A Simpler Construction of CCA2-Secure Public-Key Encryption Under General Assumptions

Yehuda Lindell

Dept. of Computer Science and Applied Math.
The Weizmann Institute of Science
Rehovot 76100, Israel.
lindell@wisdom.weizmann.ac.il

May 12, 2002

Abstract

In this paper we present a simpler construction of an encryption scheme that achieves adaptive chosen ciphertext security (CCA2), assuming the existence of trapdoor permutations. We build on previous works of Sahai and De Santis et al. and construct a scheme that we believe is the easiest to understand to date. In particular, it is only slightly more involved than the Naor-Yung encryption scheme that is secure against passive chosen-ciphertext attacks (CCA1). We stress that the focus of this paper is on *simplicity* only.

1 Introduction

One of the most basic tasks of cryptography is that of providing encryption schemes that enable the safe delivery of private messages on an open network. Such an encryption scheme should reveal no information about the plaintext to an eavesdropping adversary. However, it may be necessary to protect the privacy of messages from an adversary who has more power than just the ability to eavesdrop. In a chosen-ciphertext attack, the adversary has access to a decryption oracle and uses this is in an attempt to "break" the encryption scheme. Such attacks come in two flavours: *passive* chosen-ciphertext attacks (CCA1), where the adversary can access the decryption oracle only up until the point that it receives a challenge ciphertext, and *adaptive* chosen-ciphertext attacks (CCA2), where the adversary can even access the decryption oracle after it receives the challenge ciphertext. (In the latter case, the adversary can query the decryption oracle for any ciphertext except for the challenge itself.) This is a very strong attack; nevertheless, there are real settings in which this level of security is required (see [2] for an example of an attack on an RSA standard that was made possible due to the fact that the encryption scheme used was not CCA2-secure). We refer the reader to [16] for a survey on the importance of CCA2 security.

Feasibility and efficiency. Two rather distinct directions of research have been considered with respect to secure encryption (and cryptography in general). One direction of research focuses on proving the *feasibility* of obtaining secure schemes, while the other concentrates on doing this *efficiently*. The latter research usually relies on specific number-theoretic (or other) complexity assumptions, whereas the former prefers to only assume the existence of some general primitive,

like a trapdoor permutation. This paper considers the question of the feasibility of obtaining CCA2-secure encryption, under the assumption that trapdoor permutations exist. It is well known that such a feasibility result has already been established. However, known constructions of CCA2-secure encryption schemes under general assumptions are rather complicated. Thus, despite their importance, it is hard to teach these schemes in a course on cryptography, for example. The aim of this paper is to improve this situation.

The Naor-Yung paradigm [12]. Our encryption scheme follows the Naor-Yung paradigm for constructing CCA1-secure encryption schemes. According to this paradigm, the plaintext is encrypted twice (independently), and then a non-interactive zero-knowledge proof (NIZK) is used in order to prove that both ciphertexts are encryptions to the same plaintext. The intuition behind this idea is that if the adversary manages to obtain two encryptions of the same plaintext with independent keys, then essentially it must already "know" the plaintext. Therefore, the decryption oracle that it is given is of no help.

The history of the feasibility of CCA2-encryption. The first CCA2-secure encryption scheme was presented in a breakthrough work by Dolev, Dwork and Naor (DDN) [7]. However, their construction is rather complicated, requiring many multiple encryptions and an involved key-selection technique. An important step in the simplification of CCA2-secure encryption was taken by Sahai [14] who showed that CCA2-encryption schemes can actually be constructed using the Naor-Yung paradigm. This involved introducing a stronger notion of NIZK proofs, called *one-time simulation-sound* NIZK. Loosely speaking, one-time simulation-soundness ensures that it is not feasible for an adversary to generate an accepting NIZK proof of a false statement, even if the reference string is generated by the simulator and even if a simulated proof (of a not necessarily true statement) is observed.[1] Sahai showed that the Naor-Yung encryption scheme, with the NIZK proof system replaced by one which is one-time simulation-sound, is CCA2-secure. Unfortunately, much of the complexity of the DDN construction remained in Sahai's construction of this strong NIZK. Thus, on the one hand, the Sahai high-level construction is significantly simpler than DDN; however, when considering all the details, it is still quite involved.

Recently, De Santis et al. [6, 15] presented a very elegant and far simpler construction of simulation-sound NIZK. The aim of their work was actually to strengthen the notion even further to *many-time* simulation-soundness. (In a many-time simulation-sound NIZK, the soundness is preserved even if the adversary observes many simulated proofs.) Nevertheless, their construction also yields a simpler CCA2-secure encryption scheme.

Our contribution. In this paper, we use ideas from [6] and apply them to the problem of *one-time* simulation-soundness. Exploiting the fact that one-time simulation soundness is enough for CCA2-security, we obtain a significant simplification of the [6] construction of a many-time simulation-sound NIZK. Our construction is both intuitive and simple, and not less importantly, has a short and easy proof of correctness. By plugging our construction into the Sahai CCA2-secure encryption scheme, we obtain a scheme that is only slightly more involved than the original CCA1-secure encryption scheme of Naor and Yung. Thus, we provide an alternative (and simpler proof) to the following theorem:

[1] This is in contrast to regular NIZK proof systems where soundness is only guaranteed relative to a uniformly distributed reference string.

Theorem 1 *Assuming the existence of trapdoor permutations, there exists an encryption scheme that is secure against adaptive chosen-ciphertext* (CCA2) *attacks.*

Related work. The focus of this work is the construction of encryption schemes that are secure against adaptive chosen-ciphertext attacks, assuming only the existence of trapdoor permutations. As we have mentioned above, the first such scheme was presented by Dolev, Dwork and Naor [7]. On the other hand, our construction builds rather directly on the chain of works of Naor and Yung [12], Sahai [14] and De Santis et al. [6].

The first efficient CCA2-secure encryption scheme (proved in the standard model) was presented in a breakthrough work by Cramer and Shoup [4]. However, their construction relies on a specific complexity assumption (namely, the Decisional Diffie-Hellman assumption). Recently, they presented other CCA2-secure schemes, relying on other assumptions (some of which are more standard) [5]. (We stress that our work is incomparable to theirs. On the one hand, they achieve high efficiency while relying on specific complexity assumptions. On the other hand, we assume only the existence of trapdoor permutations, but obtain a scheme that is very inefficient due to the generic NIZK proof that is used.)

Much work on the problem of efficient CCA2-secure encryption has been carried out in the random-oracle model; the most famous of these being OAEP [1]. However, when the random oracle is replaced by a concrete hash function, the security argument becomes heuristic only. Thus, the existence of these schemes does not constitute a proof of Theorem 1.

