A preliminary version of this paper appears as part of our Eurocrypt 2009 paper with Hofheinz [BHY09]. This is the full version.

Encryption Schemes Secure under Selective Opening Attack

MIHIR BELLARE[*] SCOTT YILEK[†]

September 2008

Abstract

The existence of encryption schemes secure under selective opening attack (SOA) has remained open despite considerable interest and attention. We provide the first public key encryption schemes secure against sender corruptions in this setting. The underlying tool is lossy encryption. The schemes have short keys. (Public and secret keys of a fixed length suffice for encrypting an arbitrary number of messages.) The schemes are stateless and noninteractive, and security does not rely on erasures. The schemes are without random oracles, proven secure under standard assumptions (DDH, Paillier's DCR, QR, lattices), and even efficient. We are able to meet both an indistinguishability (IND-SO-ENC) and a simulation-style, semantic security (SEM-SO-ENC) definition.

[*]Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`. Supported in part by NSF grants CNS–0524765 and CNS–0627779 and a gift from Intel Corporation.

[†]Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: `syilek@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/syilek`. Supported in part by NSF grants CNS–0430595 and CNS–0831536.

Contents

1 Introduction

IND-CPA and IND-CCA are generally viewed as strong notions of encryption security that suffice for applications. However, there is an important setting where these standard notions do not in fact imply security and the search for solutions continues, namely, in the presence of selective-opening attack (SOA) [DNRS03, CFGN96, Nie02, DN00, CHK05, CDNO97]. Let us provide some background on SOA and then discuss our results.

1.1 Background

THE PROBLEM. Suppose a receiver with public encryption key pk receives a vector $\mathbf{c} = (\mathbf{c}[1], \ldots, \mathbf{c}[n])$ of ciphertexts, where sender i created ciphertext $\mathbf{c}[i] = \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$ by encrypting a message $\mathbf{m}[i]$ under pk and coins $\mathbf{r}[i]$ $(1 \leq i \leq n)$. It is important here that *the messages* $\mathbf{m}[1], \ldots, \mathbf{m}[n]$ *might be related*, but *the coins* $\mathbf{r}[1], \ldots, \mathbf{r}[n]$ *are random and independent*. Now, the adversary, given $\mathbf{c}$, is allowed to corrupt some size t subset $I \subseteq \{1, \ldots, n\}$ of senders (say $t = n/2$), obtaining not only their messages but *also their coins*, so that it has $\mathbf{m}[i], \mathbf{r}[i]$ for all $i \in I$. This is called a selective opening attack (SOA). The security requirement is that the privacy of the unopened messages, namely $\mathbf{m}[i_1], \ldots, \mathbf{m}[i_{n-t}]$ where $\{i_1, \ldots, i_{n-t}\} = \{1, \ldots, n\} \setminus I$, is preserved. (Meaning the adversary learns nothing more about the unopened messages than it could predict given the opened messages and knowledge of the message distribution. Formal definitions to capture this will be discussed later.) The question is whether SOA-secure encryption schemes exist.

STATUS AND MOTIVATION. One's first impression would be that a simple hybrid argument would show that any IND-CPA scheme is SOA-secure. Nobody has yet been able to push such an argument through. (And, today, regarding whether IND-CPA implies SOA-security we have neither a proof nor a counterexample.) Next one might think that IND-CCA, at least, would suffice, but even this is not known. The difficulty of the problem is well understood and documented [DNRS03, CFGN96, CHK05, Nie02, DN00, CDNO97], and whether or not SOA-secure schemes exist remains open.

Very roughly, the difficulties come from a combination of two factors. The first is that it is the random coins underlying the encryption, not just the messages, that are revealed. The second is that the messages can be related.

We clarify that the problem becomes moot if senders can erase their randomness after encryption, but it is well understood that true and reliable erasure is difficult on a real system. We will only be interested in solutions that avoid erasures.

The problem first arose in the context of multiparty computation, where it is standard to assume secure communication channels between parties [BOGW88, CCD88]. But, how are these to be implemented? Presumably, via encryption. But due to the fact that parties can be corrupted, the encryption would need to be SOA-secure. We contend, however, that there are important practical motivations as well. For example, suppose a server has SSL connections with a large number of clients. Suppose a virus corrupts some fraction of the clients, thereby exposing the randomness underlying their encryptions. Are the encryptions of the uncorrupted clients secure?

COMMITMENT. Notice that possession of the coins allows the adversary to verify that the opening is correct, since it can compute $\mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$ and check that this equals $\mathbf{c}[i]$ for all $i \in I$. This apparent commitment property has been viewed as the core technical difficulty in obtaining a proof. The view that commitment is in this way at the heart of the problem has led researchers to formulate and focus on the problem of commitment secure against SOA [DNRS03]. Here, think of the algorithm $\mathcal{E}$ in our description above as the commitment algorithm of a commitment scheme, with the public key being the empty string. The question is then exactly the same.

DEFINITIONS. Previous work [DNRS03] has introduced and used (for commitment) a simulation-based, semantic-style security formalization of security under SOA. A contribution of our paper is to provide (for encryption) an alternative indistinguishability-based formalization that we denote IND-SO-ENC. We will also refer to a simulation-based, semantic security formalization SEM-SO-ENC for encryption.

1.2 Results

We provide the first public-key encryption schemes provably secure against selective-opening attack. The schemes have short keys. (Public and secret keys of a fixed length suffice for encrypting an arbitrary number of messages.) The schemes are stateless and noninteractive, and security does not rely on erasures. The schemes are without random oracles, proven secure under standard assumptions, and even efficient. We are able to meet both the indistinguishability (IND-SO-ENC) and the semantic security (SEM-SO-ENC) definitions, although under different assumptions.

CLOSER LOOK. The main tool (that we define and employ) is lossy encryption, an encryption analogue of lossy trapdoor functions (LTDFs) [PW08] that is closely related to meaningful-meaningless encryption [KN08] and dual-mode encryption [PVW08]. We provide an efficient implementation of lossy encryption based on DDH. We also show that any (sufficiently) lossy trapdoor function yields lossy encryption. Via [PW08, BFO08, RS08] we thereby obtain lossy encryption schemes based on DDH, Paillier's DCR [Pai99] and lattices.

We then show that any lossy encryption scheme is IND-SO-ENC secure, thereby obtaining IND-SO-ENC secure schemes based on DDH, DCR and lattices. If the lossy encryption scheme has an additional property that we call efficient openability, we show that it is also SEM-SO-ENC secure. We observe that the classical quadratic residuosity-based encryption scheme of Goldwasser and Micali [GM84] is lossy with efficient openability, thereby obtaining SEM-SO-ENC secure encryption. It is interesting in this regard that the solution to a long-standing open problem is a scheme that has been known for 25 years. (Only the proof was missing until now.)

