Deniable Cloud Storage: Sharing Files via Public-key Deniability

Paolo Gasti
University of Genoa
Genoa, Italy
gasti@disi.unige.it

Giuseppe Ateniese
Johns Hopkins University
Baltimore, MD, USA
ateniese@cs.jhu.edu

Marina Blanton
University of Notre Dame
Notre Dame, IN, USA
mblanton@cse.nd.edu

ABSTRACT

Cloud computing provides users with ample computing resources, storage, and bandwidth to meet their computing needs, often at minimal cost. As such services become popular and available to a larger body of users, security mechanisms become an integral part of them. Conventional means for protecting data privacy, such as encryption, can protect communication and stored data from unauthorized access including the service provider itself. Such tools, however, are not sufficient against powerful adversaries who can force users into opening their encrypted content. In this work we introduce the concept of deniable cloud storage that guarantees privacy of data even when one's communication and storage can be opened by an adversary. We show that existing techniques and systems do not adequately solve this problem. We design the first sender-and-receiver deniable public-key encryption scheme that is both practical and is built from standard tools. Furthermore, we treat practical aspects of user collaboration and provide an implementation of a deniable shared file system, DenFS.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems

General Terms

Algorithms, Security

1. INTRODUCTION

The goal of cloud computing is to provide CPU, storage, network bandwidth and virtually unlimited scalability at low cost. In addition, cloud services provide full availability, i.e., users can access their data from any connected machine. Cloud users do not need to worry about backups or unexpected costs: if a component fails, it is the provider's responsibility to replace it and make the data available using replicas [18]. Armbrust et al. [2] identify three aspects which characterize cloud computing: 1) the illusion of infinite computing resources available on demand, and therefore the elimination of the need to plan ahead for provisioning; 2) the elimination of an up-front commitment by users; and 3) the ability to pay for use of computing resources on a short-term basis as needed. Cloud computing is said to maximize benefits of scale [2, 28].

Amazon's Simple Storage Service (S3) [22] is among the best-known storage providers, and is used by other service providers to store customers' data and by end users. A notable example of service provider relying on S3 is Dropbox [24], which provides seamless file synchronization with an on-line repository and across different computers, together with an easy interface for sharing documents with other users. Dropbox is a very powerful tool for collaboration: users do not suffer delays introduced by the network, since they always access a local copy; updates are sent and received in background and users are quickly notified when the content of a shared file changes.

With cloud storage services, encryption becomes crucial to protect data privacy, at least because the cloud provider has full access to users' data. It is recent news that Google provided the FBI all the documents of one of its users after receiving a search warrant. What is unusual is that that user was not notified of the warrant and was not aware of the search until he was arrested [29].

The goal of traditional encryption is to preserve privacy of data communication or storage in the presence of passive adversaries. Deniable encryption, on the other hand, was introduced by Canetti et al. [6] as a mechanism for maintaining data privacy in the presence of active adversaries who have the ability to coerce data senders or recipients into opening observed ciphertexts. That is, after observing a ciphertext, an adversary approaches the sender and asks her to reveal the random choices or keys used in generating the ciphertext, which expose the corresponding plaintext. A deniable encryption scheme then has a property that allows the sender to open the ciphertext in such a way as to reveal a different plaintext than the one originally used in producing the ciphertext. The type of deniable encryption resilient to sender coercion is called *sender-deniable* encryption, and *receiver-deniable* encryption achieves deniability when the ciphertext recipient is coerced into opening it. Then *sender-and-receiver deniable* schemes combine both of the above properties. We refer to the plaintext intended for the recipient as the *real* message and the plaintext that the coercer sees after opening the ciphertext as the *fake* message.

Deniable encryption can be divided into two types: *plan-ahead* and *ad-hoc* encryption. The former requires the sender to choose both the real and fake messages at the time of encryption, while the latter permits the fake message to be chosen at the time of coercion (from the entire message space). Currently, the only known constructions for ad-hoc deniable encryption correspond to the public-key setting and achieve sender-deniability.[1] Such constructions are inefficient as they require one (public-key) ciphertext per message

[1]One-time pad and the work of Ibrahim [14] are exceptions; see section 2 for details.

WPES'10, October 4, 2010, Chicago, Illinois, USA.

bit [6, 16, 13], or allow several message bits per ciphertext, but the ciphertext size is still exponential in the number of message bits transmitted [13]. Since our goal is to seek a practical solution, we concentrate on plan-ahead deniability. In particular, we are concerned with practical applications of deniable encryption and consider the following scenario as our target functionality: an adversary obtains access to a laptop computer, the owner of which keeps the stored data encrypted. Some of the encrypted data is shared, using a cloud service, with one or more collaborating users; file updates are exchanged over the Internet. The adversary, who may have intercepted the updates, may have injected new files and may have reconstructed multiple snapshots of the encrypted data, forces the owner to decrypt the content of the hard drive by revealing any key material and necessary auxiliary information.

The original formulation of deniable encryption assumes that an adversary captures only communication traffic. We, however, view as one of the most prominent uses of deniable encryption the case where encrypted contents are stored on a laptop or a desktop computer and on a remote server as well, and the adversary forces the owner of the machine to unlock its encrypted contents. This means that the only information that can stay hidden is a small secret, such as a decryption key, that the owner of the machine can hide from the coercer. Note that existing deniable encryption schemes do not necessarily work in this context since extra information must be stored in order to be able to open a ciphertext to a fake value. Once this additional information is provided to the coercer, he can use it to uncover the original message. We would like the owner to have the ability to open the data in two different ways through a usable mechanism, e.g., by entering a password that will unlock the key material and decrypt the data.

Considering that certain encryption algorithms are well-known, widely accepted and used in practice, it would be desirable to use such algorithms, if possible, instead of custom-built schemes to achieve wider adoption of deniable encryption. Since the conventional encryption schemes are not deniable, the existing deniable encryption schemes normally do not use standard tools. One of our goal, however, is to explore what deniability properties can be achieved with conventional encryption schemes.

Contributions: First, we construct a sender-deniable plan-ahead public key encryption scheme using RSA-OAEP [3] and the Damgård-Jurik generalization [9] of Paillier's encryption scheme [19] as building blocks. Then we extend it to provide non-interactive sender-and-receiver deniable plan-ahead public-key encryption. Finally, we show how to efficiently construct a public-key deniable encryption scheme using any IND-CPA encryption scheme as a black box. Our constructions have a high throughput (i.e., linear in the size of the ciphertext) for messages that can be deniably communicated. An essential component of this work is the design and implementation of DenFS, a distributed file system that allows a group of collaborators seeking deniability to view, edit, and exchange documents as if they were working on a standard file system. Finally, in the appendix we give extensions to our work that allow us to increase the bandwidth of the deniable channel and provide other efficiency improvements.

Deniability in practice. TrueCrypt [25] is a widely used disk encryption tool that provides deniability in the form of a deniable file system. TrueCrypt refers to a deniable file system as a *hidden volume*, which is created inside a non-deniable encrypted disk image, using the space marked as not allocated by the file system. To mount the hidden volume, a user must first provide the password to decrypt the non-deniable file system and then provide a second password to decrypt and mount the hidden volume. If the second password is not known, the unallocated space is indistinguishable from random data [26]. Since TrueCrypt always fills the unallocated space with random data, the existence of the hidden volume can be denied. Czeskis et al. examine in [8] the efficacy with which TrueCrypt v5.1a provides deniability. They found that Microsoft Windows and several common applications and utilities compromise the deniability of TrueCrypt hidden volumes. As an example, Windows keeps track of the serial number of each volume it mounts and also creates a link to recently opened documents. Both these features clearly compromise the deniability of a hidden volume. As with other deniable file system implementations, we assume that the OS and application are aware of such property of the file system and avoid to keep track of the content of the deniable files. This assumption allows us to consider the security of our proposal alone, alleviating the need to examine the whole system. From a practical point of view, however, we show that our deniable filesystem is immune to some of the attacks illustrated by Czeskis et al., while the impact of the remaining threats can be mitigated as discussed in section 4.

Assange and Weinmann introduced a deniable file system called Rubberhose [23], which transparently encrypts data on a storage device and allows users to hide that encrypted data. Rubberhose needs a dedicated hard disk and creates several partitions, encrypted with a different passphrase, in which blocks are not contiguous, but scattered in a pseudorandom manner across the drive.

Anderson et al. [1] proposed two deniable file systems schemes, which they call steganographic file systems (stegfs). The first initializes the disk with a large number of cover files which contain random data. Deniable files are stored as the XOR of subsets of the cover files, which are identified by a secret key. The main disadvantage of this file system is the high overhead associated with each read and write operation. With their second scheme, the disk is filled with random data and the deniable files are written at random positions; the likelihood of overwriting hidden data increases with the amount of information stored on the disk. The schemes of Anderson et al. are vulnerable against an adversary who can obtain multiple different copies of the encrypted content of the disk (snapshots) over a period of time.

In order to avoid the problem of data collisions in the scheme of Anderson et al., Pang et al. [20] proposed the use of a bitmap to track block allocation. Dummy data are added when the disk is formatted to provide deniability. Since dummy data cannot be reallocated without formatting the disk, an adversary who can access multiple snapshots of the disk can easily break the deniability of the scheme. This problem was addressed by Zhou et al. [31]. Their solution requires the use of a trusted agent to manage the dummy data. To perform this task, the trusted agent needs to know the password used to protect the deniable files.

Unfortunately, none of the available deniable file systems provide a solution for the scenario proposed earlier in this section, where an adversary obtains access to a laptop and also records communication. In particular, none of the existing schemes can be applied to the cloud environment. In this scenario we further assume that a malicious cloud provider has complete control over the user's data. Moreover, all these file systems require users to share a secret key, which may not be always possible and severely limits the flexibility of the file system, as discussed in section 4.