Organization. As we have mentioned, the technical contribution of this paper is the construction of a simple one-time simulation-sound NIZK. Therefore, the main body of the paper focuses on this issue. In particular, Section 2 contains the formal definitions of simulation-sound NIZK proof systems and the cryptographic tools necessary for our construction. Then, in Section 3 we present our construction of a one-time simulation-sound NIZK and its proof of correctness. For the sake of completeness, in Section 4 we describe the Sahai CCA2-secure encryption scheme, and in Appendix A we provide its proof of correctness. These latter results are reproduced from [14].

2 Definitions and Cryptographic Tools

2.1 Definitions

In this section, we present the definitions for adaptive non-interactive zero-knowledge (NIZK) and one-time simulation-sound adaptive NIZK. Our formal definitions are essentially taken from [10, 14]. We denote the security parameter by n, and an unspecified negligible function by $\mu(n)$ (i.e., $\mu(n)$ grows slower than $1/p(n)$ for every polynomial $p(\cdot)$). We often omit explicit reference to the security parameter n in our notations.

Adaptive non-interactive zero-knowledge. In the model of non-interactive zero-knowledge proofs [3], the prover and verifier both have access to the same uniformly distributed reference string. A proof in this model is a single string sent by the prover to the verifier, and the reference string is used for both generating proofs and verifying their validity. The soundness of the proof system is such that if the reference string is indeed uniformly distributed, then with overwhelming probability, no false theorem can be proved (even by an all-powerful cheating prover). On the other hand, the zero-knowledge property is formulated by stating that there exists a simulator who outputs a reference string and a proof, that are computationally indistinguishable from what is

viewed by a verifier in the real setting described above. Notice that the simulator generates both the reference string and the proof and is not expected to simulate proofs relative to a uniformly distributed reference string (which would not be possible to achieve). In particular, this means that the simulator may choose the reference string to be pseudorandom, and according to some specific distribution.

The *adaptivity* of a NIZK system refers both to the soundness and zero-knowledge. In both cases, an adaptive NIZK is one where the statement to be proven is chosen only *after* the reference string has been fixed. Thus, the cheating prover first receives a uniformly distributed reference string, and then attempts to find some $x \notin L$ with an accepting proof π for x. The adaptive soundness condition states that the probability that π is an accepting proof is negligible. On the other hand, the adaptive zero-knowledge property is formulated by having the simulator output a reference string *before* giving it a statement x for which it must generate a simulated proof. In the formal definition below, a function f is specified that "chooses" a statement x to be proven, based on the reference string R (for soundness, f chooses $x \notin L$, whereas for zero-knowledge f chooses $x \in L$).

Definition 2 (adaptive non-interactive zero-knowledge): *A pair of probabilistic machines (P, V) is called an* adaptive non-interactive zero-knowledge proof system *for a language L if the following holds:*

- Completeness: *For every $x \in L$,*

$$\Pr[V(x, R, P(x, R)) = 1] > 1 - \mu(|x|)$$

  where R is a random variable uniformly distributed in $\{0,1\}^{\mathrm{poly}(|x|)}$.

- Adaptive Soundness: *For every function $f : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \setminus L$ and prover P^*,*

$$\Pr[V(f(R), R, P^*(R)) = 1] < \mu(n)$$

  where R is a random variable uniformly distributed in $\{0,1\}^{\mathrm{poly}(|x|)}$.

- Adaptive Zero-Knowledge: *There exists a simulator $S = (S_1, S_2)$ such that for every function $f : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n \cap L$, the ensembles $\{f(R_n), R_n, P(f(R_n), R_n)\}_{n \in \mathsf{N}}$ and $\{S^f(1^n)\}_{n \in \mathsf{N}}$ are computational indistinguishable, where R_n is a random variable uniformly distributed in $\{0,1\}^{\mathrm{poly}(n)}$ and $S^f(1^n)$ denotes the output from the following experiment:*

  1. *$(r, s) \leftarrow S_1(1^n)$: Simulator S_1 (upon input 1^n) outputs a reference string r and some state information s to be passed on to S_2.*
  2. *$x \leftarrow f(r)$: the statement x to be proven is chosen.*
  3. *$\pi \leftarrow S_2(x, r, s)$: Simulator S_2 generates a simulated proof π that $x \in L$.*
  4. *Output (x, r, π).*

Adaptive NIZK proof systems can be constructed from any trapdoor permutation [8]. We note that any NIZK proof system is *witness-indistinguishable*, where informally speaking, witness-indistinguishability means that proofs generated using one witness are indistinguishable from proofs generated using a different witness [9].

One-time simulation-sound adaptive NIZK. Loosely speaking, a NIZK proof system is one-time simulation-sound if the soundness condition holds even with respect to a reference string generated by the simulator (and not uniformly distributed), and even after a single simulated proof (of a not necessarily correct statement) has been observed. Of course, it is always possible for a cheating prover to simply copy the simulated proof that it observed. Therefore, the requirement is that it is infeasible to efficiently compute any *other* pair of an incorrect statement and accepting proof.

Definition 3 (one-time simulation soundness): *Let (P, V) be an adaptive* NIZK *proof system for a language L, and let $S = (S_1, S_2)$ be a simulator for (P, V). Then, we say that (P, V, S) is* one-time simulation-sound*, if for every probabilistic polynomial-time adversary $A = (A_1, A_2)$, it holds that the probability that A succeeds in the following experiment is negligible:*

1. $(r, s) \leftarrow S_1(1^n)$.
2. $(x, a) \leftarrow A_1(r)$*: Adversary A_1 receives the (simulator-generated) reference string r and outputs a statement x for which it wants to see a proof, and state information a for A_2.*
3. $\pi \leftarrow S_2(x, r, s)$.
4. $(x', \pi') \leftarrow A_2(x, r, \pi, a)$*: Adversary A_2 receives the simulated proof, and outputs a statement x' and a proof π' that $x' \in L$.*
5. *We say that A* succeeds *if it output an accepting proof of a false statement (and did not copy the proof π). That is, A succeeds if $x' \notin L$, $(x', \pi') \neq (x, \pi)$ and $V(x', r, \pi') = 1$).*

In such a case, we say that the proof system (P, V) is a one-time simulation-sound adaptive NIZK*.*

As we have mentioned above, the notion of simulation-soundness was first introduced by [14] who also presented a construction for one-time simulation-soundness (and an extension to allow for any predetermined polynomial number of simulated proofs). Unbounded simulation-soundness (allowing any polynomial number of simulated proofs) was later demonstrated in [6].

2.2 Cryptographic Tools

In this section, we present informal definitions for the cryptographic tools that we use in constructing our encryption scheme. These tools are standard; however we add minor (yet important) additional requirements. We note that it is easy to obtain these additional requirements using known techniques.

