Deniable Encryption with Negligible Detection Probability: An Interactive Construction

Markus Dürmuth[1,⋆] and David Mandell Freeman[2,⋆⋆]

[1] Ruhr-University Bochum, Germany
markus.duermuth@rub.de
[2] Stanford University, USA
dfreeman@cs.stanford.edu

Abstract. *Deniable encryption*, introduced in 1997 by Canetti, Dwork, Naor, and Ostrovsky, guarantees that the sender or the receiver of a secret message is able to "fake" the message encrypted in a specific ciphertext in the presence of a coercing adversary, without the adversary detecting that he was not given the real message. To date, constructions are only known either for weakened variants with separate "honest" and "dishonest" encryption algorithms, or for single-algorithm schemes with non-negligible detection probability.

We propose the first sender-deniable public key encryption system with a single encryption algorithm and negligible detection probability. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

Keywords: Deniable encryption, electronic voting, multi-party computation.

1 Introduction

One of the central goals of cryptography is protecting the secrecy of a transmitted message. The secrecy property of an encryption scheme is usually formalized as semantic security [10], which guarantees that an adversary cannot gain even partial information about an encrypted message.

The notion of semantic security has proven to be very useful in a large number of applications. However, there are some scenarios where semantic security is not sufficient. For example, semantic security does not ensure message secrecy if the adversary can coerce the sender or the receiver of a message to reveal the secret keys and/or the randomness that was used to form an encryption. Specifically, semantic security does not prevent an encryption scheme from being committing, in the sense that if an adversary sees a ciphertext and then tries to coerce the sender to reveal all of the input to the encryption (message and randomness), any inputs that the sender can reveal that are consistent with the ciphertext must reveal the true message encrypted. In fact, many encryption schemes have only one set of possible inputs per ciphertext.

⋆ Research conducted at Stanford University.
⋆⋆ Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 610–626, 2011.

This committing property of encryption can be problematic in applications such as electronic voting [15] or keeping information secret when facing a coercer using physical force, or in the case of secure multi-party computation in the presence of an adaptive adversary [4].

Deniable encryption, introduced by Canetti, Dwork, Naor, and Ostrovsky in 1997 [3], possesses a stronger security property than semantic security that avoids these shortcomings. Informally, a (possibly interactive) public key encryption scheme is *sender-deniable* if, given an encryption $c = \mathsf{Enc}_{\mathsf{pk}}(m_1; r_1)$ of a message m_1 with randomness r_1, for every message m_2 the sender can compute alternative randomness r_2 such that $(\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1), r_1)$ and $(\mathsf{Enc}_{\mathsf{pk}}(m_2; r_2), r_2)$ are computationally indistinguishable. Thus when coerced, the sender can reveal either the "real" randomness r_1 or the "fake" randomness r_2, and the adversary cannot tell whether m_1 or m_2 was really encrypted in the ciphertext c. A similar property defines *receiver-deniability*, and a system with both properties is *sender-and-receiver-deniable* or *bi-deniable*.

Canetti et al. also considered a weaker form of deniability, called *flexible deniability*. In a flexibly deniable system, the sender has two different encryption algorithms, *honest* and *dishonest* encryption. At the time of encryption the sender chooses either honest or dishonest encryption. When choosing honest encryption he is unable to fake later when coerced, but when choosing dishonest encryption he can create randomness such that the distribution matches randomness for *honest* encryption. What makes this notion weak for practical purposes is that if the adversary believes that the sender used the dishonest encryption, then the adversary can coerce the sender to reveal randomness for the dishonest encryption, and the sender looses his ability to fake. The system's security relies on the fact that the adversary really believes that the sender used the honest encryption.

Canetti et al. constructed a non-interactive sender-deniable encryption scheme that transmits bits, based on a primitive called *translucent sets* that can be realized under trapdoor permutations and other standard assumptions. The system is semantically secure, but it has only $1/O(n)$-deniability. That is, an adversary can detect whether the user is "faking" messages and randomness with probability $1/O(n)$, where n is the system's security parameter. Canetti et al. left open the problem of constructing an encryption scheme with negligible deniability. This problem has remained open for more than 13 years.

Our Contribution. We give the first public key encryption scheme that satisfies the definition of sender-deniability in [3] with a single encryption algorithm and negligible probability of detection. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

1.1 Overview of Our Construction

The basic idea of our construction is the following. We take a public key bit encryption scheme with dense ciphertexts, in which a uniformly random ciphertext decrypts to a uniformly random message. To encrypt a bit $b \in \{0, 1\}$, we first obtain $4n + 1$ public

keys for the underlying encryption scheme. We construct $n + 1$ encryptions of b, construct n encryptions of $1 - b$, and sample $2n$ random ciphertexts, each under a different public key, and then permute the output randomly. Decrypting all ciphertexts individually and taking the majority recovers the original message with noticeable probability. Repeating the protocol multiple times in parallel reduces the decryption error.

To fake the sender's input, we claim that a constructed ciphertext encrypting b was sampled randomly. This trick has been used before, but it usually gives an adversary a non-negligible probability of detecting the faking (e.g., inverse linear in the ciphertext length as in [3]). What is unique in our construction is that we additionally construct "real-looking" randomness for a *sampled* ciphertext encrypting $1 - b$, so we obtain a distribution which is computationally indistinguishable with *negligible* advantage for any efficient adversary.

In order to compute fake randomness for a sampled ciphertext, we need two ingredients. First, the encryption scheme needs to have the property that given a ciphertext, the secret key holder can compute randomness that is indistinguishable from the randomness used to compute the ciphertext. Second, the sender must know which sampled ciphertext she should use in her faking and she must obtain a secret key for that ciphertext. (After all, she doesn't know which sampled ciphertexts encrypt b and which encrypt $1 - b$.) On the other hand, the receiver doesn't know which ciphertexts were sampled and which were constructed. Our basic idea for giving the sender the necessary information is to have the receiver send back pairs of indices for ciphertexts that decrypt to opposite plaintexts, so that with high probability one of the pairs corresponds to a constructed encryption of b and a sampled encryption of $1 - b$. The sender then indicates one such pair and obtains from the receiver the secret keys for both elements of that pair. Since the two ciphertexts encrypt opposite messages, these revealed values do not compromise the secrecy of the system.

Once in possession of the correct secret key, the sender can fake randomness. Deniability ultimately rests on the semantic security of the underlying encryption scheme, as our manipulation changes the distribution on the set of sampled ciphertexts whose indices were *not* sent back to the sender in the above process.