2. BACKGROUND AND DEFINITIONS

In this section we first review related work and then proceed with defining the model. Canetti et al. [6] introduced the notion of deniable encryption and gave several constructions for sender-deniable encryption. In particular, their work shows that public-key sender-deniable ad-hoc encryption can be built from translucent sets and trapdoor permutation, where each ciphertext encodes a single bit. It also reports that a shared-key sender-deniable plan-ahead encryption can be constructed by either concatenating encryptions of several messages under different keys (and thus increasing the ciphertext size) or sharing a key proportional in size to the message length. Additionally, [6] shows that sender-deniable encryption can be converted to receiver-deniable encryption (or vice versa) using an extra round of communication; namely, the receiver uses a sender-deniable scheme to transmit encryption of a random bit r to the sender, and the sender transmit $b \oplus r$ to the receiver in the clear, where bit b is the actual message to be transmitted. This interactive solution, however, is not applicable to the scenario of encrypted storage considered in this work.

Ibrahim [13] gives ad-hoc sender-deniable public-key encryption schemes based on the hardness of the quadratic residuosity problem for N, which is a product of two large primes. The first scheme allows transmission of one bit per ciphertext. The second scheme can transmit more than one message bits per ciphertext, but only a small number of such bits (ciphertext size is exponential in the message length).

In [14] Ibrahim gives an ad-hoc receiver-deniable public-key scheme, which has higher bandwidth than other ad-hoc deniable schemes. The scheme is based on RSA and 1-out-of-n oblivious transfer, and permits transmission of messages of $\log N - \delta$ bits long with ciphertexts of size $2 \log N$, where N is the RSA modulus and δ is a randomizing string (e.g., 128 bits long). This solution, however, requires usage of mediated RSA, where a user does not have full knowledge of her secret key, i.e., each private key is split between the owner of the key and a third party. This means that such a solution has a limited applicability and will not work in our setting. Additionally, this solution relies on trusted hardware.

A recent work of Klonowski et al. [16] extends the ad-hoc sender-deniable public-key scheme of Canetti et al. through a nested construction. The authors also show how hidden messages can be transmitted using standard ElGamal encryption, which achieves the functionality of plan-ahead receiver-deniable encryption. In particular, the sender and receiver share a secret S, and the sender also has access to the receiver's private key (associated with the public key the encryption scheme uses). This allows a ciphertext to encode, besides a fake message, an additional secret message of the same size. In fact, this provides a broad-band subliminal channel if the receiver is willing to trust the sender with her private key. Meng and Wang [17] extent the idea of Klonowski et al. [16] providing a receiver deniable encryption scheme based on BCP commitment scheme of Bresson et al. [5]. Their scheme does not require sender and receiver to exchange any pre-encryption information.

To summarize, existing public-key constructions are either flexible, but inefficient or are able to communicate several secret bits and use standard tools, but require the recipient to give up its private key. Furthermore, all suitable solutions are sender-deniable and therefore are not applicable to the problem of deniable storage. The motivation of this work thus comes from a usability perspective: can we design an efficient solution that does not require revealing the private key of the receiver? Can we achieve receiver-deniability without using mediated settings?

Following Klonowski et al., we investigate communication of hidden messages by means of an additional channel. That is, a ciphertext always encodes a fake message and can additionally contain a real hidden message. Furthermore, the assumption that the sender and the receiver share a secret appears powerful, and we investigate what can be achieved without this assumption.

The original definitions of deniable encryption in [6] are stated in terms of computational indistinguishability of views associated with real and fake messages. In particular, after encryption of real message m is transmitted, the so-called *faking algorithm* allows it to be opened to reveal a fake message m_f. The security requirement is such that, given any messages m_1 and m_2, the encryption of m_1 is computationally indistinguishable from the encryption of m_2. Additionally, the deniability property requires that the view of the adversary, that includes communication of the ciphertext c corresponding to the encryption of m, the random choices calculated by the *faking algorithm* and the opening to m_f of the ciphertext, is indistinguishable from communication of encryption of m_f, its opening and the random choices used to encrypt it. Sender (receiver) deniability means that the sender (resp., receiver) reveals her random choices/keys upon coercion. The definition of a (plan-ahead) deniable public-key encryption is strictly stronger than the standard definition of CPA-security, i.e., deniability also implies CPA-security.

In this work, we define deniable encryption as the ability to hide the presence of the real message: the adversary's view when transmitting a fake message with a hidden real message should be indistinguishable from the case when no hidden message is included.

DEFINITION 1. *A (plan-ahead) deniable public-key encryption scheme consists of the following four efficient algorithms:*

- Setup : *a probabilistic algorithm that, on input a security parameter 1^κ, outputs $(pk, sk) \in \mathcal{K}_{pub} \times \mathcal{K}_{pri}$. The private key space $\mathcal{K}_{pri}$ consists of two partitions, $\mathcal{K}^+_{pri}$ and $\mathcal{K}^-_{pri}$*
- Enc : *a probabilistic algorithm that, on input pk and messages m_f and m, outputs a ciphertext $C(m_f; m)$. We explicitly denote by r the random choices made by this algorithm.*
- Dec : *a deterministic algorithm that, on input (pk, sk) and ciphertext C, outputs message m_f encoded in the ciphertext.* Dec *also outputs m iff $sk \in \mathcal{K}^+_{pri}$*
- Open : *a deterministic algorithm that on input a ciphertext C and public key pk, outputs private information PI that opens the encryption to message m_f.*

In the above, what the opening algorithm outputs depends on what party is coerced into opening the encryption. If it is the sender, PI can consist of random choices r used in ciphertext generation. For this reason Enc stores all its random choices in a system-wide table accessible by Open. If the receiver is opening the ciphertext, PI can consist of the private key sk. Finally, if both of them are to open the ciphertext, the information PI available to the adversary will consist of the union of the sender's and receiver's information. When Open is invoked by the sender, we denote it by Open_S, and when it is invoked by the receiver, it is denoted by Open_R.

Let $\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_S, \mathsf{Open}_R)$ be a deniable public-key encryption scheme, where $(pk, sk) \leftarrow \mathsf{Setup}(1^\kappa)$. The security properties required by $\mathcal{D}$ are inspired by the security definition in [6], adapted to our setting. However, instead of using definitions based on computational indistinguishability of the view of the adversary, we provide a rigorous definition of (public-key) deniability in terms of interactive experiments.

When discussing sender deniability, we assume that a coercer can guess who the intended recipient of the transmission is (e.g., if the message is communicated via email, recipient identity can be deduced from the transmission). Thus, the sender has to use the true recipient's public key during the opening phase.

We define the security requirement of a (plan-ahead) deniable public-key encryption scheme by adapting the standard definition of IND-CPA to our setup. Let $\mathcal{E} = (Gen, E, D)$ denote a public-key encryption scheme where Gen, E and D are polynomial-time algorithms defined in the standard way (in particular, Gen and E are probabilistic algorithms while D is deterministic). Given a public key pk generated by Gen, we will denote by $\mathcal{M}_{pk}$ and $\mathcal{C}_{pk}$ the message and ciphertext spaces defined by pk. In addition, for simplicity and without loss of generality, we assume that messages in $\mathcal{M}_{pk}$ and ciphertexts in $\mathcal{C}_{pk}$, respectively, have all the same size. We next state the properties of an encryption scheme we rely on. Consider the IND-CPA experiment defined as follows:

Experiment $\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa)$

1. Run $(pk, sk) \leftarrow Gen(1^\kappa)$.
2. Adversary $\mathcal{A}$ is given pk and eventually outputs two messages $m_0, m_1 \in \mathcal{M}_{pk}$ of its choice.
3. A random bit b is drawn and the encryption $E_{pk}(m_b)$ is returned to $\mathcal{A}$.
4. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 iff $b = b'$.

DEFINITION 2 (IND-CPA SECURITY). *An encryption scheme $\mathcal{E} = (Gen, E, D)$ has indistinguishable encryptions under chosen plaintext attack if there exists a negligible function* negl *such that for any probabilistic polynomial time $\mathcal{A}$, $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.*

Recall that in a plan-ahead deniable encryption scheme $\mathcal{D}$ a ciphertext can encrypt a fake and a real messages or just the fake message. Let $C(m_f; m)$ denote a ciphertext that encrypts both of them, and $C(m_f; -)$ denote a ciphertext that encrypts only m_f. The above IND-CPA experiment for $\mathcal{D}$ will then mean that in step 2 $\mathcal{A}$ outputs two pairs of messages $(m_{f,0}, m_0), (m_{f,1}, m_1) \in \mathcal{M}_{pk}$ and obtains $\mathsf{Enc}(m_{f,b}, m_b, pk) = C(m_{f,b}; m_b)$ in step 3. Note that one of messages m_0 and m_1 or both of them can be $-$.

We also provide the definition of sampleable ciphertext space, which will be used to construct our schemes:

DEFINITION 3 (SAMPLEABLE). *An encryption scheme $\mathcal{E} = (Gen, E, D)$ has sampleable ciphertext space if there exists a polynomial time (in κ) algorithm such that, on input pk (produced by $Gen(1^\kappa)$), can choose an element of $\mathcal{C}_{pk}$ with the same distribution the algorithm $E_{pk}(\cdot)$ produces on a fixed message m.*

Note that when the encryption scheme $\mathcal{E}$ is IND-CPA-secure, the above definition will imply that the distribution of ciphertexts $E_{pk}(m)$ is computationally indistinguishable from the distribution of ciphertexts $E_{pk}(m')$ for $m \neq m'$. Then we can sample the ciphertext space by creating $E_{pk}(m)$, and the view will be indistinguishable from the distribution of encryptions of other messages.

Unlike prior literature, we specify deniability using interactive experiments to more precisely capture the adversary's abilities. We define the experiment for sender-deniability of deniable encryption scheme $\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_S)$ as follows:

Experiment $\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa)$

1. Set a system-wide table T initially empty.
2. Run $(pk, sk) \leftarrow \mathsf{Setup}(1^\kappa)$.
3. Adversary $\mathcal{A}$ is given pk and oracle access to $\mathsf{Open}_S(\cdot, pk)$ and to $\mathsf{Enc}(\cdot, \cdot, pk)$. Eventually $\mathcal{A}$ outputs a message pair (m_f, m), with $|m_f| = |m|$.
4. A random bit $b \xleftarrow{R} \{0, 1\}$ is drawn; if $b = 0$, $\mathcal{A}$ is given $C(m_f; m)$, and if $b = 1$, $\mathcal{A}$ is given $C(m_f; -)$.
5. $\mathcal{A}$ continues to have access to $\mathsf{Open}_S(\cdot, pk)$ and $\mathsf{Enc}(\cdot, \cdot, pk)$.
6. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 iff $b = b'$.