Non-interactive perfectly-binding commitment schemes. Loosely speaking, a non-interactive perfectly-binding commitment scheme is a *probabilistic* algorithm C with the following properties:

- *Hiding:* for every two strings s_1 and s_2 (such that $|s_1| = |s_2|$), it is hard to distinguish $\{C(s_1)\}$ from $\{C(s_2)\}$.
- *Binding:* for every two strings $s_1 \neq s_2$, the range of $C(s_1)$ is disjoint from the range of $C(s_2)$. (Thus, given $C(s_1)$, it is impossible to decommit to any value other than s_1.)

We denote by $C(s; r)$ the output of the commitment scheme C upon input $s \in \{0,1\}^n$ and using random coins $r \in_R \{0,1\}^{\mathrm{poly}(n)}$. (Thus, the binding property states that for $s_1 \neq s_2$, it holds that $C(s_1; r_1) \neq C(s_2; r_2)$ for every r_1 and r_2.) In addition to the above, we require the following properties:

- *Pseudorandom range:* We require that the output of the commitment algorithm be pseudorandom. This property is fulfilled by the following commitment scheme based on one-way permutations: Let f be a one-way permutation and b a hard-core of f. Then, $C(\sigma) \stackrel{\text{def}}{=} (f(U_n), b(U_n) \oplus \sigma)$, where U_n denotes the uniform distribution over $\{0,1\}^n$.
- *Negligible support:* We require that a random string is a valid commitment with only negligible probability. This is easily obtained from the above-described commitment scheme based on one-way permutations, by requiring that any commitment to s is preceded by a commitment to 0^n. That is, define $C'(s) = (\text{Commit}(0^n), \text{Commit}(s))$, where each bit is separately committed to using $C(\sigma) = (f(U_n), b(U_n) \oplus \sigma)$.

We note that the Naor commitment scheme [11] as is, has both of these above properties. (Although the [11] commitment scheme is interactive, the receiver message can be hardwired into the common reference string, and so suffices for our needs here.)

"Strong" one-time signature schemes. Loosely speaking, a one-time signature scheme is an existentially unforgeable signature scheme (secure against a chosen-message attack), with the restriction that the signer must only sign a single message with any key. Thus, such a signature scheme is defined as a triplet of algorithms $(G, \mathsf{Sign}, \mathsf{Verify})$, where G is a probabilistic generator that outputs a signing-key sk and a verification-key vk. The validity of the signature scheme fulfills that for every message m, $\mathsf{Verify}(vk, m, \mathsf{Sign}(sk, m)) = 1$, where $(vk, sk) \leftarrow G(1^n)$ (i.e., honestly generated signatures are always accepted). On the other hand, the security of the scheme states that the probability that an efficient forging algorithm S^* succeeds in generating a forgery given a single chosen signature is negligible. More formally, the following experiment is defined: The generator G is run, outputting a key-pair (vk, sk). Then, S^* is given vk and chooses a message m for which it receives a signature $\sigma = \mathsf{Sign}(sk, m)$. Following this, S^* outputs a pair (m', σ') and we require that the probability that $\mathsf{Verify}(vk, m', \sigma') = 1$ and $m' \neq m$ is negligible.

As with the commitment scheme, here we also require an additional property. The standard definition of security requires that the forger cannot generate a valid signature on any different message. We strengthen this and require that the forger cannot even generate a different valid signature on the same message. That is, we modify the above (informal) definition and require that the probability that $\mathsf{Verify}(vk, m', \sigma') = 1$ and $(m', \sigma') \neq (m, \sigma)$, is negligible. (Thus, the only the valid signature the forger can present is the exact one that it has seen.) Such a signature scheme can be constructed using universal one-way hash functions and 1–1 one way functions (in a similar fashion to the standard construction based on any one-way function [13]). By using 1–1 one-way functions, we ensure that each message has a unique signature and therefore the above strengthening is achieved.

3 Simple One-Time Simulation-Sound NIZK

The motivation behind our construction is as follows. Following [8], (and similarly to [6]) the reference string for the non-interactive proof is divided into two parts. The first part of the string is used by the simulator to simulate a proof, while the second is really used for proving (according to a given NIZK scheme). Typically, in order to prove that $x \in L$, a compound statement of the following structure is proved: either the first part of the reference string has some special property or $x \in L$. However, when the reference string is chosen at random, the first part will *not* have this special property (except with negligible probability). Therefore, if the proof is accepting, it must

be that $x \in L$. On the other hand, the simulator is able to generate a pseudorandom string that *does* have the property, and can therefore "cheat" in the proof.

In our scheme, the property used is that the first part of the reference string is a commitment to a verification-key vk of a one-time signature scheme. That is, the prover sends a verification-key vk along with the proof and proves that: either the first part of the reference string is a commitment to this verification-key vk, or $x \in L$. Furthermore, the prover signs on the proof using the associated signing-key sk (and the verifier checks the validity of this signature using vk). A real prover chooses a random pair of signature keys and generates a proof based on the fact that $x \in L$.

On the other hand, the simulator works by generating the reference string so that indeed the first part is a commitment to a verification-key vk (for which it knows the associated signing-key sk). Then, the simulator proves the proof using this fact (rather than the fact that $x \in L$). Notice that the simulator is also able to sign on the proof, as required, because it knows the associated secret-key. The fact that the scheme is simulation-sound follows from the observation that any accepting proof to a false statement must use the property that the first part of the reference string is a commitment to vk. In particular, this means that such a proof is accompanied with a signature using sk (where sk is known only to the simulator). Thus, it is only possible to generate an accepting proof to a false statement if it is possible to forge a signature.

We now formally present the proof system. In the presentation, we refer to an adaptive NIZK proof system, denoted (P, V), and to commitment and signature schemes. These commitment and signature schemes have the additional properties described in Section 2.2.

Protocol 1 (NIZK Scheme Π)

- **Common reference string:** (r_1, r_2)
- **Prover protocol** (upon input $x \in L$ and a witness w for x):
  1. *Choose a random pair of signature keys (vk, sk) for a strong one-time signature scheme.*
  2. *Let L' be the following language:*

     $$L' = \{(x, r_1, vk) \mid x \in L \text{ or } r_1 = \text{Commit}(vk)\}$$

     Then, generate a non-interactive proof (using reference string r_2) that $(x, r_1, vk) \in L'$. That is, invoke the NIZK *prover P for L' on input (x, r_1, vk), auxiliary-input w and reference string r_2, obtaining a proof p.*
  3. *Compute $\sigma = \mathsf{Sign}_{sk}(x, p)$.*
  4. *Output $\pi = (vk, x, p, \sigma)$.*
- **Verifier protocol** (upon input x and a proof $\pi = (vk, x, p, \sigma)$):
  1. *Check the signature using vk. That is, check that $\mathsf{Verify}_{vk}((x, p), \sigma) = 1$.*
  2. *Invoke the* NIZK *verifier V and check that p constitutes a correct proof that $(x, r_1, vk) \in L'$ when the reference string equals r_2 (i.e., check that $V((x, r_1, vk), r_2, p) = 1$).*
  3. *Output 1 if and only if the above two checks succeed.*