To instantiate our system, we observe that two well known encryption schemes have the properties necessary for our construction: the Goldwasser-Micali bit encryption scheme based on the quadratic residuosity assumption [10], and a simple bit encryption scheme constructed from a trapdoor permutation using a hard-core predicate.

1.2 Related Work

In addition to their sender-deniable scheme with $1/O(n)$-deniability, Canetti et al. [3] also constructed a flexible (i.e., two-algorithm) sender-deniable encryption scheme with negligible deniability. More recently, O'Neill, Peikert, and Waters [12] announced a flexible *bi*-deniable encryption scheme with negligible deniability based on lattice assumptions. We view this latter work as orthogonal to our own: it is non-interactive and achieves deniability for both sender and receiver simultaneously, but the construction uses in an essential way the fact that there are different honest and dishonest encryption algorithms.

A related concept originating from adaptive security of multi-party computation is *non-committing encryption* [4,6,8,2]. Informally, a public key bit encryption scheme is non-committing if a simulator can efficiently sample a distribution of ciphertexts c and two sets of randomness r_0, r_1 such that the ciphertext c along with r_b is indistinguishable from a legitimate encryption of b along with the true randomness. The main difference between non-committing encryption and deniable encryption is while the *simulator* can generate ciphertexts and randomness corresponding to either message, a *user* cannot, in general, compute randomness that reveals a different message than was actually encrypted. While deniable encryption implies non-committing encryption, the converse does not hold; in particular, the non-committing encryption scheme in [4] is not deniable.

A different concept is *plausible deniability*. This term usually describes engineering techniques that allow one to deny the existence of encrypted data or knowledge of the secret key. A well known example is the TrueCrypt file system encryption [17], where one can add secret containers inside an encrypted volume such that one can deny their existence. Another example [5] uses steganographic measures to hide encrypted data in cover text. These techniques come without formal proof or even definition, and attacks exist that can reveal the presence and content of encrypted data [7]. In addition, this form of deniability usually is not suitable for online applications such as electronic voting.

1.3 Outline

In Section 2 we give the formal definition of a deniable encryption scheme. In Section 3 we describe the building block for our deniable protocol, which we call a *samplable* public key bit encryption scheme. In Section 4 we construct an interactive encryption scheme from a samplable public key bit encryption scheme. We prove our scheme is secure and deniable under the assumption that the underlying scheme is samplable and semantically secure. In Section 5 we describe two samplable public key bit encryption schemes, one based on the quadratic residuosity assumption and one based on trapdoor permutations. Finally, in Section 6 we discuss some open questions related to our work.

2 Deniable Encryption

We begin by fixing some notation. If X is a set and Δ is a distribution on X, we use $x \leftarrow \Delta$ to denote an element of X sampled according to the distribution Δ, and we use $x \leftarrow y$ to denote assignment of the value y to x. If X is finite we use $x \xleftarrow{\mathrm{R}} X$ to denote an element sampled uniformly at random from X. For a two-party protocol π between S and R we write

$$(o_S, o_R, tr) \leftarrow \pi((i_S; r_S), (i_R; r_R))$$

for the execution of π with input i_S, i_R and randomness r_S, r_R from S and R, respectively, producing output o_S, o_R for the respective parties, and a public transcript tr.

A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is said to be *negligible* iff for all $c \in \mathbb{N}$ we have that $|\mu(n)| \leq \frac{1}{n^c}$ for sufficiently large n. If this is the case, we write $\mu(n) = \mathrm{negl}(n)$. A probability p is said to be *overwhelming* if $p = 1 - \mathrm{negl}(n)$.

Two sequences of random variables $(X_n)_{n\in\mathbb{N}}, (Y_n)_{n\in\mathbb{N}}$ are $\mu(n)$-*computationally indistinguishable*, and denoted by $(X_n) \approx^{\mu(n)} (Y_n)$, if for all polynomial-time adversaries $\mathcal{A}$ we have $|\Pr[\mathcal{A}(x) = 1; x \leftarrow X_n] - \Pr[\mathcal{A}(x) = 1; x \leftarrow Y_n]| \leq \mu(n)$ for sufficiently large n. We say (X_n) and (Y_n) are *statistically indistinguishable*, denoted $(X_n) \approx (Y_n)$, if the same holds for *all* adversaries $\mathcal{A}$, regardless of running time.

Our definition of deniable encryption is a slight rephrasing of the definition of Canetti et al. [3].

Definition 2.1. Let $n \in \mathbb{N}$ be a security parameter. An efficiently computable protocol[1] π between two parties S and R (*sender* and *receiver*, respectively) is called a $\mu(n)$-*sender-deniable public key bit encryption scheme* if the following three conditions are satisfied:

Correctness: We say π is *correct* if for all messages $b \in \{0,1\}$ we have

$$\Pr\left[b' \neq b;\ r_S, r_R \xleftarrow{R} \{0,1\}^*, (\cdot, b', \cdot) \leftarrow \pi((b; r_S), (n; r_R))\right] \ \leq\ \nu(n)$$

for some negligible function $\nu\colon \mathbb{N} \to \mathbb{R}$.

Passive secrecy: The two random variables tr_0, tr_1 defined by

$$\begin{aligned} r_S, r_R \xleftarrow{R} \{0,1\}^*,\ & (\cdot,\cdot,tr_0) \leftarrow \pi((0; r_S), (n; r_R)), \\ & (\cdot,\cdot,tr_1) \leftarrow \pi((1; r_S), (n; r_R)), \end{aligned} \tag{2.1}$$

are $\nu(n)$-computationally indistinguishable for some negligible $\nu(n)$.

Deniability: There is an efficient *faking algorithm* Fake that takes input a message $b \in \{0,1\}$, sender randomness $r_S \in \{0,1\}^*$, and a transcript of the protocol π, such that for any $b \in \{0,1\}$ and

$$\begin{aligned} r_S, r_R \xleftarrow{R} \{0,1\}^*,\ & (\cdot,\cdot,\widetilde{tr}) \leftarrow \pi((1-b; r_S), (n; r_R)), \\ \tilde{r}_S \leftarrow \mathsf{Fake}(b, r_S, \widetilde{tr}),\ & (\cdot,\cdot,tr) \leftarrow \pi((b; r_S), (n; r_R)), \end{aligned} \tag{2.2}$$

the following two distributions are $\mu(n)$-computationally indistinguishable:

$$(b, r_S, tr) \approx^{\mu(n)} (b, \tilde{r}_S, \widetilde{tr}). \tag{2.3}$$

The distribution on the left is produced by a real encryption of b with randomness r_S. The distribution on the right is produced when we actually encrypt $1-b$ with randomness r_S, but tell the adversary that we encrypted b with randomness $\tilde{r}_S$.

