

Protocol ENC

Proceed as follows, given security parameter k.

Step 1: The initiator, P_i, on input (P_i, P_j, s), chooses $\kappa \overset{R}{\leftarrow} \{0,1\}^k$ and sends $(P_i, s, enc_{e_j}(\kappa))$ to P_j. Next, P_i outputs the session key $\sigma = f_\kappa(P_i, P_j, s)$ under session ID s.

Step 2: Upon receipt of (P_i, s, c) the responder, P_j, computes $\kappa' = dec_{d_j}(c)$. If the decryption algorithm does not reject the ciphertext as invalid, then P_j outputs the session key $\sigma' = f_{\kappa'}(P_i, P_j, s)$ under session ID s.

Figure 5: A KE protocol based on public-key encryption.

Theorem 15 *If the encryption (gen, enc, dec) is* RCCA-secure *and the family $\{f_\kappa\}_{\kappa \in \{0,1\}^k}$ is pseudorandom, then protocol* ENC *is SK-secure (without forward secrecy) in the authenticated links model (*AM*).*

For background on the definition of SK-security as a notion of secure key-exchange protocols the reader is referred to [9]. We note that the AM is an idealized model of a perfectly-authenticated network. In order to convert the protocol ENC into a secure key exchange protocol in the realistic UM model (where all communication links are controlled by an active attacker) one can apply to ENC any *authenticator* [2]. See [9] for the details.

Fortunately, for proving the above theorem we do not need to get into the model and details of key-exchange protocols. Indeed, [9] prove this theorem by reducing its security to that of an "encryption game" they define. It is shown that if protocol ENC can be broken as a key-exchange protocol then the game can be won (with non-negligible advantage) by a feasible attacker. On the other hand, they prove that no such successful attacker against this game can exist if the encryption scheme is CCA secure. All we have to show is that *the same* encryption game cannot be won by a feasible attacker if the encryption scheme is *RCCA secure*. This is proven next after introducing the encryption game in Figure 6.

Lemma 16 *Assume that the encryption scheme (gen, enc, dec) is* RCCA-secure *and the family $\{f_\kappa\}_{\kappa \in \{0,1\}^k}$ is pseudorandom. Then if $\mathcal{B}$ runs in polynomial-time and never includes the value t^* as the t component in the pairs (c, t) queried during Phases 1 and 3, then the probability that $\mathcal{B}$ wins in the above game is no more than 1/2 plus a negligible fraction.*

Note: in [9] the condition restricting the queries of $\mathcal{B}$ in phases 1 and 3 is more relaxed: namely, that $\mathcal{B}$ is not allowed to query the specific pair (c^*, t^*) in these phases. However, it is easy to verify that the proof of SK-security in [9] remains correct under the more stringent restriction in the above Lemma.

Proof: We show how to build a successful IND-RCCA attacker $\mathcal{R}$ against the scheme (gen, enc, dec) based on an attacker $\mathcal{B}$ that wins the above game with non-negligible advantage, namely, with probability $1/2 + \varepsilon$ for some non-negligible function ε.

1. $\mathcal{R}$ is given a public key e.
2. $\mathcal{R}$ chooses two random and independent keys κ_0 and κ_1 from $\{0,1\}^k$, outputs $(\texttt{test messages}, \kappa_0, \kappa_1)$ and receives a challenge ciphertext c^*.

The encryption game

The parties to the game are $\mathcal{G}$ and $\mathcal{B}$ (for good and bad). $\mathcal{G}$ possesses a pair of public and private keys, e and d (generated via the key generation algorithm *gen*). $\mathcal{B}$ knows e but not d.
The game proceeds in phases:

Phase 0: $\mathcal{G}$ provides $\mathcal{B}$ with a challenge ciphertext $c^* = enc_e(\kappa_0)$ for $\kappa_0 \overset{\text{R}}{\leftarrow} \{0,1\}^k$.

Phase 1: $\mathcal{B}$ sends a pair (c, t) to $\mathcal{G}$ who responds with $f_\kappa(t)$ where $\kappa = dec_d(c)$. This is repeated a polynomial (in k) number of times with each pair being chosen adaptively by $\mathcal{B}$ (i.e., after seeing $\mathcal{G}$'s response to previous pairs).

Phase 2: $\mathcal{B}$ sends a test string t^* to $\mathcal{G}$. Then $\mathcal{G}$ chooses a random bit $b \overset{\text{R}}{\leftarrow} \{0,1\}$. If $b = 0$ then $\mathcal{G}$ responds with $f_{\kappa_0}(t^*)$ where κ_0 is the value encrypted by $\mathcal{G}$ in phase 0. If $b = 1$ then $\mathcal{G}$ responds with a random string r of the same length as $f_{\kappa_0}(t^*)$.