We now proceed to prove the correctness of Protocol 1:

Theorem 4 *Assume that (P, V) is a secure* adaptive NIZK *for $\mathcal{NP}$, and that the signature and commitment schemes meet the requirements as described in Section 2.2. Then, Protocol 1 constitutes a one-time simulation-sound adaptive* NIZK *proof system for $\mathcal{NP}$.*

Proof: We begin by proving that Protocol 1 is an adaptive non-interactive proof system. Completeness is immediate. Soundness follows from the fact that the commitment scheme used has negligible support, and thus a random string is a valid commitment with only negligible probability. Therefore, when r_1 is uniformly chosen, $x \notin L$ implies that $(x, vk, r_1) \notin L'$, except with negligible probability. Adaptive soundness then follows from the adaptive soundness of the NIZK proof system (P, V), for which proofs are generated using r_2 (that is uniformly chosen).

We now proceed to demonstrate the zero-knowledge property. As we mentioned in the motivating discussion, intuitively, zero-knowledge holds because a simulator can set r_1 to be a commitment to a verification-key vk, for which it knows the associated signing-key sk. Then, the simulator proves that $(x, vk, r_1) \in L'$ based on the fact that $r_1 = \text{Commit}(vk)$, and without any witness to the fact that $x \in L$. Since, the commitment scheme used has a pseudorandom range, such a r_1 is indistinguishable from a random string. Furthermore, the NIZK proof system (P, V) is witness indistinguishable and therefore the simulated proof cannot be distinguished from a real one. We now provide the exact description of the simulator. The simulator is divided into two parts: S_1 who chooses the reference string and S_2 who generates simulated proofs.

1. **Simulator S_1:**
   (a) Choose a random pair of signature keys (vk, sk) for a strong one-time signature scheme.
   (b) Compute $r_1 = \text{Commit}(vk) = C(vk; r_c)$ for a random r_c.
   (c) Choose a *uniformly* distributed string r_2.
   (d) Output (r_1, r_2) and $s = (vk, sk, r_c)$ where s is S_1's output state information to be given to S_2.
2. **Simulator** S_2 (upon input x, (r_1, r_2) and $s = (vk, sk, r_c)$):
   (a) Invoke the NIZK prover for L' (as defined in Protocol 1) on input (x, r_1, vk), auxiliary-input r_c and reference string r_2, and obtain a proof p. (Note that the witness provided to the NIZK prover is for $r_1 = \text{Commit}(vk)$, and not the witness for $x \in L$.)
   (b) Compute $\sigma = \text{Sign}_{sk}(x, p)$.
   (c) Output $\pi = (vk, x, p, \sigma)$.

There are two differences between a real proof and that provided by the simulator (where we consider the distributions $\{x, (r_1, r_2), \pi\}$ of the reference string and the proof). Firstly, the string r_1 generated by S_1 is only pseudorandom (and not random as in a real setting). Secondly, the proof p provided by S_2 is based on the witness for the fact that $r_1 = \text{Commit}(vk)$, rather than being based on the witness for $x \in L$. Intuitively, since r_1 is pseudorandom and the NIZK is witness indistinguishable, these distributions cannot be distinguished. Formally, one defines a hybrid distribution in which $r_1 = \text{Commit}(vk)$ and yet the proof p is based on the witness for $x \in L$. Then, the hybrid is indistinguishable from a real proof by the indistinguishability of r_1 from a random string (everything else is exactly the same). On the other hand, the hybrid is indistinguishable from a simulated proof by the witness indistinguishable property of the NIZK. (Notice that the reference string r_2 for this NIZK is uniformly distributed, and thus the witness indistinguishability property holds.) The indistinguishability of a simulated proof from a real one follows. (We note that from the above proof it follows that the underlying non-interactive proof need not be zero-knowledge; rather, adaptive witness indistinguishability suffices.)

One-time simulation-soundness. Until now, we have shown that Protocol 1 constitutes a non-interactive zero-knowledge proof system. It remains to show that it is also one-time simulation-sound. Intuitively, one-time simulation-soundness holds for the following reason. Let (vk, sk) be the signature keys chosen by the simulator S_1. Then, if an adversary generates a proof based on the fact that $r_1 = \text{Commit}(vk)$, it must sign on the proof using the key sk (otherwise, the verification of the signature will fail). This constitutes a successful forgery of the signature scheme and therefore can only occur with negligible probability. Details follow.

Assume by contradiction that there exists an adversary $A = (A_1, A_2)$ (as by Definition 3) who sees a simulated proof π for a statement x, and with non-negligible probability outputs a pair $(x', \pi') \neq (x, \pi)$ where $x' \notin L$ and π' is an accepting proof. That is, let (r_1, r_2) be the reference string output by S_1, let x be the statement that A_1 outputs upon receiving (r_1, r_2), and let $\pi = (vk, x, p, \sigma)$ be the simulated proof supplied by S_2 that $x \in L$. Then, by the contradicting assumption, with non-negligible probability A_2 outputs an accepting proof $\pi' = (vk', x', p', \sigma')$, such that $(x', \pi') \neq (x, \pi)$, and $x' \notin L$. We consider two different cases:

1. *Case* 1 – $vk' \neq vk$*:* First recall that by the definition of the simulator S_1, the string r_1 is such that $r_1 = \text{Commit}(vk)$. However, $vk' \neq vk$ and therefore we have that $r_1 \neq \text{Commit}(vk')$ (by the perfect binding property of the commitment scheme). Therefore, $x' \notin L$ implies that $(x, r_1, vk) \notin L'$. By the (unconditional) soundness of the underlying NIZK scheme, we have that the probability that p' (and therefore π') is an accepting proof is at most negligible.

2. *Case* 2 – $vk' = vk$*:* In this case, we use A to contradict the (strong) security of the signature scheme. Recall that A's proof π' is only accepting if $\text{Verify}_{vk'}((x', p'), \sigma') = 1$. Since $(x', \pi') \neq (x, \pi)$ and $vk' = vk$, it holds that $(x', p', \sigma') \neq (x, p, \sigma)$. Therefore, we have that A received a message and signature $((x, p), \sigma)$ and generated a valid message and signature $((x', p'), \sigma')$, where $((x', p'), \sigma') \neq ((x, p), \sigma)$. By the strong security of the signature scheme, A can succeed in doing this with only negligible probability.