$\mathsf{Enc}(m_f, m, pk)$:

1. Compute $C(m_f; m)$.
2. Store the tuple $(m_f, m, C(m_f; m), r)$ in T, where r denotes the set of the random choices used during the encryption.
3. Output $C(m_f; m)$.

$\mathsf{Open}_S(C, pk)$:

1. If C is in T return the tuple (m_f, m, C, r).
2. If C is the challenge ciphertext, return the random choices derived from r to open C as an encryption of $(m_f; -)$, regardless of whether $C = C(m_f; m)$ or $C = C(m_f; -)$.
3. Otherwise return $\perp$.

Note that we allow the adversary to open any ciphertext including the challenge ciphertext (produced on a message of its choice).

A more detailed explanation of this definition is in order. The adversary $\mathcal{A}$ is given access to an opening oracle and an encryption oracle under the public key pk. Note that even though the definition is for public-key encryption schemes we still provide access to an encryption oracle to model the ability of the coercer to choose arbitrary messages (m_f, m), have them encrypted by the sender, and have the sender reveal the random choices used in the encryption.

DEFINITION 4. *A public-key encryption scheme $\mathcal{D}$ is* sender-deniable *if there exists a negligible function* negl *such that for any probabilistic polynomial time (PPT) $\mathcal{A}$, $\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.*

The experiment for receiver-deniability of a deniable encryption scheme $\mathcal{D} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Open}_R)$ is defined as follows:

Experiment $\mathsf{RDen}_{\mathcal{A},\mathcal{D}}(\kappa)$

1. Run $(pk, sk) \leftarrow \mathsf{Setup}(1^\kappa)$.
2. $\mathcal{A}$ is given pk and oracle access to $\mathsf{Open}_R(\cdot, pk)$ and eventually outputs a message pair (m_f, m), with $|m_f| = |m|$.
3. A random bit $b \xleftarrow{R} \{0, 1\}$ is drawn; if $b = 0$, $\mathcal{A}$ is given $C(m_f; m)$, and if $b = 1$, $\mathcal{A}$ is given $C(m_f; -)$.
4. Adversary $\mathcal{A}$ continues to have access to $\mathsf{Open}_R(\cdot, pk)$.
5. $\mathcal{A}$ outputs bit b', and the experiment outputs 1 iff $b = b'$.

By allowing the adversary to have oracle access to Open_R, we make it very powerful. In particular, for existing receiver-deniable constructions, calling Open_R once will give the adversary access to the private decryption key sk. In practice, once a receiver is coerced and reveals the private key, the key pair should no longer be used and a new key will be generated. We, however, grant the adversary this level of power to show that, even after revealing the private key, the real message m remains hidden.

DEFINITION 5. *A public-key encryption scheme $\mathcal{D}$ is* receiver-deniable *if there exists a negligible function* negl *such that for any PPT $\mathcal{A}$, $\Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] \leq 1/2 + \mathsf{negl}(\kappa)$.*

Additionally, we say that a public-key encryption scheme $\mathcal{D}$ is *sender-and-receiver* deniable if it satisfies the requirements for both sender deniability and receiver deniability.

We emphasize that, in this paper, we are not considering adversaries with access to a decryption oracle.

3. DENIABLE ENCRYPTION FROM STANDARD TOOLS

3.1 A Sender-deniable Solution

First we propose an efficient sender-deniable encryption scheme, then we extend it to provide sender-and-receiver deniability. We denote with $\bar{E}(N, e, m)$ and $\bar{D}(N, d, c)$ the encryption of message m

using RSA-OAEP [3] with public key (N, e) and the decryption of the ciphertext c with the private key (N, d), i.e., $\bar{E}(N, e, m) = (\text{OAEP}(m))^e \bmod N$ and $\bar{D}(N, d, c) = \text{OAEP}^{-1}(c^d \bmod N)$. RSA-OAEP is IND-CCA-secure (and therefore it is also IND-CPA-secure). When the OAEP padding is removed, the result is either the plaintext m if the ciphertext is valid or $\perp$ otherwise. We base our solution on an efficient generalization of Paillier's probabilistic public key system [19] introduced by Damgård-Jurik in [9].

Setup: On input a security parameter 1^κ, choose an RSA modulus $N = pq$ of length κ bits, with $p \equiv q \equiv 3 \pmod 4$ and $\gcd(p-1, q-1) = 2$. Set $\lambda = (p-1)(q-1)/2$, i.e., λ is the least common multiple of $p-1$ and $q-1$. Then choose a value e such that $\gcd(e, \lambda) = 1$ and calculate d such that $e \cdot d \equiv 1 \bmod \lambda$. The public key is $pk = (N, e)$ and the secret key is $sk = (\lambda, d, p, q)$.

Enc: Given a public key $pk = (N, e)$, the sender forms a ciphertext as follows, depending on whether deniability is used or not: To deniably transmit a message m along with an innocent-looking fake message $m_f < N^s$, the sender computes

$$\mathsf{Enc}(pk, m_f, m) = ((1+N)^{m_f}) \cdot (\bar{E}(N, e, m))^{N^s} \bmod N^{s+1}$$

To transmit a message m_f without the ability to deny it, the sender picks a random r from $\mathbb{Z}_N^*$ and forms the encryption as

$$\mathsf{Enc}(pk, m_f, -) = ((1+N)^{m_f}) \cdot r^{N^s} \bmod N^{s+1}$$

Dec: Given a ciphertext c, the recipient with the public key $pk = (N, e)$ and the private key $sk = (\lambda, d, p, q)$ proceeds as follows:

1. Calculate $v = c^\lambda \bmod N^{s+1}$, i.e., $(1+N)^{m'_f \lambda} \bmod N^{s+1}$.
2. Use the technique described in [9] to compute $m_f \lambda \bmod N^s$ from v and multiply it by $\lambda^{-1} \bmod N^s$ to obtain m_f.
3. Using p and q, calculate d' such that $d' \cdot N^s \equiv 1 \bmod \lambda$.
4. Compute $c' = c \bmod N$. If $c = (1+N)^a \cdot b^{N^s} \bmod N^{s+1}$, then $c \bmod N = (1+N)^a \cdot b^{N^s} \bmod N = b^{N^s} \bmod N$.
5. Compute $m' = \bar{D}(N, d, c'^{d'} \bmod N)$. If $m' \neq \perp$, set $m = m'$, else set $m = -$.
6. Return (m_f, m).

Open_S: Suppose a coercer obtains a ciphertext c generated by the sender and requests its opening. Given the public key $pk = (N, e)$, the sender reveals m_f. If the ciphertext was encrypted as $\mathsf{Enc}(pk, m_f, -)$ then the sender outputs r as the random choices. Analogously, if the ciphertext was obtained as $\mathsf{Enc}(pk, m_f, m)$ then the sender simply outputs $\bar{E}(N, e, m) \bmod N$ as the random choices.

THEOREM 1. *Assuming that both the scheme of Damgård-Jurik and RSA-OAEP are* IND-CPA*-secure encryption schemes, the scheme above is sender-deniable.*

PROOF. In order to prove that our scheme is sender-deniable, we need to show that an adversary $\mathcal{A}$ which separately plays the IND-CPA experiment (for the security property) and the *sender deniability* experiment (for the deniability property) can have only negligible advantage in having any of the experiments to output 1, i.e. it has a negligible advantage in wining any of the experiments.

Security We show that there is no adversary with non-negligible advantage in breaking the IND-CPA security of our scheme through a sequence of experiments.

Experiment 0 This experiment is the standard IND-CPA experiment, i.e., the adversary $\mathcal{A}$ receives a public key $pk = (N, e)$ and outputs two messages $(m_{0,f}, m_0)$ and $(m_{1,f}, m_1)$. The challenger picks a random bit b and sends the encryption of $(m_{b,f}, m_b)$ to $\mathcal{A}$. $\mathcal{A}$ eventually outputs a bit b'. By definition we have that

$$\Pr[\mathsf{win}_0] = \Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{D}}(\kappa) = 1]$$

Experiment 1 In this experiment we modify the way the challenger constructs the challenge ciphertext. In particular, it constructs the challenge ciphertext as $C = ((1+N)^{m_f}) \cdot (r)^{N^s} \bmod N^{s+1}$ where $r \leftarrow \mathbb{Z}_N^*$. The adversary can only distinguish Experiment 1 from Experiment 0 with negligible probability, since in the random oracle model the ciphertexts of RSA-OAEP are distributed as random elements in $\mathbb{Z}_N^*$ for a PPT adversary. Therefore there exists a negligible function negl_0 such that

$$|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(\kappa)$$

Experiment 2 In this experiment we further modify the way the challenger constructs the ciphertext. Here the challenge ciphertext C is chosen as the encryption of a random element in $\mathcal{M}_{pk}$. Since the scheme of Damgård-Jurik is IND-CPA-secure, $\mathcal{A}$ has only negligible advantage in noticing that it is playing Experiment 2. Therefore we have that

$$|\Pr[\mathsf{win}_1] - \Pr[\mathsf{win}_2]| \leq \mathsf{negl}_1(\kappa)$$

for some negligible function negl_1. Combining the probabilities in all Experiments, we have that

$$\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] = \Pr[\mathsf{win}_2] + \mathsf{negl}(\kappa)$$

for some negligible function negl. To conclude the proof, we show that the probability $\Pr[\mathsf{win}_2]$ is exactly $1/2$. This is because the challenge ciphertext in Experiment 2 is chosen independently from bit b, therefore $\mathcal{A}$ can only guess b with probability $1/2$. For this reason the advantage of $\mathcal{A}$ into breaking the sender-deniable scheme must be negligible.