Phase 3: Same as Phase 1.

Phase 4: $\mathcal{B}$ outputs a bit b'.

And the winner is... $\mathcal{B}$ if and only if $b = b'$.

Figure 6: A game that captures the SK-security of the key-exchange protocol ENC

3. Phase 0: $\mathcal{R}$ presents $\mathcal{B}$ with public key e and challenge ciphertext c^*.
4. Phase 1: upon query (c, t) from $\mathcal{B}$, $\mathcal{R}$ outputs $(\texttt{ciphertext}, c)$ and receives back a value κ. If $\kappa = \texttt{test}$ then $\mathcal{R}$ responds to $\mathcal{B}$ with $f_{\kappa_0}(t)$, otherwise $\mathcal{R}$ responds with $f_\kappa(t)$.[15]
5. Phase 2: upon presentation of a test string t^* by $\mathcal{B}$, $\mathcal{R}$ chooses a bit $b \in_R \{0,1\}$ and answers $\mathcal{B}$ with the value $f_{\kappa_0}(t^*)$ if $b = 0$, and with a random value r of the length of $f_{\kappa_0}(t^*)$ if $b = 1$.
6. Phase 3: $\mathcal{R}$ responds to $\mathcal{B}$'s queries in phase 3 exactly as in phase 1.
7. Phase 4: when $\mathcal{B}$ outputs a bit b', $\mathcal{R}$ outputs 0 if $b' = b$ and outputs 1 otherwise.

We finish the proof of Lemma 16 by proving the following claim.

Claim 17 *If in the above run of $\mathcal{R}$, its encryption oracle chooses*

1. $c^* = enc_e(\kappa_0)$ *then the probability that $\mathcal{B}$ guesses b is $1/2 + \varepsilon - q2^{-k}$ where q is the number of queries made by $\mathcal{B}$.*
2. $c^* = enc_e(\kappa_1)$ *then the probability that $\mathcal{B}$ guesses b is at most $1/2 + \varepsilon_{prf} + 2^{-k}$ where ε_{prf} is the distinguishing probability of some feasible attacker against the family $\{f_\kappa\}_{\kappa\in\{0,1\}^k}$ and so is negligible.*

From this claim the Lemma follows since the advantage of $\mathcal{R}$ is

$$|Prob(\mathcal{R} \text{ outputs } 0 \ : \ c^* = enc_e(\kappa_0)) - Prob(\mathcal{R} \text{ outputs } 0 \ : \ c^* = enc_e(\kappa_1))|$$

[15]Here and in the sequel we will assume for simplicity that if the response from the decryption oracle is $\kappa = \texttt{invalid}$ then $f_\kappa(t)$ also returns `invalid`.

$$= |Prob(\mathcal{B} \text{ guesses } b \ : \ c^* = enc_e(\kappa_0)) - Prob(\mathcal{B} \text{ guesses } b \ : \ c^* = enc_e(\kappa_1))|$$
$$\geq |1/2 + \varepsilon - q2^{-k} - 1/2 - \varepsilon_{prf} - 2^{-k}| = |\varepsilon - \varepsilon_{prf} - (q+1)2^{-k}|$$

which is non-negligible since we assumed ε to be non-negligible (while ε_{prf} and $(q+1)2^{-k}$ are negligible).

We proceed to prove the claim.

Proof of 1: In this case, the inputs from $\mathcal{R}$ to $\mathcal{B}$ have the same probability distribution than those that the oracle $\mathcal{G}$ would have provided to $\mathcal{B}$ under a regular run of the game with keys e and d, except if in the interaction between $\mathcal{B}$ and $\mathcal{R}$ the former asks a query in which $c = enc_e(\kappa_1)$. In this case $\mathcal{R}$ answers $f_{\kappa_0}(t)$ (since its decryption oracle returns `test`) while $\mathcal{G}$ would have answered $f_{\kappa_1}(t)$. The probability of this event, however, is 2^{-k} for each such query since the value κ_1 is random and independent from the view of $\mathcal{B}$. Thus, the probability of $\mathcal{B}$ to win the game is at least $1/2 + \varepsilon - q2^{-k}$ where q is the number of queries made by $\mathcal{B}$.