We call π a *sender-deniable public key bit-encryption scheme* if it is $\mu(n)$-sender-deniable for a negligible $\mu(n)$. We can define a *receiver-deniable* encryption scheme analogously by having Fake produce fake receiver randomness $\tilde{r}_R$. A *sender-and-receiver-deniable* or *bi-deniable* scheme has both properties, with a faking algorithm for each party.

A straightforward reduction shows that deniability (with negligible $\mu(n)$) implies passive secrecy.

[1] I.e., all computations run in (expected) time polynomial in n and the number of rounds is polynomial in n.

Lemma 2.2. *A two-party protocol π that has the deniability property of Definition 2.1 for a negligible $\mu(n)$ also has the passive secrecy property.*

Proof. Assume there exists an efficient adversary $\mathcal{A}$ that can distinguish transcripts of π corresponding to messages 0 and 1 with non-negligible probability. Now if we are given a challenge (m, r_S, tr) for deniability, we can use $\mathcal{A}$ to decide if tr is an encryption of b or $1-b$, and thus can win the deniability game with the same probability and the same running time. This contradicts the hypothesis that π has the deniability property for negligible $\mu(n)$. □

3 Samplable Public Key Encryption

As a building block for our deniable encryption scheme, we use a public key bit encryption scheme for which a secret key holder can recover randomness used in the encryption. We do not require the recovered randomness to be exactly that used to encrypt, as this may be impossible to compute, but rather that it be indistinguishable from the real randomness. In this section we formalize this idea.

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption system encrypting messages in $\{0,1\}$. We write the encryption algorithm as

$$\mathsf{Enc}\colon \mathcal{PK} \times \{0,1\} \times \mathcal{R} \to \mathcal{C}$$

and the decryption algorithm as

$$\mathsf{Dec}\colon \mathcal{SK} \times \mathcal{C} \to \{0,1\},$$

where $\mathcal{PK}$, $\mathcal{SK}$, $\mathcal{R}$, and $\mathcal{C}$ are the spaces of public keys, secret keys, sender randomness, and ciphertexts, respectively. The key spaces $\mathcal{PK}$ and $\mathcal{SK}$ depend on the security parameter n, and the spaces $\mathcal{R}$ and $\mathcal{C}$ also depend on the public key pk used in the encryption; for the sake of readability we omit these dependencies from the notation.

We denote the encryption of a bit b under public key pk with randomness r by $\mathsf{Enc}_{\mathsf{pk}}(b; r)$, and the system's security parameter (input to KeyGen) by n. We let $\Delta_{\mathcal{R}}$ denote the distribution on $\mathcal{R}$ that the encryption algorithm samples. We require the usual correctness condition: for all $\mathsf{pk} \in \mathcal{PK}$, all $\mathsf{sk} \in \mathcal{SK}$, and $b \in \{0,1\}$, we have $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(b; r)) = b$ with overwhelming probability over $r \leftarrow \Delta_{\mathcal{R}}$.

Definition 3.1. We say that $\mathcal{E}$ is *samplable* if the following conditions hold:

1. The set $\mathcal{C}$ is finite, and the distribution on $\mathcal{C}$ given by
$$(\mathsf{Enc}_{\mathsf{pk}}(b; r) : b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}, r \leftarrow \Delta_{\mathcal{R}}) \tag{3.1}$$
is statistically indistinguishable from the uniform distribution on $\mathcal{C}$.
2. There is an efficient algorithm $\mathsf{SampleRand}$ that takes as input a secret key and a ciphertext and outputs a value in $\mathcal{R}$, such that if we choose
$$b \stackrel{\mathrm{R}}{\leftarrow} \{0,1\},\ r \leftarrow \Delta_{\mathcal{R}},$$
$$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(b; r),\ \tilde{r} \leftarrow \mathsf{SampleRand}(\mathsf{sk}, c),$$

then the following two distributions are statistically indistinguishable:

$$(\mathsf{sk}, c, r) \approx (\mathsf{sk}, c, \tilde{r}). \tag{3.2}$$

Note that the second condition implies that for all but a negligible fraction of $c \in \mathcal{C}$, we have

$$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}}(c); \mathsf{SampleRand}_{\mathsf{sk}}(c)) = c.$$

We present two examples of samplable encryption schemes in Section 5.

4 A Deniable Encryption Protocol

We are now ready to construct a sender-deniable encryption scheme from any samplable public key bit-encryption scheme. Let $n \in \mathbb{N}$ be a security parameter. At a high level, our protocol consists of the following exchange, iterated n times to ensure correctness of decryption. Since the "sender" and "receiver" are both sending and receiving messages, for clarity we describe our protocol in terms of two players Sarah and Ronald. Sarah has a bit b that she wishes to transmit to Ronald in a deniable manner.

To encrypt the bit b, first Ronald sends $4n + 1$ freshly generated public keys pk_i to Sarah. Sarah partitions the indices $\{0, \ldots, 4n\}$ into three random disjoint subsets and computes $4n + 1$ ciphertexts as follows:

- Choose a set A containing $n + 1$ indices and encrypt b under the key pk_i for $i \in A$.
- Choose a set B disjoint from A containing n indices and encrypt $1 - b$ under the key pk_i for $i \in B$.
- Let C denote the remaining $2n$ indices, and sample a uniformly random ciphertext c_i from $\mathcal{C}$ for $i \in C$. Definition 3.1 guarantees that approximately half of these ciphertexts decrypt to b.

Ronald computes a bit b' by decrypting all of the received ciphertexts and taking the majority of the plaintexts. We will show that the probability that $b' = b$ is at least $1/2 + 1/5\sqrt{n}$; thus repeating the protocol independently n times ensures that Ronald can compute b correctly with overwhelming probability[2].