1.3 Discussion and related work

SOA SECURE ENCRYPTION. In the version of the problem that we consider, there is one receiver and many senders. Senders may be corrupted, with the corruption exposing their randomness and message. An alternative version of the problem considers a single sender and many receivers, each receiver having its own public and secret key. Receivers may be corrupted, with corruption exposing their secret key. Previous work has mostly focused on the receiver corruption version of the problem. Canetti, Feige, Goldreich and Naor [CFGN96] introduce and implement non-committing encryption, which yields SOA-secure encryption in the receiver corruption setting. However, their scheme does not have short keys. (Both the public and the secret key in their scheme are as long as the total number of message bits ever encrypted.) Furthermore, Nielsen [Nie02] shows that this is necessary. Canetti, Halevi and Katz [CHK05] provide SOA-secure encryption schemes for the receiver corruption setting with short public keys, but they make use of (limited) erasures. (They use a key-evolving system where, at the end of every day, the receiver's key is updated and the previous version of the key is securely erased.) In the symmetric setting, Panjwani [Pan07] proves SOA-security against a limited class of attacks.

Our schemes do not suffer from any of the restrictions of previous ones. We have short public and secret keys, do not rely on erasures, and achieve strong notions of security.

A natural question is why our results do not contradict Nielsen's negative result saying that no non-interactive public key encryption scheme with short and fixed keys is SOA-secure without erasures for an unbounded number of messages [Nie02]. The reason is that we consider sender corruptions as opposed to receiver corruptions.

REMARK. It has generally been thought that the two versions of the problem (sender or receiver corruptions) are of equal difficulty. The reason is that corruptions, in either case, allow the adversary to verify an opening and appear to create a commitment. (Either the randomness or the decryption key suffices to verify an opening.) Our work refutes this impression and shows that sender corruptions are easier to handle than receiver ones. Indeed, we can fully resolve the problem in the former case, while the latter case remains open. (Achieving a simulation-based notion for receiver corruptions is ruled out by [Nie02] but achieving an indistinguishability-based notion may still be possible.)

COMMITMENT. The first explicit treatment of SOA-secure commitment is by [DNRS03]. They formalized the problem and defined a simulation style security notion that we will call SEM-SO-COM. On the negative side, they showed that the existence of a one-shot (this means non-interactive and without setup assumptions) SEM-SO-COM-secure commitment scheme implied solutions to other well-known cryptographic problems, namely, three-round ZK and "magic functions." This is evidence that simulation-based one-shot SOA-secure commitment is difficult to achieve. On the positive side [DNRS03] showed that any statistically hiding chameleon commitment scheme is SOA-secure. (This scheme would not be one-shot, which is why this does not contradict their negative results.) In the zero-knowledge (ZK) setting, [GM06] notice a selective opening attack and circumvent it by adapting the distribution of the committed messages.

In work that was independent of, and concurrent to, ours, Hofheinz [Hof08] continued the investigation of SDA-secure commitment. He showed that no one-shot or perfectly binding commitment scheme can be shown SEM-SO-COM-secure using black-box reductions to standard assumptions. On the other hand, via non-black-box techniques, he showed that there exists an interactive SEM-SO-COM-secure commitment scheme under the assumption that one-way permutations exist. He also introduced an indistinguishability style notion that we will call IND-SO-COM. He showed that no perfectly hiding commitment scheme (whether interactive or not) can be shown IND-SO-COM secure using black-box reductions to standard assumptions. On the positive side, he showed that any statistically hiding commitment scheme is IND-SO-COM secure. (We note that a special case of this result was already implicit in [BR07].) He does not consider encryption.

An obvious question is why our results for encryption do not contradict the above negative results for commitment. The answer is that our SOA-secure encryption schemes do not give rise to commitment schemes. The commitment results do show that the SOA-security of an encryption scheme cannot be proved using a black-box reduction, *but only if encryption constitutes a commitment.* Because we consider SOA-security under *sender* corruptions in the encryption setting, this is not the case. (Recall that with sender corruptions, an encryption opening does not reveal the secret key, so the information-theoretic argument of [Nie02] that any encryption scheme is committing does not apply.)

BHY. Our paper, along with that of Hofheinz, were submitted to Eurocrypt 2009. They were accepted under the condition that they be merged. The resulting merged paper appeared as [BHY09]. Full versions have, however, been written separately as the present paper and [Hof08].

2 Notation

For any integer n, let 1^n be its unary representation and let $[n]$ denote the set $\{1, \ldots, n\}$. If b is a tuple of values of size m, we will write $(b_1, \ldots, b_m) \leftarrow b$ when we mean that b is parsed into b_1 to b_m. We let $a \leftarrow_{\$} b$ denote choosing a value uniformly at random from random variable b and assigning it to a.

We say a function $\mu(n)$ is negligible if $\mu \in o(n^{-\omega(1)})$. We let $\mathsf{neg}(n)$ denote an arbitrary negligible function. If we say some $p(n) = \mathrm{poly}(n)$, we mean that there is some polynomial q such that for all sufficiently large n, $p(n) \leq q(n)$. The statistical distance between two random variable X and Y over common domain D is $\Delta(X, Y) = \frac{1}{2}\sum_{z \in D} |\Pr[X = z] - \Pr[Y = z]|$ and we say that two random variables X and Y are δ-close if their statistical distance is at most δ and if δ is negligible, we might say $X \equiv_s Y$.

We denote by ϵ the empty string. For any strings m_0 and m_1, let $m_0 \oplus m_1$ denote the bitwise xor of the two strings. We use boldface letters for vectors, and for any vector $\mathbf{m}$ of n messages (called an n-vector) and $i \in [n]$, let $\mathbf{m}[i]$ denote the ith message in $\mathbf{m}$. For a set $I \subseteq [n]$ of indices $i_1 < i_2 < \ldots < i_l$, let $\mathbf{m}[I] = (\mathbf{m}[i_1], \mathbf{m}[i_2], \ldots, \mathbf{m}[i_l])$. For any set I (resp. any vector $\mathbf{m}$)(resp. any string m), let $|I|$ (resp. $|\mathbf{m}|$) (resp. $|m|$) denote the size of the set (resp. length of the vector) (resp. length of the string). An n-message sampler $\mathcal{M}$ is an algorithm that on input a security parameter λ in unary ouputs an $n(\lambda)$-vector of messages.

All algorithms in this paper are randomized, unless otherwise specified as being deterministic. For any algorithm A, let $\mathsf{Coins}_A(x_1, x_2, \ldots)$ denote the set of possible coins A uses when run on inputs $x_1, x_2, \ldots$. Let $A(x_1, x_2, \ldots; r)$ denote running algorithm A on inputs $x_1, x_2, \ldots$ and with coins $r \in \mathsf{Coins}_A(x_1, x_2, \ldots)$. Then $A(x_1, x_2, \ldots)$ denotes the random variable $A(x_1, x_2, \ldots; r)$ with r chosen uniformly at random from $\mathsf{Coins}_A(x_1, x_2, \ldots)$. When we say an algorithm is efficient, we mean that it runs in polynomial time in its first input; if the algorithm is randomized we might also say it runs in probabilistic polynomial time (PPT). An unbounded algorithm does not necessarily run in polynomial time.