Proof of 2: Assume that when $c^* = enc_e(\kappa_1)$, $\mathcal{B}$ wins with probability $1/2+\varepsilon'$. We show how to build a distinguisher $\mathcal{F}$ against the family $\{f_\kappa\}_{\kappa \in \{0,1\}^k}$ that succeeds with advantage $\varepsilon_{prf} = \varepsilon' - 2^{-k}$ and thus $\varepsilon' = \varepsilon_{prf} + 2^{-k}$ as claimed. The distinguisher $\mathcal{F}$ has access to a function oracle f^*, implemented either by a random member of the pseudorandom family, or by a completely random function of the same output length. The idea is for $\mathcal{F}$ to simulate a run of $\mathcal{R}$ and $\mathcal{B}$ in which $c^* = enc_e(\kappa_1)$, but to replace the responses of $\mathcal{R}$ of the form $f_{\kappa_0}(t)$ with responses $f^*(t)$ received from its function oracle.

Specifically, $\mathcal{F}$ starts by generating a pair of public/private keys (e, d) under scheme (gen, enc, dec), and producing a ciphertext $c^* = enc_e(\kappa_1)$ for a randomly chosen element $\kappa_1 \in_R \{0,1\}^k$. It then activates $\mathcal{B}$ with public key e and challenge ciphertext c^*. $\mathcal{B}$'s queries (c, t) in phases 1 and 3 are answered by $\mathcal{F}$ by first computing $\kappa = dec_d(c)$ and then responding with $f^*(t)$ if $\kappa = \kappa_1$, and with $f_\kappa(t)$ otherwise. When $\mathcal{B}$ sends its test string t^* in phase 2, $\mathcal{F}$ chooses a random bit b and responds as follows. If $b = 0$ then $\mathcal{F}$ returns $f^*(t^*)$, else $\mathcal{F}$ returns a random r. Finally, $\mathcal{F}$ outputs the same bit b' as $\mathcal{B}$ does.

We claim that if f^* is a pseudorandom function f_{κ_0} (for random $\kappa_0 \in \{0,1\}^k$) then the behavior of $\mathcal{B}$ in the interaction with $\mathcal{F}$ is identical to its behavior in the game with $\mathcal{R}$ when $c^* = enc_e(\kappa_1)$ as long as the value κ_1 chosen by $\mathcal{F}$ is different than the key κ_0 used by the pseudorandom function f^* (a condition that fails with only negligible probability 2^{-k}). Indeed, for any query (c, t) by $\mathcal{B}$ in which c decrypts to a value something different than κ_0 or κ_1 the response of both $\mathcal{F}$ and $\mathcal{R}$ is $f_\kappa(t)$. While if c decrypts to either κ_0 or κ_1 (the "test" case under $\mathcal{R}$), the response of both $\mathcal{R}$ and $\mathcal{F}$ is $f_{\kappa_0}(t)$ (note that in both cases κ_0 is random over $\{0,1\}^k \setminus \{\kappa_1\}$). Thus, in this case, $\mathcal{B}$ guesses the bit b correctly with probability $1/2 + \varepsilon' - 2^{-k}$.

If f^* uses a random function then the answers to the test string t^* under $b = 0$ and $b = 1$ are both random and independent from any other value in the game (here we are using the fact that $t^* \neq t$ for all other values t queried during the game). Thus, in this case, $\mathcal{B}$ guesses b correctly with probability 1/2.

In all, we get that the advantage of $\mathcal{F}$ to distinguish a random f^* from a randomly chosen pseudorandom function f^* is $\varepsilon_{prf} = 1/2 + \varepsilon' - 2^{-k} - 1/2 = \varepsilon' - 2^{-k}$. □

Remark. A very similar proof as that of Lemma 16 can be used to prove the encryption-based authenticator from [2].

5.3 RCCA suffices for encrypted password authentication

Password authentication is very common in practice. Its typical application is for authenticating a (human) user sitting at some remote client machine (a terminal, laptop, etc.) to a centralized authentication server. The most common weakness of password authentication mechanisms is its vulnerability to dictionary attacks in which an eavesdropper can use the authentication information sent over the network to derive the user's password by exhaustive search over a given password dictionary of relatively small size (say, one million passwords which with significant probability includes the user's password). Methods for avoiding dictionary attacks against password-only attacks exist ([5] and the many subsequent papers in this area). One particular setting which is widespread in practice is one in which the server has a public encryption key e and a corresponding private decryption key d for some public-key encryption scheme S. The user (or, more typically, its client machine) has the public key e (either e is already stored at the client, or it is sent from the server together with a certificate that the client can verify). Both server and user share a secret password (usually memorized by the user).

This setting is prevalent in the Web (where the password information is usually sent under the protection of a server-authenticated SSL tunnel), in IPsec's remote access, in SSH (secure shell), etc.