   Formally, we construct a forger A' who receives vk and a single oracle query to $\text{Sign}_{sk}(\cdot)$ and successfully forges a signature. A' works exactly like the simulator except that in Step (b) of S_2's specification, it "computes" the signature by consulting its oracle. Notice that S_1 and S_2 need no knowledge of sk in order to complete all their other steps. Thus, A' can perfectly emulate the simulation setting for A. Therefore, if A outputs $\pi' = (vk, x', p', \sigma')$, where $(x', p', \sigma') \neq (x, p, \sigma)$ and σ' is a valid signature on (x', p'), then this constitutes a successful forgery of the signature scheme. This implies that A succeeds with at most negligible probability.

This completes the proof. ■

We remark that trapdoor permutations suffice for securely obtaining all the building blocks required in the construction of Protocol 1. We therefore have the following result:

Proposition 3.1 *Assuming the existence of trapdoor permutations, there exists a one-time simulation-sound adaptive* NIZK*.*

4 The Encryption Scheme

In this section, we describe the CCA2-secure public-key encryption scheme of Sahai [14]. This scheme is exactly the scheme of Naor-Yung [12], with the modification that the NIZK used is one-time simulation-sound. We stress that our contribution is in Section 3, where we present a simple

one-time simulation-sound NIZK. Thus, we directly plug our NIZK into the construction (and proof) of [14], obtaining a new (and simpler) CCA2-secure public-key encryption scheme. (The definition of chosen-ciphertext security can be found in Appendix A. For the exposition below, we assume familiarity with these concepts and definitions.)

The Naor-Yung paradigm. As we have mentioned, the [14] encryption scheme is based on the Naor-Yung paradigm [12]. According to this paradigm, the plaintext is encrypted twice with independent keys (from an encryption scheme that is secure against chosen-plaintext attacks) and then a NIZK proof is provided to ensure that both encryptions are indeed of the same plaintext. Passive chosen-ciphertext security (CCA1) or adaptive chosen-ciphertext security (CCA2) are achieved by applying NIZKs with certain "special" properties. For CCA1, the NIZK is such that soundness holds even with respect to the pseudorandom string output by the simulator (as long as a simulated proof has not been observed).[2] For CCA2, the NIZK must be one-time simulation-sound. From here on, we focus on the CCA2 case. However, we stress that the CCA2 scheme and its proof of security are almost identical to that of the CCA1 scheme. This highlights one of the conceptual advantages of the [14] approach; both CCA1 and CCA2-secure encryption schemes can be presented and proved together (and for almost the price of one).

Formal definition of the scheme. We now present the construction of the encryption scheme. Let (G, E, D) be an encryption scheme that is secure against chosen-plaintext attacks. Furthermore, let (P, V) be a one-time simulation-sound adaptive NIZK proof system for the following NP-language:

$$L = \{(c_1, c_2, pk_1, pk_2) \mid \exists m \text{ s.t. } c_1 = E_{pk_1}(m) \ \& \ c_2 = E_{pk_2}(m)\}$$

That is, L is the language of pairs of ciphertexts (and public-keys), such that both ciphertexts are encryptions of the same message (we denote $c = E_{pk}(m)$, if c is an encryption of m). Then, the CCA2-secure scheme, denoted $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, is defined as follows:

Construction 2 (adaptive chosen-ciphertext encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$):

- Key Generation: *Obtain two independent key sets from G. That is, obtain $(pk_1, sk_1) \leftarrow G(1^n)$ and $(pk_2, sk_2) \leftarrow G(1^n)$. Furthermore, choose a uniformly distributed reference string r of the correct length for the* NIZK *proof system (P, V).*
  - Public Key $\mathcal{PK} = (pk_1, pk_2, r)$
  - Secret Key $\mathcal{SK} = (sk_1, sk_2)$
- Encryption: *In order to encrypt a plaintext m, compute $c_1 = E_{pk_1}(m; r_1)$ and $c_2 = E_{pk_2}(m; r_2)$, for random strings r_1 and r_2. Then, invoke the NIZK prover P upon (c_1, c_2, pk_1, pk_2) with reference string r, obtaining a proof π. Notice that P can prove this statement efficiently when it is given the witness (m, r_1, r_2). Finally, output $\mathcal{E}(m) = (c_1, c_2, \pi)$.*
- Decryption: *In order to decrypt (c_1, c_2, π), first verify that π is an accepting proof for the statement (c_1, c_2, pk_1, pk_2) with reference string r. If yes, then decrypt c_1 and output the decryption value m.*[3]

[2] This is in contrast with standard NIZK proof systems, where soundness is guaranteed only if the reference string is uniformly distributed.

[3] Our choice of decrypting the first ciphertext c_1 is arbitrary; equivalently, one could define the decryption algorithm by having it decrypt c_2.

The fact that the above encryption scheme is secure against adaptive chosen-ciphertext attacks has been proven in [14]. That is,

Theorem 5 (Sahai [14]): *Assume that* (G, E, D) *is an encryption scheme secure against chosen-plaintext attacks, and that* (P, V) *is a one-time simulation-sound adaptive* NIZK *proof system. Then, the encryption scheme* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *of Construction* 2 *is secure against adaptive chosen-ciphertext attacks.*

Combining Theorem 5 with Proposition 3.1, we obtain the existence of CCA2-secure encryption assuming trapdoor permutations only (i.e., we obtain Theorem 1). For the sake of completeness, we reproduce the proof of Theorem 5 in Appendix A (below, we describe the basic idea behind this proof).

Motivation for the proof of security. The basic idea underlying Construction 2 of $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is that it is enough to use only one secret-key in order to decrypt ciphertexts. This is because anyone can verify the validity of a NIZK proof. Therefore, given knowledge of any of the two secret-keys, decryption can be carried out by verifying the NIZK and then decrypting. Since the NIZK proof ensures that both encryptions are to the *same* plaintext, it does not matter which secret-key is used. Now, consider an adversary A_{CPA} who carries out a chosen-plaintext attack (CPA) on the scheme (G, E, D). (Recall that in a CPA attack, the adversary gets no decryption oracle.) Adversary A_{CPA} receives a public-key pk_1 and proceeds to generate a simulated public-key for the scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ which incorporates pk_1. Specifically, A_{CPA} chooses a second key-pair (pk_2, sk_2) and a NIZK reference string r, thereby obtaining a public-key (pk_1, pk_2, r) for $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. Furthermore, A_{CPA} knows *one* of the two secret-keys (namely sk_2). Therefore, as we have discussed above, A_{CPA} is able to correctly decrypt ciphertexts. The important point is that A_{CPA} is able to correctly simulate a *decryption oracle* for a CCA2-adversary $\mathcal{A}$ who attacks $(\mathcal{G}, \mathcal{E}, \mathcal{D})$. In other words, given only chosen-plaintext ability, A_{CPA} can simulate an adaptive chosen-ciphertext attack for a CCA2-adversary $\mathcal{A}$.

