Adaptively-Secure, Non-Interactive Public-Key Encryption*

Ran Canetti[1], Shai Halevi[1], and Jonathan Katz[2]

[1] IBM T.J. Watson Research Center, NY, USA.
[2] Department of Computer Science, University of Maryland.

Abstract. Adaptively-secure encryption schemes ensure secrecy even in the presence of an adversary who can corrupt parties in an adaptive manner based on public keys, ciphertexts, and secret data of already-corrupted parties. Ideally, an adaptively-secure encryption scheme should, like standard public-key encryption, allow arbitrarily-many parties to use a single encryption key to securely encrypt arbitrarily-many messages to a given receiver who maintains only a single short decryption key. However, it is known that these requirements are impossible to achieve: no non-interactive encryption scheme that supports encryption of an unbounded number of messages and uses a single, unchanging decryption key can be adaptively secure. Impossibility holds even if secure data erasure is possible.

We show that this limitation can be overcome by *updating the decryption key over time* and making some mild assumptions about the frequency of communication between parties. Using this approach, we construct adaptively-secure, completely non-interactive encryption schemes supporting secure encryption of arbitrarily-many messages from arbitrarily-many senders. Our schemes additionally provide forward security and security against chosen-ciphertext attacks.

1 Introduction

Imagine a band of political dissidents who need to go into hiding from an oppressive regime. While in hiding, the only form of communication with the outside world is via the public media. Before going into hiding, each individual wants to publish a key that will allow *anyone* (even parties not currently known to this individual) to publish encrypted messages that only this individual can decipher. Since it is not known in advance how long these members will need to be in hiding, reasonably short public keys must suffice for encrypting an unbounded number of messages. Furthermore, messages encrypted to each dissident must remain secret even if other dissidents are caught and their secrets are extracted from them. Do encryption schemes satisfying these requirements exist?

At first glance, a standard public-key encryption scheme seems to suffice. Indeed, a public-key encryption schemes allows a receiver to publish a key that can

* This work was supported by NSF Trusted Computing Grant #0310751 and CyberTrust Grant #0430450.

then be used by anyone to send encrypted messages to the receiver. The public key is *short* (i.e., of fixed polynomial length) and can be used by arbitrary senders (potentially unknown to the receiver at the time the key is published) to securely send arbitrarily-many messages to the receiver *without further interaction.* Furthermore, senders need not maintain any state other than the receiver's public key, and the receiver similarly need not maintain any state except for his secret key.

However, standard public-key encryption schemes do not provide the desired level of security. Standard definitions of security, including semantic security against passive attacks [GM84] as well as various notions of security against active attacks [NY90,RS91,DDN00,BDPR98], only consider the case where the adversary never learns any secret key. However, when an adversary can compromise players and learn their internal states in an *adaptive* manner, possibly depending on previously-observed ciphertexts and information learned during previous corruptions, the standard notions no longer apply. In particular, in the adaptive setting *encrypting with a CCA-secure encryption scheme is not known to provide secure communication.*

To obtain provable security against adaptive adversaries, one must ensure that the information gathered by the adversary when compromising parties (namely, their secret keys) does not give the adversary any additional advantage toward compromising the security of the yet-uncorrupted parties. The standard way of formulating this is by requiring the existence of a *simulator* that can generate "dummy ciphertexts" which can be later "opened" (i.e., by revealing an appropriate secret key) as encryptions of any message; see, e.g., [CFGN96]. A scheme satisfying this additional condition is said to be *adaptively secure.*

Several methods are known for achieving adaptively-secure encrypted communication, but none can be used in the basic setting exemplified by the above toy problem. Beaver and Haber [BH92] propose an adaptively secure encryption protocol in which the sender and receiver must interact before they can securely communicate for the first time. Furthermore, the parties must maintain a shared secret key *per connection.* This key must be continually updated, with the old key being erased, as more messages are encrypted. *Non-committing encryption schemes* [CFGN96,B97,DN00] more closely mimic the functionality of standard public-key encryption, and in particular do not require maintenance of per-connection state. (In addition, these solutions also remove the need for secure data erasure.) In these schemes, however, both the public and secret keys are at least as long as the overall number of bits to be encrypted. In fact, as noted by Nielsen [N02], any adaptively-secure scheme with non-interactive encryption *must* have a decryption key which is at least as long as the number of bits to be decrypted under this key. In a nutshell, this is because the simulator must "open" the "dummy ciphertexts" as encryptions of any given sequence of messages by presenting an appropriate secret key; therefore, the number of possible secret keys must be at least the number of possible message-sequences. The unfortunate conclusion is that a public-key encryption scheme that can encrypt an unbounded number of messages with short and unchanging keys cannot be adaptively secure. This holds even if secure data erasures are possible, and even in a weaker setting where only receivers can be corrupted.

We also comment that previous work on adaptively-secure encryption did not address resistance to chosen-ciphertext attacks.

Our Contributions. This work demonstrates that we can circumvent Nielsen's negative result if the secret decryption key is allowed to periodically change, and some mild assumptions about the frequency of communication between parties are made. That is, under standard hardness assumptions, there exist adaptively-secure, non-interactive public-key encryption schemes with short keys that can handle arbitrarily-many messages and senders. In particular, our schemes solve the toy example from above in a way that is essentially the best possible under the given constraints.

This is done by considering *key-evolving* encryption schemes [CHK03] in which the secret key is locally updated by the receiver according to a globally-known schedule (say, at the end of every day), while the public key remains fixed. The secret key for the previous period is securely *erased* once it is no longer needed. Using this approach, we construct adaptively-secure, non-interactive encryption schemes that can be used to encrypt arbitrarily-many bits as long as the number of encrypted bits (for any particular key) is bounded *per time period*. As discussed above, an assumption of this sort is essential to circumvent Nielsen's negative results. Also, this assumption is reasonable in many cases: for instance, one may easily posit some known upper bound on the number of incoming e-mails processed per day.

In addition to being adaptively secure, our schemes also provide both forward security [A97,CHK03] and security against chosen-ciphertext attacks. (We comment that although forward security is reminiscent of adaptive security, neither security property implies the other.) Accordingly, we refer to schemes satisfying our security requirements as adaptively- and forward-secure encryption (AFSE) schemes. We formalize the requirements for AFSE schemes within the UC framework [C01]. That is, we present an functionality $\mathcal{F}_{\mathrm{AFSE}}$ that captures the desired properties of AFSE schemes. This functionality is a natural adaptation of the "standard" public-key encryption functionality of [C01,CKN03] to the context of key-evolving encryption. As in the non-adaptive case, $\mathcal{F}_{\mathrm{AFSE}}$ guarantees security against active adversaries, which in particular implies security against chosen-ciphertext attacks. Using the composability properties of the UC framework, our constructions are guaranteed to remain secure in any protocol environment. Indeed, the formulation of $\mathcal{F}_{\mathrm{AFSE}}$, which blends together the notions of forward security, chosen-ciphertext security, and adaptive security of public-key encryption schemes, is another contribution of this work.