This setting was formally studied in [16] who prove that the simple encryption of the password (with other user/server identification information) and some fresh information (such as a unique session id) under a public key encryption scheme ensures secure password authentication provided that the encryption scheme is CCA-secure. More generally, they show that a wider family of protocols, named Encrypted Challenge-Response protocols, provides secure one-way password authentication[16] if the encryption is CCA-secure. Their notion of secure password authentication guarantees that the attacker's best strategy is the trivial (and unavoidable) one: trying all possible passwords in *active* impersonation attacks against the server until the corresponding user's password is found (this is not only time consuming for the attacker but well designed systems will invalidate the attacked user's password after a relatively small number of failed authentication trials).

Here we show that the results from [16] hold also if the encryption scheme is RCCA (without requiring any change to the protocols from [16]). For the full details on the notion of password authentication security, and the Encrypted Challenge-Response protocol and its derivatives, the reader is referred to [16]. We have:

Theorem 18 *The Encrypted Challenge-Response protocol from [16] provides secure one-way password authentication if the public key encryption scheme is IND-RCCA secure.*

For stating and proving their result [16] define a notion of public-key encryption security against *ciphertext-verification attacks* which is implied by CCA security but is strictly stronger than RCCA. This notion is presented below and we will refer to it as IND-CVA. The proof of the above theorem consists of proving that a relaxed notion, that we call IND-RCVA, and which is implied by IND-RCCA, is sufficient to guarantee the security of the Encrypted Challenge-Response protocol. We proceed to present first the definition of IND-CVA from [16] and then introduce our relaxed variant IND-RCVA.

[16] "one-way authentication" refers to the fact that these protocols provide client authentication only. [16] also shows how to expand these protocols to provide mutual authentication and authenticated key exchange.

Definition 19 (IND-CVA [16]) *Let $S = (gen, enc, dec)$ be a public-key encryption scheme. A* ciphertext-verification attack *is defined via the following game, which involves the scheme S and an adversary A.*

1. *The key-generation algorithm is run (with security parameter k) to generate a secret/public key pair, (d, e). The adversary A is given the public key e.*
2. *The adversary adaptively generates queries (x_i, c_i), which we call* verification queries*. For each verification query (x_i, c_i) the adversary is told whether or not $x_i = dec_d(c_i)$.*
3. *The adversary generates a pair of plaintexts m_0, m_1 of the same length, and asks for an encryption of one of them. Below we call this the* test query *of A. With probability $1/2$, A gets an answer $c = enc_e(m_0)$, and with probability $1/2$ it gets $c = enc_e(m_1)$.*
4. *The adversary may adaptively generate more verification queries (x_i, c_i), subject to the constraint that $c_i \neq c$. Again, for each verification query (x_i, c_i), the adversary is told whether or not $x_i = dec_d(c_i)$.*
5. *The adversary A guesses whether c is an encryption of m_0 or of m_1.*
   We say that A wins the game *if it guesses correctly.*

An encryption scheme S is said to be secure against ciphertext-verification attacks (IND-CVA) *if the probability of any polynomial-time adversary A to win the above game is no more that $1/2$ plus a negligible fraction.*

Our relaxation of this notion is as follows

Definition 20 (IND-RCVA) *Let $S = (gen, enc, dec)$ be an encryption scheme as above. A* relaxed ciphertext-verification attack *is defined via the game in Definition 19 with the following modification. The constraint in Step 4 is changed such that the attacker is not allowed to query pairs (x_i, c_i) where $x_i \in \{m_0, m_1\}$. The scheme S is called* secure against relaxed ciphertext-verification attacks (IND-RCVA) *if the probability of any polynomial-time adversary A to win the game is no more that $1/2$ plus a negligible fraction.*

Proof of Theorem 18: Since IND-RCVA is implied by IND-RCCA it suffices to show that any efficient successful attacker against the security of the Encrypted Challenge-Response protocol (as a one-way authentication protocol) can be converted into an IND-RCVA attacker against S. The proof is almost identical to the proof in [16]. It only requires a slight modification in the argument of the proof of Lemma 5 of that paper which builds an IND-CVA attacker A against the encryption scheme S given an attacker I that breaks the Encrypted Challenge-Response protocol. The required observation in our setting is that the attacker A built in [16] is *already* an IND-RCVA attacker, namely, it never issues a verification query with a plaintext equal to one of the test messages. This can be seen by inspecting the description of the attacker A in the proof of Lemma 5 in [16]. (For the benefit of the interested reader, the specific points to observe in the description of A in [16] are that the test messages chosen in step (2) of A's run are of the form $w_t = f(\mathsf{spwd}\ ; \mathsf{U}, \mathsf{S}, \mathsf{sid}'_t)$ and $\bar{w}_t = f(0; 0)$. The verification queries issued in step (3) are of the form $z_i = f(\mathsf{spwd}'; \mathsf{U}', \mathsf{S}, \mathsf{sid}')$ with $\mathsf{U}' \neq \mathsf{U}$ and therefore different from w_t and $\bar{w}_t$ (here we use the fact that f is assumed to be "one-to-one on its components"). The one additional verification query from step (4) is $z = f(\mathsf{spwd}\ ; \mathsf{U}, \mathsf{S}, \mathsf{sid})$ which is also different from w_t and $\bar{w}_t$ due to the fact that $\mathsf{sid}'_t \neq \mathsf{sid}$.) □

Remark. In [16] the basic Encrypted Challenge-Response protocol is extended to provide mutual authentication and key exchange also using public-key encryption mechanisms. In [16] these protocols are based on CCA security, however we note that they can be shown to hold under RCCA security using the results from Section 5.2.