3 Encryption Related Definitions

3.1 Public-Key Encryption Schemes

A public-key encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of PT algorithms. The key generation algorithm $\mathcal{K}$ takes as input 1^λ and outputs a public key/secret key pair (pk, sk). The encryption algorithm $\mathcal{E}$ takes as input a public key pk and a message m and outputs a ciphertext c. The decryption algorithm takes as input a secret key sk and a ciphertext c and outputs either a message m, or $\perp$, denoting failure. We require the correctness condition that for all (pk, sk) generated by $\mathcal{K}$, and for all messages m, $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. The standard notion of security for public-key encryption scheme is indistinguishability under chosen-plaintext attack (IND-CPA) [GM84].

3.2 Encryption Security under Selective Opening Attack

We consider both indistinguishability-based and simulation-based definitions of security for encryption under selective opening which we call IND-SO-ENC and SEM-SO-ENC, respectively. In the following definitions we say that a pair of functions (n, t) is a valid SOA parameter pair if both n and t are integer functions of the security parameter, n is at most polynomial in the security parameter, and for all $\lambda \in \mathbb{N}$, $t(\lambda) \leq n(\lambda)$.

INDISTINGUISHABILITY-BASED. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, n and t be integer functions, $\mathcal{M}$ be an n-message sampler. The game INDSO in Figure 3 provides an adversary with a single **Corrupt** oracle. An indso-adversary makes exactly one query to this **Corrupt** oracle. We say the indso-advantage of an indso-adversary A with respect to $\mathcal{M}, n, t$ is

$$\mathbf{Adv}^{\text{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) = 2 \cdot \Pr\left[\, \text{INDSO}^{A}_{\mathcal{AE},\mathcal{M},n,t}(\lambda) \Rightarrow \mathsf{true} \,\right] - 1,$$

where $\mathcal{M}|_{I,\mathbf{m}_0[I]}$ returns a random n-vector $\mathbf{m}_1$ according to $\mathcal{M}$, subject to $\mathbf{m}_1[I] = \mathbf{m}_0[I]$. In other words, $\mathcal{M}|_{I,\mathbf{m}_0[I]}$ denotes conditionally resampling from the message space subject to the constraint that the messages corresponding to indices in I are equal to $\mathbf{m}_0[I]$. If there is an efficient algorithm that does this resampling (for all I, $\mathbf{m}_0$), we say $\mathcal{M}$ supports efficient conditional resampling.

We say that a public-key encryption scheme $\mathcal{AE}$ is IND-SO-ENC-secure if for any valid SOA parameter pair (n, t), any efficient n-message sampler $\mathcal{M}$ that supports efficient conditional resampling and for all efficient indso-adversaries A, the indso-advantage of A with respect to $\mathcal{M}, n, t$ is negligible.

```
procedure Initialize($\lambda$):                        procedure Corrupt($I$):                INDSO$_{\mathcal{AE},\mathcal{M},n,t}$
    $(pk, sk) \leftarrow\!\!{\scriptstyle\$}\ \mathcal{K}(1^\lambda)$                If $|I| \neq t$ return $\perp$
    $\mathbf{m}_0 \leftarrow\!\!{\scriptstyle\$}\ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\!\!{\scriptstyle\$}\ \{0,1\}$          $\mathbf{m}_1 \leftarrow\!\!{\scriptstyle\$}\ \mathcal{M}|_{I,\mathbf{m}_0[I]}$
    For $i = 1, \ldots, n(\lambda)$ do                       Return $(\mathbf{r}[I], \mathbf{m}_b)$
        $\mathbf{r}[i] \leftarrow\!\!{\scriptstyle\$}\ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_0[i])$
        $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_0[i]; \mathbf{r}[i])$        procedure Finalize($b'$):
    Return $(pk, \mathbf{c})$                                  Return $(b = b')$
```

Figure 1: Security game for indistinguishability-based definition.

In words, the game proceeds as follows. The adversary is given a public key pk and n ciphertexts $\mathbf{c}$ encrypted under public key pk. The messages corresponding to the n ciphertexts come from the joint distribution $\mathcal{M}$. The adversary can query **Corrupt** once with a set I of t ciphertexts. In response, it receives the randomness $\mathbf{r}[I]$ used to generate those ciphertexts in addition to a message vector $\mathbf{m}_b$ such that $\mathbf{m}_b[I]$ were the actual messages encrypted using $\mathbf{r}[I]$ and the rest of $\mathbf{m}_b$ depends on the bit b. If b, which the experiment chooses randomly, is 0, the rest of the messages in the vector are the actual messages used to create the ciphertexts $\mathbf{c}$ that were given to the adversary. If $b = 1$, the rest of the messages are instead resampled from $\mathcal{M}$, conditioned on I and $\mathbf{m}_b[I]$. The adversary must then try to guess the bit b.

The definition is a natural extension of IND-CPA to the selective decryption setting. Intuitively, the definition means that an adversary, after adaptively choosing to open some ciphertexts, cannot distinguish between the actual unopened messages and another set of messages that are equally likely given the opened messages that the adversary has seen.

SIMULATION-BASED. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, n and t be integer functions, $\mathcal{M}$ be an n-message sampler, and R be a relation. Game SEMSOREAL gives an adversary A a single oracle **Corrupt** and game SEMSOIDEAL gives a simulator a single oracle **Corrupt** as well; the games are found in Figure 2. A semso-adversary (resp. semso-simulator) makes a single query to its **Corrupt** oracle. The the semso-advantage of a semso-adversary A with respect to $n, t, \mathcal{M}$, R, and semso-simulator S is

$$\begin{aligned}\mathbf{Adv}^{\text{sem-so-enc}}_{A,S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) = &\Pr\left[\, \text{SEMSOREAL}^{A}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1 \,\right] \\ &- \Pr\left[\, \text{SEMSOIDEAL}^{S}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1 \,\right]\end{aligned}$$

We say that a public-key encryption scheme $\mathcal{AE}$ is SEM-SO-ENC-secure if for any valid SOA paramter pair (n, t), any efficient n-message sampler $\mathcal{M}$, any efficiently computable relation R, and any efficient semso-adversary A, there exists an efficient semso-simulator S such that the semso-advantage of A with respect to $n, t, \mathcal{M}$, R, and S is negligible in the security parameter.