Techniques and Constructions. We first note that dealing with corruption of senders is easy, since a sender can simply erase its local state upon completing the encryption algorithm. We thus concentrate on the more difficult case of receiver corruption. We then show that it suffices to consider AFSE for the case when only a *single* message is encrypted per time period, since any such construction can be extended in a generic manner to give a scheme which can be used to encrypt any bounded number of messages per time period. With this in mind, our first construction uses the paradigm of Naor-Yung and Sahai [NY90,S99] to construct an AFSE scheme based on any forward-secure encryption (FSE) scheme and any simulation-sound non-interactive zero-knowledge (NIZK) proof system [DDOPS01]. Recall that, under the Naor-Yung/Sahai paradigm, the sender encrypts messages by essentially using two independent copies of a semantically-secure encryption scheme together with an NIZK proof of consistency. To decrypt, the receiver verifies the proof and then decrypts either one of the component ciphertexts. Naor and Yung prove that this provides security against "lunch-time" (i.e., non-adaptive) chosen-ciphertext attacks when an arbitrary NIZK proof system is used, and Sahai later showed that this technique achieves full (i.e., adaptive) CCA-security if a one-time simulation-sound NIZK proof system is used. We show that if a semantically-secure FSE scheme is used as the underlying encryption scheme, and the NIZK proof system is "fully" simulation sound (as defined in [DDOPS01]), the resulting construction is also an AFSE scheme. This approach can be extended to encrypt a polynomial number of bits per ciphertext using only a single NIZK proof. (We remark that, as opposed to the case of standard CCA-secure encryption [S99], here it is not enough that the underlying NIZK is *one-time* simulation sound.)

While the above approach is conceptually simple, it is highly impractical due to the inefficiency of known NIZKs. We thus propose an alternate approach that leads to more efficient solutions based on specific, number-theoretic assumptions. As part of this approach, we first define and construct "standard" (i.e., non key-evolving) encryption schemes which are secure against lunch-time chosen-ciphertext attacks and are adaptively-secure for encryption of a *single* message (in total). We call such schemes *receiver non-committing encryption* (RNCE) schemes.[1] Our construction of an AFSE scheme proceeds by first encrypting the message using any RNCE scheme, and then encrypting the resulting ciphertext using any CCA-secure FSE scheme. Informally, this construction achieves adaptive security for an *unbounded* number of messages (as long as only one message is encrypted per time period) because the secret key of the outer FSE scheme is updated after every period and so the simulator only needs to "open" one ciphertext (i.e., the one corresponding to the current time period) as an arbitrary message. It can accomplish the latter using the "inner" RNCE scheme.

Obtaining an efficient scheme using this approach requires efficient instantiation of both components. Relatively efficient CCA-secure FSE schemes (in particular, schemes which avoid the need for NIZK proofs) are already known [CHK04,BB04]. Therefore, we focus on constructing efficient RNCE schemes based on specific number-theoretic assumptions. Our first RNCE scheme is based on the Cramer-Shoup encryption scheme [CS98] (and adapts techniques of [JL00]) and its security is predicated on the decisional Diffie-Hellman (DDH) assump-

[1] Indeed, this is a relaxation of the notion of non-committing encryption from [CFGN96]. It is similar to the relaxation studied by Jarecki and Lysyanskaya [JL00], except that we also require security against lunch-time chosen-ciphertext attacks.

tion. However, this scheme allows encryption of only a logarithmic number of bits per ciphertext. We also show a second RNCE scheme based on the schemes of [GL03,CS03] (which, in turn, build on [CS02]), whose security relies on the decisional composite residuosity assumption introduced by Paillier [P99] and which can be used to encrypt a polynomial number of bits per ciphertext.

Organization. The AFSE functionality is defined and motivated in Section 2. Our construction of AFSE using the Naor-Yung/Sahai paradigm is described in Section 3. In Section 4, we present definitions for RNCE and show two constructions of RNCE schemes based on specific number-theoretic assumptions. Finally, in Section 5 we construct an AFSE scheme from any RNCE scheme and any CCA-secure FSE scheme. In Appendix A, we include definitions of key-evolving and forward-secure encryption, while a brief overview of the UC framework and its application to secure encryption is provided in Appendix B. In this abstract we omit all proofs due to lack of space. The proofs can be found in the full version of this paper [CHK05].

2 Definition of AFSE

We define AFSE by specifying an appropriate ideal functionality in the UC security framework (cf. Appendix B). This functionality, denoted $\mathcal{F}_{\mathrm{AFSE}}$ and presented in Figure 1, is obtained by appropriately modifying the "standard" public-key encryption functionality $\mathcal{F}_{\mathrm{PKE}}$ [C01,CKN03] which is reviewed in Appendix B.1.

Intuitively, $\mathcal{F}_{\mathrm{AFSE}}$ captures the same security notions as $\mathcal{F}_{\mathrm{PKE}}$ except that it also provides a mechanism by which the receiver can "update" its secret key; $\mathcal{F}_{\mathrm{AFSE}}$ guarantees security only as long as a bounded number of messages are encrypted between key updates. In fact, for simplicity, the functionality as defined only guarantees security when a *single* ciphertext is encrypted between key updates. Say a ciphertext encrypted with respect to a particular time period t is outstanding until the receiver has updated its secret key a total of $t+1$ times. Then, if more than one outstanding ciphertext is requested, the functionality guarantees no security whatsoever for this ciphertext. (Formally, this is captured by handing the corresponding plaintext to the adversary.) Section 2.1 discusses how $\mathcal{F}_{\mathrm{AFSE}}$ can be extended to allow any bounded number of outstanding ciphertexts, which corresponds to ensuring security as long as at most this many messages are encrypted between key updates. It also presents a generic transformation from protocols secure for a single outstanding ciphertext to protocols secure for the general case.

For convenience, we highlight some differences between $\mathcal{F}_{\mathrm{AFSE}}$ and $\mathcal{F}_{\mathrm{PKE}}$. First, an additional parameter — a time period t — is introduced. An encryption request now additionally specifies a time period for the encryption called the "sender time", and the functionality maintains a variable t^* called the "receiver time". The receiver time is initialized to 0, and is incremented by the receiver R^* using an Update request. A ciphertext generated for sender time t is only

Functionality $\mathcal{F}_{\mathrm{AFSE}}$

$\mathcal{F}_{\mathrm{AFSE}}$ proceeds as follows, when parameterized by message domain ensemble $\mathcal{D} = \{D_k\}_{k\in\mathbb{N}}$ and security parameter k.

Key Generation: Upon receiving a request $(\mathtt{KeyGen}, sid)$ from party R^*, do: Verify that $sid = (sid', R^*)$ (i.e., that the identity R^* is encoded in the session ID). If not, then ignore this input. If yes:

1. Hand $(\mathtt{KeyGen}, sid)$ to the adversary.
2. Receive a value pk^* from the adversary, and hand pk^* to R^*. Initialize $t^* \leftarrow 0$ and $\mathsf{messages\text{-}outstanding} \leftarrow 0$.

Encryption: Upon receiving from some party P a tuple $(\mathtt{Encrypt}, sid, \mathsf{pk}, t, m)$ proceed as follows:

1. If $m \in D_k$, $\mathsf{pk} = \mathsf{pk}^*$, and either $t < t^*$ or $\mathsf{messages\text{-}outstanding} = 0$, then send $(\mathtt{Encrypt,}sid, \mathsf{pk}, t, P)$ to the adversary. In all other cases, send $(\mathtt{Dummy\text{-}Encrypt}, sid, \mathsf{pk}, t, m, P)$ to the adversary (i.e., reveal the plaintext to the adversary).
2. Receive a reply c from the adversary and send $(\mathtt{ciphertext,}c)$ to P. In addition, if $m \in D_k$, $\mathsf{pk} = \mathsf{pk}^*$, and $t \geq t^*$, then do:
   (a) If $\mathsf{messages\text{-}outstanding} = 0$, set $\mathsf{messages\text{-}outstanding} \leftarrow 1$ and $\mathsf{flag} \leftarrow \mathsf{outstanding}$. Else, set $\mathsf{flag} \leftarrow \mathsf{dummy}$.
   (b) record the tuple (m, t, c, flag) in the list of ciphertexts.