When coerced to reveal her randomness, Sarah will reveal sets $\tilde{A}, \tilde{B}, \tilde{C}$ in which a real encryption of b from the set A is exchanged with a sampled encryption of $1-b$ from the set C. She must also reveal appropriately distributed randomness for the sampled encryption. However, to do this she needs Ronald's help; in particular, she needs a secret key for an index $i \in C$ such that c_i encrypts $1 - b$. Since Ronald cannot tell which encryptions were sampled randomly, he sends back $n/2$ random pairs of indices (u_i, v_i) such that c_{u_i} and c_{v_i} decrypt to different messages. Almost certainly, he will choose at least one pair (u_j, v_j) such that $u_j \in A$ corresponds to an encryption of b and $v_j \in C$ corresponds to an encryption of $1 - b$, or vice versa. Sarah indicates such a pair, and Ronald sends the corresponding secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$. Knowing the secret key for a randomly generated ciphertext enables Sarah to use the $\mathsf{SampleRand}$ algorithm to produce randomness which is indistinguishable from "real" randomness that would produce the given ciphertext.

We now formally describe our scheme.

[2] More precisely, we can obtain correctness with overwhelming probability by repeating the protocol $f(n)$ times for any $\omega(\sqrt{n \log n})$ function f; we use $f(n) = n$ for simplicity.

4.1 The Protocol

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SampleRand})$ be a samplable public key bit encryption scheme. Define a protocol $\pi_{\mathcal{E}}$ between a sender Sarah and a receiver Ronald as follows:

1. Ronald's input is a security parameter $n \in \mathbb{N}$; we assume for simplicity that n is even.
   (a) Choose key pairs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^n)$ for $i = 0, \ldots, 4n$.
   (b) Send $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{4n})$ to Sarah.
2. Sarah's input is a bit $b \in \{0, 1\}$. Sarah computes $n + 1$ encryptions of b and n encryptions of $1 - b$ under different keys, and chooses $2n$ additional random ciphertexts:
   (a) Choose a random partition of $\{0, \ldots, 4n\}$ into disjoint subsets A, B, C of cardinality $n + 1$, n, and $2n$, respectively.
   (b) For $i \in A$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i = b$, and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.
   (c) For $i \in B$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i = 1 - b$, and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.
   (d) For $i \in C$, choose random ciphertexts $c_i \overset{\mathrm{R}}{\leftarrow} \mathcal{C}$.
   (e) Send $(c_0, \ldots, c_{4n})$ to Ronald.
3. Ronald decrypts all ciphertexts: he sets $\beta'_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i)$ for $i = 0, \ldots, 4n$, and outputs the majority b'.
4. Ronald sends pairs of indices corresponding to ciphertexts with opposite messages back to Sarah.
   (a) Set $I = \varnothing$. Do the following for $i = 1, \ldots, n/2$.
      i. Choose a random pair of indices $(u_i, v_i) \in \{0, \ldots, 4n\}$ such that $u_i, v_i \notin I$ and $\beta'_{u_i} \neq \beta'_{v_i}$.
      ii. Set $I \leftarrow I \cup \{u_i, v_i\}$.
   (b) Ronald sends $(u_1, v_1), \ldots, (u_{n/2}, v_{n/2})$ to Sarah.
5. Sarah chooses a pair of indices such that one index is in the set A and one is in the set C:
   (a) Choose a random $j \in \{1, \ldots, n/2\}$ such that either $u_j \in A$ and $v_j \in C$ or $u_j \in C$ and $v_j \in A$. If no such index exists, then abort the protocol.
   (b) Send j to Ronald.
6. Ronald sends the secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ to Sarah.

This completes the description of the protocol.

The Transcript. The protocol's transcript consists of:
1. The public keys $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{4n})$ sent by Ronald in Step 1,
2. The ciphertexts $(c_0, \ldots, c_{4n})$ sent by Sarah in Step 2e,
3. The tuples (u_i, v_i) sent by Ronald in Step 4,
4. The index j Sarah has chosen in Step 5 (or the decision to abort), and
5. The secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ sent by Ronald in step 6.

The Faking Algorithm. The algorithm Fake is defined as follows: suppose we are given a message b, randomness

$$r_S = \left(A, B, C, (\alpha_i)_{i\in A\cup B}, (c_i)_{i\in C}\right) \tag{4.1}$$

and a transcript tr. If tr indicates that the protocol has aborted in Step 5a, then Fake outputs $\tilde{r}_S = \perp$. Otherwise, do the following:

1. Let $y = \{u_j, v_j\} \cap A$ and $z = \{u_j, v_j\} \cap C$.
   (In particular, we have $\beta'_y = b$ and $\beta'_z = 1 - b$.)
2. Compute
   - $\tilde{A} \leftarrow B \cup \{z\}, \quad \tilde{B} \leftarrow A \setminus \{y\}, \quad \tilde{C} \leftarrow (C \setminus \{z\}) \cup \{y\}$.
   - $\tilde{\alpha}_i \leftarrow \alpha_i$ for $i \in (A \setminus \{y\}) \cup B$.
   - $\tilde{\alpha}_z \leftarrow \mathsf{SampleRand}_{\mathsf{sk}_z}(c_z)$.
   - $\tilde{c}_i \leftarrow c_i$ for $i \in \tilde{C}$.
3. Output
$$\tilde{r}_S = \left(\tilde{A}, \tilde{B}, \tilde{C}, (\tilde{\alpha}_i)_{i\in\tilde{A}\cup\tilde{B}}, (\tilde{c}_i)_{i\in\tilde{C}}\right). \tag{4.2}$$

We denote the n-fold independent execution of the encryption protocol by $\pi^n_{\mathcal{E}}$, where the final output is determined as the majority of the output bits of the individual instances.

It is clear that the protocol runs in time polynomial in n. We now show that the protocol almost never aborts in Step 5a.

Proposition 4.1. *The probability that Sarah aborts the protocol $\pi_{\mathcal{E}}$ in Step 5a is negligible in n.*

Proof. For $\beta \in \{0, 1\}$, let $C_\beta := C \cap \{i : \mathsf{Dec}_{\mathsf{sk}_i}(c_i) = \beta\}$. By Property 1 of Definition 3.1 and Chernoff bounds [14, Ch. 8, Prop. 5.3] we have, with overwhelming probability, $\frac{7n}{8} \le |C_\beta| \le \frac{9n}{8}$ for $\beta \in \{0, 1\}$. Now in each iteration of Step 4(a)i, the probability of choosing a pair (u_i, v_i) with either $u_i \in A$ and $v_i \in C$ or $v_i \in A$ and $u_i \in C$ is at least

$$\frac{|A| - n/2}{|A| + |C_b|} \cdot \frac{|C_{1-b}| - n/2}{|C_{1-b}| + |B|} \ge \frac{n/2}{18n/8} \cdot \frac{3n/8}{17n/8} = \frac{2}{51}$$

whenever $n \ge 8$. Since $n/2$ pairs are chosen in total, by another application of Chernoff bounds we obtain that the probability that no suitable pair is chosen is negligible in n. □

