Efficient and Non-Malleable Proofs of Plaintext Knowledge and Applications

(Extended Abstract*)

JONATHAN KATZ†

Abstract

We describe very efficient protocols for *non-malleable* (interactive) proofs of plaintext knowledge for the RSA, Rabin, Paillier, and El-Gamal encryption schemes whose security can be proven in the standard model. We also highlight some important applications of these protocols, where we take care to ensure that our protocols remain secure when run in an asynchronous, concurrent environment:

- *Chosen-ciphertext-secure, interactive encryption.* In some settings where both parties are on-line (e.g., SSL), an interactive encryption protocol may be used. We construct chosen-ciphertext-secure interactive encryption schemes based on any of the schemes above. In each case, the improved scheme requires only a small overhead beyond the original, semantically-secure scheme.
- *Password-based authenticated key exchange.* We provide efficient protocols for password-based authenticated key exchange in the public-key model [27, 5]. Security of our protocols may be based on any of the cryptosystems mentioned above.
- *Deniable authentication.* We demonstrate deniable authentication protocols satisfying the strongest notion of security. These are the first efficient constructions based on, e.g., the RSA or computational Diffie-Hellman assumptions.

Our techniques provide a general methodology for constructing efficient *non-malleable* (zero-knowledge) proofs of knowledge when shared parameters are available (for our intended applications, these parameters can simply be included as part of users' public keys). Thus, non-malleable proofs of knowledge are easy to achieve "in practice".

1 Introduction

Given an instance of a public-key encryption scheme with public key pk and secret key sk, a *proof of plaintext knowledge* (PPK) allows a sender $\mathcal{S}$ to prove knowledge of the contents m of some ciphertext $C = \mathcal{E}_{pk}(m)$ to a receiver $\mathcal{R}$ (a formal definition appears in Section 2.1). To be useful, a PPK should additionally ensure that no information about m is revealed, either to the receiver — which is important if the receiver does not have sk — or to an eavesdropper. As we show here, such PPKs have applications to the construction of chosen-ciphertext-secure (CCA2) public-key encryption schemes [29, 32], password-based

*The full version of this work appears in [28, Chapter 5].

†`jkatz@cs.columbia.edu`. Department of Computer Science, Columbia University.

authentication and key exchange (password-AKE) protocols in the public-key model [27, 5], and deniable authentication [15, 17].

We note that PPKs may be achieved using completeness results for zero-knowledge (ZK) proofs [25] and proofs of knowledge [20]; non-interactive PPKs are also possible (assuming appropriate public parameters are included as part of pk) based on the completeness results for non-interactive zero-knowledge (NIZK) [21] and NIZK proofs of knowledge [12]. For common encryption schemes such as Rabin [31], RSA [33], Paillier [30], or El-Gamal [19], the well-studied Σ-protocols [6] for these schemes (e.g., [23, 35, 7]) may be adapted to give PPKs, although modifications are needed to ensure security against a cheating verifier/receiver. For the intended applications listed above, however, solutions such as these are not enough; the following considerations additionally need to be taken into account:

- Non-malleability. An active adversary $\mathcal{M}$ may be controlling all communication between the honest parties in a classic "man-in-the-middle" attack. We then need to ensure that the adversary cannot divert the proof of knowledge being given by $\mathcal{S}$ to $\mathcal{R}$. For example, $\mathcal{S}$ may be giving a PPK of C, yet $\mathcal{M}$ might be able to change this to a PPK of some C' *even though* $\mathcal{M}$ *has no knowledge of the real decryption of* C'. In many contexts this is unacceptable.
- Concurrency. A receiver may be interacting asynchronously and concurrently with many senders who are simultaneously giving PPKs corresponding to different ciphertexts. The protocol should remain secure even when executed in this environment.

Some solutions to the above problems exist (cf. Section 1.2). However, these solutions — particularly in the case of non-malleability — are extremely inefficient. The non-malleable PPKs given here, on the other hand, are very efficient. In fact, the chosen-ciphertext-secure encryption, password-AKE, and deniable authentication schemes we build from our non-malleable PPKs are roughly as efficient as (in some cases, more efficient than) existing provably-secure solutions which are based on different assumptions or consider weaker notions of security.

1.1 Applications and Results

We informally describe the principal applications of the non-malleable PPKs we give here. Formal definitions are given in Appendix A.

Interactive public-key encryption. When a sender and receiver are both on-line, it may be possible to use an interactive encryption protocol[1] if this protocol confers other advantages (e.g., stronger security properties, greater efficiency, etc.). For example, known (non-interactive) CCA2 encryption schemes are either impractical [15, 34, 13, 10] or are based on a specific, *decisional* assumption [9, 10]. Hence, it is reasonable to look for efficient constructions of interactive CCA2 encryption schemes which may be based on various *computational* assumptions.

Interactive schemes for CCA1[2] encryption based on PPKs have been proposed previously [22, 26, 24]. For such an approach to yield *adaptive* CCA2 security, however, a

[1] Note that interaction is generally taking place already (e.g., to establish a TCP connection) as part of the larger protocol in which encryption is taking place.

[2] I.e., secure against *non-adaptive* chosen-ciphertext attacks; the notation follows [2].

non-malleable PPK (as introduced here) must be used. A relatively efficient CCA2 interactive encryption scheme (which does not use proofs of knowledge) is given by [15]; however, their protocol requires a signature from the receiver, making it unsuitable for use in a deniable authentication protocol (see below). Moreover, the protocol requires the receiver — for each encrypted message — to (1) compute an existentially unforgeable signature and (2) run the key generation procedure for a public-key encryption scheme (often the most computationally-intensive step).

Using the non-malleable PPKs presented here, we construct efficient interactive encryption protocols which are provably secure (in the standard model) against chosen-ciphertext attacks. The efficiency of our protocols is comparable to that of the most efficient non-interactive scheme [9] which is based on the DDH assumption. In each case, our protocols require only a small computational overhead beyond a basic, semantically-secure scheme; furthermore, most of this extra computation may be done in a preprocessing stage. The security of our protocols may be based on a variety of assumptions; e.g., the RSA assumption, the *computational* composite residuosity assumption [30], or the hardness of factoring.

Password-based authentication and key exchange. The usefulness of interactive encryption schemes is even more apparent when such schemes are used to encrypt messages sent as part of a larger protocol in which interaction is *already* taking place. As an example, consider password-based authentication and key exchange in the public-key model [27, 5]. In this setting, a client and a server share a weak, human-memorizable password which they use to authenticate each other and to derive a key for securing future communication; additionally, the server has a public key which is known by the client. Since the password is short, off-line dictionary attacks must be explicitly prevented. Previous work [27, 5] gives elegant (interactive[3]) protocols for securely realizing this task using any CCA2 public-key cryptosystem; our techniques allow the first efficient implementation of these protocols based on, e.g., the factoring assumption.

Deniable authentication. A deniable authentication protocol [15, 17, 16] allows a prover $\mathcal{P}$ (who has a public key) to authenticate a message to a verifier $\mathcal{V}$ such that the transcript of the interaction cannot be used as evidence that $\mathcal{P}$ actually took part in the protocol (i.e., $\mathcal{P}$ can *deny* that the authentication took place). The strongest notion of deniability requires the existence of a simulator which, given access only to the verifier, can output a transcript indistinguishable from an interaction of the actual prover and verifier. In addition to deniability, we require (informally) that an adversary who interacts with the prover — where the prover authenticates messages $m_1, \ldots$ of the adversary's choice — should be unable to falsely authenticate any message $m' \notin \{m_1, \ldots\}$ for an honest verifier.

Constructions of deniable authentication protocols based on any CCA2 encryption scheme are known [17, 18, 16]. However, these protocols are not secure (in general) when a CCA2 *interactive* encryption scheme is used. For example, the interactive encryption scheme of [15] requires a signature from the prover and hence the resulting authentication protocol (using the above constructions) is no longer deniable; this problem is pointed out explicitly in [17].

The only previously-known deniable authentication protocol which is both practical and satisfies the strongest notion of deniability use the construction of [17] instantiated with the

[3]Interaction is necessary in any such protocol in order to prevent replay attacks.

practical CCA2 encryption scheme of [9]. We present here the first efficient deniable authentication protocols based on factoring (or related assumptions) which are secure under the strongest notion of security. Our constructions have the same round-complexity and require roughly the same computation as the most efficient previous solution, but the security of our protocols may be based on a wider class of assumptions.

Non-malleable proofs of knowledge. Efficient, non-malleable (interactive) proofs of knowledge are of general interest, and have applicability in a variety of settings. In contrast to non-interactive proofs (even in the random oracle setting), interactive proofs automatically prevent replay attacks. In addition, they may prove useful in protocol design whenever man-in-the-middle attacks are a concern.

Previous constructions of non-malleable proofs of knowledge [15, 34, 13] — especially in the interactive case where a logarithmic number of rounds are needed — are very inefficient. The present work shows that, when public information is available, non-malleable proofs of knowledge can be obtained easily. In particular, for applications such as those outlined above, the public information may be published along with the already necessary public keys (and there is no motivation for the party publishing the information to cheat). Our results also extend to the auxiliary string model. Thus, analogous to the results of Damgård [11] for concurrent zero-knowledge, we demonstrate that non-malleable proofs of knowledge are easy to obtain "in practice". Yet, our results do not address the important theoretical question of obtaining constant-round non-malleable ZK proofs *without* extra setup assumptions.

Concurrent proofs of knowledge. Although the issue of concurrency in the context of zero-knowledge proofs has been investigated extensively (following the initial work by Dwork, Naor, and Sahai [17]), we are not aware of any previous work dealing with concurrency in the context of proofs of knowledge. In the context of zero-knowledge, the difficulty is to ensure that the interaction of a single prover with multiple verifiers (in a concurrent fashion) can be simulated in expected polynomial time. In the context of proofs of knowledge, on the other hand, the difficulty arises from having to extract (in expected polynomial time) witnesses from multiple *provers* all proving different statements and interacting concurrently with a single verifier. Following [17], we use timing constraints to ensure the security of our protocols when they are run in a concurrent, asynchronous environment.

1.2 Related Work

Proofs of plaintext knowledge are explicitly considered by Aumann and Rabin [1], who provide an elegant solution for *any* public-key encryption scheme. Our solutions improve upon theirs in many respects: (1) by working with specific, number-theoretic assumptions we obtain simple, 3-round schemes that are vastly more efficient; (2) we explicitly consider malleability and ensure that our protocols are non-malleable; (3) our protocols are secure even against a dishonest verifier, whereas Aumann and Rabin consider only security against an honest verifier (i.e., the intended recipient); (4) we explicitly handle concurrency and our protocols remain provably-secure under asynchronous, concurrent composition.

Non-malleable zero-knowledge (interactive) proofs were defined and constructed by Dolev, Dwork, and Naor [15]; subsequently, Sahai [34] and De Santis, et al. [13] provide definitions and constructions for non-malleable, non-interactive zero-knowledge proofs and proofs of knowledge.[4] Thus, in principal, solutions to our problem exist. Yet, these solutions are based on general assumptions and are completely impractical. In particular, known non-malleable interactive proofs [15] require a poly-logarithmic number of rounds, while in the non-interactive setting we do not even know how to give efficient NIZK proofs — let alone non-malleable ones — for number-theoretic problems of interest (i.e., without reducing the problem to a general NP-complete language).