Deniability We show that there is no adversary with non-negligible advantage in breaking the sender-deniability of our scheme through a sequence of experiments.

Experiment 0 This experiment is the standard SDen experiment, i.e., adversary $\mathcal{A}$ receives a public key $pk = (N, e)$ and answers to its Enc and Open_S queries, and then outputs a message (m_f, m). The challenger picks a random bit b and sends the challenge ciphertext either as $C(m_f, m)$ if $b = 0$ or $C(m_f, -)$ if $b = 1$ to $\mathcal{A}$. $\mathcal{A}$ eventually outputs its choice for b'. By definition we have that

$$\Pr[\mathsf{win}_0] = \Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1]$$

Experiment 1 In this experiment we modify the way the challenger constructs the challenge ciphertext. In particular, it constructs the challenge ciphertext as $c = ((1+N)^{m_f}) \cdot (r)^{N^s} \bmod N^{s+1}$ where $r \leftarrow \mathbb{Z}_N^*$. The adversary can only distinguish Experiment 1 from Experiment 0 with negligible probability, since in the random oracle model the ciphertexts of RSA-OAEP are distributed as random elements in $\mathbb{Z}_N^*$ for a PPT adversary. Note that the challenger can still respond to the Enc and Open queries properly. Therefore there exists a negligible function negl_0 such that

$$|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(\kappa)$$

Combining the probabilities in Experiments 0 and 1, we have that

$$\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] = \Pr[\mathsf{win}_1] + \mathsf{negl}(\kappa)$$

for some negligible function negl. To conclude the proof, we show that the probability $\Pr[\mathsf{win}_1]$ is exactly $1/2$. This is because the challenge ciphertext in Experiment 1 is constructed independently from the bit b, so $\mathcal{A}$ can only guess b with probability $1/2$. For this reason the advantage of $\mathcal{A}$ into breaking the sender-deniable scheme must be negligible. □

Note that by fixing the value e for all keys (e.g., $e = 3$) we can remove e from the public key and set $pk = (N)$, i.e., a standard

public key for the scheme of Damgård-Jurik [9]. In this case the coercer has no way to distinguish our deniable scheme from the standard non-deniable scheme of Damgård-Jurik and the claim made by the sender that an observed ciphertext is the sole encryption of m_f is hard to contradict.

This scheme does not provide receiver deniability. Upon coercion, the receiver would have to surrender the factorization of N, which allows the coercer to decrypt the deniable message m. Receiver deniability can be obtained using the technique described by Canetti et al. in [6]; however, this would make the scheme interactive. To achieve receiver deniability, we construct another non-interactive encryption scheme which is detailed next.

3.2 Sender-and-receiver Deniability

The idea behind this scheme is to use two ciphertexts of the scheme of Damgård-Jurik to transport an ElGamal encryption. Even if the receiver surrenders the factorization of N, the coercer still needs another secret value to decrypt the ElGamal encryption. In order to achieve receiver deniability, the receiver should claim that it does not know such secret information. To achieve this result, we use a well known generalization of ElGamal based on quadratic residues modulo a composite (see, e.g., [15]). For simplicity, we will also employ a public redundancy function $\mathsf{red}(\cdot)$ in conjunction with the ElGamal encryption. Such function maps elements from a generic message space to a redundancy space which is a subset of $\mathbb{QR}(N)$ (the set of quadratic residues modulo N). Both $\mathsf{red}(\cdot)$ and $\mathsf{red}^{-1}(\cdot)$ are efficiently computable. Moreover, a random quadratic residue is in the redundancy space with negligible probability. In this way it is trivial for a receiver which knows how to decrypt the ElGamal ciphertext to distinguish the encryption of (m'_f, m''_f, m) from the encryption of $(m'_f, m''_f, -)$.

Setup: On input a security parameter 1^κ, choose an RSA modulus $N = pq$ of length κ where p and q are safe primes, i.e., $p = 2p' + 1$ and $q = 2q' + 1$ for primes p', q'. Let $\lambda = 2p'q'$ and g be a random generator of $\mathbb{QR}(N)$. Either choose $x \xleftarrow{R} \mathbb{Z}_{p'q'}$ and set $h = g^x$ or set $h \xleftarrow{R} \mathbb{QR}(N)$. The public key is $pk = (N, g, h)$; the secret key is either $sk = (\lambda, x)$ or $sk = (\lambda, \perp)$ depending on how h was chosen.

The public key pk is disseminated. If such keys get registered or certified, the certifying authority will request a proof of knowledge of the private key corresponding to λ, but not to x.

Enc: Given a public key $pk = (N, g, h)$ and a value $r \xleftarrow{R} \mathbb{Z}_{N^2}$, the sender forms a ciphertext as follows, depending on whether deniability is used or not: To deniably transmit a message m along with two innocent-looking fake messages m'_f, m''_f, the sender computes

$$\mathsf{Enc}(pk, m'_f, m''_f, m) = (c_1, c_2) = \\ ((1+N)^{m'_f}(\bar{c}_1)^{N^s},\ (1+N)^{m''_f}(\bar{c}_2)^{N^s}) \pmod{N^{s+1}}$$

where $\bar{c}_1 = g^r \bmod N$ and $\bar{c}_2 = \mathsf{red}(m) \cdot h^r \bmod N$. To transmit two messages m'_f, m''_f without the ability to deny them, the sender chooses two random values r_1, r_2 from $\mathbb{QR}(N)$ and sets $\bar{c}_1 = r_1$ and $\bar{c}_2 = r_2$ in the equation above.

Dec: Given a ciphertext (c_1, c_2), the recipient with the public key $pk = (N, g, h)$ and a private key $sk = (\lambda, x)$ proceeds as follows:

1. Calculate $v = c_1^\lambda \bmod N^{s+1} = (1+N)^{m'_f \lambda} \bmod N^{s+1}$ and use the technique described in [9] to compute $m'_f \lambda \bmod N^s$. The message m'_f is calculated by multiplying the result by λ^{-1}.
2. Proceed analogously to calculate m''_f.
3. If $sk = (\lambda, x)$, compute $m' = (c_2 \bmod N) \cdot (c_1 \bmod N)^{-x} \bmod N$. If m' is in the redundancy space, i.e., the co-domain of the function $\mathsf{red}(\cdot)$, set $m = \mathsf{red}^{-1}(m')$, otherwise set $m = -$.
4. Return (m', m'', m).

Open_S: Suppose a coercer obtains a ciphertext (c_1, c_2) generated by the sender and requests its opening. The sender exposes messages m'_f and m''_f. Then it reveals the associated random choices by claiming that $\bar{c}_1$ and $\bar{c}_2$ are random values uniformly chosen from $\mathbb{QR}(N)$.

Open_R: Suppose a coercer obtains a ciphertext (c_1, c_2) generated by the sender. Given the public key $pk = (N, g, h)$ and the private key $sk = (\lambda, x)$, the receiver opens its private key as $sk = (\lambda, \perp)$ such that λ matches N. Basically the receiver claims that it does not have knowledge of a value x such that the $h = g^x$.

Note that Open_R reveals the factorization of N to the adversary.

THEOREM 2. *Assuming that the scheme of Damgård-Jurik is an* IND-CPA*-secure encryption scheme and ElGamal over $\mathbb{QR}(N)$ is an* IND-CPA*-secure encryption scheme given the factorization of N, our scheme is sender-deniable.*

PROOF. In order to prove that our scheme is sender-deniable, we need to show that an adversary $\mathcal{A}$ which separately plays the IND-CPA experiment (for the security property) and the *sender deniability* experiment (for the deniability property) can win any of them with only negligible advantage.

Security We show that there is no adversary with non-negligible advantage in breaking the IND-CPA security of our sender-and-receiver deniable scheme through a sequence of experiments.

Experiment 0 This experiment is the standard IND-CPA experiment, i.e., the adversary $\mathcal{A}$ receives a public key $pk = (N, g, h)$ and outputs two messages $(m'_{0,f}, m''_{0,f}, m_0)$ and $(m'_{1,f}, m''_{1,f}, m_1)$. The challenger picks a random bit and sends the challenge ciphertext corresponding to the encryption of $(m'_{b,f}, m''_{b,f}, m_b)$ to $\mathcal{A}$. $\mathcal{A}$ eventually outputs a bit b'. By definition we have that

$$\Pr[\mathsf{win}_0] = \Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{D}}(\kappa) = 1]$$

Experiment 1 In this experiment we modify the way the challenger constructs the challenge ciphertext. In particular, it constructs the challenge ciphertext as

$$C = (((1+N)^{m'_f}) \cdot (r_1)^{N^s}, ((1+N)^{m''_f}) \cdot (r_2)^{N^s}) \pmod{N^{s+1}}$$

where both r_1 and r_2 are random elements from $\mathbb{QR}(N)$. The adversary can only distinguish Experiment 1 from Experiment 0 with negligible probability, since the elements r_1 and r_2 are distributed properly for any PPT adversary. Therefore there exists a negligible function negl_0 such that

$$|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(\kappa)$$

Experiment 2 In this experiment we further modify the way the challenger constructs the ciphertext. Here the challenge ciphertext is constructed as $C = (c_1, c_2)$ where c_1 and c_2 are chosen as the encryption of two random element in $\mathcal{M}_{pk}$. Since the scheme of Damgård-Jurik is IND-CPA-secure, $\mathcal{A}$ has only negligible advantage in noticing that it is playing Experiment 2. Thus, we have that

$$|\Pr[\mathsf{win}_1] - \Pr[\mathsf{win}_2]| \leq \mathsf{negl}_1(\kappa)$$

for some negligible function negl_1. Combining the probabilities in all Experiments, we have that

$$\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] = \Pr[\mathsf{win}_2] + \mathsf{negl}(\kappa)$$

for some negligible function negl. To conclude the proof, we show that the probability $\Pr[\mathsf{win}_2]$ is exactly $1/2$. This is because the

challenge ciphertext in Experiment 2 is chosen independently from bit b, therefore $\mathcal{A}$ can only guess b with probability $1/2$. For this reason the advantage of $\mathcal{A}$ into breaking the sender-deniable scheme must be negligible.