4.2 Correctness

Lemma 4.2. *Let $\mathcal{E}$ be a samplable public key bit encryption scheme, and let b' be Ronald's output computed by $\pi_{\mathcal{E}}((b, r_S), (n, r_R))$. Then the probability (over r_S and r_R) that $b' = b$ is at least $1/2 + 1/(5\sqrt{n})$.*

Proof. A single instance of $\pi_{\mathcal{E}}$ outputs the majority of messages obtained from decrypting all $4n + 1$ ciphertexts c_i. Out of these, c_i encrypts b for $i \in A$ and c_i encrypts $1 - b$ for $i \in B$. This means that $b' = b$ if and only if least half of the remaining $2n$ ciphertexts $\{c_i : i \in C\}$ decrypt to b.

Property 1 of Definition 3.1 implies that a single c_i decrypts to b with probability $p(n)$ satisfying $|1/2 - p(n)| = \nu(n)$ for some negligible $\nu(n)$. The probability that at least half of the ciphertexts $\{c_i : i \in C\}$ decrypt to b is thus

$$\begin{aligned}
\sum_{i=n}^{2n} \binom{2n}{i} p(n)^i \cdot (1-p(n))^{2n-i} &\geq \sum_{i=n}^{2n} \binom{2n}{i} \left(\frac{1}{2} - \nu(n)\right)^{2n} \\
&= \left(\frac{1}{2} - \nu(n)\right)^{2n} \cdot \frac{1}{2}\left(\binom{2n}{n} + \sum_{i=0}^{2n} \binom{2n}{i}\right) \\
&\overset{(*)}{\geq} (1 - 2\nu(n))^{2n} \cdot \left(\frac{1}{4\sqrt{n}} + \frac{1}{2}\right) \\
&\geq 1/2 + \frac{1}{5\sqrt{n}},
\end{aligned}$$

where $(*)$ follows from the inequality $\binom{2n}{n} \geq 2^{2n-1}/\sqrt{n}$, which in turn follows from Stirling's approximation $n! \sim \sqrt{2\pi n} \cdot e^{-n} \cdot n^n$ [16, p. 13]. □

4.3 Deniability

Our main result is the following:

Theorem 4.3. *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. If $\mathcal{E}$ is semantically secure and samplable, then $\pi_{\mathcal{E}}^n$ is a sender-deniable encryption scheme.*

Proof. We must show that the n-fold repetition of $\pi_{\mathcal{E}}$ satisfies the three conditions of Definition 2.1 for negligible $\mu(n)$. Lemma 4.2 shows that the probability that Ronald's message b' is equal to Sarah's message b is at least $1/2 + 1/(5\sqrt{n})$. Thus repeating the protocol n times and taking the majority of the b' gives the correct answer b with overwhelming probability by using Chernoff bounds [14, Ch. 8, Prop. 5.3].

Next, Lemma 2.2 shows that passive secrecy follows from deniability. It thus only remains to prove deniability. We show that if $\mathcal{E}$ is semantically secure and samplable, then for a *single* execution of $\pi_{\mathcal{E}}$, the two distributions of (2.3) are $\mu(n)$-computationally indistinguishable for some negligible $\mu(n)$. Assuming this is the case, a standard hybrid argument shows that the corresponding distributions for the n-fold parallel repetition of $\pi_{\mathcal{E}}$ are $\mu'(n)$-computationally indistinguishable for some negligible $\mu'(n)$.

We now consider a single execution of $\pi_{\mathcal{E}}$ and define a series of games. Game_0 will output the distribution of the left hand side of (2.3), while Game_8 will output the distribution of the right hand side of (2.3). We will then show that for all i, the outputs of Game_i and Game_{i+1} are $\mu_i(n)$-computationally indistinguishable for some negligible $\mu_i(n)$. By the triangle inequality, this implies that the two distributions in (2.3) are $\mu(n)$-computationally indistinguishable for some negligible $\mu(n)$.

We now define our series of games. In each game Ronald's randomness r_R and the security parameter n are taken to be the same. Unless otherwise stated, the output of Game_i is the same as that of Game_{i-1}.

Game_0: The two parties run the protocol $\pi_{\mathcal{E}}$ as defined in Section 4.1, with Sarah's input message b and randomness r_S given by (4.1). The game outputs the message b, the randomness r_S, and the transcript tr.

Game_1: The two parties run the protocol as in Game_0, but change Step 2d as follows:

(2d)' For $i \in C$, choose random $\beta_i \xleftarrow{\mathrm{R}} \{0,1\}$ and $\alpha_i \leftarrow \Delta_{\mathcal{R}}$ and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.

The ciphertexts in tr are still the ciphertexts $c_0, \ldots, c_{4n}$.

Game_2: The two parties run the protocol as in Game_1, but change Steps 2e and 3 as follows:

(2e)' Send $(\beta_0, \ldots, \beta_{4n})$ to Ronald.

(3)' Ronald sets $\beta'_i \leftarrow \beta_i$ for all i and outputs the majority b' of the β'_i.

The ciphertexts $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\beta_i; \alpha_i)$ are now computed at the time the transcript is output. The output is the same as in Game_1.

Game_3: The two parties run the protocol as in Game_2, with an additional step that flips the bit of one ciphertext:

7. (a) Choose a random $x \in C$, not equal to any of the indices sent in Step 4, with $\beta_x = b$.
   (b) Compute $\tilde{c}_x \leftarrow \mathsf{Enc}_{\mathsf{pk}_x}(1 - \beta_x; \alpha_x)$.

The output is the same as in Game_2, except the ciphertext $\tilde{c}_x$ is output instead of c_x.

Game_4: The two parties run the protocol as in Game_3, but now Sarah flips all of the bits that she sends to Ronald (including the bit in ciphertext c_x), while still computing the ciphertexts in the transcript from the original bits (with only β_x flipped). For easier readability we restate most of the protocol.