Non-malleable PPKs were considered, *inter alia*, by Cramer, Damgård, and Nielsen [7] in the context of communication-efficient multi-party computation; they also present an efficient construction suitable for their intended application (we note that no definitions of non-malleable PPKs are given in [7]). Here, in addition to constructions, we give formal definitions and also show applications to a number of other cryptographic protocols. Furthermore, we note some important differences between our approaches. First, the solution of [7] relies in an essential way upon the fact that the set of participants (and thus their number and their identities) is fixed and publicly known. Our protocols, on the other hand, do not require any notion of user identities and we assume no bound on the number of potential participants. When the set of users *is* known (e.g., in the context of multi-party computation), our approach improves upon previous work in that the size of our public parameters is independent of the number of users n (instead of $\mathcal{O}(n)$, as in [7]). Finally, Cramer, et al. assume a synchronous communication model; here, we allow for asynchronous and concurrent communication which is more realistic in the context of, e.g., public-key encryption.

2 Definitions and Preliminaries

Definitions for CCA2 interactive encryption and deniable authentication appear in Appendix A.

2.1 Non-Malleable Proofs of Plaintext Knowledge

The definitions given in this section focus on proofs of plaintext knowledge, yet they may be easily extended to proofs of knowledge for general NP-relations. We assume a non-interactive public-key encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The encryption of message m under public key pk using randomness r to give ciphertext C is denoted as $C := \mathcal{E}_{pk}(m; r)$. In this case, we say that tuple (m, r) is a *witness to the decryption of C under pk*. For convenience, we assume that $|pk| = k$, where k is the security parameter. We let the notation $\langle A(a), B(b)\rangle(c)$ be the random variable denoting the output of B following an execution of an interactive protocol between A (with private input a) and B (with private input b) on joint input c, where A and B have uniformly-distributed random tapes.

A *proof of plaintext knowledge* (PPK) allows a sender $\mathcal{S}$ to prove knowledge of a witness to the decryption of some ciphertext C to a receiver $\mathcal{R}$. Both $\mathcal{S}$ and $\mathcal{R}$ have an additional joint input σ generated by some algorithm $\mathcal{G}(pk)$; in practice, σ will be generated by $\mathcal{R}$ and published along with $\mathcal{R}$'s public key pk.[5] To ensure that no information about m is revealed, we require our PPKs to be zero-knowledge in the following sense: We require the existence

[4] Interestingly, ours is the first work to explicitly consider non-malleable *interactive* proofs *of knowledge*.

[5] It is important to note that there will be no incentive for $\mathcal{R}$ to cheat when choosing σ.

of a simulator $\mathcal{SIM}$ which takes pk as input and outputs parameters σ whose distribution will be equivalent to the output of $\mathcal{G}(pk)$. Furthermore, given any valid ciphertext C (but no witness to its decryption), $\mathcal{SIM}$ must be able to perfectly simulate a PPK of C with any *malicious* receiver $\mathcal{R}'$ using parameters σ. The zero-knowledge property we require is actually quite strong: $\mathcal{SIM}_2$ must achieve a perfect simulation *without rewinding* $\mathcal{R}'$. This definition is met by our constructions.

Our definitions build on the now-standard one for proofs of knowledge [3], except that our protocols are technically *arguments* of knowledge and we therefore restrict ourselves to consideration of provers running in probabilistic, polynomial time.

Definition 1 *Let $\Pi = (\mathcal{G}, \mathcal{S}, \mathcal{R})$ be a tuple of* PPT *algorithms. We say Π is a* proof of plaintext knowledge (PPK) *for encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ if the following conditions hold:*

(Completeness) *For all pk output by $\mathcal{K}(1^k)$, all σ output by $\mathcal{G}(pk)$, and all C with witness w to the decryption of C under pk we have $\langle \mathcal{S}(w), \mathcal{R}\rangle(pk, \sigma, C) = 1$ (when $\mathcal{R}$ outputs 1 we say it* accepts*).*

(Perfect zero-knowledge) *There exists a* PPT simulator $\mathcal{SIM} = (\mathcal{SIM}_1, \mathcal{SIM}_2)$ *such that, for all pk output by $\mathcal{K}(1^k)$, all computationally-unbounded $\mathcal{R}'$, and all m, r, the following distributions are equivalent:*

$$\{\sigma \leftarrow \mathcal{G}(pk); C := \mathcal{E}_{pk}(m; r) : \langle \mathcal{S}(m, r), \mathcal{R}'\rangle(pk, \sigma, C)\}$$

$$\{(\sigma, s) \leftarrow \mathcal{SIM}_1(pk); C := \mathcal{E}_{pk}(m; r) : \langle \mathcal{SIM}_2(s), \mathcal{R}'\rangle(pk, \sigma, C)\}.$$

(Witness extraction) *There exists a function $\kappa : \{0,1\}^* \to [0,1]$, a negligible function $\varepsilon(\cdot)$, and an expected-polynomial-time* knowledge extractor $\mathcal{KE}$ *such that, for all* PPT *algorithms $\mathcal{S}'$, with all but negligible probability over pk output by $\mathcal{K}(1^k)$, σ output by $\mathcal{G}(pk)$, and uniformly-distributed r, machine $\mathcal{KE}$ satisfies the following:*

> *Denote by $p_{pk,\sigma,r}$ the probability that $\mathcal{R}$ accepts when interacting with $\mathcal{S}'$ (using random tape r) on joint input pk, σ, C (where C is chosen by $\mathcal{S}'$). On input pk, σ, and access to $\mathcal{S}'_r$, the probability that $\mathcal{KE}$ outputs a witness to the decryption of C under pk is at least:*
>
> $$p_{pk,\sigma,r} - \kappa(pk) - \varepsilon(|pk|).$$

Our definition of interactive non-malleable PPKs builds on the ideas of De Santis, et al. [13], who define a similar notion in the non-interactive setting. Informally, a non-malleable PPK should satisfy the intuition that "anything proven by a man-in-the-middle adversary $\mathcal{M}$ is known by $\mathcal{M}$ (unless $\mathcal{M}$ simply copies a proof)." To formalize this idea, we allow $\mathcal{M}$ to interact with a simulator (whose existence is guaranteed by Definition 1) while simultaneously interacting with a receiver $\mathcal{R}$. $\mathcal{M}$ chooses (adaptively) ciphertexts C, C' and then executes a PPK of C' to $\mathcal{R}$ while the simulator executes a PPK of C to $\mathcal{M}$. The following definition states (informally) that if $\mathcal{R}$ accepts $\mathcal{M}$'s proof — yet the transcripts of the two proofs are different — then a knowledge extractor $\mathcal{KE}^*$ can extract a witness to the decryption of C'. The reason we have $\mathcal{M}$ interact with the simulator instead of the real sender $\mathcal{S}$ is that we must ensure that the knowledge is actually extracted from $\mathcal{M}$ and not from the real sender.

Definition 2 *PPK* $(\mathcal{G}, \mathcal{S}, \mathcal{R})$ *is* non-malleable *if there exists a simulator* $\mathcal{SIM}$ *(satisfying the relevant portion of Definition 1), a function* $\kappa^* : \{0,1\}^* \rightarrow [0,1]$*, a negligible function* $\varepsilon^*(\cdot)$*, and an expected-polynomial-time* knowledge extractor $\mathcal{KE}^*$ *such that, for all* PPT *algorithms* $\mathcal{M}$*, with all but negligible probability over* pk *output by* $\mathcal{K}(1^k)$*,* σ, s *output by* $\mathcal{SIM}_1(pk)$*, and uniformly-distributed* r, r'*, machine* $\mathcal{KE}^*$ *satisfies the following:*

> *Assume* $\mathcal{M}$ *(using random tape* r'*) acts as a receiver with* $\mathcal{SIM}_2(s; r)$ *on joint input* pk, σ, C *and simultaneously as a sender with* $\mathcal{R}$ *on joint input* pk, σ, C' *(where* C *is a valid ciphertext and* C, C' *are adaptively chosen by* $\mathcal{M}$*). Let the transcripts of these two interactions be* π *and* π'*. Denote by* p^* *the probability (over the random tape of* $\mathcal{R}$*) that* $\mathcal{R}$ *accepts in the above interaction and* $\pi \neq \pi'$*. On input* pk, σ, s, r*, and access to* $\mathcal{M}_{r'}$*, the probability that* $\mathcal{KE}^*$ *outputs a witness to the decryption of* C' *under* pk *is at least:*
>
> $$p^* - \kappa^*(pk) - \varepsilon^*(|pk|).$$

Our definitions of zero-knowledge (in Definition 1) and non-malleability (in Definition 2) both consider the single-theorem case only. The definitions may be modified for the multi-theorem case; however, the present definitions suffice for our intended applications.

2.2 A Note on Complexity Assumptions

Our number-theoretic complexity assumptions are with respect to adversaries permitted to run in *expected polynomial time*. For example, we assume that RSA inverses cannot be computed with more than negligible probability by any expected polynomial-time algorithm. The reason for this is our reliance on constant-round proofs of knowledge, for which only expected polynomial-time knowledge extractors are currently known. For simplicity, security definitions of *protocols* are stated with respect to PPT adversaries. Thus, a deniable authentication protocol is called secure (cf. Appendix A) if any PPT adversary attacking the protocol has negligible success probability.

3 A Non-Malleable Proof of Plaintext Knowledge

In the present abstract we describe the RSA-based construction only. Non-malleable PPKs for the Rabin (based on the hardness of factoring), Paillier (based on the computational composite residuosity assumption), and El Gamal (based on the computational Diffie-Hellman assumption) encryption schemes appear in the full version of this work [28, Chapter 5].

We begin with an overview of our technique. Recall the parameter σ which is shared by the sender and receiver and which is used as a common input during execution of the PPK. Embedded in σ will be a particular value y for which the simulator knows a witness x such that $R(y, x) = 1$. A PPK for ciphertext C will be a witness indistinguishable proof of knowledge of *either* a witness to the decryption of C *or* a witness for y, using the generic techniques for constructing such proofs [8]. Note that soundness of the protocol is not affected since a PPT adversary cannot derive a witness for y, while ZK simulation is easy for a simulator who knows the witness x.

As stated, this simple approach does not suffice to achieve non-malleability. To see why, consider a simulator interacting with man-in-the-middle $\mathcal{M}$ while $\mathcal{M}$ simultaneously interacts with verifier $\mathcal{V}$. Since the simulator must simulate a proof of "w or x" (where w is the witness to the decryption of the ciphertext C chosen by the adversary), the simulator must know x. However, if we use the knowledge extractor to extract from $\mathcal{M}$, who is proving "w' or x" (where w' is the witness to the decryption of C'), there is nothing which precludes extracting x! Note that without initial knowledge of x the simulator cannot properly perform the simulation; yet, if the simulator initially knows x, there is no contradiction in extracting this value from $\mathcal{M}$. Thus, a more careful approach is needed.