5.4 RCCA hybrid encryption

Public key encryption is often too slow for encrypting long messages. As a consequence the regular way to encrypt long messages (e.g. email) uses the so called hybrid encryption method (also called the "envelope method") in which a PK encryption scheme is used to encrypt a symmetric key and this key used to encrypt the message under a symmetric-key encryption scheme. Hybrid encryption schemes can be made to be CCA-secure if the PK encryption scheme in use is CCA-secure. One well-known method is as follows. Let $\textsc{penc}$ be a public-key encryption scheme, $\textsc{senc}$ be a symmetric-key encryption scheme, and $\textsc{mac}$ be a MAC algorithm. For any plaintext m define $\textsc{eem}(m) = (c_1, c_2, c_3)$ where $c_1 = \textsc{penc}_e(\kappa_1, \kappa_2), c_2 = \textsc{senc}_{\kappa_1}(m), c_3 = \textsc{mac}_{\kappa_2}(c_2)$ where κ_1, κ_2 are random keys (chosen by the encryptor) for the schemes $\textsc{senc}$ and $\textsc{mac}$, respectively. If $\textsc{penc}$ is CCA-secure, $\textsc{senc}$ is semantically secure against chosen-plaintext attacks (CPA), and $\textsc{mac}$ is a strong[17] MAC function, then $\textsc{eem}$ is CCA-secure. Here we show that the scheme $\textsc{eem}$ constitutes an RCCA-secure public-key encryption scheme provided that $\textsc{penc}$ is RCCA-secure, $\textsc{senc}$ is secure against chosen-plaintext attacks, and $\textsc{mac}$ is a regular secure MAC function (i.e. no need for the "strong" property). Actually, we will show a more general result:

Theorem 21 *If* $\textsc{penc}$ *is an RCCA-secure public-key encryption scheme and* $\textsc{senc}$ *an IND-RCCA secure symmetric encryption scheme then the hybrid scheme* $\textsc{henc}$ *defined as* $\textsc{henc}(m) = (\textsc{penc}(k), \textsc{senc}_k(m))$, *where* k *is a random key for the scheme* $\textsc{senc}$, *is RCCA-secure (as a public-key encryption scheme).*

Note that this Theorem uses the notion of IND-RCCA security for symmetric encryption. Next we sketch such a definition as a simple adaptation of the IND-RCCA notion from Section 3.2 to the symmetric-key setting. The only difference with the public key case is that in the symmetric key scenario the attacker is not provided with an encryption key but with access to an encryption oracle. As in the public-key case the attacker also has access to a decryption oracle. Both oracles provide their responses using the same (random and unknown to the attacker) key; in particular, on input a ciphertext produced by the encryption oracle on message m, the decryption oracle will output m. The definition of IND-RCCA remains unchanged from the public key case except for the addition of the encryption oracle. There are no restriction on the messages that the attacker can query from the encryption oracle or the decryption oracle. The test message query from the attacker is answered by the encryption oracle by returning the ciphertext of one of the two test messages (chosen at random by the oracle). After a test message query is issued by the attacker, any ciphertext presented to the decryption oracle that decrypts to one of the test messages is answered with `test`. The attacker wins the game if it guesses correctly which test message was encrypted.

A simple example of an IND-RCCA secure symmetric encryption scheme is given by the "encrypt-then-authenticate" paradigm: it is easy to show that the symmetric encryption scheme $EtA_{(\kappa_1,\kappa_2)}(m) = (c_1, c_2)$, where $c_1 = \textsc{senc}_{\kappa_1}(m)$ and $c_2 = \textsc{mac}_{\kappa_2}(c_1)$ is IND-RCCA secure if $\textsc{senc}$ is secure against chosen-plaintext attacks (CPA), and $\textsc{mac}$ is a regular (not necessarily strong) secure

[17]The term "strong MAC" refers to a strengthening of the regular notion of MAC security, namely, in addition to the standard security notion against chosen message attacks, it is required that given pairs of messages and their mac values, it is infeasible to find a different valid mac value for any of these messages.