In words, the games proceed as follows. In the game SEMSOREAL, the adversary A is given a public key pk and n ciphertexts $\mathbf{c}$ encrypted under public key pk. The messages corresponding to the n ciphertexts come from the joint distribution $\mathcal{M}$. The adversary can then make one query to **Corrupt** consisting of a set I of t ciphertexts. In return, the adversary receives the messages $\mathbf{m}[I]$ and randomness $\mathbf{r}[I]$ used to generate those ciphertexts. The adversary then outputs a string w to **Finalize** and the output of the game is $\mathsf{R}(\mathbf{m}, w)$, the relation applied to the message vector and adversary's output. In the game SEMSOIDEAL, a vector $\mathbf{m}$ of messages is chosen and the simulator is given only the security parameter. The simulator is allowed one query to **Corrupt**, which is a set I of t indices. In response, the simulator is given $\mathbf{m}[I]$, the messages corresponding to the index set I it queried. Finally, the simulator outputs a string w and the output of the game is $\mathsf{R}(\mathbf{m}, w)$.

```
procedure Initialize($\lambda$):                    procedure Corrupt($I$):  SEMSOREAL$_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}$
$(pk, sk) \leftarrow_{\$} \mathcal{K}(1^\lambda)$                          If $|I| \neq t$ return $\perp$
$\mathbf{m} \leftarrow_{\$} \mathcal{M}(1^\lambda)$                              Return $(\mathbf{r}[I], \mathbf{m}[I])$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow_{\$} \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}[i])$          procedure Finalize($w$):
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$             Return $\mathsf{R}(\mathbf{m}, w)$
Return $(pk, \mathbf{c})$
```

```
procedure Initialize($\lambda$):                    procedure Corrupt($I$):  SEMSOREAL$_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}$
$\mathbf{m} \leftarrow_{\$} \mathcal{M}(1^\lambda)$                              If $|I| \neq t$ return $\perp$
Return $1^\lambda$                                   Return $\mathbf{m}[I]$

                                                procedure Finalize($w$):
                                                Return $\mathsf{R}(\mathbf{m}, w)$
```

Figure 2: Security games for simulation-based definition.

4 Lossy Encryption

The main tool we use in our results is what we call a *Lossy Encryption Scheme*. Informally, a lossy encryption scheme is a public-key encryption scheme with a standard key generation algorithm (which produces 'real' keys) and a lossy key generation algorithm (which produces 'lossy' keys), such that encryptions with real keys are committing, while encryptions with lossy keys are not committing. Peikert, Vaikuntanathan, and Waters [PVW08] called such lossy keys "messy keys", for *me*ssage lo*ssy*, while defining a related notion called Dual-Mode Encryption. The notion of Lossy Encryption is also similar to Meaningful/Meaningless Encryption [KN08], formalized by Kol and Naor.

More formally, a *lossy public-key encryption scheme* $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ is a tuple of PT algorithms defined as follows. The key generation algorithm $\mathcal{K}$ takes as input the security parameter 1^λ and outputs a keypair (pk, sk); we call public keys generated by $\mathcal{K}$ real public keys. The lossy key generation algorithm $\mathcal{K}_\ell$ takes as input the security parameter and outputs a keypair (pk, sk); we call such pk lossy public keys. The encryption algorithm $\mathcal{E}$ takes as input a public key pk (either from $\mathcal{K}$ or $\mathcal{K}_\ell$) and a message m and outputs a ciphertext c. The decryption algorithm takes as input a secret key sk and a ciphertext c and outputs either a message m, or $\perp$ in the case of failure. We require the following properties from $\mathcal{AE}$:

1. *Correctness on real keys.* For all $(pk, sk) \leftarrow_{\$} \mathcal{K}$ it must be the case that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$. In other words, when the real key generation algorithm is used, the standard public-key encryption correctness condition must hold.

2. *Indistinguishability of real keys from lossy keys.* No polynomial-time adversary can distinguish between the first outputs of $\mathcal{K}$ and $\mathcal{K}_\ell$. We call the advantage of an adversary A distinguishing between the two the los-key-advantage of A and take it to mean the obvious thing, i.e., the probability that A outputs 1 when given the first output of $\mathcal{K}$ is about the same as the probability it outputs 1 when given the first output of $\mathcal{K}_\ell$.

3. *Lossiness of encryption with lossy keys.* For any $(pk, sk) \leftarrow \mathcal{K}_\ell$ and two distinct messages m_0, m_1, it must be the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. We say the advantage of an adversary A in distinguishing between the two is the los-ind advantage of A and take it to mean the advantage of A in the standard ind-cpa game *when the public key pk in the ind-cpa game is lossy*. Notice that because the ciphertexts are *statistically* close, even an unbounded distinguisher will have low advantage. We sometimes call ciphertexts created with lossy public keys *lossy ciphertexts*.

4. *Possible to claim any plaintext.* There exists a (possibly unbounded) algorithm Opener that, given a lossy public key pk_ℓ, message m, and ciphertext $c = \mathcal{E}(pk_\ell, m)$, will output $r' \in_R \mathsf{Coins}_{\mathcal{E}}(pk_\ell, m)$ such that $\mathcal{E}(pk_\ell, m; r') = c$. In other words, Opener will find correctly distributed randomness to open a lossy ciphertext to the plaintext it encrypts. It then directly follows from the lossiness of encryption that with high probability the opener algorithm can successfully open *any* ciphertext to *any* plaintext.

We note that the fourth property is already implied by the first three properties; the canonical (inefficient) Opener algorithm will, given pk_ℓ, m, and c, simply try all possible coins to find the set of all r such that $\mathcal{E}(pk_\ell, m; r) = c$ and output a random element of that set. Nevertheless, we explicitly include the property because it is convenient in the proofs, and later we will consider variations of the definition which consider other (more efficient) opener algorithms.

We also note that the definition of lossy encryption already implies IND-CPA. We next provide two instantiations of lossy public-key encryption, one from DDH and one from lossy trapdoor functions.

4.1 Instantiation from DDH

We now describe a lossy public-key encryption scheme based on the DDH assumption. Recall that the DDH assumption for cyclic group $\mathbb{G}$ of order prime p says that for random generator $g \in \mathbb{G}^*$ (we use $\mathbb{G}^*$ to denote the generators of $\mathbb{G}$), the tuples (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are computationally indistinguishable, where $a, b, c \leftarrow_{\$} \mathbb{Z}_p$.

The scheme we describe below is originally from [NP01], yet some of our notation is taken from the similar dual-mode encryption scheme of [PVW08]. The scheme has structure similar to ElGamal.

Let $\mathbb{G}$ be a prime order group of order prime p. The scheme $\mathcal{AE}_{\mathsf{ddh}} = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ is a tuple of polynomial-time algorithms defined as follows:

```
Algorithm $\mathcal{K}(1^\lambda)$
$g \leftarrow_{\$} \mathbb{G}^*;\ x, r \leftarrow_{\$} \mathbb{Z}_p$
$pk \leftarrow (g, g^r, g^x, g^{rx})$
$sk \leftarrow x$
Return $(pk, sk)$
```

```
Algorithm $\mathcal{E}(pk, m)$
$(g, h, g', h') \leftarrow pk$
$(u, v) \leftarrow_{\$} \mathsf{Rand}(g, h, g', h')$
Return $(u, v \cdot m)$
```

```
Algorithm $\mathcal{D}(sk, c)$
$(c_0, c_1) \leftarrow c$
Return $c_1 / c_0^{sk}$
```

```
Algorithm $\mathcal{K}_\ell(1^\lambda)$
$g \leftarrow_{\$} \mathbb{G}^*;\ r, x \neq y \leftarrow_{\$} \mathbb{Z}_p$
$pk \leftarrow (g, g^r, g^x, g^{ry})$
$sk \leftarrow \perp$
Return $(pk, sk)$
```

```
Subroutine $\mathsf{Rand}(g, h, g', h')$
$s, t \leftarrow_{\$} \mathbb{Z}_p$
$u \leftarrow g^s h^t;\ v \leftarrow (g')^s (h')^t$
Return $(u, v)$
```

We show that $\mathcal{AE}_{\mathsf{ddh}}$ satisfies the four properties of lossy encryption schemes.