1. Ronald sends $4n + 1$ key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ to Sarah.
2. Sarah does the following:
   (a) Choose a random partition of $\{0, \ldots, 4n\}$ into disjoint subsets A, B, C of cardinality $n + 1$, n, and $2n$, respectively.
   (2b)' For $i \in A$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i \leftarrow b$ and $\tilde{\beta}_i \leftarrow 1 - b$.
   (2c)' For $i \in B$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i \leftarrow 1 - b$ and $\tilde{\beta}_i = b$.
   (2d)" For $i \in C$ choose random $\beta_i \xleftarrow{\mathrm{R}} \{0,1\}$ and $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\tilde{\beta}_i \leftarrow 1 - \beta_i$.
   (2e)" Send $(\tilde{\beta}_0, \ldots, \tilde{\beta}_{4n})$ to Ronald.

3." Ronald sets $\beta'_i = \tilde{\beta}_i$ and outputs the majority b'.

4. Ronald sends pairs of indices corresponding to ciphertexts with opposite messages back to Sarah. (This step is identical to the corresponding step in the real protocol.)
5. Sarah chooses a pair of indices such that one index is in the set A and one is in the set C. (This step is identical to the corresponding step in the real protocol.)
6. Ronald sends the secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ to Sarah.
7. (a) Choose a random $x \in C$, not equal to any of the indices sent in Step 4, with $\beta_x = b$.
   (b) Let $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$ for all $i \neq x$, and let $\tilde{c}_x \leftarrow \mathsf{Enc}_{\mathsf{pk}_x}(1 - \beta_x; \alpha_x)$.

The output is the same as in Game_3.

Game_5: The two parties run the protocol as in Game_4, but now Sarah uses the flipped bits $\tilde{\beta}_i$ to compute all ciphertexts (including the ciphertext c_x).

7. Set $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\tilde{\beta}_i; \alpha_i)$ for all i.
8. The game outputs the protocol's transcript and computes fake randomness:
   - $\tilde{A} \leftarrow B \cup \{z\}, \quad \tilde{B} \leftarrow A \setminus \{y\}, \quad \tilde{C} \leftarrow (C \setminus \{z\}) \cup \{y\}$.
   - $\tilde{\alpha}_i \leftarrow \alpha_i$ for $i \in \tilde{A} \cup \tilde{B}$.
   - $\tilde{c}_i = c_i$ for $i \in \tilde{C}$.

   We define $\tilde{r}_S$ as in (4.2), and the game outputs $\tilde{r}_S$ instead of r_S.

Game_6: The two parties run the protocol as in Game_5, but Sarah fakes the randomness for the ciphertext with index z by choosing $\tilde{\alpha}_z \leftarrow \mathsf{SampleRand}_{\mathsf{sk}_z}(c_z)$. (Recall $z = \{u_j, v_j\} \cap C$.) This $\tilde{\alpha}_z$ is output in the appropriate position of $\tilde{r}_S$. Other than this change, the output is the same as in Game_5.

Game_7: We reverse the change made between Game_1 and Game_2, by setting

(2b–2d) Set $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\tilde{\beta}_i; \alpha_i)$ for all i.
(2e) Send $(c_0, \ldots, c_{4n})$ to Ronald.
(3) Set $\beta'_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i)$ for $i = 0, \ldots, 4n$, and output the majority b'.

Game_8: We reverse the change made between Game_0 and Game_1, by setting

(2d) For $i \in C$, choose random ciphertexts $c_i \overset{\mathrm{R}}{\leftarrow} \mathcal{C}$.

Therefore the output of Game_8 is the message b, the transcript $\widetilde{tr}$ of a real encryption of $1 - b$ using Sarah's randomness r_S, and the fake randomness $\tilde{r}_S = \mathsf{Fake}(b, r_S, \widetilde{tr})$.

Since the output of Game_0 is distributed as the left hand side of (2.3) and the output of Game_8 is distributed as the right hand side of (2.3), it suffices to show that for each i in $1, \ldots, 8$, if there is an efficient adversary that can distinguish the output of Game_i from that of Game_{i-1}, then this adversary can be used to solve some problem that we have assumed to be (computationally or statistically) infeasible. We now consider each pair of games in turn.

$\mathsf{Game}_0 \rightarrow \mathsf{Game}_1$: The fact that $\mathcal{E}$ is samplable implies that the outputs of these two games are statistically indistinguishable. To show this, we put an ordering on C and define hybrid games for $j = 0, \ldots, 2n$ in which the first j ciphertexts c_i for $i \in C$ are chosen at random from C, and the last $2n - j$ are chosen as real encryptions of random bits. The 0th hybrid is Game_0 and the $2n$th hybrid is Game_1.

Suppose we are given a ciphertext X chosen as either a real encryption of a random bit or as a uniformly random ciphertext. Let x be the index of the ciphertext that changes between the jth and $(j+1)$th hybrid. We can simulate the protocol by choosing secret keys sk_i for all $i \neq x$ and using X as c_x. When X is a random ciphertext for pk_x we are in the jth hybrid, and when X is a real encryption of a random bit under pk_x we are in the $(j+1)$th. Thus any adversary that can distinguish these two hybrids can distinguish a random ciphertext from a real encryption of a random bit, which contradicts the assumption that $\mathcal{E}$ is samplable.

$\mathsf{Game}_1 \to \mathsf{Game}_2$: Note that the output of Ronald in Step 4, as well as the remainder of the protocol, depends only on the *plaintexts* of the messages he receives in Step 2e. Since Sarah now knows all of the plaintexts, she can send these plaintexts instead in Step 2e and compute the c_i for the transcript later. Thus the output distributions of Game_1 and Game_2 are identical.

$\mathsf{Game}_2 \to \mathsf{Game}_3$: The fact that $\mathcal{E}$ is semantically secure implies that the outputs of these two games are $\nu(n)$-computationally indistinguishable for some negligible $\nu(n)$. To show this, first note that the indices Ronald sends in Step 4 are now all determined before Sarah computes any ciphertexts. Thus we can choose a random $x \in C$ not equal to *any* of these indices (assuming such an x exists) without using any public or secret keys.

Now let X be a semantic security challenge for pk_x. We can simulate the protocol by choosing secret keys sk_i for all $i \neq x$ and setting $c_x = X$. When X is an encryption of b we are in Game_2, and when X is an encryption of $1-b$ we are in Game_3. Thus any adversary that can distinguish the outputs of Game_2 and Game_3 can be used to break the semantic security of $\mathcal{E}$.

Finally, we show that an index x as above exists with overwhelming probability. Let C_β be defined as in the proof of Proposition 4.1, and recall that, with overwhelming probability, we have $\frac{7n}{8} \leq |C_\beta| \leq \frac{9n}{8}$. Since Ronald chooses $n/2$ indices in Step 4 that correspond to encryptions of b, with overwhelming probability there remain at least $3n/8$ indices in C_b from which to choose x.