To overcome this obstacle, we borrow a technique used previously by Di Crescenzo, et al. [14]. Namely, the value y will depend on a parameter α which $\mathcal{M}$ cannot re-use; thus, the simulator proves knowledge of "w or x_α" while $\mathcal{M}$ is forced to prove knowledge of "w' or $x_{\alpha'}$", for some $\alpha' \neq \alpha$. The resulting proof will be non-malleable if the following conditions hold: (1) it is possible for the simulator to know the witness x_α; yet (2) learning $x_{\alpha'}$ for *any* $\alpha' \neq \alpha$ results in a contradiction; furthermore, (3) $\mathcal{M}$ cannot duplicate the value α used by the simulator. Details follow in the remainder of this section.

We briefly review the RSA cryptosystem, extended to allow encryption of ℓ-bit messages using the techniques of Blum and Goldwasser [4]. The public key N is chosen as a product of two random $k/2$-bit primes (where k is the security parameter), and e is a prime number such that $|e| = O(k)$.[6] Let $hc(\cdot)$ be a hard-core bit [24] for the RSA permutation (so that, given r^e, $hc(r)$ is computationally indistinguishable from random; note that $hc(\cdot)$ may depend on information included with the public parameters), and define $hc^*(r) \stackrel{\text{def}}{=} hc(r^{e^{\ell-1}}) \circ \cdots \circ hc(r^e) \circ hc(r)$. Encryption of ℓ-bit message m is done by choosing a random element $r \in \mathbb{Z}_N^*$, computing $C = r^{e^\ell} \bmod N$, and sending $\langle C, c \stackrel{\text{def}}{=} hc^*(r) \oplus m \rangle$. It is easily shown that this scheme is semantically secure under the RSA assumption.

Our protocol uses a Σ-protocol for proving knowledge of e^ℓ-th roots based on the Σ-protocol for proving knowledge of e-th roots [23]. To prove knowledge of $r = C^{1/e^\ell}$, the prover chooses a random element $r_1 \in \mathbb{Z}_N^*$ and sends $A = r_1^{e^\ell}$ to the verifier. The verifier replies with a challenge q selected randomly from $\mathbb{Z}_e$. The prover responds with $R = r^q r_1$ and the receiver verifies that $R^{e^\ell} \stackrel{?}{=} C^q A$. To see that special soundness holds, consider two accepting conversations (A, q, R) and (A, q', R'). Since $R^{e^\ell} = C^q A$ and $(R')^{e^\ell} = C^{q'} A$ we have $(R/R')^{e^\ell} = C^{q-q'}$. Noting that $|q - q'|$ is relatively prime to e^ℓ, standard techniques may be used to compute the desired witness C^{1/e^ℓ} [23]. Special honest-verifier zero knowledge is demonstrated by the simulator which, on input C and a "target" challenge q, chooses random $R \in \mathbb{Z}_N^*$, computes $A = R^{e^\ell}/C^q$, and outputs the transcript (A, q, R).

We now describe the non-malleable PPK in detail (cf. Figure 1). Parameters σ are generated by selecting two random elements $g, h \in \mathbb{Z}_N^*$. Additionally, a function $H : \{0,1\}^* \to \mathbb{Z}_e$ from a family of universal one-way hash functions is chosen at random. Once σ is established, a PPK for ciphertext $\langle C, c \rangle$ proceeds as follows: first, a key-generation algorithm for a one-time signature scheme is run to yield verification key VK and signing key SK, and $\alpha = H(\mathsf{VK})$ is computed. The PPK will be a witness indistinguishable proof of knowledge of *either* $r = C^{1/e^\ell}$ *or* $x_\alpha \stackrel{\text{def}}{=} (g^\alpha h)^{1/e}$. In more detail, the sender chooses

[6] A 3-round protocol for the case of small e (e.g., $e = 3$) is also possible, but we omit details here.

Public key: N; prime e

$\sigma:\ g,h \in \mathbb{Z}_N^*;\ H:\{0,1\}^* \to \mathbb{Z}_e$

$\mathcal{S}$ (input $m \in \{0,1\}^\ell$)		$\mathcal{R}$
$(\mathsf{VK},\mathsf{SK}) \leftarrow \mathsf{SigGen}(1^k)$		
$r, r_1, R_2 \leftarrow \mathbb{Z}_N^*; q_2 \leftarrow \mathbb{Z}_e$		
$C := r^{e^\ell}; c := hc^*(r) \oplus m$		
$\alpha := H(\mathsf{VK})$		
$A_1 := r_1^{e^\ell}$		
$A_2 := R_2^e / (g^\alpha h)^{q_2}$	$\xrightarrow{\mathsf{VK}, C, c, A_1, A_2}$	
	$\xleftarrow{q}$	$q \leftarrow \mathbb{Z}_e$
$q_1 := q - q_2 \bmod e$		
$R_1 := r^{q_1} r_1$		
$s \leftarrow \mathsf{Sign}_{\mathsf{SK}}(\text{transcript})$	$\xrightarrow{q_1, R_1, R_2, s}$	Verify: $R_1^{e^\ell} \stackrel{?}{=} C^{q_1} A_1$
		$R_2^e \stackrel{?}{=} \left(g^{H(\mathsf{VK})} h\right)^{(q - q_1 \bmod e)} A_2$
		$\mathsf{Vrfy}_{\mathsf{VK}}(\text{transcript}, s) \stackrel{?}{=} 1$

Figure 1: Non-malleable PPK for the RSA cryptosystem.

random elements $r_1, R_2 \in \mathbb{Z}_N^*$ along with a random element $q_2 \in \mathbb{Z}_e$. The sender computes $A_1 = r_1^{e^\ell}$ and $A_2 = R_2^e/(g^\alpha h)^{q_2}$. These values are sent (along with VK, C, c) as the first message of the PPK. The receiver chooses challenge $q \in \mathbb{Z}_e$ as before. The sender sets $q_1 = q - q_2 \bmod e$ and answers with $R_1 = r^{q_1} r_1$ (completing the "real" proof of knowledge with challenge q_1) and R_2 (completing the "simulated" proof of knowledge with challenge q_2). The values q_1, R_1, R_2 are sent to the receiver. To complete the proof, the sender signs a transcript of the entire execution (including C, c) using SK and sends the signature to the receiver. The receiver verifies the correctness of the proofs by checking that $R_1^{e^\ell} \stackrel{?}{=} C^{q_1} A_1$ and $R_2^e \stackrel{?}{=} (g^\alpha h)^{(q - q_1 \bmod e)} A_2$. Finally, the receiver verifies the correctness of the signature on the transcript.

For greater efficiency, it is possible to modify the protocol for the case of small e (e.g., $e = 3$) without increasing the round complexity. We omit details in the present abstract.

Theorem 1 *Assuming the hardness of the RSA problem for expected-polynomial-time algorithms, the protocol of Figure 1 is a non-malleable PPK (with $\kappa^*(pk) = 1/e$) for the RSA encryption scheme outlined above.*

Proof That the protocol is indeed a PPK will follow from the stronger properties we prove below. We use the following simulator: $\mathcal{SIM}_1(N, e)$ chooses random hash function H, runs $\mathsf{SigGen}(1^k)$ to generate $(\mathsf{VK}, \mathsf{SK})$, and computes $\alpha = H(\mathsf{VK})$. Random elements $g, x \in \mathbb{Z}_N^*$ are chosen, and h is set equal to $g^{-\alpha} x^e$. Finally, $\sigma = \langle g, h, H \rangle$ is output along with state information $\mathsf{state} = \langle \mathsf{VK}, \mathsf{SK}, x \rangle$. Note that σ output by $\mathcal{SIM}_1$ has the correct distribution. Furthermore, given state, $\mathcal{SIM}_2$ can simulate the witness indistinguishable proof of Figure 1 for any ciphertext: simply use verification key VK and then the witness $x = (g^\alpha h)^{1/e}$ is known. Note that the resulting simulation is perfect and is achieved without rewinding the (potentially) dishonest verifier.

Fix pk, σ, state, and randomness r for $\mathcal{SIM}_2$. We are given adversary $\mathcal{M}$ using (unknown) random tape r' who interacts with both $\mathcal{SIM}_2(\mathsf{state}; r)$ and honest receiver $\mathcal{R}$. Recall that ciphertext $\langle C, c \rangle$, for which $\mathcal{SIM}_2$ will be required to prove a witness, is chosen adaptively by $\mathcal{M}$. Once the challenge q' of $\mathcal{R}$ is fixed, the entire interaction is completely determined; thus, we may define $\pi(q')$ as the transcript of the conversation between $\mathcal{SIM}_2(\mathsf{state}; r)$ and $\mathcal{M}_{r'}$ when q' is the challenge sent by $\mathcal{R}$; analogously, we define $\pi'(q')$ as the transcript of the conversation between $\mathcal{M}_{r'}$ and $\mathcal{R}$ when q' is the challenge of $\mathcal{R}$.

The knowledge extractor $\mathcal{KE}^*$ is given $pk, \sigma, \mathsf{state}, r$, and access to $\mathcal{M}_{r'}$. When we say that $\mathcal{KE}^*$ *runs* $\mathcal{M}_{r'}$ *with challenge* q we mean that $\mathcal{KE}^*$ interacts with $\mathcal{M}_{r'}$ by running algorithm $\mathcal{SIM}_2(\mathsf{state}; r)$ and sending challenge q for $\mathcal{R}$. We stress that interleaving of messages (i.e., scheduling of messages to/from $\mathcal{R}$ and $\mathcal{SIM}_2$) is completely determined by $\mathcal{M}_{r'}$. $\mathcal{KE}^*$ operates as follows: First, $\mathcal{KE}^*$ picks a random value ${q'}^1 \in \mathbb{Z}_e$ and runs $\mathcal{M}_{r'}$ with challenge ${q'}^1$ (cf. Figure 2). If $\pi'({q'}^1)$ is not accepting, or if $\pi'({q'}^1) = \pi({q'}^1)$, stop and output $\perp$. Otherwise, run the following:

For $i = 0$ to $e - 1$:
  ${q'}^2 \leftarrow \mathbb{Z}_e$
  Run $\mathcal{M}_{r'}$ with challenge ${q'}^2$
  If $\pi'({q'}^2)$ is accepting and $\pi({q'}^2) \neq \pi'({q'}^2)$ and ${q'}^2 \neq {q'}^1$:
    Output $\pi'({q'}^2)$ and stop
  Run $\mathcal{M}_{r'}$ with challenge i
  If $\pi'(i)$ is accepting and $\pi(i) \neq \pi'(i)$ and $i \neq {q'}^1$:
    Output $\pi'(i)$ and stop
Output $\perp$ and stop.

At this point, $\mathcal{KE}^*$ has either output $\perp$ or has the transcripts $\pi({q'}^1), \pi'({q'}^1), \pi'({q'}^2)$, where $\pi({q'}^1) \neq \pi'({q'}^1)$ and $\pi'({q'}^1), \pi'({q'}^2)$ are accepting transcripts with ${q'}^1 \neq {q'}^2$ (see Figure 2).