Deniability We show that there is no adversary with non-negligible advantage in breaking the sender-deniability of our scheme through a sequence of experiments.

Experiment 0 This experiment is the standard SDen experiment, i.e., $\mathcal{A}$ receives a public key $pk = (N, g, h)$ and answers to its Enc and Open_S queries according to the experiment. $\mathcal{A}$ outputs a triplet $(m'_{0,f}, m''_{0,f}, m_0)$ in $\mathcal{M}_{pk}$. The challenger picks a random bit b and sends to $\mathcal{A}$ the challenge ciphertext corresponding to the encryption of $(m'_{b,f}, m''_{b,f}, m)$ if $b = 0$ and $(m'_{b,f}, m''_{b,f}, -)$ otherwise. $\mathcal{A}$ eventually outputs a bit b'. By definition we have that

$$\Pr[\mathsf{win}_0] = \Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1]$$

Experiment 1 In this experiment we modify the way the challenger constructs the challenge ciphertext. In particular, it constructs the challenge ciphertext as

$$C = (((1+N)^{m'_f})\cdot(r_1)^{N^s}, ((1+N)^{m''_f})\cdot(r_2)^{N^s}) \pmod{N^{s+1}}$$

where both r_1 and r_2 are random elements from $\mathbb{QR}(N)$. The adversary can only distinguish Experiment 1 from Experiment 0 with negligible probability, since the elements r_1 and r_2 are distributed properly for any PPT adversary. Note that the challenger can still respond to the Enc and Open queries properly. Therefore there exists a negligible function negl_0 such that

$$|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(\kappa)$$

Combining the probabilities in Experiments 0 and 1, we have that

$$\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{D}}(\kappa) = 1] = \Pr[\mathsf{win}_1] + \mathsf{negl}(\kappa)$$

for some negligible function negl. To conclude the proof, we show that the probability $\Pr[\mathsf{win}_1]$ is exactly $1/2$. This is because the challenge ciphertext in Experiment 1 is constructed independently from messages b, so $\mathcal{A}$ can only guess b with probability $1/2$. For this reason the advantage of $\mathcal{A}$ into breaking the sender-deniable scheme must be negligible. □

THEOREM 3. *Assuming that the scheme of Damgård-Jurik is an* $\mathsf{IND\text{-}CPA}$*-secure encryption scheme and ElGamal over* $\mathbb{QR}(N)$ *is an* $\mathsf{IND\text{-}CPA}$*-secure encryption scheme given the factorization of* N*, our scheme is receiver-deniable.*

PROOF. Since we already proved that the security property holds for our scheme, in order to prove that it is receiver-deniable, we need to show that an adversary $\mathcal{A}$ can win the RDen experiment only with negligible advantage. To prove it, we show that if adversary $\mathcal{A}$ wins the *receiver deniability* experiment with probability non-negligible larger than $1/2$, then we can build an efficient algorithm $\mathcal{B}$ that wins in the $\mathsf{IND\text{-}CPA}$ experiment against ElGamal over $\mathbb{QR}(N)$ with non-negligible advantage. $\mathcal{B}$ engages in the $\mathsf{IND\text{-}CPA}$ experiment with the challenger, obtains the public key $\overline{pk} = (g, N = pq, p, q, h = g^x)$ and sets $pk = (N, g, h)$, $\lambda = (p-1)(q-1)/2$. Then $\mathcal{B}$ uses pk to setup the *receiver deniability* experiment for $\mathcal{A}$. $\mathcal{B}$ responds to Open_R queries from $\mathcal{A}$ with $sk = (\lambda, \perp)$. $\mathcal{A}$ eventually outputs its choice for (m'_f, m''_f, m). $\mathcal{B}$ outputs m_0, m_1 to the challenger where $m_0 = \mathsf{red}(m)$, and $m_1 = R$ is a random message from the message space such that $|R| = |\mathsf{red}(m)|$. The challenger picks a random bit b and returns $(\bar{c}_1, \bar{c}_2) = \mathrm{ElGamal}_{\overline{pk}}(m_b)$. $\mathcal{B}$ builds the challenge ciphertext for $\mathcal{A}$ as $\overline{C} = ((1+N)^{m'_f}(\bar{c}_1)^{N^s}, (1+N)^{m''_f}(\bar{c}_2)^{N^s})$. $\mathcal{B}$ continues to respond to $\mathsf{Open}_R(C, pk)$ queries with $(\lambda, \perp)$. Eventually $\mathcal{A}$ outputs b', and $\mathcal{B}$ outputs the same guess to the challenger.

It is not hard to see that $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{B},\mathrm{ElGamal}}(\kappa) = 1] = \Pr[\mathsf{RDen}_{\mathcal{A},\Gamma}(\kappa) = 1]$. That is, if $b = 0$, $\mathcal{B}$ receives the encryption of m and $\mathcal{A}$ receives the encryption of (m'_f, m''_f, m), in which case $\mathcal{B}$ answers the challenge correctly if and only if $\mathcal{A}$ answers its challenge correctly. If $b = 1$, $\mathcal{B}$ receives the encryption of R and $\mathcal{A}$ receives the encryption of (m'_f, m''_f, R), which is interpreted as $(m'_f, m''_f, -)$ since R is a random message. Therefore $\mathcal{B}$ guesses correctly if and only if $\mathcal{A}$ guesses correctly. Since $\mathcal{B}$ cannot guess correctly non-negligibly more than half of the times, $\delta \leq \mathsf{negl}(\kappa)$ for some negligible function negl. $\mathcal{A}$ cannot detect the simulation for the same reasons as in theorem 2. □

3.3 Deniability from any Semantically Secure Encryption Scheme

In this section we show that any $\mathsf{IND\text{-}CPA}$-secure encryption scheme can be used to construct a sender-and-receiver deniable scheme. The main idea behind this solution is simple: a receiver's public key consists of two parts, where the first part is a public key and the second part could be another public key or a randomly chosen string. To achieve deniability, the second half of the key is claimed to be random. Let $\mathcal{E} = (Gen, E, D)$ denote a $\mathsf{IND\text{-}CPA}$-secure encryption scheme. Let us also denote the domain of all possible public keys output by $Gen(1^\kappa)$ as $\mathcal{PK}(\kappa)$. The redundancy function $\mathsf{red}(\cdot)$ now maps elements from a generic message space to a subset of the message space of $\mathcal{E}$.

Setup: On input a security parameter 1^κ, a user generates a key pair $(pk_1, sk_1) \leftarrow Gen(1^\kappa)$ and also either generates a second key pair $(pk_2, sk_2) \leftarrow Gen(1^\kappa)$ or chooses a random value from $\mathcal{PK}(\kappa)$. The public key of the user is then $pk = (k_1, k_2)$, where $k_1 = pk_1$ and k_2 is either pk_2 or the random value. The user's corresponding private key is $sk = (s_1, s_2)$, where $s_1 = sk_1$ and s_2 is either sk_2 or the empty string $\perp$. As in the previous scheme, the public key $pk = (k_1, k_2)$ is disseminated and, upon request, only a proof of knowledge of the private key corresponding to k_1 is provided.

Enc: Given a public key $pk = (k_1, k_2)$, the sender forms a ciphertext as follows, depending on whether deniability is used or not. To deniably transmit a message m along with an innocent fake message m_f, the sender forms the encryption as

$$\mathsf{Enc}(pk, m_f, m) = (c_1, c_2) = (E_{k_1}(m_f), E_{k_2}(\mathsf{red}(m))).$$

To transmit a message m without the ability to deny it, the sender forms the encryption using R chosen according to the distribution defined by $\mathsf{Enc}(pk; \cdot)$ on $\mathcal{C}_{k_2}$ as

$$\mathsf{Enc}(pk; m) = (c_1, c_2) = (E_{k_1}(m), R),$$

Dec: Given a ciphertext (c_1, c_2), the recipient with the public key $pk = (k_1, k_2)$ and private key $sk = (s_1, s_2)$ proceeds as follows:

1. Set $m_1 = D_{s_1}(c_1)$.
2. If $s_2 \neq \perp$, compute $m_2 = D_{s_2}(c_2)$.
3. If m_2 is in the redundancy space then return $(m_1, \mathsf{red}^{-1}(m_2))$, otherwise return $(m_1, -)$.

Open_S: Suppose a coercer obtains a ciphertext (c_1, c_2) generated by the sender and requests its opening. Given public key $pk = (k_1, k_2)$, the sender opens the random choices made in generating c_1. The sender also claims that c_2 was chosen from $\mathcal{C}_{k_2}$ according to the distribution defined by $\mathsf{Enc}(pk; \cdot)$. Thus, the actual message the sender claims to have sent is $(m_1, -)$.

Open_R: Suppose a coercer obtains a ciphertext (c_1, c_2) that the recipient received. Given the recipient's public key $pk = (k_1, k_2)$,

the recipient opens its private key as $sk = (s_1, \perp)$ such that s_1 matches k_1. In other words, the recipient claims that the second part of the public key was chosen at random from $\mathcal{PK}(\kappa)$.

THEOREM 4. *Assuming that $\mathcal{E} = (Gen, E, D)$ is an* IND-CPA*-secure encryption scheme with security parameter κ, the scheme above is sender-deniable.*

THEOREM 5. *Assuming that $\mathcal{E} = (Gen, E, D)$ is an* IND-CPA*-secure encryption scheme with security parameter κ, the scheme above is receiver-deniable.*

The proof of theorems 4 and 5 is provided in Appendix A.

4. A DENIABLE SHARED FILE SYSTEM

Cloud computing is a powerful tool for improving productivity and ease of collaboration among users. It is not surprising that there is a growing interest towards these technologies, although privacy and security concerns are still mostly an open problem (see e.g. [21]). Many companies are offering practical storage and synchronization services at a very low cost or even for free. Thanks to these services, users can store their documents on line and easily synchronize them between a laptop and a desktop computer, or other users. In order to protect their privacy, users can encrypt their files prior to sending them using available encryption software. Unfortunately most current encryption software are designed to protect against adversaries who have no access to the decryption key, while in our model we assume that the coercer will be able to obtain access to such private information.