1. *Correctness on real keys.* To see the correctness property is satisfied, consider a (real) public key $pk = (g, g^r, g^x, g^{rx})$ and corresponding secret key $sk = x$. Then, for some message $m \in \mathbb{G}$

$$\begin{aligned} \mathcal{D}(sk, \mathcal{E}(pk, m)) &= \mathcal{D}(sk, (g^{s+rt}, g^{xs+rxt} \cdot m)) \\ &= (g^{xs+rxt} \cdot m)/(g^{s+rt})^x \\ &= m \end{aligned}$$

2. *Indistinguishability of real keys from lossy keys.* This follows from the assumption that DDH is hard in the groups we are using, since the first output of $\mathcal{K}$ is (g, g^r, g^x, g^{rx}) and the first output of $\mathcal{K}_\ell$ is (g, g^r, g^x, g^{ry}) for $y \neq x$.

3. *Lossiness of encryption with lossy keys.* We need to show that for any lossy public key pk generated by $\mathcal{K}_\ell$, and any messages $m_0 \neq m_1 \in \mathbb{G}$, it is the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. The results of Peikert, Vaikuntanathan, and Waters can be applied here (specifically Lemma 4 from their paper [PVW08]). We repeat their lemma for completeness.

   Lemma 4.1 [Lemma 4 from [PVW08]] Let $\mathbb{G}$ be an arbitrary multiplicative group of prime order p. For each $x \in \mathbb{Z}_p$, define $\mathsf{DLOG}_{\mathbb{G}}(x) = \{(g, g^x) \ : \ g \in \mathbb{G}\}$. There is a probabilistic algorithm Rand that takes generators $g, h \in \mathbb{G}$ and elements $g', h' \in \mathbb{G}$, and outputs a pair $(u, v) \in \mathbb{G}^2$ such that:

   - If $(g, g'), (h, h') \in \mathsf{DLOG}_{\mathbb{G}}(x)$ for some x, then (u, v) is uniformly random in $\mathsf{DLOG}_{\mathbb{G}}(x)$.
   - If $(g, g') \in \mathsf{DLOG}_{\mathbb{G}}(x)$ and $(h, h') \in \mathsf{DLOG}_{\mathbb{G}}(y)$ for $x \neq y$, then (u, v) is uniformly random in $\mathbb{G}^2$.

   The Rand procedure mentioned in the lemma is exactly our Rand procedure defined above. As [PVW08] proves, this lemma shows that encryptions under a lossy key are statistically close, since such encryptions are just pairs of uniformly random group elements.

4. *Possible to claim any plaintext.* The unbounded algorithm Opener is simply the canonical opener mentioned above. Specifically, on input lossy public key $pk = (g, h, g', h')$, message $m \in \mathbb{G}$, and ciphertext $(c_1, c_2) \in \mathbb{G}^2$, it computes the set of all $s, t \in \mathbb{Z}_p$ such that $\mathsf{Rand}(g, h, g', h'; s, t)$ outputs $(c_1, c_2/m)$. It then outputs a random element of this set.

4.2 Instantiation from Lossy TDFs

Before giving our scheme we will recall a few definitions.

Definition 4.2 [Pairwise Independent Function Family] A family of functions $\mathcal{H}_{n,m}$ from $\{0,1\}^n$ to $\{0,1\}^m$ is *pairwise-independent* if for any distinct $x, x' \in \{0,1\}^n$ and any $y, y \in \{0,1\}^m$,

$$\Pr_{h \leftarrow\$ \mathcal{H}_{n,m}} [h(x) = y \wedge h(x') = y'] = \frac{1}{2^{2m}}.$$

For our results, we make use of *lossy trapdoor functions*, a primitive recently introduced by Peikert and Waters [PW08]. Informally, a lossy trapdoor function is similar to a traditional injective trapdoor function, but with the extra property that the trapdoor function is indistinguishable from another function that loses information about its input. We recall the definition from Peikert and Waters (with minor notational changes):

Definition 4.3 [Collection of (n, k) Lossy Trapdoor Functions] Let λ be a security parameter, $n = n(\lambda) = \mathrm{poly}(\lambda)$, and $k = k(\lambda) \leq n$. A *collection of* (n, k)*-lossy trapdoor functions* $\mathcal{L}_{n,k} = (S_{\mathsf{tdf}}, S_{\mathsf{loss}}, F_{\mathsf{tdf}}, F_{\mathsf{tdf}}^{-1})$ is a tuple of algorithms with the following properties:

1. *Easy to sample, compute, and invert given a trapdoor, an injective trapdoor function.* The sampler S_{tdf}, on input 1^λ outputs (s, t), algorithm F_{tdf}, on input index s and some point $x \in \{0,1\}^n$, outputs $f_s(x)$, and algorithm F_{tdf}^{-1}, on input t and y outputs $f_s^{-1}(y)$.

2. *Easy to sample and compute lossy functions.* Algorithm S_{loss}, on input 1^λ, outputs $(s, \perp)$, and algorithm F_{tdf}, on input index s and some point $x \in \{0,1\}^n$, outputs $f_s(x)$, and the image size of f_s is at most $2^r = 2^{n-k}$.

3. *Difficult to distinguish between injective and lossy.* The function indices outputted by the sampling algorithms S_{tdf} and S_{loss} should be computationally indistinguishable. We say the advantage of distinguishing between the indices is the ltdf-advantage.

We now describe an instantiation of lossy encryption based on lossy trapdoor functions.