We first verify that the expected running time of $\mathcal{KE}^*$ until this point is polynomial in k. Fix $pk, \sigma, \mathsf{state}$, and r as above. Let p^* be the probability (over challenges q sent by $\mathcal{R}$) that $\mathcal{M}_{r'}$ gives a valid proof and $\pi(q) \neq \pi'(q)$. If $p^* > 1/e$, the expected number of iterations of the loop above (assuming this loop is executed) is at most $2/p^*$; furthermore, the probability of executing this loop (which is only done following an initial success) is exactly p^*. Thus, the expected running time is upper-bounded by $p^* \cdot \frac{poly(k)}{p^*} = poly(k)$, where $poly(k)$ is an upper bound on the running time of $\mathcal{M}$. On the other hand, if $p^* \leq 1/e$, the number of iterations of the loop above is at most e, yet the probability of executing the loop is at most $1/e$. Thus, the expected running time of $\mathcal{KE}^*$ in this case is at most $\frac{1}{e} \cdot e \cdot poly(k) = poly(k)$.

Let Good be the event that $\mathcal{KE}^*$ does not output $\perp$. In this case, let the transcripts be as indicated in Figure 2. Note that the probability of event Good is exactly p^* when $p^* > 1/e$ and 0 otherwise. In either case, we have $\Pr[\mathsf{Good}] \geq p^* - 1/e$.

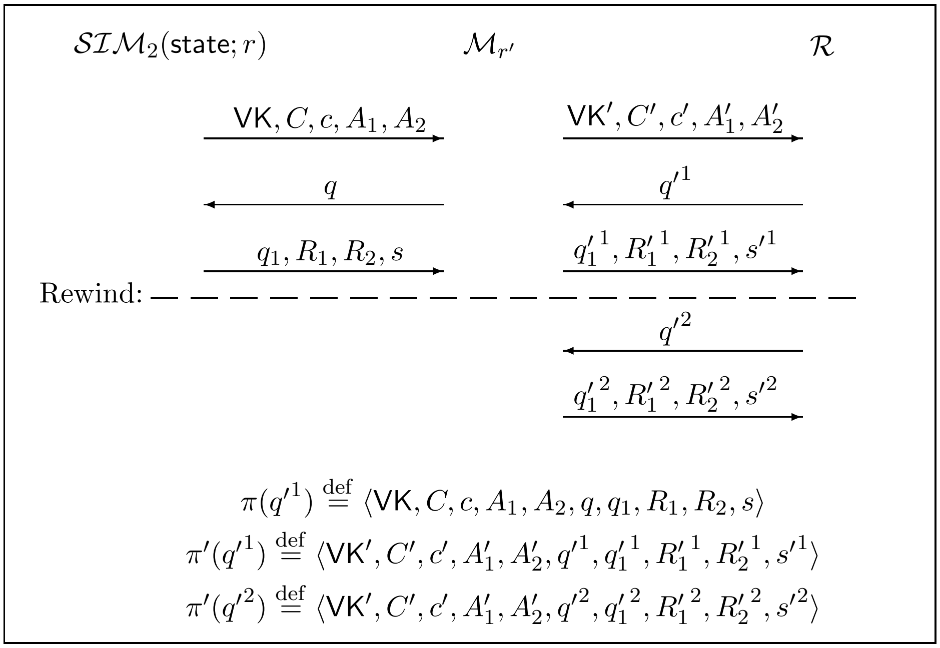

Figure 2: Knowledge extraction.

Assuming event Good occurs, $\pi'({q'}^1)$ and $\pi'({q'}^2)$ are accepting transcripts with ${q'}^1 \neq {q'}^2$ and therefore we must have either ${q'_1}^1 \neq {q'_1}^2$ or ${q'}^1 - {q'_1}^1 \neq {q'}^2 - {q'_1}^2 \bmod e$ (or possibly both). In case ${q'}^1 - {q'_1}^1 = {q'}^2 - {q'_1}^2 \bmod e$ (denote this event by Real) and hence ${q'_1}^1 \neq {q'_1}^2$, we have the two equations:

$$\begin{aligned} \left({R'_1}^1\right)^{e^\ell} &= (C')^{{q'_1}^1} A'_1 \bmod N \\ \left({R'_1}^2\right)^{e^\ell} &= (C')^{{q'_1}^2} A'_1 \bmod N. \end{aligned}$$

Therefore, $(\Delta_R)^{e^\ell} = (C')^{\Delta_q}$, where $\Delta_R \stackrel{\text{def}}{=} {R'_1}^1/{R'_1}^2$ (we assume all values in $\mathbb{Z}_N$ are invertible; if not, N may be factored and e^{th} roots easily computed) and $\Delta_q \stackrel{\text{def}}{=} {q'_1}^1 - {q'_1}^2$. Since Δ_q and e^ℓ are relatively prime, we can efficiently compute integers A, B such that $A\Delta_q + Be^\ell = 1$; we may then compute $r' \stackrel{\text{def}}{=} (\Delta_R)^A (C')^B = (C')^{1/e^\ell}$. Once r' is known, $\mathcal{KE}^*$ may compute $hc^*(r')$ and hence determine the witness to the decryption of ciphertext $\langle C', c' \rangle$.

On the other hand, if ${q'}^1 - {q'_1}^1 \neq {q'}^2 - {q'_1}^2 \bmod e$ (denote this event by Fake), we have the two equations:

$$\begin{aligned} \left({R'_2}^1\right)^{e} &= \left(g^{\alpha'} h\right)^{{q'}^1 - {q'_1}^1} A'_2 \\ \left({R'_2}^2\right)^{e} &= \left(g^{\alpha'} h\right)^{{q'}^2 - {q'_1}^2} A'_2, \end{aligned}$$

where $\alpha' \stackrel{\text{def}}{=} H(\mathsf{VK}')$, from which $\mathcal{KE}^*$ may compute $y' \stackrel{\text{def}}{=} (g^{\alpha'} h)^{1/e}$ (using the same techniques as above). Since $\mathsf{Good} = \mathsf{Real} \cup \mathsf{Fake}$, we have $\Pr[\mathsf{Real}] + \Pr[\mathsf{Fake}] \geq p^* - 1/e$. To complete the proof (since $\mathsf{Real} \cap \mathsf{Fake} = \emptyset$), we show that $\Pr[\mathsf{Fake}]$ (where the probability is over the random tape ω of $\mathcal{KE}^*$) is less than some negligible function with all but negligible probability over choice of $pk, \sigma, \mathsf{state}, r, r'$.

Claim 1 $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[\mathsf{VK} = \mathsf{VK}' \wedge \mathsf{Good}]$ *is negligible.*

Consider algorithm Forge which takes input VK, has access to a signing oracle, and runs as follows: emulating $\mathcal{KE}^*$, algorithm Forge generates pk, σ, and parameters state and r for $\mathcal{SIM}_2$. However, Forge does not run the key-generation procedure for the one-time signature scheme, but instead uses VK. Next, Forge runs the initial portion of $\mathcal{KE}^*$ by choosing random r' for $\mathcal{M}$, choosing q' randomly from $\mathbb{Z}_e$, and running $\mathcal{M}_{r'}$ with challenge q'. When a signature under VK is needed (during the single execution of $\mathcal{SIM}_2$), Forge obtains the required signature from its signing oracle. Once $\mathcal{M}_{r'}$ completes its execution, Forge stops and outputs the transcript $\pi'(q')$.

The probability that the transcript output by Forge contains a valid forgery under VK is at least $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[\mathsf{VK} = \mathsf{VK}' \wedge \mathsf{Good}]$. However, the security of the one-time signature scheme guarantees that this is negligible. Note that, since no rewinding is involved and consequently Forge runs in strict polynomial time, the signature scheme need only be secure against PPT adversaries. □

Claim 2 $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[\mathsf{VK} \neq \mathsf{VK}' \wedge H(\mathsf{VK}) = H(\mathsf{VK}') \wedge \mathsf{Good}]$ *is negligible.*

The proof is similar to that of Claim 1, but is based on the security of the family of universal one-way hash functions. As in Claim 1, the family of universal one-way hash functions need only be secure against PPT adversaries. Details omitted. □

Claim 3 $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[H(\mathsf{VK}) \neq H(\mathsf{VK}') \wedge \mathsf{Fake}]$ *is negligible.*

Note that algorithm $\mathcal{KE}^*$, as described previously, does not require any secret information about N, e, g in order to run. Thus, we can consider the expected polynomial-time algorithm $\mathcal{KE}'$ which takes as input a modulus N, a prime e, and a (random) element $g \in \mathbb{Z}_N^*$ and otherwise runs identically to $\mathcal{KE}^*$. Clearly, the probability that both Fake and $H(\mathsf{VK}) \neq H(\mathsf{VK}')$ occur remains unchanged.

Let $\alpha = H(\mathsf{VK})$ and $\alpha' = H(\mathsf{VK}')$; define $\Delta \stackrel{\text{def}}{=} \alpha' - \alpha \neq 0$. By definition of event Fake, $\mathcal{KE}'$ may compute y' such that $(y')^e = g^{\alpha'} h$; but then:

$$y' \stackrel{\text{def}}{=} \left(g^{\alpha'} h\right)^{1/e} = \left(g^{\Delta} x^e\right)^{1/e} = \left(g^{\Delta}\right)^{1/e} x,$$

and therefore $y \stackrel{\text{def}}{=} y'/x$ satisfies $y^e = g^{\Delta}$. Note that $|\Delta|$ and e are relatively prime since $\Delta \in (-e, e)$. The same techniques used above allow efficient computation of $g^{1/e}$. In other words, whenever Fake and $\alpha' \neq \alpha$ occur, algorithm $\mathcal{KE}'$ inverts the given RSA instance g. Therefore, under the RSA assumption, $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[H(\mathsf{VK}) \neq H(\mathsf{VK}') \wedge \mathsf{Fake}]$ must be negligible. □

Claims 1–3 imply that $\Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[\mathsf{Fake}] < \varepsilon(k)$ for some negligible function $\varepsilon(\cdot)$. But then:

$$\varepsilon(k) > \Pr_{pk,\sigma,\mathsf{state},r,r',\omega}[\mathsf{Fake}] > \sqrt{\varepsilon(k)} \cdot \Pr_{pk,\sigma,\mathsf{state},r,r'}\left[\Pr_{\omega}[\mathsf{Fake}] > \sqrt{\varepsilon(k)}\right],$$

and hence $\Pr_{pk,\sigma,\mathsf{state},r,r'}\left[\Pr_\omega[\mathsf{Fake}] \leq \sqrt{\varepsilon(k)}\right] \geq 1 - \sqrt{\varepsilon(k)}$. Thus, with all but negligible probability over choice of $pk, \sigma, \mathsf{state}, r, r'$, the probability of event Fake is negligible ■

4 Applications

In this section, we discuss applications of non-malleable PPKs to the construction of (1) interactive CCA2 encryption protocols, (2) password-AKE in the public-key model, and (3) strong deniable-authentication protocols. We describe protocols based on the PPK of Figure 1 whose security is therefore based on the RSA assumption; analogous constructions based on the PPKs given in the full version of this work [28] yield protocols whose security is based on alternate assumptions (hardness of factoring, etc.).