To solve this problem, a user can rely on a disk encryption tools with support for deniability, like TrueCrypt. This solution is clearly not acceptable in our scenario: TrueCrypt hidden volumes are easy to uncover by comparing multiple snapshots of the container. Moreover, the owner of the shared data may not want all users to be able to access portions of content encrypted by other members, unless explicitly specified. In order to overcome this, several hidden volumes can be created and shared using distinct secret keys. Unfortunately, if we let users share symmetric keys for access control then key management soon becomes a nightmare.

With our public-key deniable scheme we provide a reasonable solution to the problems above. In particular we use our plan-ahead deniable encryption scheme in the context of cloud storage and file sharing among collaborating users, by relying on a cloud provider – such as Amazon S3 and, on top of it, Dropbox – for the storage. Such provider does not have access to the decryption keys for the user's data, and is not trusted by the users. Any participant that is approached by the coercer and forced to decrypt one or more shared files can just open the corresponding non-deniable (*fake*) files without revealing any sensitive information.

4.1 High-level Description

In order to explore the practicality of our solution, we designed and implemented a prototype application. The purpose of our prototype is to implement DenFS, a file system which transparently and deniably encrypts all its content, and stores it *in the cloud*. In this way we intend to provide one of the basic tools for coordinating a collaboration between users. One of the design goals of DenFS is to provide a seamless experience, allowing users to interact with objects stored in the deniable file system as if they were stored on a standard file system: it is possible to read and write files and directories using standard tools and applications with no modifications. The content of the deniable file system can be shared with an arbitrary number of users. DenFS considers the cloud storage as untrusted and doesn't rely on ACLs for access control. Instead, DenFS implements access control through public key encryption: users can read only the files for which they hold the appropriate decryption key. When a new file is created, it is encrypted under the public key of one or more users. In this way DenFS allows content providers to select which users can read a specific file. When several users are collaborating, their files can shared by encrypting them under the the public keys of all collaborators. DenFS uses hybrid encryption, as discussed below, for higher efficiency. Different users may have a different view of the deniable file system, since DenFS shows only the encrypted files for which the appropriate decryption key was provided at mount time. The synchronization between files and folders in DenFS and the cloud storage happens in background. Files are also stored locally in encrypted form to provide a buffer which hides all the delays introduced by the network when reading or writing a file. Moreover, this buffer allows users to interact with files without being constantly connected. A synchronization service takes care of updating both the local cache and the on-line repository. Conflicts are shown to the user who can decide how to deal with them. File updates propagate with a speed proportional to users' available bandwidth. There is no significant performance downside when a relatively large number of users share the same folder, because each modification is sent only once to the cloud and then pushed concurrently to each user.

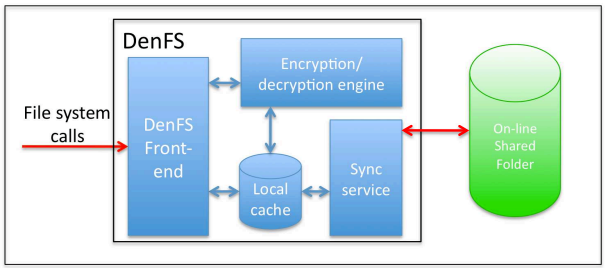

Figure 1: Components of the DenFS file system.

We designed our prototype to be efficient, secure, and easy to use. To mount a DenFS file system, a user simply specifies the mount point, the backend directory, and a password. The backend directory is the directory where encrypted files are stored. In order to synchronize files between users, the backend directory must be inside a Dropbox folder and must be shared. Each user's secret key is encrypted with a symmetric algorithm; the decryption password is prompted when mounting the file system. If a user does not enter the decryption password, it will only be able to encrypt new files. All file system operations, like reading or writing a file, are automatically mapped to the backend folder and therefore to the cloud.

Before mounting the file system, each user must create a directory containing sensitive looking files which will be used to fill the non-deniable part of a deniable encryption. This directory is called *non-deniable pool*. DenFS can be mounted in one of the two modes of operation available: *deniable* and *non-deniable*. When mounting the file system in non-deniable mode, random data is used to fill the deniable part of the ciphertext. In deniable mode, the non-deniable part is filled with the encryption of a file selected from the *non-deniable pool*. When the file system is mounted in non-deniable mode, reading only involves the non-deniable part while writing involves both the deniable and non-deniable part. We believe that this approach provides a reasonably good usability. Improving usability issues for a deniable filesystem and in particular for DenFS is still an open problem. Every time the non-deniable content is

Table 1: Sequential Output on a 3GB Dataset

	Per byte		Block		Rewrite	
	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU
Ext3	31657	39%	34455	5%	14432	2%
DenFS	22579	37%	34056	5%	14442	3%

Table 2: Sequential and Random Input on a 3GB Dataset

	Seq. per byte		Seq. per block		Random seeks	
	K/sec	% CPU	K/sec	% CPU	K/sec	% CPU
Ext3	35108	40%	53882	3%	4137	6%
DenFS	35447	52%	51758	2%	143.2	0%

Table 3: Small Files Sequential Performance – 512K Files

	Create		Read		Delete	
	files/s	CPU	files/s	CPU	files/s	CPU
Ext3	45643	65%	551512	99%	1227	1%
DenFS	5459	76%	22212	85%	17059	15%

Table 4: Small Files Random Performance – 512K Files

	Create		Read		Delete	
	files/s	CPU	files/s	CPU	files/s	CPU
Ext3	19388	27%	502471	99%	561	0%
DenFS	4902	67%	25766	84%	22280	22%

written in the file, the deniable part of the same file is overwritten with random data from the proper set. When the file system is mounted in deniable mode and a file is modified, the non-deniable part of the same file is overwritten choosing new data from the non-deniable pool. In this way an adversary who has access to several snapshots of the encrypted content of the shared folder is unable to distinguish between deniable and non-deniable operations by comparing snapshots.

DenFS is divided into several components. The architecture of DenFS is summarized in figure 1. The front-end provides a standard interface for file system calls, such as open, read and write. We based our implementation on FUSE (File system in Userspace) which is a library and a kernel module that allow non-privileged users to implement a fully functional file system in a userspace program [10]. The system calls which deal with file permissions, access and modification time, and file ownership implemented by the front-end are simply forwarded to the back-end (local cache). All the other system calls are mediated by a cryptographic engine which implements our deniable encryption scheme. We rely on OpenSSL [27] for the implementation of the ciphers and hash functions used by DenFS.

Czeskis et al. highlight in [8] that both the operating system and applications leak information about the deniably encrypted files. As an example, Microsoft Windows stores the serial number of each volume when it is mounted. With DenFS users do not resort to deniability to hide a volume, as in the case of TrueCrypt hidden volumes. Similarly, file history does not compromise the deniability of DenFS, since the coercer knows the name of each file in the deniable volume. Applications auto-saving feature should be instructed so that temporary files are stored on the deniable file system. Among the threats identified by Czeskis et al., a desktop search application (such as Google Desktop) seems to be the worst menace against the deniability of DenFS: when a DenFS volume is mounted in deniable mode, a desktop search application is able to construct its index from the deniable content of the files. Since the coercer has access to such index, the deniability of the encrypted documents is seriously compromised. We envision two countermeasures: 1) users can store the index constructed by the desktop search application on a deniably encrypted volume. In this case the coercer doesn't have access to the index corresponding to the deniable content of the DenFS volume; 2) users can disable the desktop search functionality on the deniably encrypted volumes.

4.2 Low-level Description

Our file system provides a transparent translation layer between the deniably encrypted local cache and the deniable mount point. When a new file is added to the file system, a file with the same name is encrypted and stored in the backend folder. For efficiency reasons, we encrypt each file using a hybrid encryption algorithm based on the scheme in section 3.3. For the session key encapsulation we employ RSA-OAEP, while for the data encapsulation we use AES with a 128-bit key in counter mode. The data encapsulation is an instantiation of the scheme in section 3.3, and the key encapsulation is an instantiation of the same scheme where the fake message is the key used to encrypt the fake message, while the deniable message is the key used to encrypt the real message. The structure of the files in the backend folder is:

header length	*header*	*size*	*non-deniable data*	*deniable data*

The fields *header length* and *size* are 32-bit little-endian unsigned integers and indicate the length of the header and the file size, respectively. Both *header length* and *size* are expressed in bytes, therefore the maximum file size is 4GB. This, however, requires the underlying file system in the backend folder to allow the creation of a file slightly larger than 8GB to accommodate the deniable and non-deniable data and the other fields. Dropbox does not impose any limit on the file size, although the free account only allows users to upload up to 2GB of data and therefore up to about 1GB of deniably encrypted data. The header also contains the session keys used to encrypt the non-deniable and the deniable messages, the two initialization vectors, and the two MAC tags. The default key size for the asymmetric cipher is 1024 bits and can vary from 1024 to 4096 bits. Users keep a local file which maps a directory on the deniable file system to a user's key. Adding a new file to a directory also means encrypting it under a specific key, which is chosen from the local map. If no key is specified for a folder, a new file stored in that folder is encrypted under the user's public key.

For every new encrypted file, the session key is saved in a buffer. This allows a user who writes a new file to be able to read it back even if it did not encrypt it under its public key. This feature allows the file system to mimic more closely the behavior of a standard file system in which, after writing a file, users can re-read it. By disabling this feature, a new file disappears after being saved if it was not encrypted under the creator's public key. The key buffer is erased when the file system is unmounted.

When the file system is used in non-deniable mode and a user creates a new file, our prototype writes a file with the same name in the backend. Then it picks a random session key, saves it in the key buffer and encrypts it. The ciphertext is stored in the file header. A write operation on the deniable folder translates to an encryption and a write operation in the associated file on the backend. The *size* field is updated accordingly. After that, the deniable part of the backend file is filled with random data and a random value is inserted in the header as a deniable encryption key.