Decryption: Upon receiving a tuple $(\mathtt{Decrypt}, sid, c)$ from player P, if $P \neq R^*$ then ignore this input. Otherwise:

1. If the list of ciphertexts contains a tuple $(m, t, c, \star)$ with the given ciphertext c and $t = t^*$, then return m to R^*.
2. Otherwise send a message $(\mathtt{Decrypt}, sid, t^*, c)$ to the adversary, receive a reply m, and forward m to R^*.

Update: Upon receiving $(\mathtt{Update}, sid)$ from player P, if $P = R^*$ do:

1. Send a message $(\mathtt{Update}, sid)$ to the adversary.
2. Remove from the list of ciphertexts all the tuples $(m, t^*, c, \mathsf{flag})$ with the current time t^*. If any of these tuple has $\mathsf{flag} = \mathsf{outstanding}$, then reset $\mathsf{messages\text{-}outstanding} \leftarrow 0$.
3. Set $t^* \leftarrow t^* + 1$.

Corruptions: Upon corruption of party P, if $P = R^*$ then send to the adversary all tuples $(m, t, c, \star)$ in the list of ciphertexts with $t \geq t^*$. (If $P \neq R^*$ then do nothing.)

Fig. 1. The AFSE functionality, $\mathcal{F}_{\mathrm{AFSE}}$.

decrypted by $\mathcal{F}_{\mathrm{AFSE}}$ (upon request of the appropriate receiver) when the current receiver time is $t^* = t$.

Second, $\mathcal{F}_{\mathrm{AFSE}}$ limits the information gained by the adversary upon corruption of parties in the system. When corrupting parties other than R^*, the adversary learns nothing. When corrupting R^* at some "receiver time" t^*, the adversary does not learn any information about messages that were encrypted at "sender times" $t < t^*$. (This is akin to the level of security provided by forward-secure encryption schemes, and in fact strengthens the usual notion of adaptive security which potentially allows an adversary to learn all past messages upon corruption of a party.) In addition, *adaptive* security is guaranteed for a single message encrypted at some sender time $t \geq t^*$ (i.e., a single outstanding message).

The fact that security is guaranteed only for a single outstanding message is captured via the variable $\mathsf{messages\text{-}outstanding}$, which is initialized to 0 and is set to 1 when a message is encrypted for time period t with $t \geq t^*$. When the receiver's time unit t^* advances beyond the time unit t of the outstanding ciphertext, the variable $\mathsf{messages\text{-}outstanding}$ is reset to 0. If another encryption request arrives with time period $t \geq t^*$ while $\mathsf{messages\text{-}outstanding}$ is equal to 1, then $\mathcal{F}_{\mathrm{AFSE}}$ *discloses the entire plaintext to the adversary* (and thus does not ensure any secrecy in this case).

We remark that $\mathcal{F}_{\mathrm{AFSE}}$ can be used in a natural way to realize a variant of the "secure message transmission functionality" [C01,AF04] in synchronous networks with respect to adaptive adversaries. We omit further details.

2.1 Handling Multiple Outstanding Ciphertexts

While the functionality $\mathcal{F}_{\mathrm{AFSE}}$ and all the constructions in this work are described assuming a bound of at most one outstanding ciphertext, both the functionality and the constructions can be generalized to the case of any *bounded* number of outstanding ciphertexts (corresponding to a bounded number of messages encrypted per time period). Generalizing the functionality is straightforward, so we do not describe it here. As for constructions, any AFSE scheme which is secure for the case of a single outstanding ciphertext can be extended generically so as to be secure for any bounded number ℓ of outstanding ciphertext in the following way: The public key of the new scheme consists of ℓ independent keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell$ generated using the original scheme. To encrypt a message m, the sender computes the "nested encryption" $E_{\mathsf{pk}_1}(E_{\mathsf{pk}_2}(\cdots E_{\mathsf{pk}_\ell}(m)\cdots))$ and sends the resulting ciphertext to the receiver. One can show that this indeed realizes $\mathcal{F}_{\mathrm{AFSE}}$ for at most ℓ outstanding ciphertexts. The formal proof, however, is more involved and is omitted.

2.2 Realizing $\mathcal{F}_{\mathrm{AFSE}}$ Using Key-Evolving Encryption Schemes

We present our constructions as key-evolving encryption *schemes* (i.e., as a collection of algorithms) rather than as protocols (as technically required by the UC framework). For completeness, we describe the (obvious) transformation from key-evolving encryption schemes to protocols geared toward realizing $\mathcal{F}_{\text{AFSE}}$.

Recall that a key-evolving encryption scheme consists of four algorithms $(\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$, where $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption routines (as in a standard encryption scheme, except that the encryption and decryption routines also take as input a time period t), and Upd is the secret-key update algorithm that takes as input the current secret key and time unit, and outputs the secret key for the next time unit. The definition is reviewed in Appendix A.

Given a key evolving encryption scheme $S = (\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$, one may construct the protocol π_S as follows: An activation of π_S with input message `KeyGen`, `Update`, `Encrypt`, or `Decrypt` is implemented via calls to the algorithms $\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}$, or Dec, respectively. The only state maintained by π_S between activations is the secret key that was generated by Gen (and that is modified in each activation of `Update`), and the current time period. Any other local variables that are temporarily used by any of the algorithms are erased as soon as the activation completes. With this transformation we can now define an AFSE scheme:

Definition 1. *A key-evolving encryption scheme S is an adaptively- and forward-secure encryption (AFSE) scheme if the protocol π_S resulting from the transformation above securely realizes $\mathcal{F}_{\text{AFSE}}$ with respect to adaptive adversaries.*

3 AFSE Based on Forward-Secure Encryption

In this section we show how to construct an AFSE scheme from any FSE scheme secure against chosen-plaintext attacks along with any simulation-sound NIZK proof system. (See Appendix A for definitions of key-evolving encryption and forward security, both against chosen-plaintext and chosen-ciphertext attacks.) We describe in detail a construction that allows encryption of only a single bit per ciphertext and then discuss how this may be generalized to allow for encryption of any polynomial number of bits per ciphertext. Our construction uses a simple twist of the Naor-Yung/Sahai transformation [NY90,S99]; when applied to two FSE schemes, the resulting scheme yields not only CCA security but also security against adaptive corruptions. We comment that, as opposed to the case of non-adaptive CCA security, "one-time" simulation sound NIZK proofs are *not sufficient* to achieve security against adaptive corruptions; instead, we require NIZK proofs satisfying the stronger notion of unbounded simulation soundness [DDOPS01].