The above shows how the decryption oracle in a CCA2-attack can be simulated by A_{CPA}. However, A_{CPA} must also be able to generate a challenge ciphertext for $\mathcal{A}$ from $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, given its own challenge ciphertext from (G, E, D). That is, during its attack, A_{CPA} receives some challenge ciphertext c_1. Based on c_1, A_{CPA} must provide $\mathcal{A}$ with a challenge. Furthermore, it must be shown that if $\mathcal{A}$ can distinguish ciphertexts in $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ with non-negligible probability, then A_{CPA} can use this to also distinguish ciphertexts in (G, E, D). Loosely speaking, A_{CPA} generates the needed ciphertext by simply computing $c_2 = E_{pk_2}(0^n)$ (i.e., c_2 is an encryption to garbage) and then providing a proof π that c_1 and c_2 are encryptions of the same message. Of course, this statement may not be true (since A_{CPA} does not know if c_1 is an encryption of 0^n or of some other message). Nevertheless, such a proof can be generated using the NIZK simulator, and this will be indistinguishable from a real ciphertext. Thus, $\mathcal{A}$ receives the challenge ciphertext (c_1, c_2, π) in its CCA2-attack. The point is that $\mathcal{A}$'s challenge ciphertext contains c_1 and therefore any "information" learned by $\mathcal{A}$ about its challenge ciphertext (c_1, c_2, π) can be used by A_{CPA} to derive information about its own challenge ciphertext c_1. Observe, however, that by the way A_{CPA} constructs the challenge ciphertext, it follows that $\mathcal{A}$ receives a *simulated* NIZK proof π during its attack. Furthermore, $\mathcal{A}$ is able to ask for more decryptions of ciphertexts after seeing this simulated proof, and these ciphertexts contain NIZK proofs. In order for the decryption simulation of A_{CPA} described above to be correct, it must hold that $\mathcal{A}$ cannot generate accepting proofs of false statements, even in such a setting. This is where the one-time simulation-soundness of the NIZK is utilized. Thus we have that A_{CPA} can simulate a complete CCA2-attack for $\mathcal{A}$.

Acknowledgements

We thank Amit Sahai for pointing out to us that the soundness of the simulation-sound NIZK can be made unconditional by using a commitment scheme with negligible support. We also thank Oded Goldreich for helpful discussions and for encouraging us to write this work.

References

[1] M. Bellare and P. Rogaway. Optimal asymmetric encryption – How to encrypt with RSA. In *EUROCRYPT'94*, Springer-Verlag (LNCS 950), pages 92–111, 1994.

[2] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1. In *CRYPTO'98*, Springer-Verlag (LNCS 1462), pages 1–12, 1998.

[3] M. Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. In *20th STOC*, pages 103–112, 1988.

[4] R. Cramer and V. Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*, Springer-Verlag (LNCS 1462), pages 13–25, 1998.

[5] R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *EUROCRYPT 2002*, Springer-Verlag (LNCS 2332), pages 45–64, 2002.

[6] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-interactive Zero-Knowledge. In *CRYPTO 2001*, Springer-Verlag (LNCS 2139), pages 566–598, 2001.

[7] D. Dolev, C. Dwork and M. Naor. Non-malleable Cryptography. In *SIAM Journal of Computing*, Vol. 30 (20), pages 391–437, 2000.

[8] U. Feige, D. Lapidot and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, Vol. 29 (1), pages 1–28, 1999.

[9] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd STOC*, pages 416–426, 1990.

[10] O. Goldreich. *Foundation of Cryptography – Basic Tools.* Cambridge University Press, 2001.

[11] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[12] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd STOC*, pages 427–437, 1990.

[13] J. Rompel. One-way functions are necessary and efficient for secure signatures. In *22nd STOC*, pages 387–394, 1990.

[14] A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *40th FOCS*, pages 543–553, 1999.

[15] A. Sahai. Simulation-Sound Non-Interactive Zero Knowledge. Manuscript, 2000.

[16] V. Shoup. Why chosen ciphertext security matters. *IBM Research Report* RZ 3076, November, 1998.

A Proof of Theorem 5

In this appendix, for the sake of completeness, we reproduce the proof of Theorem 5 from [14]. We begin by providing the formal definition of public-key encryption schemes that are secure under chosen-ciphertext attacks. We assume familiarity with the notion and motivation behind chosen-ciphertext attacks, and therefore go straight to presenting the formal definition:

Definition 6 (indistinguishability of encryptions under adaptive chosen-ciphertext attacks): *A public-key encryption scheme* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *is* indistinguishable under adaptive chosen-ciphertext attacks (CCA2) *if for every pair of probabilistic polynomial-time oracle machines* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,*

$$|\Pr[\mathsf{Expt}_{\mathcal{A}}(0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}(1) = 1]| < \mu(n)$$

where $\mathsf{Expt}_{\mathcal{A}}(b)$ *is defined as follows for* $b \in \{0, 1\}$ *(below,* $\mathcal{D}_{sk}$ *is an oracle that decrypts according to the secret-key* sk *and,* $\mathcal{D}^{\neg c}_{sk}$ *is an analogous oracle that decrypts any ciphertext except for* c*):*

1. $(pk, sk) \leftarrow \mathcal{G}(1^n)$*: generate a public and secret key-pair.*
2. $(m_0, m_1, a) \leftarrow \mathcal{A}_1^{\mathcal{D}_{sk}}(pk)$, where $|m_0| = |m_1|$: $\mathcal{A}_1$ *receives a decryption oracle and outputs a pair of plaintexts for the challenge, and state information* a *for* $\mathcal{A}_2$*.*
3. $c \leftarrow \mathcal{E}_{pk}(m_b)$*: compute the challenge ciphertext*
4. $b' \leftarrow \mathcal{A}_2^{\mathcal{D}^{\neg c}_{sk}}(c, a)$: $\mathcal{A}_2$ *receives the challenge ciphertext, access to a (restricted) decryption oracle and the state information* a *from* A_1*, and outputs a guess* b' *for* b*.*
5. *Output* b'*.*

We note that if $\mathcal{A}_2$ is not given access to a decryption oracle and everything else remains the same, then the encryption scheme is said to satisfy *passive* chosen-ciphertext security (CCA1). If in addition, no decryption oracle is given to $\mathcal{A}_1$, then the scheme is said to satisfy chosen-plaintext security (CPA).

We now proceed to formally prove that Construction 2 constitutes a CCA2-secure public-key encryption scheme. (Motivating discussion can be found at the end of Section 4.)