Let λ be a security parameter and let $(S_{\mathsf{tdf}}, S_{\mathsf{loss}}, F_{\mathsf{tdf}}, F_{\mathsf{tdf}}^{-1})$ define a collection of (n, k)-lossy trapdoor functions. Also let $\mathcal{H}$ be a collection of pair-wise independent hash functions from n bits to ℓ bits; the message space of the cryptosystem will then be $\{0,1\}^\ell$. The parameter ℓ should be such that $\ell \leq k - 2\log(1/\delta)$, where δ is a negligible function in the security parameter λ. The scheme $\mathcal{AE}_\ell = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ is then defined as follows:

```
Algorithm $\mathcal{K}(1^\lambda)$
$(s,t) \leftarrow_\$ S_{\mathsf{tdf}}(1^\lambda)$
$h \leftarrow_\$ \mathcal{H}$
$pk \leftarrow (s,h); sk \leftarrow (t,h)$
Return $(pk, sk)$

Algorithm $\mathcal{E}(pk, m)$
$(s,h) \leftarrow pk$
$x \leftarrow_\$ \{0,1\}^n$
$c_1 \leftarrow F_{\mathsf{tdf}}(s,x)$
$c_2 \leftarrow m \oplus h(x)$
Return $(c_1, c_2)$

Algorithm $\mathcal{D}(sk, c)$
$(t,h) \leftarrow sk$
$(c_1, c_2) \leftarrow c$
$x \leftarrow F_{\mathsf{tdf}}^{-1}(t, c_1)$
Return $h(x) \oplus c_2$
```

The $\mathcal{K}_\ell$ algorithm is simply the same as $\mathcal{K}$, but using S_{loss} instead of S_{tdf}. (In this case, the trapdoor t will be $\perp$.)

We now show that $\mathcal{AE}_\ell$ satisfies the four properties of lossy encryption schemes.

1. *Correctness on real keys.* This follows since when $pk = (s,h)$ was generated by $\mathcal{K}$, s is such that $(s,t) \leftarrow_\$ S_{\mathsf{tdf}}(1^\lambda)$ and $h \leftarrow_\$ \mathcal{H}$ so that

$$\begin{aligned}\mathcal{D}(sk, \mathcal{E}(pk,m)) &= h(F_{\mathsf{tdf}}^{-1}(t, F_{\mathsf{tdf}}(s,x))) \oplus (m \oplus h(x)) \\ &= h(x) \oplus m \oplus h(x) \\ &= m\end{aligned}$$

2. *Indistinguishability of real keys from lossy keys.* We need to show that any efficient adversary has low los-key advantage in distinguishing between a real public key (s,h) and a lossy key (s', h'), where $(s,h) \leftarrow_\$ \mathcal{K}(1^\lambda)$ and $(s',h') \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$. Since s is the first output of S_{tdf} and s' is the first output of S_{loss}, we use the third property of lossy trapdoor functions, specifically that the function indices outputted by S_{tdf} and S_{loss} are computationally indistinguishable.

3. *Lossiness of encryption with lossy keys.* We need to show that for any lossy public key pk generated by $\mathcal{K}_\ell$, and any messages $m_0 \neq m_1 \in \{0,1\}^\ell$, it is the case that $\mathcal{E}(pk, m_0) \equiv_s \mathcal{E}(pk, m_1)$. As Peikert and Waters show in [PW08], this is true because of the lossiness of f_s (where s is part of pk, generated by S_{loss}). Specifically, they show that the average min-entropy $\tilde{\mathrm{H}}_\infty(x|(c_1, pk))$ of the random variable x, given $f_s(x)$ and pk is at least k, and since $\ell \leq k - 2\log(1/\delta)$, it follows that $h(x)$ will be δ-close to uniform and $m_b \oplus h(x)$ will also be δ-close to uniform for either bit b.

4. *Possible to claim any plaintext.* Again, the opener is simply the canonical opener that is guaranteed to be correct by the first three properties. Specifically, the (unbounded) algorithm Opener, on input a public key $pk = (s,h)$, message $m' \in \{0,1\}^\ell$, and ciphertext $c = (c_1, c_2) = (f_s(x), h(x) \oplus m)$ for some $x \in \{0,1\}^n$ and $m \in \{0,1\}^\ell$, must output $x' \in \{0,1\}^n$ such that $f_s(x') = c_1$ and $h(x') \oplus m' = c_2$. To do so, Opener enumerates over all $\{0,1\}^n$ and creates a set $X = \{x' \in \{0,1\}^n \ : \ f_s(x') = c_1 \wedge h(x') = m' \oplus c_2\}$ before returning a random $x \in X$.

4.3 An Extension: Efficient Opening

Recall that in the above definition of lossy encryption, the Opener algorithm could be unbounded. We will now consider a refinement of the definition that will be useful for achieving the simulation-based selective opening definition. We say that a PKE scheme $\mathcal{AE}$ is a *lossy encryption scheme with efficient opening* if it satisfies the following four properties:

1. *Correctness on real keys.* For all $(pk, sk) \leftarrow\$ \mathcal{K}$ it must be the case that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$.

2. *Indistinguishability of real keys from lossy keys.* No polynomial-time adversary can distinguish between the first outputs of $\mathcal{K}$ and $\mathcal{K}_\ell$.

3. *Lossiness of encryption with lossy keys.* For any $(pk_\ell, sk_\ell) \leftarrow \mathcal{K}_\ell$ and two distinct messages m_0, m_1, it must be the case that $sk_\ell, \mathcal{E}(pk_\ell, m_0) \equiv_s sk_\ell, \mathcal{E}(pk_\ell, m_1)$. We call the advantage of an adversary in the ind-cpa game when given the lossy public key and lossy secret key the $\mathrm{los\text{-}ind2}$ advantage.

4. *Possible to efficiently claim any plaintext.* There exists an efficient algorithm Opener that on input lossy keys sk_ℓ and pk_ℓ, message m, and ciphertext $c = \mathcal{E}(pk_\ell, m)$, outputs an $r' \in_R \mathsf{Coins}_{\mathcal{E}}(pk_\ell, m)$ such that $\mathcal{E}(pk_\ell, m; r') = c$. It again follows from the above properties that Opener can open ciphertexts to arbitrary plaintexts with high probability.

We emphasize that it is important for the opener algorithm to take as input the lossy secret key. This may seem strange, since in the two schemes described above the lossy secret key was simply $\perp$, but this need not be the case. We also emphasize that the third property is slightly modified to allow the adversary access to the lossy secret key when trying to distinguish ciphertexts.

4.4 The GM Probabilistic Encryption Scheme

The Goldwasser-Micali Probabilistic encryption scheme [GM84] is an example of a lossy encryption scheme with efficient opening. We briefly recall the GM scheme. Let Par be an algorithm that efficiently chooses two large random primes p and q and outputs them along with their product N. Let $\mathcal{J}_p(x)$ denote the Jacobi symbol of x modulo p. We denote by QR_N the group of quadratic residues modulo N and we denote by QNR_N^{+1} the group of quadratic non-residues x such that $\mathcal{J}_N(x) = +1$. Recall that the security of the GM scheme is based on the Quadratic Residuosity Assumption, which states that it is difficult to distinguish a random element of QR_N from a random element of QNR_N^{+1}. The scheme $\mathcal{AE}_{GM} = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ is defined as follows.

```
Algorithm $\mathcal{K}(1^\lambda)$
$(N, p, q) \leftarrow\$ \mathsf{Par}(1^\lambda)$
$x \leftarrow\$ \mathsf{QNR}_N^{+1}$
$pk \leftarrow (N, x)$
$sk \leftarrow (p, q)$
Return $(pk, sk)$
```

```
Algorithm $\mathcal{E}(pk, m)$
$(N, x) \leftarrow pk$
For $i = 1$ to $|m|$
  $r_i \leftarrow\$ \mathbb{Z}_N^*$
  $\mathbf{c}[i] \leftarrow r_i^2 \cdot x^{m_i} \bmod N$
Return $\mathbf{c}$
```

```
Algorithm $\mathcal{D}(sk, \mathbf{c})$
$(p, q) \leftarrow sk$
For $i = 1$ to $|\mathbf{c}|$
  If $\mathcal{J}_p(\mathbf{c}[i]) = \mathcal{J}_q(\mathbf{c}[i]) = +1$
    $m_i \leftarrow 0$
  Else $m_i \leftarrow 1$
Return $m$
```

The algorithm $\mathcal{K}_\ell$ is the same as $\mathcal{K}$ except that x is chosen at random from QR_N instead of QNR_N^{+1}; in the lossy case the secret key is still the factorization of N.