The construction. Let $\mathcal{E}' = (G', U', E', D')$ be a key-evolving encryption scheme, and let $\mathcal{P} = (\ell, P, V)$ be an NIZK proof system (where $\ell(k)$ is the length of the common random string for security parameter k) for the following NP language

$$\begin{aligned} L_{\mathcal{E}'} \stackrel{\text{def}}{=} \{&(t, \mathsf{pk}'_0, c'_0, \mathsf{pk}'_1, c'_1) \ : \\ &\exists\, m, r_0, r_1 \text{ s.t. } c'_0 = E'(\mathsf{pk}'_0, t; m; r_0),\ c'_1 = E'(\mathsf{pk}'_1, t; m; r_1)\}. \end{aligned}$$

We construct a new key-evolving encryption scheme $\mathcal{E} = (G, U, E, D)$ as follows:

Key generation, G. On security parameter 1^k, run two independent copies of the key generation algorithm of $\mathcal{E}'$ to obtain $(\mathsf{pk}'_0, \mathsf{sk}'_0) \leftarrow G'(1^k)$ and $(\mathsf{pk}'_1, \mathsf{sk}'_1) \leftarrow G'(1^k)$. Choose a random bit $b \in \{0,1\}$ and a random $\ell(k)$-bit string $\mathsf{crs} \in \{0,1\}^{\ell(k)}$. The public key is the triple $(\mathsf{pk}'_0, \mathsf{pk}'_1, \mathsf{crs})$, and the secret key is (b, sk'_b). Erase the other key $\mathsf{sk}'_{\bar{b}}$.

Key update, U. Key update is unchanged, namely $U(t, (b, \mathsf{sk}')) = (b, U'(t, \mathsf{sk}'))$.

Encryption, E. To encrypt a bit $m \in \{0,1\}$ at time t, first pick two independent random strings r_0, r_1 as needed for the encryption algorithm E' and compute $c'_0 \leftarrow E'(\mathsf{pk}'_0, t; m; r_0)$, $c'_1 \leftarrow E'(\mathsf{pk}_1, t; m; r_1)$, and a proof that $(t, \mathsf{pk}'_0, c'_0, \mathsf{pk}'_1, c'_1) \in L_{\mathcal{E}'}$; namely $\pi \leftarrow P(\mathsf{crs}; t, \mathsf{pk}'_0, c'_0, \mathsf{pk}'_1, c'_1; m, r_0, r_1)$. The ciphertext is the triple $c = (c'_0, c'_1, \pi)$.

Decryption, D. To decrypt a ciphertext $c = (c'_0, c'_1, \pi)$ at time t, first run the verifier $V(\mathsf{crs}; t, \mathsf{pk}'_0, c'_0, \mathsf{pk}'_1, c'_1)$. If V rejects, the output is $\perp$. Otherwise, the recipient uses (b, sk'_b) to recover $m \leftarrow D'(\mathsf{sk}'_b; c'_b)$.

We claim the following theorem:

Theorem 1. *If $\mathcal{E}'$ is forward-secure against chosen-plaintext attacks (*fs-CPA*, cf. Definition 4) and if (P, V) is an unbounded simulation-sound NIZK proof system* [DDOPS01, Def. 6], *then $\mathcal{E}$ is an AFSE scheme.*

The proof appears in the full version, but we provide some intuition here. Underlying our analysis is the observation that a simulator (who can generate proofs for false assertions) can come up with a valid-looking "dummy ciphertext" whose component ciphertexts encrypt *different* messages (i.e., both 0 and 1). The simulator, who also knows both underlying decryption keys, can thus open the dummy ciphertext as an encryption of either 0 or 1, depending on which decryption key is presented to an adversary. (Note further that the adversary will be unable to generate dummy ciphertexts of this form due to the simulation soundness of the NIZK proof system.) The above argument demonstrates adaptive security for a single encrypted bit. Adaptive security for an unbounded number of bits (as long as only one ciphertext is outstanding) holds since the secret keys of the underlying FSE schemes evolve after each encryption. We remark that one-time simulation soundness for (P, V) would not be sufficient here, since the simulator must generate *multiple* "fake ciphertexts" and the hybrid argument that works in the non-adaptive case (see [S99]) does not work here.

AFSE for longer messages. To obtain a construction of an AFSE scheme for n-bit messages, one can simply use n pairs of public keys generated using $\mathcal{E}'$ (the receiver now chooses at random one secret key from each pair to store, while the other is erased). The rest is an obvious extension of the proof intuition from above, with the only subtle point being that the resulting ciphertext contains a single NIZK proof computed over the entire vector of n ciphertext pairs (with the language being defined appropriately).

4 Receiver Non-Committing Encryption

This section defines and constructs receiver non-committing encryption (RNCE) that is secure against "lunch-time attacks" (aka CCA1-secure). We note that RNCE was considered in [JL00] for the more basic case of chosen-plaintext attacks. Section 5 shows how to combine any RNCE scheme with any FSE scheme secure against chosen-ciphertext attacks to obtain a secure AFSE scheme. Since our proposed constructions of RNCE schemes are quite efficient (and since relatively-efficient constructions of FSE schemes secure against chosen-ciphertext attacks are known [CHK03,CHK04,BB04]), we obtain (relatively) efficient AFSE schemes.

On a high level, a receiver non-committing encryption scheme is one in which a simulator can generate a single "fake ciphertext" and later "open" this ciphertext (by showing an appropriate secret key) as any given message. These "fake ciphertexts" should be indistinguishable from real ciphertexts, even when an adversary is given access to a decryption oracle before the fake ciphertext is known.

4.1 Definition of RNCE

Formally, a receiver non-committing encryption (RNCE) scheme consists of five PPT algorithms $(G, E, D, \tilde{F}, \tilde{R})$ such that:

- G, E, and D are the key-generation, encryption, and decryption algorithms. These are defined just as for a standard encryption scheme, except that the key generation algorithm also outputs some auxiliary information z in addition to the public and secret keys pk and sk.
- The fake encryption algorithm $\tilde{F}$ takes as input $(\mathsf{pk}, \mathsf{sk}, z)$ and outputs a "fake ciphertext" $\tilde{c}$.
- The reveal algorithm $\tilde{R}$ takes as input $(\mathsf{pk}, \mathsf{sk}, z)$, a "fake ciphertext" $\tilde{c}$, and a message $m \in \mathcal{D}$. It outputs a "secret key" $\widetilde{\mathsf{sk}}$. (Intuitively, $\widetilde{\mathsf{sk}}$ is a secret key for which $\tilde{c}$ decrypts to m.)

We make the standard correctness requirement; namely, for any $\mathsf{pk}, \mathsf{sk}, z$ output by G and any $m \in \mathcal{D}$, we have $D(\mathsf{sk}; E(\mathsf{pk}; m)) = m$.

Our definition of security requires, informally, that for any message m an adversary cannot distinguish whether it has been given a "real" encryption of m along with a "real" secret key, or a "fake" ciphertext along with a "fake" secret key under which the ciphertext decrypts to m. This should hold even when the adversary has non-adaptive access to a decryption oracle. We now give the formal definition.

Definition 2 (RNC-security). *Let $\mathcal{E} = (G, E, D, \tilde{F}, \tilde{R})$ be an RNCE scheme. We say that $\mathcal{E}$ is RNC-secure (or simply "secure") if the advantage of any* PPT *algorithm A in the game below is negligible in the security parameter k.*

1. *The key generation algorithm $G(1^k)$ is run to get $(\mathsf{pk}, \mathsf{sk}, z)$.*

2. *The algorithm A is given 1^k and* pk *as input, and is also given access to a decryption oracle* $D(\mathsf{sk}; \cdot)$. *It then outputs a challenge message* $m \in \mathcal{D}$.
3. *A bit b is chosen at random. If $b = 1$ then a ciphertext* $c \leftarrow E(\mathsf{pk}; m)$ *is computed, and A receives* (c, sk). *Otherwise, a "fake" ciphertext* $\tilde{c} \leftarrow \tilde{F}(\mathsf{pk}, \mathsf{sk}, z)$ *and a "fake" secret key* $\tilde{\mathsf{sk}} \leftarrow \tilde{R}(\mathsf{pk}, \mathsf{sk}, z; \tilde{c}, m)$ *are computed, and A receives* $(\tilde{c}, \tilde{\mathsf{sk}})$. *(After this point, A can no longer query its decryption oracle.) A outputs a bit b'.*

The advantage of A is defined as $2 \cdot \left|\Pr[b' = b] - \frac{1}{2}\right|$.