Concurrent composition. In our intended applications, the man-in-the-middle adversary may conduct multiple PPKs in an asynchronous and concurrent fashion. Witness extraction will be required for each such execution; furthermore, extraction of this witness will be required as soon as the relevant proof is completed. If arbitrary interleaving of the proofs is allowed, extracting all witnesses in the naive way may require exponential time due to the nested rewinding of the prover (a similar problem is encountered in simulation of concurrent zero-knowledge proofs [17]). To avoid this problem, we introduce *timing constraints* [17] in our protocols. These are explained in detail below.

4.1 Interactive CCA2 Encryption

The non-malleable PPK of the previous section (cf. Figure 1) immediately yields an interactive encryption scheme: The receiver generates N, e as in the original RSA scheme and also runs $\mathcal{G}(pk)$ to give parameters σ. The public key pk is (N, e, σ) and the secret key is the factorization of N. To encrypt message m under public key pk, the sender computes $C = r^{e^\ell}$ and $c = hc^*(r) \oplus m$, sends $\langle C, c\rangle$ to the receiver, and then executes algorithm $\mathcal{S}$ for the ciphertext using parameters σ (i.e., the sender proves knowledge of a witness to the decryption of C). To decrypt, the receiver uses $\mathcal{R}$ to determine whether to accept or reject the proof; if the receiver accepts, the receiver decrypts $\langle C, c\rangle$ as in the standard RSA scheme. If the proof is rejected, the receiver outputs $\perp$.

This interactive scheme, as described, is secure against adaptive chosen-ciphertext attacks when the adversary is given sequential access to the decryption oracle. To ensure security against an adversary given *concurrent* access to the decryption oracle, timing constraints are necessary. In particular, we require that $\mathcal{S}$ respond to the challenge (i.e., send the third message of the protocol) within time α from when the second message of the protocol is sent. If $\mathcal{S}$ does not respond in this time, the proof is rejected. Additionally, a fourth message is sent from the receiver to the sender; this message is simply an *acknowledgment* message which is ack if the sender's proof was verified to be correct and $\perp$ otherwise. Furthermore, $\mathcal{R}$ delays the sending of this message until at least time β has elapsed from when the second message of the protocol was sent (with $\beta > \alpha$). We stress that, when concurrent access to the decryption oracle is allowed, the decryption oracle enforces the above timing constraints by (1) rejecting any proofs for which more than time α has elapsed between sending the second message and receiving the third message, and (2) the decrypted ciphertext is not returned to the adversary until after the acknowledgment message is sent (in particular, until time β has elapsed since sending the second message).

Theorem 2 *Assuming the hardness of the RSA problem for expected-polynomial-time algorithms, the protocol of Figure 1 (with $|e| = \Theta(k)$) is an interactive encryption scheme secure against sequential chosen-ciphertext attacks. If timing constraints are enforced as outlined above, the protocol is secure against concurrent chosen-ciphertext attacks.*

The proof appears in Appendix B.

The resulting encryption scheme is computationally efficient, especially in comparison with the basic semantically-secure scheme. For example, the scheme of Figure 1 for encryption of ℓ-bit messages requires only $2\ell+4$ exponentiations compared to the ℓ exponentiations required by the underlying RSA scheme. Furthermore, most of the additional computation may be done in a preprocessing stage before the message to be sent is known.

4.2 Password-Based Authentication and Key Exchange

As shown by Halevi and Krawczyk [27] and Boyarsky [5], protocols for password-based authentication may be constructed from any CCA2 encryption scheme as follows: Let pw be the password of the user which is stored by the server. A password-based authentication protocol using a (non-interactive) CCA2 encryption scheme has the server send a random, sufficiently-long nonce n to the user, who replies with an encryption of $pw \circ n$ (actually, this brief description suppresses details which are unimportant for the discussion which follows; see [27, 5]). The server decrypts and verifies correctness of the password and the nonce; the nonce is necessary to prevent replay attacks. If desired, a key K may be exchanged by encrypting $K \circ pw \circ n$ instead.

Our interactive CCA2 encryption schemes allow the first efficient implementations of these protocols based on, for example, the RSA or factoring assumptions. Note that these protocols already require interaction and therefore using an interactive encryption scheme is not a serious drawback. Furthermore, when an interactive encryption scheme is used the nonce is not necessary if the probability that a server repeats its messages is negligible [5]; in this case, authentication (for example) may be achieved by simply having the user perform a (random) encryption of pw. Thus, our constructions require only one more round than those of [27, 5].

4.3 Deniable Authentication

Our non-malleable PPKs may be easily adapted to give deniable-authentication protocols whose security is based on the *one-wayness* of the appropriate encryption scheme; semantic security is not required. This allows for very efficient deniable-authentication protocols since, for example, we may use the "simple" RSA encryption scheme in which r is encrypted as $r^e \bmod N$ (under the RSA assumption, this scheme is one-way). Furthermore, efficient deniable-authentication protocols may be constructed using weaker assumptions; e.g., the CDH assumption [28] (previous efficient constructions require the DDH assumption).

$$
\begin{array}{rcl}
\multicolumn{3}{c}{\text{Public key: } N;\ \text{prime } e;\ g,h \in \mathbb{Z}_N^*;\ H:\{0,1\}^* \to \mathbb{Z}_e} \\
\mathcal{V}\ (\text{input } m \in \{0,1\}^*) & & \mathcal{P}\ (\text{input } d) \\
(\mathsf{VK},\mathsf{SK}) \leftarrow \mathsf{SigGen}(1^k) & & \\
y, r_1, R_2 \leftarrow \mathbb{Z}_N^*; q_2 \leftarrow \mathbb{Z}_e & & \\
C := y^e; \alpha := H(\mathsf{VK}) & & \\
A_1 := r_1^e & & \\
A_2 := R_2^e / (g^\alpha h)^{q_2} & \xrightarrow{\ \mathsf{VK}, m, C, A_1, A_2\ } & \\
 & \xleftarrow{\qquad q \qquad} & q \leftarrow \mathbb{Z}_e \\
q_1 := q - q_2 \bmod e & & \\
R_1 := y^{q_1} r_1 & & \\
s \leftarrow \mathsf{Sign}_{\mathsf{SK}}(\text{transcript}) & \xrightarrow{\ q_1, R_1, R_2, s\ } & \text{Verify: } R_1^e \stackrel{?}{=} C^{q_1} A_1 \\
 & & R_2^e \stackrel{?}{=} \left(g^{H(\mathsf{VK})} h\right)^{(q - q_1 \bmod e)} A_2 \\
 & \xleftarrow{\qquad C^d \qquad} & \mathsf{Vrfy}_{\mathsf{VK}}(\text{transcript}, s) \stackrel{?}{=} 1
\end{array}
$$

Figure 3: A deniable-authentication protocol based on RSA.

We first present the paradigm for the construction of our deniable authentication protocols. The basic idea is for the receiver to give a non-malleable PPK for a ciphertext C encrypted using an encryption scheme which is one-way for random messages. Additionally, the message m which is being authenticated is included in the transcript and is signed along with everything else. Assuming the verifier's proof succeeds, the prover authenticates the message by responding with the decryption of C.

Figure 3 shows an example of this approach applied to the non-malleable PPK of Figure 1. The public key of the prover $\mathcal{P}$ is an RSA modulus N, a prime e (with $|e| = \Theta(k)$), elements $g, h \in \mathbb{Z}_N^*$, and a hash function H chosen randomly from a family of universal one-way hash functions. Additionally, the prover has secret key d such that $de = 1 \bmod \varphi(N)$. The verifier $\mathcal{V}$ has message m taken from an arbitrary message space (of course, $|m|$ must be polynomial in the security parameter). To have m authenticated by $\mathcal{P}$, the verifier chooses a random $y \in \mathbb{Z}_N^*$, computes $C = y^e$, and then performs a non-malleable proof of knowledge of the witness y to the decryption of C (as in Figure 1). Additionally, the message m is sent as the first message of the protocol, and is signed along with the rest of the transcript. If the verifier's proof succeeds, the prover computes C^d and sends this value to the verifier. If the proof does not succeed, the prover simply replies with $\perp$.

As in the case of interactive encryption, timing constraints are needed when concurrent access to the prover is allowed. In this case, we require that the verifier respond to the challenge (i.e., send the third message of the protocol) within time α from when the challenge was sent. If $\mathcal{V}$ does not respond within this time, the proof is rejected. Additionally, the last message of the protocol is not sent by the prover until at least time β has elapsed since sending the challenge (clearly, we must have $\beta > \alpha$).

Theorem 3 *Assuming the hardness of the RSA problem for expected-polynomial-time algorithms, the protocol of Figure 3 is a strong deniable-authentication protocol (over an arbitrary message space) for adversaries given sequential access to the prover. If timing constraints are enforced as outlined above, the protocol is a strong ε-deniable-authentication protocol for adversaries given concurrent access to the prover.*

The proof is similar to that of Theorem 2, and appears in the full version [28].

We stress that the resulting deniable authentication protocols are quite practical. For example, the full version of this work [28] shows a deniable authentication protocol based on the CDH assumption which has the same round-complexity, requires fewer exponentiations, has a shorter public key, and is based on a weaker assumption than the most efficient previously-known protocol for strong deniable authentication (i.e., the protocol of [17] instantiated with the Cramer-Shoup encryption scheme [9]). Furthermore, no previous efficient protocols were known based on the RSA, factoring, or computational/decisional composite residuosity assumptions.

We also remark that interactive CCA2 encryption schemes do *not*, in general, yield deniable authentication protocols using this paradigm. For example, when using the chosen-ciphertext-secure encryption scheme of [15], two additional rounds are necessary just to achieve *weak* deniable authentication (our use of the terms "strong" and "weak" is explained in Appendix A.2).

Further efficiency improvements are possible; however, the resulting protocols are provably secure for polynomial-sized message spaces only. In Figure 4, we illustrate the improvement for the deniable authentication protocol of Figure 3. In the improved protocol, we rely on the fact that the adversary cannot re-use the value m (since, to break the security of the scheme, the adversary must authenticate a *new* message which the prover does not authenticate). This eliminates the need for a verification key and a one-time signature on the transcript. However, since the adversary chooses m (and this value must be guessed by the simulator in advance), the scheme is only provably secure when the message space is polynomial-size.

Theorem 4 *Assuming the hardness of the RSA problem for expected-polynomial-time algorithms, the protocol of Figure 4 is a strong deniable-authentication protocol (over any polynomial-size message space $M_k \subset \mathbb{Z}_e$) for adversaries given sequential access to the prover. If timing constraints are enforced as outlined above, the protocol is a strong ε-deniable-authentication protocol for adversaries given concurrent access to the prover.*

The proof appears in the full version of this work [28].

Acknowledgments. I am grateful to Moti Yung and Rafail Ostrovsky for their many helpful comments and suggestions regarding the work described here.

References

[1] Y. Aumann and M.O. Rabin. A Proof of Plaintext Knowledge Protocol and Applications. Manuscript. June, 2001.