It is easy to see that the scheme $\mathcal{AE}_{GM}$ meets the first three properties of lossy PKE schemes with efficient opening: the correctness of the scheme under real keys was shown in [GM84], the indistinguishability of real keys from lossy keys follows directly from the Quadratic Residuosity Assumption, and encryptions under lossy keys are lossy since in that case all ciphertexts are just sequences of random quadratic residues.

We claim that $\mathcal{AE}_{GM}$ is also efficiently openable. To see this consider the (efficient) algorithm Opener that takes as input secret key $sk = (p, q)$, public key $pk = (N, x)$, plaintext m, and encryption $\mathbf{c}$. For simplicity, say m has length n bits. For each $i \in [n]$, Opener uses p and q to efficiently compute the four square roots of $\mathbf{c}[i]/x^{m_i}$ and lets $\mathbf{r}[i]$ be a randomly chosen one of the four. The output of Opener is the sequence $\mathbf{r}$, which is just a sequence of random elements in $\mathbb{Z}_N^*$.

5 SOA-Security from Lossy Encryption

We now state our main results for encryption: any lossy public-key encryption scheme is IND-SO-ENC-secure, and any lossy public-key encryption scheme with efficient opening is SEM-SO-ENC-secure.

Theorem 5.1 [Lossy Encryption implies IND-SO-ENC security] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ be any *lossy public-key encryption scheme*, (n, t) any valid SOA parameters, $\mathcal{M}$ any efficient n-message sampler that supports efficient resampling, and A be any efficient indso-adversary. Then, there exists an unbounded los-ind adversary C and an efficient los-key adversary B such that

$$\mathbf{Adv}^{\text{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) \le 2n \cdot \mathbf{Adv}^{\text{los-ind}}_{C,\mathcal{AE}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) \ .$$

Proof: Without loss consider an indso-adversary A that never makes a query to **Corrupt** that results in $\perp$. We will prove the theorem using a sequence of game transitions. We start with a game that is simply the game INDSO run with A, and end with a game in which A has no advantage, showing that each subsequent game is either computationally or statistically indistinguishable from the previous game. Now, we know that

$$\mathbf{Adv}^{\text{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) = 2 \cdot \Pr\left[\, \text{INDSO}^A_{\mathcal{AE},\mathcal{M},n,t}(\lambda) \Rightarrow \mathsf{true} \,\right] - 1$$

by the definition of IND-SO-ENC-security (see Section 3.2). We let G_0 be the game INDSO. We will now explain the game transitions.

G_1: The only change from G_0 is that **Initialize** uses the lossy key generation, and thus the adversary is given a lossy public key and lossy ciphertexts.

H_0: In the **Corrupt** procedure, instead of opening the ciphertexts corresponding to index I by revealing the actual coins used to generate the ciphertexts, H_0 runs the Opener algorithm on the actual messages and ciphertexts and returns the output. By the definition of the Opener algorithm (see Section 4), the coins will still be correctly distributed and consistent with the ciphertexts.

H_j: We generalize H_0 with a sequence of hybrid games. In the jth hybrid game, the first j ciphertexts returned by **Initialize** are encryptions of dummy messages instead of the first j messages outputted by $\mathcal{M}$. Yet, in **Corrupt** the game still opens the ciphertexts *to the actual messages produced by* $\mathcal{M}$ by using the Opener algorithm.

H_n: In the last hybrid game, **Initialize** returns encryptions of only the dummy message, yet **Corrupt** returns openings of the ciphertexts to the actual messages generated by $\mathcal{M}$ (again, by using the Opener algorithm).

$G_\star$: The same as H_n except the choice of $\mathbf{m}_0$ and the bit b are moved to the **Corrupt** procedure since they are no longer needed in **Initialize**.

More detailed code for the games can be found in Figure 3. Now, we first claim that there is an efficient adversary B such that

$$\Pr\left[\, G_0^A \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_1^A \Rightarrow \mathsf{true} \,\right] = \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) \ . \qquad (1)$$

To see this consider a B that is given a challenge public key pk^* and must decide whether or not it is lossy. The adversary uses the IND-SO-ENC-adversary A and executes exactly the same as G_0 and G_1, giving the adversary the challenge key pk^* and ciphertexts generated using pk^*. It is important for the conditional resamplability of $\mathcal{M}$ to be efficient in order for adversary B to be efficient.

Next, we claim that

$$\Pr\left[\, G_1^A \Rightarrow \mathsf{true} \,\right] = \Pr\left[\, H_0^A \Rightarrow \mathsf{true} \,\right] \; . \tag{2}$$

Recall that H_0 opens ciphertexts $\mathbf{c}[i] = \mathcal{E}(pk, \mathbf{m}_0[i])$ by using the Opener procedure. The key point is that in H_0, $\mathbf{c}[i]$ is still opened to $\mathbf{m}_0[i]$. This ensures us that Opener will always succeed in finding coins that open the ciphertext correctly, and ensures us that the output of Opener is identically distributed to the actual coins used to encrypt $\mathbf{m}$. Thus, the claim follows.

We can now use a standard hybrid argument to show there is an *unbounded* adversary C such that

$$\Pr\left[\, H_0^A \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, H_n^A \Rightarrow \mathsf{true} \,\right] \leq n \cdot \mathbf{Adv}^{\text{los-ind}}_{C,\mathcal{AE}}(\lambda) \; . \tag{3}$$

Adversary C, on input a lossy public key pk^*, will operate the same as H_j (for some guess j) except that it will use the challenge key, and for the jth ciphertext it will use the result of issuing an IND-CPA challenge consisting of the dummy message $\mathbf{m}_{dum}$ and the real message $\mathbf{m}_0[j]$. The adversary C needs to be unbounded because it runs the (possibly inefficient) procedure Opener. With standard IND-CPA, the unbounded nature of C would be problematic. However, in the case of lossy encryption, the encryptions of two lossy ciphertexts are *statistically close* instead of just computationally indistinguishable, so C will still have only negligible advantage.