It is easy to see that the RNC-security of $(G, E, D, \tilde{F}, \tilde{R})$ according to Definition 2 implies in particular that the underlying scheme (G, E, D) is secure against non-adaptive chosen-ciphertext attacks. It is possible to augment Definition 2 so as to grant the adversary access to the decryption oracle even after the ciphertext is known, but we do not need this stronger definition for our intended application (Section 5). We also comment that the Naor-Yung construction [NY90] is RNC-secure for 1-bit messages (if the secret key is chosen at random from the two underlying secret keys); a proof can be derived from [NY90] as well as our proof of Theorem 1.

4.2 A Secure RNCE Scheme for Polynomial-Size Message Spaces

Here, we show that the Cramer-Shoup cryptosystem [CS98] can be modified to give a secure RNCE scheme for *polynomial-size* message spaces. Interestingly, because our definition of security only involves non-adaptive chosen-ciphertext attacks, we can base our construction on the simpler and more efficient "Cramer-Shoup lite" scheme. In fact, the only difference is that we encode a message m by the group element g^m, rather than encoding it directly as the element m. (This encoding is essential for the reveal algorithm $\tilde{R}$.[2])

In what follows, we let $\mathcal{G} = \{\mathbb{G}_k\}_{k \in \mathbb{N}}$ be a family of finite, cyclic groups (written multiplicatively), where each group $\mathbb{G}_k$ has (known) prime order q_k and $|q_k| = k$. For simplicity, we describe our RNCE scheme for the message space $\{0, 1\}$; however, we will comment briefly afterward how the scheme can be extended for any polynomial-size message space.

Key generation, G. Given the security parameter 1^k, let $\mathbb{G}$ denote $\mathbb{G}_k$ and q denote q_k. Choose at random $g_1 \leftarrow \mathbb{G} \setminus \{1\}$, and also choose random $\alpha, x_1, x_2, y_1, y_2 \leftarrow \mathbb{Z}_q$. Set $g_2 = g_1^{\alpha}; h = g_1^{x_1} g_2^{x_2}$; and $d = g_1^{y_1} g_2^{y_2}$. The public key is $\mathsf{pk} = (g_1, g_2, h, d)$, the secret key is $\mathsf{sk} = (x_1, x_2, y_1, y_2)$, and the auxiliary information is $z = \alpha$.

Encryption, E. Given a public key $\mathsf{pk} = (g_1, g_2, h, d)$ and a message $m \in \{0, 1\}$, choose a random $r \in \mathbb{Z}_q$, compute $u_1 = g_1^r\ u_2 = g_2^r$, $e = g_1^m h^r$ and $v = d^r$. The ciphertext is $\langle u_1, u_2, e, v \rangle$.

[2] Looking ahead, it is for this reason that the present construction only handles *polynomial-size* message spaces: the receiver only directly recovers g^m, and must search through the message space to find the corresponding message m.

Decryption, D**.** Given a ciphertext $\langle u_1, u_2, e, v\rangle$ and secret key $\mathsf{sk} = (x_1, x_2, y_1, y_2)$, proceed as follows: First check whether $u_1^{y_1} u_2^{y_2} = v$. If not, then output $\perp$. Otherwise, compute $w = e/u_1^{x_1} u_2^{x_2}$. If $w = 1$ (i.e., the group identity), output 0; if $w = g_1$, output 1. (If $w \notin \{1, g_1\}$ then output $\perp$.)

Fake encryption, $\tilde{F}$**.** Given $\mathsf{pk} = (g_1, g_2, h, d)$ and $\mathsf{sk} = (x_1, x_2, y_1, y_2)$, choose at random $r \in \mathbb{Z}_q$. Then compute $\tilde{u}_1 = g_1^r$, $\tilde{u}_2 = g_1 g_2^r$, $\tilde{e} = g_1^{x_2} h^r$ and $\tilde{v} = \tilde{u}_1^{y_1} \tilde{u}_2^{y_2}$, and output the "fake" ciphertext $\tilde{c} = \langle \tilde{u}_1, \tilde{u}_2, \tilde{e}, \tilde{v}\rangle$.

Reveal algorithm, $\tilde{R}$**.** Given $\mathsf{pk} = (g_1, g_2, h, d)$, $\mathsf{sk} = (x_1, x_2, y_1, y_2)$, $z = \alpha$, a "fake" ciphertext $\langle \tilde{u}_1, \tilde{u}_2, \tilde{e}, \tilde{v}\rangle$, and a message $m \in \{0, 1\}$, set $x_2' = x_2 - m$ and $x_1' = x_1 + m\alpha$ (both in $\mathbb{Z}_q$) and output the "fake" secret key $\tilde{\mathsf{sk}} = (x_1', x_2', y_1, y_2)$.

One can check that the secret key $\tilde{\mathsf{sk}}$ matches the public key pk, since

$$g_1^{x_1'} g_2^{x_2'} = g_1^{x_1 + m\alpha} g_2^{x_2 - m} = (g_1^{x_1} g_2^{m}) g_2^{x_2 - m} = g_1^{x_1} g_2^{x_2} = h;$$

moreover, $\tilde{\mathsf{sk}}$ decrypts the "fake" ciphertext $\langle \tilde{u}_1, \tilde{u}_2, \tilde{e}, \tilde{v}\rangle$ to m, since

$$\frac{e}{\tilde{u}_1^{x_1'} \tilde{u}_2^{x_2'}} = \frac{g_1^{x_2} (g_1^{x_1'} g_2^{x_2'})^r}{(g_1^r)^{x_1'} (g_1 g_2^r)^{x_2'}} = \frac{g_1^{x_2 + r x_1'} g_2^{r x_2'}}{g_1^{r x_1' + x_2'} g_2^{r x_2'}} = g_1^{x_2 - x_2'} = g_1^m.$$

The above scheme can be immediately extended to support any polynomial-size message space: encryption, fake encryption, and reveal would be exactly the same, and decryption would involve computation of w, as above, followed by an exhaustive search through the message space to determine $m \stackrel{\text{def}}{=} \log_{g_1} w$. A proof of the following appears in the full version:

Theorem 2. *If the DDH assumption holds for* $\mathcal{G}$*, then the above scheme is RNC-secure.*

4.3 A Secure RNCE Scheme for Exponential-Size Message Spaces

The RNCE scheme in the previous section can be used only for message spaces of size polynomial in the security parameter, as the decryption algorithm works in time linear in the size of the message space. We now show a scheme that supports message spaces of size exponential in the security parameter. Just as in the previous section, we construct our scheme by appropriately modifying a (standard) cryptosystem secure against chosen-ciphertext attacks. Here, we base our construction on schemes developed independently by Gennaro and Lindell [GL03] and Camenisch and Shoup [CS03], building on earlier work by Cramer and Shoup [CS02]. Security of our scheme, as in these earlier schemes, is predicated on the decisional composite residuosity (DCR) assumption [P99].