When a user creates a new file on the file system mounted in deniable mode, a temporary file – *filename.den* – is created in a local

temporary folder for efficiency reasons. Note that when writing a new file, the size of the file is not necessarily know so it is impossible to predict the offset for the encryption of the deniable data. The temporary file contains the encryption of the deniable part of the document. The session keys are chosen and stored similarly to the non-deniable case. Every write operation on the front-end translates in a write operation in the temporary deniable file. When the document is closed, a file from the non-deniable pool is encrypted and stored in the non-deniable part of the document. The file is also removed from the non-deniable pool. The encryption of the deniable content is stored after the encryption of the file from the non-deniable pool. When a *stat* request is made to the file system, the attributes of the file in the backend are reported, except from the file size, which is read from the appropriate field. Hard links are not supported in the current implementation.

4.3 Performance Evaluation

We performed a combination of synthetic and real-world benchmarks to verify the performance impact of DenFS over a regular non-encrypted file system (Ext3 in our tests). We used Bonnie++ [4], a file system benchmark tool, and the OpenOffice productivity suite to evaluate the performance of our prototype. Bonnie++ performs sequential and random input/output tests. We measured sequential output performance writing a single character at a time, a whole block and rewriting one block at a time for 3GB of data. We also performed a sequential reading test, where we read one character at a time and then one block at a time from the same data. Finally we performed random seeks within a large files and measured how many seeks per second we were able to perform. Then we measured the overhead for small files by creating, reading and deleting 512,000 files in both sequential and random order.

The test machine is based on an Intel Core 2 Duo T9300 processor running at 2.6 GHz. The available RAM is 4GB and the hard disk is a 320GB, 5400 rpm Toshiba MK3252GSX. The operating system is Ubuntu Linux with kernel 2.6.31. The key length for the asymmetric ciphers was set to 1024 bits in all tests. The file system was mounted in deniable mode. The results for sequential and random access on a large file are summarized in tables 1 and 2. The results clearly show that when dealing with large files, all tests except random seeks are disk limited. In most cases, there is a small loss of performance introduced by DenFS which we believe is not going to be noticeable in everyday use. The most pronounced performance penalty is in the random read performance which decreases from about 4MB/s to 143KB/s.

Note that in some of the tests DenFS uses slightly less CPU than Ext3. This may appear odd at first, since DenFS is an encrypted filesystem. However the CPU usage in Table 1 (per-byte access) and 2 (sequential per-block access) can be justified by small variation between different runs. The CPU is clearly not the bottleneck in these tests, and therefore such variation is hard to eliminate. The behavior of the benchmark in table 3 (read) and 4 (read) – both the throughput in terms of files/second and the CPU usage are lower with DenFS than with Ext3 – is a clear indication that in this part of the test the performance of DenFS depends much more on the disk than on the CPU. Note that in our test DenFS was capable of deleting files ten to forty times faster than Ext3 because with DenFS files are not deleted immediately, but put in a queue and processed in background. Deleted files disappear immediately from the file system from an application point of view.

We investigated further to determine the amount of the overhead introduced by the cryptographic algorithms with small files. More specifically, we were interested to see how the file size impacted the read and write speed. Therefore, we created a large set of small

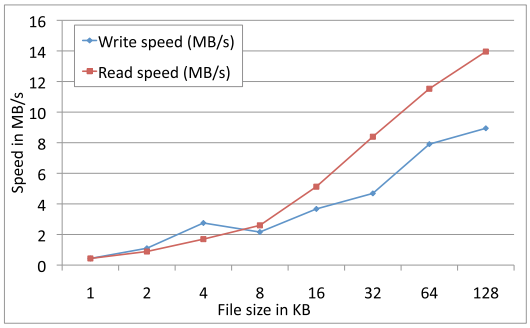

Figure 2: Read and write speed for small files.

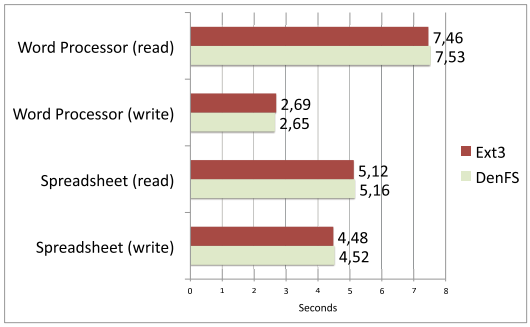

Figure 3: Performance comparison between Ext3 and DenFS with OpenOffice. The reported values are measured in seconds.

files and we measured the read and write speed when copying the files from and to the deniable file system. The size of the files composing our testbed varied from 1KB to 128KB. The results are summarized in figure 2, which shows the average speed in MB/s for a given file size. With small files, the overhead introduced by our file system is non-negligible. When the file size increases, both write and read performance improve noticeably as expected.

We also performed a real-world test with OpenOffice 3.0.1 build 9379. In order to simulate access to complex documents, we created a 306 pages OpenOffice Writer document and a spreadsheet of an approximate size of 940KB. The spreadsheet contained several thousand cells and ten graphs, while the word processor document, which was all written using the same font, also contained some bitmap images. We tried to open, modify, and save them on both Ext3 and DenFS. A copy of both OpenOffice Writer and Calc was running in the background to get rid of the application loading time in the measurements. The values shown on figure 3 represent the average of ten measurements. We believe that the performance of the two file systems are virtually indistinguishable for a user.

5. CONCLUSIONS

Deniable encryption allows the sender or receiver to open a ciphertext to a different value than what it encrypts. This work brings deniable encryption to practical use by developing the first efficient sender-and-receiver deniable *public-key* encryption scheme from standard tools alone and introducing the concept of deniable cloud storage. Deniable cloud storage allows users to collaborate and store data in such a way that if an adversary coerces a user into opening all encrypted data, the information still stays secret. We develop a prototype of a deniable file system DenFS which realizes this functionality.

6. REFERENCES

[1] R. Anderson, R. Needham, and A. Shamir. The Steganographic File System. In *International Workshop on Information Hiding*, pages 73–82, 1998.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katzand, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[3] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption. In *EUROCRYPT*, pages 92–111, 1994.

[4] Bonnie++: a free and open source filesystem benchmark. `http://www.coker.com.au/bonnie++/`.

[5] E. Bresson, D. Catalano, and D. Pointcheval. A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In *Advances in Cryptology - ASIACRYPT'03*, pages 37–54, 2003.

[6] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable Encryption. In *Adnvances in Cryptology – CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104, 1997.

[7] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. *Journal of Cryptology*, 20(3):265–294, 2007.

[8] A. Czeskis, D. J. St. Hilaire, K. Koscher, S. D. Gribble, T. Kohno, and B. Schneier. Defeating Encrypted and Deniable File Systems: TrueCrypt v5.1a and the Case of the Tattling OS and Applications. In *USENIX Workshop on Hot Topics in Security (HotSec'08)*, 2008.

[9] I. Damgard and M. Jurik. A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography (PKC'01)*, pages 119–136, 2001.

[10] Filesystem in Userspace. `http://fuse.sourceforge.net/`.

[11] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT'02*, pages 548–566, 2002.

[12] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In *Advances in Cryptology – EUROCRYPT'02*, pages 466–481, 2002.

[13] M. Ibrahim. A Method for Obtaining Deniable Public-key Encryption. *International Journal of Network Security*, 8(1):1–9, 2009.

[14] M. Ibrahim. Receiver-deniable Public-key Encryption. *International Journal of Network Security*, 8(2):159–165, 2009.

[15] A. Kiayias and M. Yung. Efficient Secure Group Signatures with Dynamic Joins and Keeping Anonymity Against Group Managers. In *Progress in Cryptology – Mycrypt'05*, pages 151–170, 2005.

[16] M. Klonowski, P. Kubiak, and M. Kutylowski. Practical Deniable Encryption. In *SPFSEM'08*, volume 4910 of *LNCS*, pages 599–609, 2008.

[17] B. Meng and J. Wang. A Receiver Deniable Encryption Scheme. In *International Symposium on Information Processing (ISIP'09)*, 2009.

[18] N. Mirzaei. Cloud Computing. Technical report, Indiana University, 2008.

[19] P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology – EUROCRYPT'99*, pages 223–238, 1999.

[20] H. Pang, K. Tan, and X. Zhou. StegFS: A Steganographic File System. In *International Conference on Data Engineering (ICDE'03)*, page 657, 2003.

[21] S. Pearson. Taking Account of Privacy when Designing Cloud Computing Services. In *ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD '09)*, pages 44–52, 2009.

[22] Amazon S3 `http://aws.amazon.com/s3/`.

[23] Rubberhose Project `http://iq.org/~proff/rubberhose.org/`.

[24] Dropbox `http://www.dropbox.com/`.

[25] Truecrypt `http://www.truecrypt.org/`.

[26] TrueCrypt Hidden Volumes `http://www.truecrypt.org/hiddenvolume`.

[27] The OpenSSL Project website `http://www.openssl.org/`.

[28] M. Vouk. Cloud Computing: Issues, Research and Implementations. In *International Conference on Information Technology Interfaces (ITI'08)*, pages 31–40, 2008.

[29] Wired: "Spam Suspect Uses Google Docs; FBI Happy" `http://www.wired.com/threatlevel/2010/04/cloud-warrant/`.

[30] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In *ACM Conference on Computer and Communications Security (CCS'04)*, pages 354–363, 2004.

[31] X. Zhou, H. Pang, and K. Tan. Hiding Data Accesses in Steganographic File System. In *International Conference on Data Engineering (ICDE'04)*, page 572, 2004.

APPENDIX

A. SECURITY PROOFS

PROOF OF THEOREM 4. In order to prove that our scheme is deniable, we need to show that an adversary $\mathcal{A}$ which separately plays the IND-CPA experiment (for the security property) and the *sender deniability* experiment (for the deniability property) can have only negligible advantage in winning any of the experiments. First we show that there is no adversary with non-negligible advantage in breaking the IND-CPA security of our scheme through a sequence of experiments.