Theorem 7 (Theorem 5 – restated): *Let* (G, E, D) *be an encryption scheme that is indistinguishable under chosen plaintext attack and let* (P, V) *be a one-time simulation sound adaptive* NIZK *proof system. Then, the encryption scheme* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *of Construction* 2 *is indistinguishable under adaptive chosen-ciphertext attack.*

Proof: Let $\mathcal{A}$ be any (probabilistic polynomial-time) CCA2-adversary. We begin by describing a mental experiment in which $\mathcal{A}$ runs its attack. The difference between the mental experiment and a real attack is that the challenge ciphertext received by $\mathcal{A}$ is not properly generated. Rather, the NIZK proof is generated by the NIZK simulator $S = (S_1, S_2)$ and the two encryptions may not be to the same message. The experiment is defined as follows (the item numbers match those of $\mathsf{Expt}_A(b)$ in Definition 6):

Experiment $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$**:**

1. A public and secret key-pair is generated:
   - $(r, s) \leftarrow S_1(1^n)$: choose a simulator-generated reference string r for the public-key.
   - $(pk_1, sk_1), (pk_2, sk_2) \leftarrow G(1^n)$: generate two public and secret key pairs from the CPA-secure encryption scheme.

   Define $pk = (pk_1, pk_2, r)$ and $sk = (sk_1, sk_2)$.
2. $(m_0, m_1, a) \leftarrow \mathcal{A}_1^{D_{sk}}(pk)$, where $|m_0| = |m_1|$. (We note that the decryption oracle here is defined exactly as in Construction 2. That is, it first verifies the NIZK proof using reference string r, and then decrypts using sk_1.)
3. Set up the challenge ciphertext:
   - $c_1 \leftarrow E_{pk_1}(m_{b_1})$
   - $c_2 \leftarrow E_{pk_2}(m_{b_2})$
   - $\pi \leftarrow S_2((c_1, c_2, pk_1, pk_2), r, s)$ (π is a simulated NIZK proof of the fact that $D_{sk_1}(c_1) = D_{sk_2}(c_2)$).

   Define $c = (c_1, c_2, \pi)$
4. $b' \leftarrow \mathcal{A}_2^{D^{\neg c}_{sk}}(c, a)$.
5. Output b'.

There are two differences between $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$ (the mental experiment) and $\mathsf{Expt}_{\mathcal{A}}(b)$ (from Definition 6). First, in the mental experiment, the challenge ciphertext is made up of an encryption to m_{b_1} and an encryption to m_{b_2}, where b_1 may not equal b_2. On the other hand, in $\mathsf{Expt}_{\mathcal{A}}(b)$, the challenge ciphertext contains two encryptions of the same message m_b. Second, in the mental experiment, the NIZK reference string and the proof that is included in the challenge ciphertext are simulator-generated (and not real). We first claim that if $b_1 = b_2$, then the fact that the NIZK is simulated essentially makes no difference. That is,

Lemma A.1 *For every probabilistic polynomial-time $\mathcal{A}$ and for any $b \in \{0, 1\}$*

$$\left|\Pr[\mathsf{Expt}_{\mathcal{A}}(b) = 1] - \Pr[\mathsf{Expt}^S_{\mathcal{A}}(b, b) = 1]\right| < \mu(n)$$

Proof: Intuitively, this follows from the zero-knowledge property of the NIZK proof system (the only difference between the experiments relates to this point). That is, we construct a distinguisher D for the (P, V) proof system that distinguishes a simulated proof from a real one with the same probability as the difference between the experiments in the claim. Distinguisher D works as follows: D receives a reference string r, chooses $(pk_1, sk_1), (pk_2, sk_2) \leftarrow G(1^n)$ and invokes $\mathcal{A}$ on the public-key $pk = (pk_1, pk_2, r)$, while emulating the decryption oracle D_{sk} for $\mathcal{A}$. Notice that D is able to carry out this decryption because it knows the secret key sk_1, and the NIZK proofs can be verified by anyone. When $\mathcal{A}$ outputs (m_0, m_1) for the challenge, D computes $c_1 = E_{pk_1}(m_b)$ and $c_2 = E_{pk_2}(m_b)$ and asks for a proof π of the correct statement (c_1, c_2, pk_1, pk_2). Distinguisher D then gives $\mathcal{A}$ the challenge ciphertext (c_1, c_2, π) and outputs whatever $\mathcal{A}$ does.

Notice that if D receives a real proof, then it *perfectly* simulates $\mathsf{Expt}_{\mathcal{A}}(b)$. On the other hand, if D receives a simulated proof, then it *perfectly* simulates $\mathsf{Expt}^S_{\mathcal{A}}(b, b)$. Thus, D's advantage in distinguishing simulated proofs from real proofs is exactly $|\Pr[\mathsf{Expt}_{\mathcal{A}}(b) = 1] - \Pr[\mathsf{Expt}^S_{\mathcal{A}}(b, b) = 1]|$. By the adaptive zero-knowledge property of (P, V), this advantage is negligible. It thus follows that the difference in the probabilities in the lemma is also negligible. ■

Before proceeding, we prove a claim stating that when $\mathcal{A}$ participates in $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$, with overwhelming probability all ciphertext queries to its decryption oracle that contain accepting NIZK proofs, are of valid ciphertexts. Formally, we say that a ciphertext $c' = (c'_1, c'_2, \pi')$ is *invalid with an accepting* NIZK, if $D_{sk_1}(c'_1) \neq D_{sk_2}(c'_2)$, and yet $V((c'_1, c'_2, pk_1, pk_2), r, \pi') = 1$. Intuitively, if any of $\mathcal{A}$'s ciphertext queries are of the above form, then this means that $\mathcal{A}$ generated an accepting NIZK of a false statement. Thus, $\mathcal{A}$ could be used to contradict the one-time simulation-soundness of (P, V).

Claim A.2 *Let $\mathcal{A}$ be participating in $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$ for some $b_1, b_2 \in \{0, 1\}$. Then, the probability that during the experiment, $\mathcal{A}$ queries its oracle with an invalid ciphertext that has an accepting* NIZK *proof (and is not the challenge), is negligible.*

Proof: This claim holds because if $\mathcal{A}$ succeeds in producing a false proof, then it can be used by an adversary A to contradict the one-time simulation-soundness of (P, V). In particular, A receives a simulator reference string r and sets up a public-key (pk_1, pk_2, r) (where it knows both sk_1 and sk_2). Then, A runs the rest of $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$ in the same way as D in the proof of Lemma A.1. That is, A invokes $\mathcal{A}$ on the public-key $pk = (pk_1, pk_2, r)$, while emulating the decryption oracle D_{sk} for $\mathcal{A}$ (A knows the secret key sk_1 and can verify NIZK proofs; therefore it can carry out this decryption). When $\mathcal{A}$ outputs (m_0, m_1) for the challenge, A computes $c_1 = E_{pk_1}(m_{b_1})$ and $c_2 = E_{pk_2}(m_{b_2})$ and asks for a *simulated* proof π of the statement (c_1, c_2, pk_1, pk_2). A then gives $\mathcal{A}$ the challenge ciphertext (c_1, c_2, π) and completes the simulation of $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$. It is easy to see that this simulation of the experiment by A is perfect.