MAC function. Thus, the above theorem proves, in particular, that the hybrid public-key encryption scheme EEM discussed above is RCCA-secure if PENC is RCCA-secure, SENC is CPA-secure, and MAC a (regularly) secure MAC function.

Proof (sketch): The proof of Theorem 21 is very similar in spirit to the proof of Lemma 16. We describe the main elements of the proof here. Let $\mathcal{H}$ be an adversary that wins the IND-RCCA game against the scheme HENC with non-negligible advantage ε_H. We build an IND-RCCA adversary $\mathcal{P}$ which wins the IND-RCCA game against the public-key encryption scheme PENC with the same advantage up to a negligible difference. We assume that the encryption SENC uses keys of length k, and that PENC is capable of encrypting strings of length k.

1. $\mathcal{P}$ is given a public key e.
2. $\mathcal{P}$ chooses two random and independent keys κ_0 and κ_1 from $\{0,1\}^k$, outputs $(\texttt{test messages}, \kappa_0, \kappa_1)$ and receives a challenge ciphertext c^*.
3. $\mathcal{P}$ runs $\mathcal{H}$ with public key e.
4. Upon ciphertext query (c,t) from $\mathcal{H}$, $\mathcal{P}$ outputs $(\texttt{ciphertext}, c)$ and receives back a value κ. If $\kappa = \texttt{test}$ then $\mathcal{P}$ responds to $\mathcal{H}$ with $\textsc{senc}_{\kappa_0}^{-1}(t)$, otherwise $\mathcal{P}$ responds with $\textsc{senc}_{\kappa}^{-1}(t)$ (if $\kappa = \texttt{invalid}$ or $\textsc{senc}_{\kappa}^{-1}(t) = \texttt{invalid}$ then $\mathcal{P}$ returns $\texttt{invalid}$).
5. Upon presentation of a pair of test messages m_0, m_1 by $\mathcal{H}$, $\mathcal{P}$ chooses a bit $b \in_R \{0,1\}$ and answers $\mathcal{H}$ with the ciphertext $(c^*, \textsc{senc}_{\kappa_0}(m_b))$.
6. When $\mathcal{H}$ outputs a bit b', $\mathcal{P}$ outputs 0 if $b' = b$ and outputs 1 otherwise.

We finish the proof of the theorem by proving the following claim.

Claim 22 *If in the above run of $\mathcal{P}$, its encryption oracle chooses*

1. $c^* = \textsc{penc}_e(\kappa_0)$ *then the probability that $\mathcal{H}$ guesses b is $1/2 + \varepsilon_H - q2^{-k}$ where q is the number of ciphertext queries made by $\mathcal{H}$.*
2. $c^* = \textsc{penc}_e(\kappa_1)$ *then the probability that $\mathcal{H}$ guesses b is at most $1/2 + \varepsilon_s + 2^{-k}$ where ε_s is the advantage of some feasible attacker against the IND-RCCA security of scheme* SENC.

From this claim the Theorem follows since the advantage of $\mathcal{P}$ is

$$\begin{aligned}
&|Prob(\mathcal{P} \text{ outputs } 0 \ : \ c^* = \textsc{penc}_e(\kappa_0)) - Prob(\mathcal{P} \text{ outputs } 0 \ : \ c^* = \textsc{penc}_e(\kappa_1))| \\
&\qquad = |Prob(\mathcal{H} \text{ guesses } b \ : \ c^* = \textsc{penc}_e(\kappa_0)) - Prob(\mathcal{H} \text{ guesses } b \ : \ c^* = \textsc{penc}_e(\kappa_1))| \\
&\qquad \geq |1/2 + \varepsilon_H - q2^{-k} - 1/2 - \varepsilon_s - 2^{-k}| = |\varepsilon_H - \varepsilon_s - (q+1)2^{-k}|
\end{aligned}$$

which is non-negligible since we assumed ε_H to be non-negligible and ε_s to be negligible.

Thus it only remains to prove the claim which (again) is an adaptation of the proof of Claim 17.

<u>Proof of 1</u>: In this case, the inputs from $\mathcal{P}$ to $\mathcal{H}$ have (almost) the same probability distribution than those that would have been generated by the oracles of $\mathcal{H}$ under a regular run of its IND-RCCA game against the HENC scheme with keys e and d. The only exception is if in the interaction between $\mathcal{H}$ and $\mathcal{P}$ the former asks a ciphertext query (c,t) in which $c = \textsc{penc}_e(\kappa_1)$. In this case $\mathcal{P}$ answers $\textsc{senc}_{\kappa_0}^{-1}(t)$ (since its decryption oracle returns $\texttt{test}$) while a real HENC decryption oracle would have answered $\textsc{senc}_{\kappa_1}^{-1}(t)$. The probability of this event, however, is 2^{-k} for each such query

since the value κ_1 is random and independent from the view of $\mathcal{H}$. Thus, the probability of $\mathcal{H}$ to win the game in its interaction with $\mathcal{P}$ is at least $1/2 + \varepsilon_H - q2^{-k}$ where q is the number of queries made by $\mathcal{H}$.