$\mathsf{Game}_3 \to \mathsf{Game}_4$: First, the distribution of the β_i's does not change, and consequently the distribution of the c_i's does not change. Second, note that choosing the process of choosing the pairs (u_i, v_i) in Step 4 is identical in Game_3 and Game_4: since Ronald chooses pairs of indices with opposite plaintexts, flipping all of Ronald's bits does not affect these choices. Furthermore, since Sarah's computation of j in Step 5 depends only on the location of the indices u_i, v_i in the sets A, B, C, Sarah's computation of j using flipped bits is exactly the same as her computation of j in Game_3. Thus the output distributions of Game_3 and Game_4 are identical.

$\mathsf{Game}_4 \to \mathsf{Game}_5$: Let σ be a permutation of order 2 acting on the set $\{0, \ldots, 4n\}$, such that

$$\sigma(y) = z, \quad \sigma(x) = x, \quad \sigma(A \setminus \{y\}) = B.$$

Note that in particular, this means $\sigma(A) = \tilde{A}$, $\sigma(B) = \tilde{B}$, and $\sigma(C) = \tilde{C}$.

For all $i \in A$, we have $\beta_i = b$ and $\beta_{\sigma(i)} = 1 - b$, and for all $i \in B \cup \{z\}$ we have $\beta_i = 1 - b$ and $\beta_{\sigma(i)} = b$. The index x only appears in the output as the ciphertext $\tilde{c}_x$, so the output of Game_4 is distributed as if $\beta_x = 1 - b$; furthermore, we have $\tilde{\beta}_x = 1 - b$ in Game_5.

Next, observe that the random bits β_i for $i \in C$ are distributed such that inverting all of them does not change their distribution. Because $\beta_x = b$ and $\beta_z = 1 - b$, the distribution on the β_i for $i \in C \setminus \{x, z\}$ is also symmetric, so permuting just these indices does not change the distribution of the corresponding β_i; the same is true for flipping just these bits. We thus conclude that the following two distributions are identical:

- Run Game_4 to encrypt bit b with randomness r_S, producing transcript tr, and output $\sigma(tr)$ and $\sigma(r_s)$. (That is, we apply the permutation σ to all of the indices of the computed elements.)
- Run Game_5 to encrypt bit $1-b$ with randomness r_S, producing transcript $\widetilde{tr}$, and output $\widetilde{tr}$ and $\tilde{r}_S$.

Since the sets A, B, C are uniformly random disjoint subsets of $\{0, \ldots, 4n\}$ of cardinality $n+1$, n, and $2n$, respectively, applying a permutation to the indices cannot change the output distribution of Game_4. It follows that the output distributions of Game_4 and Game_5 are identical.

$\mathsf{Game}_5 \to \mathsf{Game}_6$: The fact that $\mathcal{E}$ is samplable implies that these two distributions are statistically indistinguishable.

Suppose we are given a secret key sk^*, a ciphertext X, and randomness R from one of the two distributions of (3.2). We can simulate the protocol by using sk^* as sk_z, X as c_z, and R as $\tilde{\alpha}_z$. When R is chosen from $\Delta_{\mathcal{R}}$ we are in Game_5, and when R is computed using $\mathsf{SampleRand}$ we are in Game_6. Thus any adversary that can distinguish these two games can distinguish the two distributions of (3.2), which contradicts the assumption that $\mathcal{E}$ is samplable.

$\mathsf{Game}_6 \to \mathsf{Game}_7$: By the same argument as above for $\mathsf{Game}_1 \to \mathsf{Game}_2$, the outputs of these two games are identical.

$\mathsf{Game}_7 \to \mathsf{Game}_8$: By the same argument as above for $\mathsf{Game}_0 \to \mathsf{Game}_1$, the fact that $\mathcal{E}$ is samplable implies that the outputs of these two games are statistically indistinguishable. □

5 Instantiations

5.1 Quadratic Residuosity

Our first example of a samplable encryption system is the Goldwasser-Micali bit encryption system [10], which is secure under the quadratic residuosity assumption.

For a positive integer N that is a product of two distinct odd primes, we define the set $\mathcal{J}(N) = \{x \in \mathbb{Z}_N^* : \left(\frac{x}{N}\right) = 1\}$ and let $\mathcal{Q}(N)$ be the subgroup of squares in $\mathbb{Z}_N^*$, which has index 2. The *quadratic residuosity assumption* states that when N is the product of two randomly chosen n-bit primes, the two distributions obtained by sampling uniformly at random from $\mathcal{Q}(N)$ and from $\mathcal{J}(N) \setminus \mathcal{Q}(N)$ are $\mu(n)$-computationally indistinguishable for negligible $\mu(n)$.

Construction 5.1.

- $\mathsf{KeyGen}(n)$: Compute $N = pq$, where p, q are n-bit primes. Choose a quadratic non-residue $g \in \mathcal{J}(N) \setminus \mathcal{Q}(N)$. The public key is $\mathsf{pk} = (N, g)$ and the secret key is $\mathsf{sk} = p$.
- $\mathsf{Enc}_{\mathsf{pk}}(b; r)$: Choose $r \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$ and output $c = g^b r^2 \pmod N$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Let $\mathsf{sk} = p$. If $\left(\frac{c}{p}\right) = 1$, output 0; otherwise output 1.
- $\mathsf{SampleRand}_{\mathsf{sk}}(c)$: If $\mathsf{Dec}_{\mathsf{sk}}(c) = 0$, output a random solution to $X^2 = c \pmod N$; otherwise output a random solution to $X^2 = c/g \pmod N$.

It is a standard result [10] that the encryption scheme is semantically secure under the quadratic residuosity assumption. We now show the samplable properties.

Proposition 5.2. *The public key encryption scheme of Construction 5.1 is samplable.*

Proof. In the notation of Definition 3.1, the set $\mathcal{R}$ is $\mathbb{Z}_N^*$, and the set $\mathcal{C}$ is $\mathcal{J}(N)$. The distribution $\Delta_{\mathcal{R}}$ is the uniform distribution on $\mathbb{Z}_N^*$. Since $\mathcal{J}(N) = (\mathbb{Z}_N^*)^2 \cup g \cdot (\mathbb{Z}_N^*)^2$, it follows that the distribution (3.1) is the uniform distribution on $\mathcal{J}(N)$.