Public key: N; prime e; $g, h \in \mathbb{Z}_N^*$

$\mathcal{V}$ (input $m \in M_k \subset \mathbb{Z}_e$)		$\mathcal{P}$ (input d)
$y, r_1, R_2 \leftarrow \mathbb{Z}_N^*; q_2 \leftarrow \mathbb{Z}_e$		
$C := y^e; A_1 := r_1^e$		
$A_2 := R_2^e / (g^m h)^{q_2}$	$\xrightarrow{m, C, A_1, A_2}$	
	$\xleftarrow{q}$	$q \leftarrow \mathbb{Z}_e$
$q_1 := q - q_2 \bmod e$		
$R_1 := y^{q_1} r_1$	$\xrightarrow{q_1, R_1, R_2}$	Verify: $R_1^e \stackrel{?}{=} C^{q_1} A_1$
	$\xleftarrow{C^d}$	$R_2^e \stackrel{?}{=} (g^m h)^{(q - q_1 \bmod e)} A_2$

Figure 4: A deniable-authentication protocol with improved efficiency.

[2] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. Crypto '98.

[3] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. Crypto '92.

[4] M. Blum and S. Goldwasser. An Efficient Probabilistic Public-Key Encryption Scheme which Hides All Partial Information. Crypto '84.

[5] M. Boyarsky. Public-Key Cryptography and Password Protocols: The Multi-User Case. CCCS '99.

[6] R. Cramer. Modular Design of Secure Yet Practical Cryptographic Protocols. PhD Thesis, CWI and U. Amsterdam, 1996.

[7] R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. Eurocrypt '01.

[8] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. Crypto '94.

[9] R. Cramer and V. Shoup. A Practical Public-Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. Crypto '98.

[10] R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. Eurocrypt '02, to appear.

[11] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. Eurocrypt '00.

[12] A. De Santis and G. Persiano. Zero-Knowledge Proofs of Knowledge Without Interaction. FOCS '92.

[13] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-Interactive Zero Knowledge. Crypto '01.

[14] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and Non-Interactive Non-Malleable Commitment. Eurocrypt '01.

[15] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM J. Computing* 30(2): 391–437 (2000).

[16] C. Dwork and M. Naor. Zaps and Their Applications. FOCS '00.

[17] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. STOC '98.

[18] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. Crypto '98.

[19] T. El Gamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory* 31(4): 469–472 (1985).

[20] U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology* 1(2): 77–94 (1988).

[21] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Based on a Single Random String. FOCS '90.

[22] Z. Galil, S. Haber, and M. Yung. Symmetric Public-Key Encryption. Crypto '85.

[23] L.C. Guillou and J.-J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing Both Transmission and Memory. Eurocrypt '88.

[24] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, Cambridge, UK, 2001.

[25] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing but Their Validity or: All Languages in NP Have Zero-Knowledge Proof Systems. *JACM* 38(3): 691–729 (1991).

[26] S. Haber. Multi-Party Cryptographic Computations: Techniques and Applications. PhD Thesis, Columbia University, 1987.

[27] S. Halevi and H. Krawczyk. Public-Key Cryptography and Password Protocols. *ACM Transactions on Information and System Security*, 2(3): 230–268, 1999. Preliminary version in ACM CCCS '98.

[28] J. Katz. Efficient Cryptographic Protocols Preventing "Man-in-the-Middle" Attacks. PhD Thesis, Columbia University, 2002.

[29] M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure Against Chosen-Ciphertext Attack. STOC '90.

[30] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt '99.

[31] M. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, January 1979.

[32] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen-Ciphertext Attack. Crypto '91.

[33] R. Rivest, A. Shamir, and L.M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* 21(2): 120–126 (1978).

[34] A. Sahai. Non-Malleable Non-Interactive Zero-Knowledge and Adaptive Chosen-Ciphertext Security. FOCS '99.

[35] C.P. Schnorr. Efficient Identification and Signatures for Smart Cards. Crypto '89.

A Additional Definitions

A.1 Chosen-Ciphertext-Secure, Interactive Encryption

A number of definitional approaches to chosen-ciphertext security in the interactive setting are possible. For example, the notion of non-malleability [15] may be extended for the case of interactive encryption. An oracle-based definition is also possible, and we sketch such a definition here.[7]

We have a sender, a receiver (where the receiver has published public-key pk), and a man-in-the-middle adversary $\mathcal{M}$ who controls all communication between them. To model this, we define an *encryption oracle* and a *decryption oracle* to which $\mathcal{M}$ is given access. The encryption oracle $\mathcal{E}_{b,pk}$ plays the role of the sender. The adversary may interact with this oracle multiple times at various points during its execution, and may interleave requests to this oracle with requests to the decryption oracle in an arbitrary manner. At the outset of protocol execution, a bit b is chosen at random. An instance of the adversary's interaction with the encryption oracle proceeds as follows: first, the adversary chooses two messages m_0, m_1 and sends these to $\mathcal{E}_{b,pk}$. The oracle then executes the encryption protocol for message m_b. The adversary, however, need not act as an honest receiver. The oracle maintains state between the adversary's oracle calls, and the adversary may have multiple concurrent interactions with the oracle. When $\mathcal{E}_{b,pk}$ sends the final message for a given instance of its execution, we say that instance is *completed*.

The *decryption oracle* $\mathcal{D}_{sk}$ plays the role of a receiver. This oracle also maintains appropriate state between oracle calls, and the adversary may again have multiple concurrent interactions with this oracle. Furthermore, the adversary need not act as an honest sender. Each time a given decryption-instance is completed, the decryption oracle decrypts using sk and following the correct decryption protocol and then sends the result (i.e., a message or $\perp$) to the adversary.

The adversary succeeds if it can guess the bit b. Clearly, some limitations must be placed on the adversary's access to the decryption oracle or else the adversary may simply

[7] To obtain an equivalent definition using the language of non-malleability, we would need to define a notion of non-malleability with respect to *vectors* of ciphertexts. Such a definition becomes cumbersome in the interactive setting.

forward messages between $\mathcal{E}_{b,pk}$ and $\mathcal{D}_{sk}$ and therefore trivially determine b. At any point during the adversary's execution, the set of transcripts of completed encryption-instances of $\mathcal{E}_{b,pk}$ is well defined. Upon completing a decryption-instance, let $\{\pi_1, \ldots, \pi_\ell\}$ denote the transcripts of all completed encryption-instances. We allow the adversary to receive the decryption corresponding to a decryption-instance with transcript π' only if $\pi' \neq \pi_i$ for $1 \leq i \leq \ell$.

Definition 3 *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an interactive, public-key encryption scheme. We say that* Π *is* CCA2-secure *if, for any* PPT *adversary* A*, the following is negligible (in* k*):*

$$\left|\Pr\left[(sk, pk) \leftarrow \mathcal{K}(1^k); b \leftarrow \{0,1\} : A^{\mathcal{E}_{b,pk}, \mathcal{D}_{sk}}(1^k, pk) = b\right] - 1/2\right|,$$

where A*'s access to* $\mathcal{D}_{sk}$ *is restricted as discussed above.*

A.2 Deniable Authentication

We review the definition given in [17, 16]. Formal definitions appear in the full version of this work [28]. We have a prover P who has established a public key and is willing to authenticate messages to a verifier $\mathcal{V}$; however, P is *not* willing to allow the verifier to convince a third party (after the fact) that P authenticated anything. This is formalized by ensuring that any transcript of an execution of the authentication protocol could have been simulated by a verifier alone (without any access to P). Furthermore, an adversary $\mathcal{M}$ (acting as man-in-the-middle between P and a verifier) should not be able to authenticate a message m to the verifier which P does not authenticate for $\mathcal{M}$. More formally, a *strong* deniable authentication protocol should satisfy the following:

- Completeness. For any message m, if the prover and a verifier follow the protocol for authenticating m, then the verifier accepts.
- Soundness. Assume P concurrently authenticates any polynomial number of messages $m_1, m_2, \ldots$ chosen adaptively by a PPT adversary $\mathcal{M}$. Then $\mathcal{M}$ will succeed with at most negligible probability in authenticating a message $m \notin \{m_1, \ldots\}$ to an honest verifier.
- Strong Deniability. Assume P concurrently authenticates any polynomial number of messages chosen adaptively by a PPT adversary $\mathcal{V}'$. Then there exists an expected-polynomial-time simulator that, given black-box access to $\mathcal{V}'$, can output a transcript indistinguishable from a transcript of a real execution between $\mathcal{V}'$ and P.

A relaxation of the above definition which has been considered previously (e.g., [17]) allows the simulator to have access to the real P when producing the simulated transcript, but P authenticates some fixed sequence of messages independent of those chosen by $\mathcal{V}'$. We call this *weak* deniable authentication. In practice, weak deniability may not be acceptable because the protocol then leaves an undeniable record that P authenticated *something* (even if not revealing *what*). However, P may want to deny that any such interaction ever took place. Note that any solution based on non-malleable encryption [15, 17, 18, 16] which uses the interactive non-malleable encryption scheme suggested by [15] (which requires a signature from P) will achieve only weak deniability.

The notion of ε-*deniability* [17] requires that for any given $\varepsilon > 0$, there exists a simulator whose expected running time is polynomial in k and $1/\varepsilon$ and which outputs a simulated transcript such that the advantage of any poly-time algorithm in distinguishing real transcripts from simulated transcripts is negligibly close to ε.

B Proof of Theorem 2

A straightforward hybrid argument shows that it is sufficient to consider adversaries with only a single access to the encryption oracle. We prove security for the more challenging case of concurrent access to the decryption oracle. The protocol Π of Figure 1 is a PPK for encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ in which $\mathcal{K}(1^k)$ outputs as the public key a k-bit modulus N and a k-bit prime e. Encryption of ℓ-bit message m is done by choosing random $r \in \mathbb{Z}_N^*$ and sending $\tilde{C} = \langle r^{e^\ell}, hc^*(r) \oplus m \rangle$, where $hc^*(\cdot)$ is a hard-core function for the RSA permutation. Assuming the hardness of the RSA problem for expected-polynomial-time algorithms, $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is semantically secure against expected-polynomial-time adversaries. We transform any PPT adversary $\mathcal{A}$ mounting a CCA2 attack against Π into an expected-polynomial-time adversary $\mathcal{A}'$ attacking the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Furthermore, we show that the advantage of $\mathcal{A}'$ is not negligible if the advantage of $\mathcal{A}$ is not negligible. This will immediately imply CCA2 security of Π.

Let $t(k)$, which is polynomial in k, be a bound on the number of times $\mathcal{A}$ accesses the decryption oracle when run on security parameter 1^k; without loss of generality, we assume that $t(k) \geq k$ (for convenience, in the remainder of the proof we suppress the dependence on k and simply write t). In the real experiment Expt_0, the final output b' of $\mathcal{A}$ is completely determined by $pk' = \langle pk, \sigma \rangle$, random coins r' for $\mathcal{A}$, the vector of challenges $\vec{q} = q_1, \ldots, q_t$ used during the t instances $\mathcal{A}$ interacts with the decryption oracle, and the randomness used by the encryption oracle (this includes the bit b, the randomness ω used for the encryption, and the randomness used for execution of the PPK). Let Succ denote the event that $b' = b$, and let $\Pr_0[\mathsf{Succ}]$ denote the probability of this event in the real experiment.