Let p, q, p', q' be distinct primes with $p = 2p' + 1$ and $q = 2q' + 1$ (i.e., p, q are *strong* primes). Let $n = pq$ and $n' = p'q'$, and observe that the group $\mathbb{Z}_{n^2}^*$ can be decomposed as the direct product $\mathbb{G}_n \cdot \mathbb{G}_{n'} \cdot \mathbb{G}_2 \cdot \mathbf{T}$, where each $\mathbb{G}_i$ is a cyclic group of order i and $\mathbf{T}$ is the order-2 subgroup of $\mathbb{Z}_{n^2}^*$ generated by $(-1 \bmod n^2)$.

This implies that there exist homomorphisms $\phi_n, \phi_{n'}, \phi_2, \phi_T$ from $\mathbb{Z}^*_{n^2}$ onto $\mathbb{G}_n$, $\mathbb{G}_{n'}$, $\mathbb{G}_2$, and $\mathbf{T}$, respectively, and every $x \in \mathbb{Z}^*_{n^2}$ is uniquely represented by the 4-tuple $(\phi_n(x), \phi_{n'}(x), \phi_2(x), \phi_T(x))$. We use also the fact that the element $\gamma \stackrel{\text{def}}{=} (1+n) \bmod n^2$ has order n in $\mathbb{Z}^*_{n^2}$ (i.e., it generates a group isomorphic to $\mathbb{G}_n$) and furthermore $\gamma^a \bmod n^2 = 1 + an$, for any $0 \le a < n$.

Let $\mathbf{P}_n \stackrel{\text{def}}{=} \{x^n \bmod n^2 : x \in \mathbb{Z}^*_{n^2}\}$ denote the subgroup of $\mathbb{Z}^*_{n^2}$ consisting of all n^{th} powers; note that $\mathbf{P}_n$ is isomorphic to the direct product $\mathbb{G}_{n'} \cdot \mathbb{G}_2 \cdot \mathbf{T}$. The DCR assumption (informally) is that, given n, it is hard to distinguish a random element of $\mathbf{P}_n$ from a random element of $\mathbb{Z}^*_{n^2}$.

Our RNCE scheme is defined below. In this description, we let $\mathcal{G}$ be an algorithm that on input 1^k randomly chooses two primes p', q' as above with $|p'| = |q'| = k$. Also, for a positive real number r we denote by $[r]$ the set $\{0, \ldots, \lfloor r \rfloor - 1\}$.

Key generation, G. Given the security parameter 1^k, use $\mathcal{G}(1^k)$ to select two random k-bit primes p', q' for which $p = 2p'+1$ and $q = 2q'+1$ are also prime, and set $n = pq$ and $n' = p'q'$. Choose random $x, y \in [n^2/4]$ and a random $g' \in \mathbb{Z}^*_{n^2}$, and compute $g = (g')^{2n}$, $h = g^x$, and $d = g^y$. The public key is $\mathsf{pk} = (n, g, h, d)$, the secret key is $\mathsf{sk} = (x, y)$, and the auxiliary information is $z = n'$.

Encryption, E. Given a public key as above and a message $m \in [n]$, choose random $r \in [n/4]$, compute $u = g^r$, $e = \gamma^m h^r$, and $v = d^r$ (all in $\mathbb{Z}^*_{n^2}$), and output the ciphertext $c = \langle u, e, v \rangle$.

Decryption, D. Given a ciphertext $\langle u, e, v \rangle$ and secret key (x, y), check whether $u^{2y} = v^2$; if not, output $\perp$. Then, set $\hat{m} = (e/u^x)^{n+1}$. If $\hat{m} = 1 + mn$ for some $m \in [n]$, then output m; otherwise, output $\perp$.

Correctness follows, since for a valid ciphertext $\langle u, e, v \rangle$ we have $u^{2y} = (g^r)^{2y} = d^{2r} = v^2$, and also $(e/u^x)^{n+1} = (\gamma^m h^r / g^{rx})^{n+1} = (\gamma^m)^{n+1} = \gamma^m = 1 + mn$ (using for the third equality the fact that the order of γ is n).

Fake encryption, $\tilde{F}$. Given $\mathsf{pk} = (n, g, h, d)$ and $\mathsf{sk} = (x, y)$, choose at random $r \in [n/4]$, compute $\tilde{u} = \gamma \cdot g^r$, $\tilde{e} = \tilde{u}^x$, and $\tilde{v} = \tilde{u}^y$ (all in $\mathbb{Z}^*_{n^2}$), and output the "fake" ciphertext $\tilde{c} = \langle \tilde{u}, \tilde{e}, \tilde{v} \rangle$.

Reveal algorithm, $\tilde{R}$. Given $\mathsf{pk} = (n, g, h, d)$, $\mathsf{sk} = (x, y)$, $z = n'$, a "fake" ciphertext $\langle \tilde{u}, \tilde{e}, \tilde{v} \rangle$ as above, and a message $m \in [n]$, proceed as follows: Using the Chinese Remainder Theorem and the fact that $\gcd(n, n') = 1$, find the unique $x' \in [nn']$ satisfying $x' = x \bmod n'$, and $x' = x - m \bmod n$, and output the secret key $\tilde{\mathsf{sk}} = (x', y)$.

It can be verified that the secret key $\tilde{\mathsf{sk}}$ matches the public key pk and also decrypts the "fake" ciphertext to the required message m: For the second component y this is immediate and so we focus on the first component x'. First, the order of g divides n' and so $g^{x'} = g^{x' \bmod n'} = g^{x \bmod n'} = g^x = h$. Furthermore, using also the fact that the order of γ in $\mathbb{Z}^*_{n^2}$ is n, we have

$$\left(\frac{\tilde{e}}{\tilde{u}^{x'}}\right)^{n+1} = \left(\frac{\gamma^x g^{rx}}{\gamma^{x'} g^{rx'}}\right)^{n+1} = \left(\gamma^{x - x' \bmod n}\right)^{n+1} = \gamma^m.$$

In the full version we define the decisional composite residuosity assumption (DCR) with respect to $\mathcal{G}$ (cf. [P99]), and show:

Theorem 3. *If the DCR assumption holds for $\mathcal{G}$, then the above scheme is RNC-secure.*

5 AFSE Based on Receiver Non-Committing Encryption

We describe a construction of an AFSE scheme based on any secure RNCE scheme and any FSE scheme secure against chosen-ciphertext attacks. Let $\mathcal{E}' = (G', E', D', \tilde{F}, \tilde{R})$ be an RNCE scheme, and let $\mathcal{E}'' = (G'', U'', E'', D'')$ be a key-evolving encryption scheme. The message space of $\mathcal{E}'$ is $\mathcal{D}$, and we assume that ciphertexts of $\mathcal{E}'$ belong to the message space of $\mathcal{E}''$. We construct a new key-evolving encryption scheme $\mathcal{E} = (G, U, E, D)$ with message space $\mathcal{D}$ as follows:

Key generation, G**.** On security parameter 1^k, run the key-generation algorithms of both schemes, setting $(\mathsf{pk}', \mathsf{sk}', z) \leftarrow G'(1^k)$ and $(\mathsf{pk}'', \mathsf{sk}''_0) \leftarrow G''(1^k)$. The public key is $(\mathsf{pk}', \mathsf{pk}'')$ and the initial secret key is $(\mathsf{sk}', \mathsf{sk}''_0)$. (The extra information z is ignored.)