Throughout the entire simulation, A checks every ciphertext decryption request (c'_1, c'_2, π') from $\mathcal{A}$. (We note that if (c'_1, c'_2, π') equals the challenge ciphertext (c_1, c_2, π), then A ignores the query.) If $D_{sk_1}(c'_1) \neq D_{sk_2}(c'_2)$, and yet $V((c'_1, c'_2, pk_1, pk_2), r, \pi') = 1$, then A outputs $((c'_1, c'_2, pk_1, pk_2), \pi')$ and halts (A can check this because it knows *both* decryption keys and because it can verify the validity of the NIZK proof π'). Therefore, if $\mathcal{A}$ outputs an invalid ciphertext with an accepting proof in $\mathsf{Expt}^S_{\mathcal{A}}(b_1, b_2)$, it follows that A outputs a false statement with an accepting proof. Furthermore, since the simulation by A is perfect, A outputs such a false statement and an accepting proof with the same probability as $\mathcal{A}$ does in the experiment. By the one-time simulation soundness of the NIZK proof system (P, V), it follows that $\mathcal{A}$ queries its decryption oracle with an invalid ciphertext that has an accepting proof, with only negligible probability. ■

We are now ready to complete the proof of Theorem 5. By Lemma A.1, it suffices to show that,

$$\left|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(0,0)=1]-\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,1)=1]\right|<\mu(n) \tag{1}$$

We consider the hybrid experiment $\mathsf{Expt}^S_{\mathcal{A}}(1,0)$ and prove Eq. (1) by showing that both the following hold:

$$\left|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(0,0)=1]-\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,0)=1]\right|<\mu(n)$$

and

$$\left|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,0)=1]-\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,1)=1]\right|<\mu(n)$$

Both of these equations are proven using analogous arguments. We therefore present the proof of the first only. That is,

Lemma A.3 *For every probabilistic polynomial-time $\mathcal{A}$,*

$$\left|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(0,0)=1]-\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,0)=1]\right|<\mu(n)$$

Proof: This lemma is proven by showing that there exists a CPA-adversary A_{CPA} who carries out a chosen-plaintext attack and distinguishes encryptions from (G,E,D) with the same probability as the difference between the experiments in the lemma. Adversary A_{CPA} works by simulating the experiment $\mathsf{Expt}^S_{\mathcal{A}}(*,0)$ for $\mathcal{A}$. Recall that $\mathcal{A}$ is allowed an adaptive chosen ciphertext attack in this experiment, whereas A_{CPA} has no decryption oracle at all. Therefore, A_{CPA} must simulate the decryption oracle for $\mathcal{A}$ during the attack.

Adversary A_{CPA}:

1. *Key generation:* In the beginning of A_{CPA}'s chosen plaintext attack, it receives a public-key pk from the scheme (G,E,D). The first thing that A_{CPA} does is set up a public-key for $\mathcal{A}_{\text{CCA2}}$ from $(\mathcal{G},\mathcal{E},\mathcal{D})$. It does this by setting $pk_1=pk$, choosing another (independent) key-pair (pk_2,sk_2), and obtaining the *simulator* reference string r for the one-time simulation-sound NIZK (by running $S_1(1^n)$). Then, A_{CPA} invokes $\mathcal{A}_{\text{CCA2}}$ on the public-key (pk_1,pk_2,r).

2. *Pre-challenge decryption:* For every decryption request (c_1,c_2,π) given by $\mathcal{A}_{\text{CCA2}}$ to A_{CPA}, adversary A_{CPA} decrypts by first checking that the proof π is correct, and then decrypting c_2. (Recall that A_{CPA} knows the decryption key sk_2 and is therefore able to decrypt c_2.)

3. *Challenge generation:* Following the above stage, $\mathcal{A}_{\text{CCA2}}$ outputs two plaintext messages m_0 and m_1. A_{CPA} also outputs these messages and receives back a challenge ciphertext $c_1=E_{pk_1}(m_b)$ for $b\in_R\{0,1\}$. Then, A_{CPA} computes $c_2=E_{pk_2}(m_0)$. Furthermore, A_{CPA} runs the NIZK simulator to obtain a proof π that c_1 and c_2 are encryptions to the same message (although this may not be true). A_{CPA} then passes the challenge ciphertext (c_1,c_2,π) to $\mathcal{A}_{\text{CCA2}}$.

4. *Post-challenge decryption:* A_{CPA} continues to decrypt messages for $\mathcal{A}_{\text{CCA2}}$ in the same way as in the preprocessing stage.[4]

5. *Output guess:* Finally, $\mathcal{A}_{\text{CCA2}}$ outputs a guess b' for the bit b. Likewise, A_{CPA} outputs this same bit b'.

[4]This step in the simulation is essentially different to [12]. In their scheme, the NIZK used is not strong enough to also ensure soundness in the post-challenge decryption phase. Therefore, the simulation must halt after the "challenge generation" phase. The result is CCA1 security.

The main point of the simulation is as follows: by Claim A.2, all ciphertext queries made by $\mathcal{A}$ that have accepting NIZKs are also valid (except with negligible probability). Therefore, in order to simulate the decryption oracle, it is enough to verify the NIZK proof and decrypt using one of *either of the two* secret-keys. This is in contrast to the actual decryption oracle who always uses a specified key; in Construction 2, this key was specified to be sk_1. Therefore, despite the fact that A_{CPA} uses sk_2 in the simulation and not sk_1, its simulation of the decryption oracle is correct (except with negligible probability).[5]

Now, if the challenge ciphertext received by A_{CPA} is an encryption of m_0, then the probability that the output of A_{CPA}'s simulation equals 1 is negligibly close to that of $\mathsf{Expt}^S_{\mathcal{A}}(0,0)$. On the other hand, if the challenge received by A_{CPA} is an encryption of m_1, then the probability that the output of A_{CPA}'s simulation equals 1 is negligibly close to that of $\mathsf{Expt}^S_{\mathcal{A}}(1,0)$. It follows that A_{CPA}'s advantage in distinguishing between encryptions of m_0 and encryptions of m_1 is negligibly close to $|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(0,0)=1] - \Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,0)=1]|$. By the chosen-plaintext security of (G,E,D), we have that this difference is negligible, as required. ■

This completes the proof of the theorem. ■

[5]We note that the proof of the fact that $\left|\Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,0)=1] - \Pr[\mathsf{Expt}^S_{\mathcal{A}}(1,1)=1]\right| < \mu(n)$ is easier regarding this point. There, A_{CPA} is defined analogously except that it uses sk_1 to decrypt. Therefore, its simulation of the decryption oracle in $\mathsf{Expt}^S_{\mathcal{A}}(1,*)$ is perfect, and Claim A.2 is not needed.