Proof of 2: Assume that when $c^* = \text{PENC}_e(\kappa_1)$, $\mathcal{H}$ wins with probability $1/2 + \varepsilon'$. We show how to build a IND-RCCA attacker $\mathcal{S}$ against the symmetric scheme SENC that succeeds with advantage $\varepsilon' - 2^{-k}$ and thus $\varepsilon' \le \varepsilon_s + 2^{-k}$ as claimed. The attacker $\mathcal{S}$ has access to a pair of encryption and decryption oracles for SENC, denoted $\mathcal{O}_{enc}$ and $\mathcal{O}_{dec}$, respectively. The idea is for $\mathcal{S}$ to simulate a run of $\mathcal{P}$ and $\mathcal{H}$ in which $c^* = \text{PENC}_e(\kappa_1)$, but to replace the responses of $\mathcal{P}$ of the form $\text{SENC}^{-1}_{\kappa_0}(t)$ with $\mathcal{O}_{dec}(t)$ (i.e., the result returned by the decryption oracle of $\mathcal{S}$ on input t).

Specifically, $\mathcal{S}$ starts by generating a pair of public/private keys (e, d) under scheme PENC, and producing a ciphertext $c^* = \text{PENC}_e(\kappa_1)$ for a randomly chosen element $\kappa_1 \in_R \{0,1\}^k$. It then activates $\mathcal{H}$ with public key e. $\mathcal{H}$'s ciphertext queries (c, t) are answered by $\mathcal{S}$ by first decrypting c using the private key d to obtain a plaintext κ, and then responding with $\mathcal{O}_{dec}(t)$ if $\kappa = \kappa_1$, and with $\text{SENC}^{-1}_{\kappa}(t)$ otherwise (we omit the straightforward details of dealing with invalid ciphertexts). When $\mathcal{H}$ presents its test messages m_0, m_1, $\mathcal{S}$ uses these as its own test messages with $\mathcal{O}_{enc}$. Then, the response from $\mathcal{S}$ to $\mathcal{H}$'s test messages is the ciphertext (c^*, t^*), where t^* denotes the response from $\mathcal{O}_{enc}$ to $\mathcal{S}$'s test. Finally, $\mathcal{S}$ outputs the same bit b' as $\mathcal{H}$ does.

We claim that the behavior of $\mathcal{H}$ in the interaction with $\mathcal{S}$ is identical to its behavior in the game with $\mathcal{P}$ when $c^* = \text{PENC}_e(\kappa_1)$ as long as the value κ_1 chosen by $\mathcal{S}$ is different than the key, which we denote by κ_0, used by the oracles $\mathcal{O}_{enc}$ and $\mathcal{O}_{dec}$ (a condition that fails with only negligible probability 2^{-k}). Indeed, for any query (c, t) by $\mathcal{H}$ in which c decrypts to a value κ different than κ_0 or κ_1 the response of both $\mathcal{S}$ and $\mathcal{P}$ is $\text{SENC}^{-1}_{\kappa}(t)$. While if c decrypts to either κ_0 or κ_1 (the `test` case under $\mathcal{P}$), the response of both $\mathcal{P}$ and $\mathcal{S}$ is $\text{SENC}^{-1}_{\kappa_0}(t)$ (in the former case κ_0 is the key chosen by $\mathcal{P}$, in the later, it is the $\mathcal{O}_{dec}$ key; in both cases κ_0 is random over $\{0,1\}^k \setminus \{\kappa_1\}$). Also, the response to $\mathcal{H}$ for its test query is $(c^*, \text{SENC}_{\kappa_0}(m_b))$ for random b in the interaction with $\mathcal{P}$ and in the interaction with $\mathcal{S}$. Thus, in this case, $\mathcal{H}$ guesses the bit b correctly with probability $\varepsilon_s = 1/2 + \varepsilon' - 2^{-k}$ and so does $\mathcal{S}$. In other words, the advantage of $\mathcal{H}$ is at most $\varepsilon' = \varepsilon_s + 2^{-k}$, where ε_s is negligible as $\mathcal{S}$ is a legitimate IND-RCCA attacker against SENC. □

6 Open Questions

The new notions and results in this paper leave some questions open. Here we mention some.