Key update, U**.** The key-update operation is derived as one would expect from $\mathcal{E}''$; namely: $U(t; \mathsf{sk}', \mathsf{sk}''_t) = (\mathsf{sk}', U''(t; \mathsf{sk}''_t))$.

Encryption, E**.** To encrypt a message $m \in \mathcal{D}$ at time t, first compute $c' \leftarrow E'(\mathsf{pk}'; m)$ and then $c \leftarrow E''(\mathsf{pk}'', t; c')$. The resulting ciphertext is just c.

Decryption, D**.** To decrypt a ciphertext c, set $c' \leftarrow D''(\mathsf{sk}''_t; c)$ and then compute $m \leftarrow D'(\mathsf{sk}'; c')$.

Theorem 4. *If $\mathcal{E}'$ is RNC-secure, and if $\mathcal{E}''$ is forward-secure against chosen-ciphertext attacks, then the combined scheme given above is an AFSE scheme.*

We provide some informal intuition behind the proof of the above theorem. The most interesting scenario to consider is what happens upon player corruption, when the adversary obtains the secret key for the current time period t^*. We may immediately note that messages encrypted for prior time periods $t < t^*$ remain secret; this follows from the FSE encryption applied at the "outer" layer. Next, consider adaptive security for the (at most one) outstanding ciphertext which was encrypted for some time period $t \geq t^*$. Even though the adversary can "strip off" the outer later of the encryption (because the adversary now has the secret key for time period t^*), RNC security of the inner layer ensures that a simulator can open the inner ciphertext to any desired message. The main point here is that the simulator only needs to "fake" the opening of *one* inner ciphertext, and thus RNC security suffices. (Still, since the simulator does not know in advance what ciphertext it will need to open, it actually "fakes" *all* inner ciphertexts.) Chosen-ciphertext attacks are dealt with using the chosen-ciphertext security of the outer layer, as well as the definition of RNC security (where "lunch-time security" at the inner layer is sufficient). Also, we note that reversing the order of encryptions does not work: namely, using $RNCE(FSE(m))$ does not yield adaptive security, even if the RNCE scheme is fully CCA secure.

References

[AF04] M. Abe and S. Fehr. Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography. *Crypto 2004*, LNCS vol. 3152, pp. 317–334, 2004. Full version available at `eprint.iacr.org/2004/119`.

[A97] R. Anderson. Two Remarks on Public Key Cryptology. Invited lecture, given at *ACM CCCS '97*. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`.

[B97] D. Beaver. Plug and Play Encryption. *Crypto 1997*, LNCS vol. 1294, pp. 75–89, 1997.

[BH92] D. Beaver and S. Haber. Cryptographic Protocols Provably Secure Against Dynamic Adversaries. *Eurocrypt 1992*, LNCS vol. 658, pp. 307–323, 1992.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. *Crypto 1998*, LNCS vol. 1462, pp. 26–45, 1998.

[BB04] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. *Eurocrypt 2004*, LNCS vol. 3027, pp. 223–238, 2004.

[CS03] J. Camenisch and V. Shoup. Practical Verifiable Encryption and Decryption of Discrete Logarithms. *Crypto 2003*, LNCS vol. 2729, pp. 126–144, 2003.

[C01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. *42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 136–145, 2001. Also available as ECCC TR 01-16, or from `http://eprint.iacr.org/2000/067`.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Computation. *28th ACM Symposium on Theory of Computing (STOC)*, pp. 639–648, 1996. Full version in MIT-LCS-TR #682, 1996.

[CHK03] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. *Eurocrypt 2003*, LNCS vol. 2656, pp. 255–271, 2003. Full version available at `http://eprint.iacr.org/2003/083`.

[CHK04] R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *Eurocrypt 2004*, LNCS vol. 3027, pp. 207–222, 2004. Full version available at `http://eprint.iacr.org/2003/182`.

[CHK05] R. Canetti, S. Halevi, and J. Katz. Adaptively-Secure, Non-Interactive Public-Key Encryption. Full version available at `http://eprint.iacr.org/2004/317`.

[CKN03] R. Canetti, H. Krawczyk, and J.B. Nielsen. Relaxing Chosen Ciphertext Security. *Crypto 2003*, LNCS vol. 2729, pp. 565–582, 2003. Full version available at `http://eprint.iacr.org/2003/174`.

[CS98] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Chosen Ciphertext Attack. *Crypto 1998*, LNCS vol. 1462, pp. 13–25, 1998.

[CS02] R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. *Eurocrypt 2001*, LNCS vol. 2332, pp. 45–63, 2001.

[DN00] I. Damgård and J. B. Nielsen. Improved Non-Committing Encryption Schemes Based on General Complexity Assumptions. *Crypto 2000*, LNCS vol. 1880, pp. 432–450, 2000.

[DDOPS01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-Interactive Zero Knowledge. *Crypto 2001*, LNCS vol. 2139, pp. 566–598, 2001.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM. J. Computing* 30(2): 391-437, 2000.
[GL03] R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *Eurocrypt 2003*, LNCS vol. 2656, pp. 524–543, 2003. Full version available at `http://eprint.iacr.org/2003/032`.
[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Computer System Sciences* 28(2): 270-299, 1984.
[HMS03] Dennis Hofheinz, Joern Mueller-Quade, and Rainer Steinwandt. On Modeling IND-CCA Security in Cryptographic Protocols. Available at `http://eprint.iacr.org/2003/024`.
[JL00] S. Jarecki and A. Lysyanskaya. Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures. *Eurocrypt 2000*, LNCS vol. 1807, pp. 221–242, 2000.
[NY90] M. Naor and M. Yung. Public-Key Cryptosystems Provably-Secure against Chosen-Ciphertext Attacks. *22nd ACM Symposium on Theory of Computing (STOC)*, pp. 427–437, 1990.
[N02] J.B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. *Crypto 2002*, LNCS vol. 2442, pp. 111–126, 2002.
[P99] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. *Eurocrypt 1999*, LNCS vol. 1592, pp. 223–238, 1999.
[RS91] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Crypto 1991*, LNCS vol. 576, pp. 433–444, 1991.
[S99] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 543–553, 1999.

A Key-Evolving and Forward-Secure Encryption

We review the definitions of key-evolving and forward-secure encryption schemes from [CHK03].

Definition 3. *A (public-key) key-evolving encryption (*ke-PKE*)* scheme *is a 4-tuple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- *The* key generation algorithm Gen *takes as input a security parameter* 1^k *and the total number of time periods* N. *It returns a public key* pk *and an initial secret key* sk_0.
- *The* key update algorithm Upd *takes as input* pk*, an index* $t < N$ *of the current time period, and the associated secret key* sk_t. *It returns the secret key* sk_{t+1} *for the following time period.*
- *The* encryption algorithm Enc *takes as input* pk*, an index* $t \leq N$ *of a time period, and a message* M. *It returns a ciphertext* C.
- *The* decryption algorithm Dec *takes as input* pk*, an index* $t \leq N$ *of the current time period, the associated secret key* sk_t*, and a ciphertext* C. *It returns a message* M.

We require that $\mathsf{Dec}(\mathsf{sk}_t; t; \mathsf{Enc}(\mathsf{pk}_t, t, M)) = M$ *holds for all* $(\mathsf{pk}, \mathsf{sk}_0)$ *output by* Gen*, all time periods* $t \leq N$*, all correctly generated* sk_t *for this* t*, and all messages* M.