For the second condition, we observe that for a given $c \in \mathcal{J}(N)$ the value $x = \mathsf{SampleRand}_{\mathsf{sk}}(c)$ is distributed uniformly amongst the four possible values of r such that $c = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}}(c), r)$. Since real randomness comes from the uniform distribution on $\mathbb{Z}_N^*$, it follows that the two distributions (3.2) are identical. □

5.2 Trapdoor Permutations

Our second samplable encryption system is the bit encryption scheme built from a generic trapdoor permutation (cf. [11, Construction 10.27]):

Construction 5.3.

- $\mathsf{KeyGen}(n)$: Let $f : \mathcal{R} \to \mathcal{R}$ be sampled from a family $\mathcal{F}$ of trapdoor permutations (using n as the security parameter) and let $g = f^{-1}$. Let $H : \mathcal{R} \to \{0,1\}$ be a hard-core predicate for f. The public key is $\mathsf{pk} = (f, H)$ and the secret key is $\mathsf{sk} = g$.
- $\mathsf{Enc}_{\mathsf{pk}}(b; r)$: Choose $r \xleftarrow{\mathrm{R}} \mathcal{R}$ and output $c = (f(r), H(r) \oplus b) \in \mathcal{R} \times \{0,1\}$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Write $c = (y, z) \in \mathcal{R} \times \{0,1\}$. Output $H(g(y)) \oplus z$.
- $\mathsf{SampleRand}_{\mathsf{sk}}(c)$: Write $c = (y, z)$ and output $g(y)$.

It is a standard result (see e.g. [11, Theorem 10.28]) that if $\mathcal{F}$ is a family of trapdoor permutations, then the encryption scheme is semantically secure. In particular, under the RSA assumption we can let f be the function $x \mapsto x^e \pmod N$ with the hard-core predicate $H(x) = \mathsf{lsb}(x)$ [1].

Proposition 5.4. *The public key encryption scheme of Construction 5.3 is samplable.*

Proof. Since r is sampled uniformly from $\mathcal{R}$ and f is a permutation, the first component of the ciphertext is uniformly distributed in $\mathcal{R}$. Furthermore, if b is a uniformly random bit, the second component of the ciphertext is random and independent of the first component. Thus the distribution (3.1) is the uniform distribution on $\mathcal{R} \times \{0,1\}$.

For the second condition, since f is a permutation the map from $\mathcal{R} \times \{0,1\}$ to itself given by $(r, b) \mapsto \mathsf{Enc}_{\mathsf{pk}}(b; r)$ is a bijection. Thus there is a unique value of the randomness r for any ciphertext (which is exactly what is recovered by $\mathsf{SampleRand}$), and the two distributions (3.2) are identical. □

6 Conclusion and Open Problems

We have presented a sender-deniable public key encryption scheme that has a single (interactive) protocol for encryption and has negligible detection probability. It is the

first construction that satisfies these strict requirements. The security of our construction is based on well established assumptions; we give one construction based on the hardness of deciding quadratic residuosity and one based on the existence of trapdoor permutations.

Receiver-deniable encryption can be obtained from sender-deniable encryption by a straightforward construction [3]; basically, the roles of sender and receiver are reversed, and the receiver encrypts a bit that is used by the sender as a one-time pad. It is an open problem to construct a bi-deniable encryption scheme with a single encryption algorithm and no trusted third party. (The recent work of O'Neill, Peikert, and Waters [12] achieves this goal for a weaker notion of deniability that allows two encryption algorithms.) Another open problem arising from our work is to remove the interaction from our encryption protocol.

Both instantiations of our scheme rely on the hardness of factoring. It is an open problem to construct deniable encryption schemes from other assumptions. A promising direction for this problem is cryptosystems based on the hardness of Learning With Errors (LWE), a lattice-related problem. In particular, the variant of Regev's LWE cryptosystem [13] presented by Gentry, Peikert, and Vaikuntanathan [9, §8] has the property that a trapdoor can be embedded in the secret key that allows the key holder to efficiently sample from the distribution of randomness used in the encryption. However, the scheme does not have a dense ciphertext space, so the scheme is not samplable according to our definition. So far, all methods we have looked at to modify the cryptosystem to overcome this difficulty either induce a non-uniform distribution on real ciphertexts, introduce non-negligible decryption error, or destroy the system's security.

Finally, the overhead of our construction is significant: for a security parameter n, one execution of our protocol requires transmission of $O(n)$ secret keys and ciphertexts, each of length n (not to mention the computational cost of generating $O(n)$ keys and ciphertexts); furthermore, to ensure correctness we must repeat the protocol n times. Several straightforward improvements are possible, since we optimized our presentation for clarity rather than efficiency. However, a significantly more efficient construction seems to require substantial new ideas.

Acknowledgments

The authors thank Dan Boneh, Chris Peikert, Brent Waters, and the anonymous referees for helpful discussions and/or feedback on earlier versions of this work.

References

1. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA and Rabin functions: Certain parts are as hard as the whole. SIAM J. Comput. 17(2), 194–209 (1988)
2. Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
3. Canetti, R., Dwork, C., Naor, M., Ostrovsky, R.: Deniable encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 90–104. Springer, Heidelberg (1997)
4. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC. pp. 639–648 (1996)

5. Chapman, M., Davida, G.: Plausible deniability using automated linguistic stegonagraphy. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 276–287. Springer, Heidelberg (2002)
6. Choi, S., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
7. Czeskis, A., Hilaire, D.J.S., Koscher, K., Gribble, S.D., Kohno, T., Schneier, B.: Defeating encrypted and deniable file systems: TrueCrypt v5.1a and the case of the tattling OS and applications. In: 3rd Usenix Workshop on Hot Topics in Security (2008)
8. Damgård, I., Nielsen, J.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
9. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) STOC, pp. 197–206. ACM, New York (2008)
10. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984); preliminary version in 14th Annual ACM Symposium on Theory of Computing (STOC)
11. Katz, J., Lindell, Y.: Introduction to modern cryptography. Chapman & Hall/CRC, Boca Raton (2008)
12. O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption (2010) (manuscript)
13. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM, New York (2005)
14. Ross, S.: A First Course in Probability, 5th edn. Prentice-Hall, Englewood Cliffs (1998)
15. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
16. Spitzer, F.: Principles of Random Walks, 2nd edn. Graduate Texts in Mathematics, vol. 34. Springer, Heidelberg (1976)
17. TrueCrypt: Free open-source disk encryption software, `http://www.truecrypt.org/`