We modify the real experiment, giving Expt_1, as follows. Key generation is done by running $\mathcal{K}(1^k)$ to generate pk, sk. Additionally, $\mathcal{SIM}_1(pk)$ (cf. the proof of Theorem 1) is run to generate parameters σ and state (in the real experiment σ was generated by $\mathcal{G}(pk)$). The public key pk' is $\langle pk, \sigma \rangle$ and the secret key is sk. The adversary's calls to the decryption oracle are handled as in Expt_0 (in particular, any ciphertext $\tilde{C}$ may be decrypted since sk is known), but the adversary's encryption oracle call will be handled differently. When $\mathcal{A}$ calls the encryption oracle on messages m_0, m_1, we pick b randomly, compute $\tilde{C}^* = \mathcal{E}_{pk}(m_b; \omega)$ for random ω, and simulate the PPK for $\tilde{C}^*$ using algorithm $\mathcal{SIM}_2(\mathsf{state}; r)$ with randomly-chosen r. Now, the final output b' of $\mathcal{A}$ is completely determined by $pk' = \langle pk, \sigma \rangle$, random coins r' for $\mathcal{A}$, the vector of challenges $\vec{q}$ used by the decryption oracle, the values b and ω used in computing $\tilde{C}^*$, and the values state and r used by $\mathcal{SIM}_2$ in simulating the encryption oracle. Since $\mathcal{SIM}$ yields a perfect simulation of a real execution of the PPK, we have $\Pr_1[\mathsf{Succ}] = \Pr_0[\mathsf{Succ}]$, where the first probability refers to the probability of an event in Expt_1.

We now describe our adversary $\mathcal{A}'$ attacking the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Given the public key pk, adversary $\mathcal{A}'$ runs $\mathcal{SIM}_1(pk)$ to generate parameters σ and state. $\mathcal{A}'$ then fixes the randomness r' of $\mathcal{A}$, and runs $\mathcal{A}$ on input $pk' = \langle pk, \sigma \rangle$. Simulation of the encryption oracle for $\mathcal{A}$ is done as follows: when $\mathcal{A}$ submits two messages m_0, m_1, adversary $\mathcal{A}'$ simply forwards these to its encryption oracle and receives in return a ciphertext $\tilde{C}^*$ (note that the encryption oracle thus implicitly defines values b^* and ω^*). Then, $\mathcal{A}'$ simulates the PPK for $\tilde{C}^*$ using algorithm $\mathcal{SIM}_2(\mathsf{state}; r)$ for randomly-chosen coins r. Simulation of the decryption oracle for $\mathcal{A}$ is done by choosing a random vector of queries $\vec{q}^*$ and attempting to extract the relevant witnesses (in expected polynomial time) from the PPKs given by $\mathcal{A}$. Details of the simulation are described below. In case the simulation is successful, the final output b' is just the final output of $\mathcal{A}$; if the simulation is not successful, the final output b' is a randomly-chosen bit. As we show below, the simulation will succeed with sufficiently high probability such that if the advantage of $\mathcal{A}$ (in attacking the CCA2 security of Π) is not negligible then the advantage of $\mathcal{A}'$ (in attacking the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is not negligible as well. This will complete the proof.

It remains to show how to simulate the decryption oracle. Our proof requires techniques used in an analysis of concurrent composition of zero-knowledge proofs [17]. We assume that $\mathcal{A}$ controls the scheduling of all messages to and from all the oracles; so, for example, the decryption oracle does not send its next message until $\mathcal{A}$ requests it. Define the i^{th} *instance* of the decryption oracle as the i^{th} time $\mathcal{A}$ requests the second message (i.e., the challenge) of the PPK be sent by the decryption oracle. In any transcript of the execution of $\mathcal{A}$, we let C_i denote the ciphertext sent by $\mathcal{A}$ in the i^{th} instance of the decryption oracle. For any instance of the decryption oracle, we say the instance *succeeds* if (1) an honest receiver would accept the instance, (2) the transcript of the instance is different from the transcript (if it yet exists) of the interaction of $\mathcal{A}$ with the encryption oracle, and (3) the timing constraints are satisfied for that instance. Otherwise, we say the instance *fails.*

Recall that the simulator has values $\langle pk, \sigma, r', \vec{q}^*, \mathsf{state}, r\rangle$ and has access to an encryption oracle which, on input m_0, m_1, outputs $\mathcal{E}_{pk}(m_{b^*}; \omega^*)$ for random b^* and ω^*. Note that the value state defines a key VK which is used by $\mathcal{SIM}_2$ when giving its simulated proof. The values $pk, \sigma, r', \mathsf{state}$, and r are fixed throughout the simulation. When we say the simulator *interacts with* $\mathcal{A}$ *using* $\langle \vec{q}, \omega, b\rangle$ we mean that the simulator runs $\mathcal{A}$ as in Expt_1; that is, encryption oracle query m_0, m_1 is answered by encrypting m_b using randomness ω and then running $\mathcal{SIM}_2(\mathsf{state}; r)$, and the challenge sent by the i^{th} instance of the decryption oracle is q_i. We note that decryption requests cannot be immediately satisfied; this will not be a problem, as we show below.

We begin with simulation of the first instance. The simulator chooses random $\vec{q}, \omega, b$ and interacts with $\mathcal{A}$ using $\langle \vec{q}, \omega, b\rangle$. If $\mathcal{A}$ makes a call m_0, m_1 to the encryption oracle *before* q_1 is requested, the simulator forwards m_0, m_1 to its encryption oracle and receives in return a ciphertext $\tilde{C}^*$; we then say the *ciphertext is defined at instance 1.* Once the ciphertext is defined, the simulator no longer needs to choose values ω, b and, in effect, interacts with $\mathcal{A}$ using $\langle \vec{q}, \omega^*, b^*\rangle$. If the verification key VK_1 used by $\mathcal{A}$ in the first instance is equal to VK defined by state, the first instance is declared *conditionally delinquent* and the simulator proceeds to simulation of instance 2.

If the ciphertext is not defined at instance 1, the simulator interacts with $\mathcal{A}$ using $\langle \vec{q}, \omega, b\rangle$ until the first instance either succeeds or conclusively fails. Note that decryption of ciphertexts $\tilde{C}_i$ with $i > 1$ is not required since such a request would imply that time β has elapsed since the sending of the second message of instance i, but this would mean that time α has already elapsed since the second message of instance 1 was sent (and therefore the first instance has either succeeded or failed by that point). If the first instance succeeds, the simulator proceeds with witness extraction as described below. If the first instance fails, the simulator chooses new, random $\vec{q}, \omega, b$ and interacts with $\mathcal{A}$ using $\langle \vec{q}, \omega, b \rangle$. This is repeated for a total of at most t^4/τ times (using new, random $\vec{q}, \omega, b$ each time) or until the first instance succeeds, where $\tau = \tau(k)$ is an inverse polynomial whose value we will fix at the end of the proof and $t = t(k)$ is a bound on the number of times $\mathcal{A}$ interacts with the decryption oracle. If the first instance ever succeeds, the simulator proceeds with witness extraction. Otherwise, the first instance is declared *conditionally delinquent* and the simulator proceeds to simulation of the second instance.

If the ciphertext is defined at instance 1, the simulator proceeds as above, but uses values ω^*, b^* to interact with $\mathcal{A}$ (where these values are defined by ciphertext $\tilde{C}^*$ received from the simulator's encryption oracle, as discussed previously).

If the first instance ever succeeds, witness extraction will be performed. Assume the first instance succeeded when interacting with $\mathcal{A}$ using $\langle \vec{q}, \omega, b \rangle$. The simulator then does the following:

For $n = 0$ to $e - 1$:
  $q_1' \leftarrow \mathbb{Z}_e$
  Interact with $\mathcal{A}$ using $\langle q_1', q_2, \ldots, q_t, \omega, b \rangle$
    If the first instance succeeds and $q_1' \neq q_1$, output the transcript and stop
  Interact with $\mathcal{A}$ using $\langle n, q_2, \ldots, q_t, \omega, b \rangle$
    if the first instance succeeds and $n \neq q_1$, output the transcript and stop
Output $\perp$ and stop

If $\perp$ is not output, the simulator attempts to compute a witness to the decryption of $\tilde{C}_1$ as in the proof of Theorem 1. If such a witness is computed, we say the first instance is *extracted* and the simulator proceeds with simulation of instance 2. If $\perp$ is output, or if $\perp$ is not output but a witness to the decryption of $\tilde{C}_1$ cannot be computed, the entire simulation is aborted; we call this a *failure to extract*.

In general, when we are ready to simulate the i^{th} instance (assuming the entire simulation has not been aborted), each of the first $i-1$ instances has been classified as either extracted or conditionally delinquent. If instance j is extracted, the simulator knows the decryption of $\tilde{C}_j$ and can send it to $\mathcal{A}$ upon successful completion of that instance in the current simulation. On the other hand, if instance j is classified as conditionally delinquent, then with sufficiently high probability that instance will never succeed.

We say the *ciphertext is defined before instance* i if, for some $j \leq i$, the ciphertext is defined at j. At the beginning of simulation of the i^{th} instance, if the ciphertext is not defined before instance $i - 1$, the simulator chooses random $q_i, \ldots, q_t, \omega, b$ and interacts with $\mathcal{A}$ using $\langle q_1^*, \ldots q_{i-1}^*, q_i, \ldots, q_t, \omega, b \rangle$. If $\mathcal{A}$ makes a call m_0, m_1 to the decryption oracle before q_i is requested, the simulator forwards m_0, m_1 to its encryption oracle and receives in return a ciphertext $\tilde{C}^*$; we then say the ciphertext is defined at instance i. If the verification key VK_i used by $\mathcal{A}$ during the i^{th} instance of the decryption oracle is equal to VK defined by state, the i^{th} instance is declared *conditionally delinquent* and the simulator proceeds with simulation of the next instance.

If the ciphertext is not defined before instance i, the simulator continues to interact with $\mathcal{A}$ using $\langle q_1^*, \ldots q_{i-1}^*, q_i, \ldots, q_t, \omega, b \rangle$ until the i^{th} instance either succeeds or conclusively fails. If a success occurs, witness extraction is performed as described below. In case the i^{th} instance fails, the simulator chooses new, random $q_i, \ldots, q_t, \omega, b$ and interacts with $\mathcal{A}$ using $\langle q_1^*, \ldots, q_{i-1}^*, q_i, \ldots, q_t, \omega, b\rangle$. This is repeated for a total of at most t^4/τ times or until the i^{th} instance succeeds. If the i^{th} instance ever succeeds, the simulator proceeds with witness extraction as described below. Otherwise, the i^{th} instance is declared *conditionally delinquent* and the simulator proceeds to simulation of the next instance.