Definition 4. *A ke-PKE scheme is* forward-secure against chosen plaintext attacks *(*fs-CPA*) if for all polynomially-bounded functions $N(\cdot)$, the advantage of any* PPT *adversary in the following game is negligible in the security parameter:*

Setup*:* $\mathsf{Gen}(1^k, N(k))$ *outputs* (PK, SK_0)*. The adversary is given* PK*.*

Attack*: The adversary issues one* breakin(i) *query and one* challenge(j, M_0, M_1) *query, in either order, subject to* $0 \leq j < i < N$*. These queries are answered as follows:*

- *On query* breakin(i)*, key* SK_i *is computed via* $\mathsf{Upd}(PK, i-1, \cdots \mathsf{Upd}(PK, 0, SK_0) \cdots)$*. This key is then given to the adversary.*
- *On query* challenge(j, M_0, M_1)*, a random bit* b *is selected and the adversary is given* $C^* = \mathsf{Enc}(PK, j, M_b)$*.*

Guess*: The adversary outputs a guess* $b' \in \{0,1\}$*; it* succeeds *if* $b' = b$*. The adversary's* advantage *is the absolute value of the difference between its success probability and* $1/2$*.*

Forward security against (adaptive) chosen-ciphertext attacks (fs-CCA security) is defined by the natural extension of the above definition in which the adversary is given decryption oracle access during both the "Attack" and "Guess" stages.

B The UC Framework, Abridged

We provide a brief review of the universally composable security framework [C01]. The framework allows for defining the security properties of cryptographic tasks so that security is maintained under general composition with an unbounded number of instances of arbitrary protocols running concurrently. Definitions of security in this framework are called universally composable (UC).

In the UC framework, the security requirements of a given task (i.e., the functionality expected from a protocol that carries out the task) are captured via a set of instructions for a "trusted party" that obtains the inputs of the participants and provides them with the desired outputs (in one or more iterations). Informally, a protocol securely carries out a given task if running the protocol with a realistic adversary amounts to "emulating" an ideal process where the parties hand their inputs to a trusted party with the appropriate functionality and obtain their outputs from it, without any other interaction.

The notion of emulation in the UC framework is considerably stronger than that considered in previous models. Traditionally, the model of computation includes the parties running the protocol and an adversary $\mathcal{A}$ that controls the communication channels and potentially corrupts parties. "Emulating an ideal process" means that for any adversary $\mathcal{A}$ there should exist an "ideal process adversary" (or simulator) $\mathcal{S}$ that causes the *outputs* of the parties in the ideal process to have similar distribution to the outputs of the parties in an execution of the protocol. In the UC framework the requirement on $\mathcal{S}$ is more stringent. Specifically, an additional entity, called the environment $\mathcal{Z}$, is introduced. The environment generates the inputs to all parties, reads all outputs, and in addition interacts with the adversary in an arbitrary way throughout the computation. A protocol is said to securely realize functionality $\mathcal{F}$ if for any "real-life" adversary $\mathcal{A}$ that interacts with the protocol and the environment there exists an "ideal-process adversary" $\mathcal{S}$, such that *no environment* $\mathcal{Z}$ can tell whether it is interacting with $\mathcal{A}$ and parties running the protocol, or with $\mathcal{S}$ and parties that interact with $\mathcal{F}$ in the ideal process. In a sense, $\mathcal{Z}$ serves as an "interactive distinguisher" between a run of the protocol and the ideal process with access to $\mathcal{F}$.

The following *universal composition theorem* is proven in [C01]. Consider a protocol π that operates in the $\mathcal{F}$-hybrid model, where parties can communicate as usual and in addition have ideal access to an unbounded number of *copies* of the functionality $\mathcal{F}$. Let ρ be a protocol that securely realizes $\mathcal{F}$ as sketched above, and let π^ρ be identical to π with the exception that the interaction with *each copy* of $\mathcal{F}$ is replaced with an interaction with a *separate instance* of ρ. Then, π and π^ρ have essentially the same input/output behavior. In particular, if π securely realizes some functionality $\mathcal{I}$ in the $\mathcal{F}$-hybrid model then π^ρ securely realizes $\mathcal{I}$ in the standard model (i.e., without access to any functionality).

B.1 The Public-Key Encryption Functionality $\mathcal{F}_{\text{PKE}}$

(This section is taken almost verbatim from [CKN03].) Within the UC framework, public-key encryption is defined via the public-key encryption functionality, denoted $\mathcal{F}_{\text{PKE}}$ and presented in Figure 2. Functionality $\mathcal{F}_{\text{PKE}}$ is intended to capture the functionality of public-key encryption and, in particular, is written in a way that allows realizations consisting of three non-interactive algorithms without any communication. (The three algorithms correspond to the key generation, encryption, and decryption algorithms in traditional definitions.)

Referring to Figure 2, we note that *sid* 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 instances of $\mathcal{F}_{\text{PKE}}$). It also encodes the identity of the decryptor for this instance. The "public key value" pk 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 this value 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, while 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 about legitimately-encrypted messages, we are guaranteed that illegitimate ciphertexts will be decrypted to values that are independent from these messages.) Note that the same illegitimate ciphertext can be decrypted to different values in different activations. This

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

$\mathcal{F}_{\mathrm{PKE}}$ proceeds as follows, when parameterized by message domain ensemble $\mathcal{D} = \{D_k\}_{k\in\mathbf{N}}$ and security parameter k.

Key Generation: Upon receiving a value $(\texttt{KeyGen}, sid)$ from some party R^*, verify that $sid = (sid', R^*)$. If not, then ignore the input. Otherwise:

1. Hand $(\texttt{KeyGen}, sid)$ to the adversary.
2. Receive a value pk^* from the adversary, and hand pk^* to R^*.
3. If this is the first `KeyGen` request, record R^* and pk^*.

Encryption: Upon receiving from some party P a value $(\texttt{Encrypt}, sid, \mathsf{pk}, m)$ proceed as follows:

1. If $m \notin D_k$ then return an error message to P.
2. If $m \in D_k$ then hand $(\texttt{Encrypt}, sid, \mathsf{pk}, P)$ to the adversary. (If $\mathsf{pk} \neq \mathsf{pk}^*$ or pk^* 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 send $(\texttt{ciphertext}, c)$ to P. (If $\mathsf{pk} \neq \mathsf{pk}^*$ or pk^* is not yet defined then do *not* record the pair (c, m).)

Decryption: Upon receiving a value $(\texttt{Decrypt}, sid, c)$ from R^* (and R^* only), proceed as follows:

1. If there is a recorded pair (c, m) then hand m to R^*. (If there is more than one such pair then use the first one.)
2. Otherwise, hand the value $(\texttt{Decrypt}, sid, c)$ to the adversary. When receiving a value m' from the adversary, hand m' to R^*.

Fig. 2. The public-key encryption functionality, $\mathcal{F}_{\mathrm{PKE}}$

provision allows the decryption algorithm to be non-deterministic with respect to ciphertexts that were not legitimately generated.

Another characteristic of $\mathcal{F}_{\mathrm{PKE}}$ is that, when activated with a `KeyGen` request, it always responds with an (adversarially-chosen) encryption key pk'. 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). Other keys may provide correct encryption and decryption, but do not guarantee any security (see [CKN03] for further discussion about possible alternative formulations of the functionality).