1. Are NM-RCCA and UC-RCCA equivalent for polynomial-size message spaces?
2. Can RCCA be further relaxed and still provide a useful general notion of security against active attackers?
3. Can we build significantly-more-practical (or simpler, or based on more relaxed specific assumptions, etc) RCCA-secure schemes than CCA-secure ones?
4. Are there schemes that are RCCA but not sd-RCCA?
5. Are there "randomizable RCCA schemes"? (see Remark 12). This question is related to questions 1 and 4. If there exists "randomizable RCCA schemes", then question 4 is answered in the affirmative. If furthermore such a scheme has polynomial-size message space and invalid ciphertexts can be efficiently recognized given just the public key, then question 1 is answered in the negative.

6. Are there "natural" schemes that are sd-RCCA but not pd-RCCA?

References

[1] J.H. An, Y. Dodis, and T. Rabin, "On the Security of Joint Signature and Encryption", in *Eurocrypt '02*, pages 83–107, 2002. LNCS No. 2332.

[2] M. Bellare, R. Canetti and H. Krawczyk, "A modular approach to the design and analysis of authentication and key-exchange protocols", *30th STOC,* 1998.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes", *Advances in Cryptology - CRYPTO'98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998.

[4] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", *Advances in Cryptology - ASIACRYPT'00 Proceedings*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto, ed., Springer-Verlag, 2000.

[5] Bellovin, S. M. and Merritt, M., "Encrypted key exchange: Password- based protocols secure against dictionary attacks", In *Proceedings of the IEEE. Computer Society Symposium on Research in Security and Privacy* 1992, pp. 72–84.

[6] Bleichenbacher, D., "Chosen Ciphertext Attacks against Protocols Based on RSA Encryption Standard PKCS #1", *Advances in Cryptology - CRYPTO'98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998, pp. 1–12.

[7] R. Canetti, "Universally Composable Security: A new paradigm for cryptographic protocols", http://eprint.iacr.org/2000/067. Extended Abstract appears in *42nd FOCS*, 2001.

[8] R. Canetti and S. Goldwasser, "A practical threshold cryptosystem resilient against adaptive chosen ciphertext attacks", *Eurocrypt'99*, 1999.

[9] Canetti, R., and Krawczyk, H., "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", *Eurocrypt 01,* 2001.
Full version in: *Cryptology ePrint Archive* (http://eprint.iacr.org/), Report 2001/040.

[10] R. Cramer and V. Shoup, "A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack", *Advances in Cryptology - CRYPTO'98 Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998.

[11] D. Dolev, C. Dwork and M. Naor, Non-malleable cryptography, *SIAM. J. Computing,* Vol. 30, No. 2, 2000, pp. 391-437. Preliminary version in *23rd Symposium on Theory of Computing (STOC),* ACM, 1991.

[12] T. ElGamal, A Public-Key cryptosystem and a Signature Scheme based on Discrete Logarithms, *IEEE Transactions*, Vol. IT-31, No. 4, 1985, pp. 469–472.

[13] O. Goldreich, "Foundations of Cryptography: Basic Tools", Cambridge Press, 2001.

[14] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM,* Vol. 33, No. 4, 210–217, (1986).

[15] S. Goldwasser and S. Micali, Probabilistic encryption, *JCSS*, Vol. 28, No 2, 1984.

[16] S. Halevi, and H. Krawczyk, "Public-Key Cryptography and Password Protocols", *ACM Transactions on Information and System Security*, Vol. 2, No. 3, August 1999, pp. 230–268.

[17] Dennis Hofheinz and Joern Mueller-Quade and Rainer Steinwandt, "On Modeling IND-CCA Security in Cryptographic Protocols", http://eprint.iacr.org/2003/024. 2003.

[18] H. Krawczyk, "The order of encryption and authentication for protecting communications (Or: how secure is SSL?)", Crypto 2001. http://eprint.iacr.org/2001/045

[19] M. Krohn, "On the definitions of cryptographic security: Chosen-Ciphertext attack revisited," Senior Thesis, Harvard U., 1999.

[20] M. Naor and M. Yung, "Public key cryptosystems provably secure against chosen ciphertext attacks". *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 1990.

[21] Jesper B. Nielsen, " Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case", in M. Yung, editor, *Advances in Cryptology - Crypto 2002*, pages 111–126,Lecture Notes in Computer Science Volume 2442.

[22] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack", *CRYPTO '91*, 1991.

[23] V. Shoup, "A Proposal for an ISO Standard for Public Key Encryption", Crypto Eprint archive entry 2001:112, http://eprint.iacr.org, 2001.

[24] A. Sahai, "Non malleable, non-interactive zero knowledge and adaptive chosen ciphertext security", FOCS 99.