Note that during simulation of instance i, decryption of ciphertexts C_j with $j > i$ is not required since such a request would imply that time β has elapsed since the sending of the second message of instance j, but this would mean that time α has already elapsed since the second message of instance i was sent (and therefore the i^{th} instance has either succeeded or failed by that point). However, the simulator may be required to decrypt ciphertext $\tilde{C}_j$ with $j < i$. In case instance j is extracted, this is no problem, since the simulator knows the witness to the decryption of $\tilde{C}_j$. On the other hand, when j is conditionally delinquent, there is a problem. We handle this as follows: if conditionally delinquent instance j succeeds *before* q_i is sent, the entire simulation is aborted; we call this a *classification failure*. If a conditionally delinquent instance j succeeds *after* q_i is sent, we consider this an *exceptional event at instance j during simulation of i*, and do not include it in the count of failed trials. However, if $3t^3/\tau$ such exceptional events occur for any j, the entire simulation is aborted; we call this an *exception at j during simulation of i*.

If the ciphertext is defined before instance i, the simulator proceeds as above but using ω^*, b^* (where these values are defined by the ciphertext C^* obtained from the simulator's encryption oracle, as discussed previously).

If the i^{th} instance ever succeeds, witness extraction will be performed. Assume the first instance succeeded when interacting with $\mathcal{A}$ using $\langle \vec{q}, \omega, b\rangle$. The simulator then does the following (*success* here means that the i^{th} instance succeeds and no exceptional events occurred during the interaction with $\mathcal{A}$):

For $n = 0$ to $e - 1$:
  $q_i' \leftarrow \mathbb{Z}_e$
  Interact with $\mathcal{A}$ using $\langle q_1, q_{i-1}, q_i', q_{i+1}, \ldots, q_t, \omega, b\rangle$
    If the first instance succeeds and $q_i' \neq q_i$, output the transcript and stop
  Interact with $\mathcal{A}$ using $\langle q_1, q_{i-1}, n, q_{i+1}, \ldots, q_t, \omega, b\rangle$
    if the first instance succeeds and $n \neq q_i$, output the transcript and stop
Output $\perp$ and stop

If $\perp$ is not output, the simulator attempts to compute a witness to the decryption of $\tilde{C}_i$ as in the proof of Theorem 1. If such a witness is computed, we say the i^{th} instance is *extracted* and proceed with simulation of the next instance. If $\perp$ is output, or if $\perp$ is not output but a witness to the decryption of $\tilde{C}_i$ cannot be computed, the entire simulation is aborted; we call this a *failure to extract*.

Once all t instances have been simulated, if the ciphertext is not defined before t, the simulator simply interacts with $\mathcal{A}$ on input $\langle \vec{q}^*, \perp, \perp\rangle$ until $\mathcal{A}$ makes call m_0, m_1 to the encryption oracle. The simulator forwards these values to its encryption oracle, receiving in return $\tilde{C}^*$. Simulation of the encryption oracle for $\mathcal{A}$ is done using $\mathcal{SIM}_2$, as above. In this case, we say the ciphertext is defined at $t + 1$.

As long as the entire simulation is not aborted, the result is a perfect simulation of the view of $\mathcal{A}$ in Expt_1 with random variables $\Omega^* \stackrel{\text{def}}{=} \langle pk, \sigma, r', \mathsf{state}, r, w^*, b^* \rangle$. We now show that: (1) the expected running time of the above simulation is polynomial in t and $1/\tau$, and (2) with all but negligible probability over Ω^* and for some negligible function $\mu(\cdot)$, the simulation fails with probability at most $3\tau/4 + \mu(k)$. Fixing $\tau(k)$ to an appropriate inverse polynomial function then yields a correct simulation with sufficiently high probability.

Claim 4 *The expected running time of the simulation is polynomial in t and $1/\tau$.*

For simulation of each instance, at most $4t^4/\tau$ trials are run before either aborting, declaring the instance conditionally delinquent, or attempting to extract. Say extraction is attempted at instance i because the i^{th} instance succeeded (and no exceptional events occurred) when interacting with $\mathcal{A}$ using $\langle \vec{q}, \omega, b \rangle$. Let p_i denote the probability, over choice of q_i', that the i^{th} instance succeeds and no exceptional events occur when interacting with $\mathcal{A}$ using $\langle q_1, \ldots, q_{i-1}, q_i', q_{i+1}, \ldots, q_t, \omega, b \rangle$. If $p_i > 1/e$, extraction requires expected number of steps at most $2/p_i$. Since extraction with these values of $\vec{q}, \omega, b$ is performed with probability at most $p_i \cdot (4t^4/\tau)$, the contribution to the expected running time is at most $(4t^4 p_i/\tau) \cdot 2/p_i = 8t^4/\tau$. If $p_i \leq 1/e$, extraction requires at most e steps, and the contribution to the expected running time is then at most $(4t^4/e\tau) \cdot e = 4t^4/\tau$. In either case, the contribution to the expected running time for simulation of any instance is polynomial in t and $1/\tau$. Since there are at most t instances, the entire simulation has expected running time polynomial in t and $1/\tau$.
□

Claim 5 *With all but negligible probability over Ω^*, the probability of a* classification failure *is at most $\tau/4 + \varepsilon_4(k)$, where $\varepsilon_4(\cdot)$ is negligible.*

The analysis follows [17]. Say the ciphertext is defined at instance v, where $1 \leq v \leq t+1$. For $i < v$, define the *values defined before* i as $q_1^*, \ldots, q_{i-1}^*$ and the *variables not defined before* i as variables $q_i, \ldots, q_t, \omega, b$; for $i \geq v$, define the *values defined before* i as $q_1^*, \ldots, q_{i-1}^*, \omega^*, b^*$ and the *variables not defined before* i as $q_i, \ldots, q_t$. For each i, recursively define d_i as the probability (over variables not defined before i) that instance i succeeds, conditioned on the values defined before i and on the event that no delinquent instance j ($j < i$) succeeds. Define an instance i to be *delinquent* if d_i is at most $\tau/4t^3$. We now compute the probability that an instance which is not delinquent is classified as conditionally delinquent.

If an instance i is declared conditionally delinquent because $\mathsf{VK}_i = \mathsf{VK}$, then, with all but negligible probability over Ω^*, the probability that instance i succeeds is negligible. If not, the security of the one-time signature scheme is violated with non-negligible probability during Expt_1 (details omitted). If an instance is declared conditionally delinquent for failing too many trials, then, if $d_i > \tau/4t^3$, the probability that none of the t^4/τ trials succeeded is at most

$$\left(1 - \frac{\tau}{4t^3}\right)^{t^4/\tau} \leq e^{-t/4},$$

which is negligible.

Assuming that all instances declared conditionally delinquent are in fact delinquent, the probability of a classification failure during a given instance is at most $\tau/4t^3$, and hence the probability of a classification failure occurring is at most $t \cdot \tau/4t^3 \leq \tau/4$.
□

Claim 6 *With all but negligible probability over Ω^*, the probability of a* failure to extract *is negligible.*

A failure to extract occurs for one of two reasons: (1) the extraction algorithm cannot generate two different accepting transcripts or (2) the extraction algorithm generates two different accepting transcripts but cannot extract a witness to the decryption of the relevant ciphertext. Say extraction is attempted at instance i because the i^{th} instance succeeded (and no exceptional events occurred) when interacting with $\mathcal{A}$ using $\langle \vec{q}, \omega, b \rangle$. Let p_i denote (as in Claim 4) the probability, over choice of q'_i, that the i^{th} instance succeeds and no exceptional events occur when interacting with $\mathcal{A}$ using $\langle q_1, \ldots, q_{i-1}, q'_i, q_{i+1}, \ldots, q_t, \omega, b \rangle$. If case (1) occurs, this implies that $p_i = 1/e$. But in this case, extraction at this instance and with these values of $\vec{q}, \omega, b$ is performed only with probability at most $(4t^4/\tau) \cdot (1/e)$, which is negligible. Furthermore, since $\mathsf{VK}_i \neq \mathsf{VK}$ (otherwise instance i is declared conditionally delinquent), the techniques of the proof of Theorem 1 imply that, with all but negligible probability over Ω^*, case (2) occurs with only negligible probability. (If not, an RSA root may be extracted in expected polynomial time with non-negligible probability.) $\square$

Claim 7 *With all but negligible probability over Ω^*, the probability of abort due to an* exception at j during simulation of i *(for any i, j) is at most $\tau/2 + \varepsilon_5(k)$, where $\varepsilon_5(\cdot)$ is negligible.*

Fix i and j with $i > j$. Define $d'_{i,j}$ as the probability (over variables not defined before i) that instance j succeeds (conditioned on the values defined before i). An exception at j during simulation of i means that, during simulation of i, conditionally delinquent instance j succeeded at least $3t^3/\tau$ times out of at most $4t^4/\tau$ trials. We claim that if this happens, then, with all but negligible probability, $d'_{i,j}$ is at least $1/2t$. If this were not the case, letting X be a random variable denoting the number of successes in $4t^4/\tau$ trials and μ denote the actual value of $d'_{i,j}$, the Chernoff bound shows that:

$$\begin{aligned} \Pr[X > \frac{3}{2} \cdot \mu 4t^4/\tau] &< \Pr[X > 3t^3/\tau] \\ &< (e/(3/2)^3)^{t^3/\tau}, \end{aligned}$$

and since $e/(3/2)^3 < 1$, this expression is negligible. Assuming instance j is in fact delinquent (which is true with all but negligible probability; see the proof of Claim 5), the probability (over variables not defined before j) that $d'_{i,j} \geq 1/2t$ is at most $\tau/2t^2$. Summing over all t^2 possible choices of i and j yields the desired result. $\square$

Assume the advantage of $\mathcal{A}$ in Expt_1 is not negligible. This implies the existence of some constant c such that, for infinitely many values of k,

$$\left| \Pr_1[\mathsf{Succ}] - \frac{1}{2} \right| > 1/k^c.$$

Set $\tau(k) = 1/2k^c$. The probability of a correct simulation is then at least $1 - 1/2k^c - \mu(k)$ for some negligible function $\mu(\cdot)$. Let Sim denote the event that a successful simulation occurs. Then:

$$\mathsf{Adv}_{\mathcal{A}'}(k) = \left| \Pr[\mathsf{Succ} \wedge \mathsf{Sim}] + \Pr[\mathsf{Succ} \wedge \overline{\mathsf{Sim}}] - \frac{1}{2} \right|$$

$$
\begin{aligned}
&= \left|\Pr_1[\mathsf{Succ} \wedge \mathsf{Sim}] + \Pr[\mathsf{Succ}|\overline{\mathsf{Sim}}]\Pr[\overline{\mathsf{Sim}}] - \frac{1}{2}\right| \\
&\geq \left|\Pr_1[\mathsf{Succ}] - \Pr_1[\mathsf{Succ} \wedge \overline{\mathsf{Sim}}] - \frac{1}{2}\right| - \frac{1}{2} \cdot (\frac{1}{2k^c} + \mu(k)) \\
&\geq \left|\Pr_1[\mathsf{Succ}] - \frac{1}{2}\right| - \frac{3}{2} \cdot (\frac{1}{2k^c} + \mu(k)),
\end{aligned}
$$

and then for infinitely many values of k we have $\mathsf{Adv}_{\mathcal{A}'}(k) \geq 1/8k^c$ so that this quantity is not negligible.

This completes the proof of the Theorem. ■