Experiment 0 This experiment is the standard IND-CPA experiment, i.e., the adversary $\mathcal{A}$ receives a public key pk and outputs two messages $(m_{0,f}, m_0)$ and $(m_{1,f}, m_1)$. The challenger picks a random bit b and sends the encryption of $(m_{b,f}, m_b)$ to $\mathcal{A}$. $\mathcal{A}$ eventually outputs a bit b'. By definition we have that $\Pr[\mathsf{win}_0] = \Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa) = 1]$.

Experiment 1 In this experiment we modify the way the challenger constructs the challenge ciphertext. In particular, now it constructs the challenge ciphertext as $C = (c_1, c_2)$ where c_1 and c_2 are the encryption of two random messages from $\mathcal{M}_{pk}$. The adversary can only distinguish Experiment 1 from Experiment 0 with negligible probability, since $\mathcal{E}$ is IND-CPA-secure. Therefore there exists a negligible function negl_0 such that $|\Pr[\mathsf{win}_0] - \Pr[\mathsf{win}_1]| \leq \mathsf{negl}_0(\kappa)$. Combining the probabilities from Experiments 0 and 1, we have that $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] = \Pr[\mathsf{win}_1] + \mathsf{negl}(\kappa)$ for some negligible function negl. Note that the probability $\Pr[\mathsf{win}_1]$ is exactly $1/2$ because the challenge ciphertext in Experiment 1 is

chosen independently from bit b, therefore $\mathcal{A}$ can only guess b with probability $1/2$, therefore the advantage of $\mathcal{A}$ must be negligible.

To prove the deniability property we show that if adversary $\mathcal{A}$ wins the *sender deniability* experiment with probability non-negligibly larger than $1/2$ then we can build an efficient algorithm $\mathcal{B}$ that wins in the IND-CPA experiment for $\mathcal{E}$ with non-negligible advantage. Let $\Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] - 1/2 = \delta$. $\mathcal{B}$ engages in the IND-CPA experiment with a challenger, obtains the public key $\overline{pk}$ and uses it to setup the *sender deniability* experiment for $\mathcal{A}$ by constructing $pk = (k_1, k_2)$ where k_1 is generated using $Gen(1^\kappa)$ and $k_2 = \overline{pk}$ (i.e., $\mathcal{B}$ sends also k_2 to $\mathcal{A}$ without knowing the corresponding private key). $\mathcal{B}$ responds to the $\mathsf{Enc}(m_i, m_j, pk)$ queries with $C(m_i\,;\ m_j)$. Let (r_i, r_j) be the randomness used by $\mathcal{B}$ to respond to the Enc query. $\mathcal{A}$ eventually outputs its choice for the pair (m_f, m). $\mathcal{B}$ outputs m_0, m_1 to the challenger where $m_0 = m$, and $m_1 = R$ is a random message from the message space such that $|R| = |m|$. The challenger picks a random bit b and returns $\bar{c} = E_{\overline{pk}}(m_b)$. $\mathcal{B}$ builds the challenge ciphertext for $\mathcal{A}$ as $\overline{C} = (c_1\,;\ \bar{c})$, where $c_1 = E_{k_2}(m_f)$. Let r be the randomness used by $\mathcal{B}$ to encrypt c_1. $\mathcal{B}$ responds to $\mathsf{Open}_S(C, pk)$ queries with r if $C = \overline{C}$, with (r_i, r_j) if $C = C(m_i, m_j)$, and with $\perp$ otherwise. Eventually $\mathcal{A}$ outputs b', and $\mathcal{B}$ outputs the same guess to the challenger. Clearly $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{B},\mathcal{E}}(\kappa) = 1] = \Pr[\mathsf{SDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1]$. That is, if $b = 0$, $\mathcal{B}$ receives the encryption of m and $\mathcal{A}$ receives the encryption of (m_f, m), in which case $\mathcal{B}$ answers the challenge correctly if and only if $\mathcal{A}$ answers its challenge correctly. If $b = 1$, $\mathcal{B}$ receives the encryption of R and $\mathcal{A}$ receives the encryption of (m_f, R), which is interpreted as $(m_f, -)$ since R is a random message. Therefore $\mathcal{B}$ guesses correctly if and only if $\mathcal{A}$ guesses correctly. Since $\mathcal{B}$ cannot guess correctly non-negligibly more than half of the times, $\delta \leq \mathsf{negl}(\kappa)$ for some negligible function negl. □

PROOF OF THEOREM 5. For the security property see proof of theorem 4. For the deniability problem, given an IND-CPA-secure scheme $\mathcal{E} = (Gen, E, D)$, we show that if adversary $\mathcal{A}$ wins in the *receiver deniability* experiment with probability non-negligibly larger than $1/2$, then we can build an efficient algorithm $\mathcal{B}$ that wins in the IND-CPA experiment with non-negligible advantage.

Let $\Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1] - 1/2 = \delta$. $\mathcal{B}$ engages in the IND-CPA experiment with a challenger, obtains the public key $\overline{pk}$ and uses it to setup the *receiver deniability* experiment for $\mathcal{A}$ by constructing $pk = (k_1, k_2)$ where k_1 is generated using $Gen(1^\kappa)$ and $k_2 = \overline{pk}$ Let s_1 be the private key associated with k_1. $\mathcal{B}$ responds to any $\mathsf{Open}_R(C, pk)$ query with $sk = (s_1, \perp)$, regardless of whether C encrypts one messages or two. $\mathcal{A}$ eventually outputs its choice for the pair (m_f, m). $\mathcal{B}$ outputs m_0, m_1 to the challenger where $m_0 = m$, and $m_1 = R$ is a random message from the message space such that $|R| = |m|$. The challenger picks a random bit b and returns $\bar{c} = E_{\overline{pk}}(m_b)$. $\mathcal{B}$ constructs the challenge ciphertext for $\mathcal{A}$ as $C = (c_1\,;\ \bar{c})$, where $c_1 = E_{k_1}(m_f)$. Eventually $\mathcal{A}$ outputs b', and $\mathcal{B}$ outputs the same guess to the challenger.

Clearly $\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{B},\mathcal{E}}(\kappa) = 1] = \Pr[\mathsf{RDen}_{\mathcal{A},\mathcal{E}}(\kappa) = 1]$. Indeed, if $b = 1$, $\mathcal{B}$ receives the encryption of R and $\mathcal{A}$ receives the encryption $C(m_f;\ R)$, which is interpreted as $C(m_f;\ -)$ since R is a random message. Thus $\mathcal{B}$ answers the challenge correctly if and only if $\mathcal{A}$ answers its challenge correctly.

If $b = 0$, $\mathcal{B}$ receives the encryption of m_f and $\mathcal{A}$ receives the encryption of (m_f, m). $\mathsf{Open}_R(C, pk)$ returns $sk = (s_1, \perp)$ which exposes m_f. Thus even in this case $\mathcal{B}$ guesses correctly if and only if $\mathcal{A}$ guesses correctly. Since $\mathcal{B}$ cannot guess correctly non-negligibly more than half of the times, $\delta \leq \mathsf{negl}(\kappa)$ for some negligible function negl. □

B. RESILIENCE TO PHISHING ATTACKS

In order to obtain receiver-deniability, our sender-and-receiver deniable schemes require the receiver to conceal the existence of the second private key (if present), so that it does not have to reveal it to the coercer. In practice, it may be possible for an adversary to obtain information about the existence of such secret key. A coercer can send to the receiver a deniable message which asks to perform a specific action that the coercer will be able to notice, such as opening a maliciously crafted web page. Even if we assume that the receiver can avoid any action that the adversary might notice, the coercer may obtain the same result by exploiting a vulnerability in the receiver's client. As far as we can ascertain, this type of phishing attack has not been considered before in the deniability literature. Phishing attacks have obvious bearing in the cloud storage setting: a malicious cloud provider is in a favorable position to perform this kind of attack, since it could determine the existence of a private key by adding a (public-key) encrypted file to the cloud storage and observing the resulting behavior of the targeted user.

As a trivial solution to this problem, instead of using the same key pair for all encryptions the receiver generates several independent key pairs so that each message is encrypted using a different and independent public key. Even if the coercer establishes the existence of a specific private key, it does not learn any information about any other. Unfortunately, this trivial solution has a very high cost in terms of key distribution and imposes a limit on the number of messages which can be sent to a receiver.

Forward-secure schemes may seem to provide a more efficient solution. A forward-secure public key encryption scheme [7] protects secret keys from exposure by evolving them at regular intervals. When a key evolves into a new key, the old secret key is deleted and cannot be reconstructed, and therefore cannot be provided to the coercer. Unfortunately, the existence of a private key at a specific time interval implies the existence of private keys corresponding to all previous time intervals, making this solution invalid.

A better solution is constructed using an identity-based encryption scheme (IBE) to "compress" the public keys into a single value. Roughly, the receiver acts as the IBE trusted authority (PKG) itself, divides the time in intervals, and generates an IBE key under his identity for a random *subset* of all intervals; then the recipient is supposed to delete any master secrets of the PKG. The sender encrypts messages for the receiver with a key valid at a specific time. If the coercer establishes the existence of the private key for a time interval, only the messages encrypted in that interval are compromised. The problem with this approach is that the secret key must consist of large number of values. To avoid this, the receiver can claim to delete the master secret keys of the IBE trusted authority but keep a copy instead deniably encrypted locally, so that new IBE keys can be generated when needed.

A more elegant solution can be built using a forward secure hierarchical identity-based encryption scheme (FS-HIBE) [30]. A hierarchical IBE (HIBE) [11, 12] allows a PKG to delegate the generation of private keys to lower-level PKGs. A FS-HIBE scheme allows each user in the hierarchy to refresh its private key periodically while keeping the public key the same. Using a FS-HIBE scheme, each user can construct a single path *Username* → *Year* → *Month* → *Week* and have some secret key to evolve while others get deleted. As with the previous solution, the exposure of a secret key compromises only the messages encrypted in one time interval.

The existence of more efficient solutions is still an open problem. We do not investigate this further at this time and we point out that phishing attacks are relevant in this context and they seemingly affect any plan-ahead public-key (i.e., *non-interactive*) deniable encryption scheme.