It is easy to see that

$$\Pr\left[\, H_n^A \Rightarrow \mathsf{true} \,\right] = \Pr\left[\, G_\star^A \Rightarrow \mathsf{true} \,\right] \; ,$$

since $G_\star$ is just a syntactic rewrite of H_n where the choice of messages and challenge bit are moved to the **Corrupt** procedure. They can be moved because **Initialize** no longer uses them.

Finally, we claim that

$$\Pr\left[\, G_\star^A \Rightarrow \mathsf{true} \,\right] = 1/2 \; , \tag{4}$$

which is true since in $G_\star$ the adversary is given encryptions of dummy messages and the challenge bit is not even chosen until **Corrupt**.

Combining the above equations, we see that

$$\mathbf{Adv}^{\text{ind-so-enc}}_{A,\mathcal{AE},\mathcal{M},n,t}(\lambda) \leq 2n \cdot \mathbf{Adv}^{\text{los-ind}}_{C,\mathcal{AE}}(\lambda) + 2 \cdot \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) \; ,$$

which proves the theorem. ▌

Theorem 5.2 [Lossy Encryption with Efficient Opening implies SEM-SO-ENC security] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{K}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ be any *lossy public-key encryption scheme with efficient opening*, (n, t) any valid SOA parameter pair, $\mathcal{M}$ any efficient n-message sampler, R an efficiently computable relation, and A be any polynomial-time semso-adversary. Then, there exists an efficient semso-simulator S, an efficient los-key adversary B, and an unbounded los-ind2 adversary C such that

$$\mathbf{Adv}^{\text{sem-so-enc}}_{A,S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \leq \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) + n \cdot \mathbf{Adv}^{\text{los-ind2}}_{C,\mathcal{AE}}(\lambda) \; .$$

Proof: The proof of Theorem 5.2 is very similar to the proof of Theorem 5.1. Let $\mathcal{AE}$ be a lossy encryption scheme with efficient opening, (n, t) be arbitrary valid SOA parameters, $\mathcal{M}$ an efficient n-message sampler, R an efficiently computable relation, and A be some poly-time semso-adversary. We will prove the theorem

```
$G_1$
procedure Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathcal{K}_\ell(1^\lambda)$
$\mathbf{m}_0 \leftarrow\$ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_0[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_0[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$

procedure Corrupt($I$):
$\mathbf{m}_1 \leftarrow\$ \mathcal{M}|_{I, \mathbf{m}_0[I]}$
Return $(\mathbf{r}[I], \mathbf{m}_b)$

procedure Finalize($b'$):
Return $(b = b')$
```

```
$H_0$
procedure Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathcal{K}_\ell(1^\lambda)$
$\mathbf{m}_0 \leftarrow\$ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_0[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_0[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$

procedure Corrupt($I$):
$\mathbf{m}_1 \leftarrow\$ \mathcal{M}|_{I, \mathbf{m}_0[I]}$
For $i \in I$ do
    $\mathbf{r}'[i] \leftarrow\$ \mathsf{Opener}(pk, \mathbf{m}_0[i], \mathbf{c}[i])$
Return $(\mathbf{r}'[I], \mathbf{m}_b)$

procedure Finalize($b'$):
Return $(b = b')$
```

```
$H_j$
procedure Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathcal{K}_\ell(1^\lambda)$
$\mathbf{m}_0 \leftarrow\$ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1, \ldots, n(\lambda)$ do
    If $i \leq j$ then
        $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_{dum}[i])$
        $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$
    Else
        $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_0[i])$
        $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_0[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$

procedure Corrupt($I$):
$\mathbf{m}_1 \leftarrow\$ \mathcal{M}|_{I, \mathbf{m}_0[I]}$
For $i \in I$ do
    $\mathbf{r}'[i] \leftarrow\$ \mathsf{Opener}(pk, \mathbf{m}_0[i], \mathbf{c}[i])$
Return $(\mathbf{r}'[I], \mathbf{m}_b)$

procedure Finalize($b'$):
Return $(b = b')$
```

```
$H_n$
procedure Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathcal{K}_\ell(1^\lambda)$
$\mathbf{m}_0 \leftarrow\$ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_{dum}[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$

procedure Corrupt($I$):
$\mathbf{m}_1 \leftarrow\$ \mathcal{M}|_{I, \mathbf{m}_0[I]}$
For $i \in I$ do
    $\mathbf{r}'[i] \leftarrow\$ \mathsf{Opener}(pk, \mathbf{m}_0[i], \mathbf{c}[i])$
Return $(\mathbf{r}'[I], \mathbf{m}_b)$

procedure Finalize($b'$):
Return $(b = b')$
```

```
$G_\star$
procedure Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathcal{K}_\ell(1^\lambda)$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow\$ \mathsf{Coins}_{\mathcal{E}}(pk, \mathbf{m}_{dum}[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$

procedure Corrupt($I$):
$\mathbf{m}_0 \leftarrow\$ \mathcal{M}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
$\mathbf{m}_1 \leftarrow\$ \mathcal{M}|_{I, \mathbf{m}_0[I]}$
For $i \in I$ do
    $\mathbf{r}'[i] \leftarrow\$ \mathsf{Opener}(pk, \mathbf{m}_0[i], \mathbf{c}[i])$
Return $(\mathbf{r}'[I], \mathbf{m}_b)$

procedure Finalize($b'$):
Return $(b = b')$
```

Figure 3: Game transitions for proof of Theorem 5.1

using a sequence of game transitions. We will start with the game SEMSOREAL run with A and eventually end up with an efficient semso-simulator S in game SEMSOIDEAL. Let G_0 be the game SEMSOREAL and consider the following sequence of games:

G_1: The only change from G_0 is that **Initialize** uses the lossy key generation, and thus the adversary is given a lossy public key and lossy ciphertexts.

H_0: In the **Corrupt** procedure, instead of opening the ciphertexts corresponding to index I by revealing the actual coins used to generate the ciphertexts, H_0 runs the Opener algorithm on the actual messages and ciphertexts and returns the output. By the definition of the Opener algorithm (see Section 4), the coins will still be correctly distributed and consistent with the ciphertexts.

H_j: We generalize H_0 with a sequence of hybrid games. In the jth hybrid game, the first j ciphertexts returned by **Initialize** are encryptions of dummy messages instead of the first j messages outputted by $\mathcal{M}$. Yet, in **Corrupt** the game still opens the ciphertexts *to the actual messages produced by* $\mathcal{M}$ by using the Opener algorithm.

H_n: In the last hybrid game, **Initialize** returns encryptions of only the dummy message, yet **Corrupt** returns openings of the ciphertexts to the actual messages generated by $\mathcal{M}$ (again, by using the Opener algorithm).

$G_\star$: The same as H_n except the choice of $\mathbf{m}$ is moved to the **Corrupt** procedure since the vector is no longer needed in **Initialize**.

More detailed descriptions of the games can be found in Figure 2. Now, by definition we know that

$$\Pr\left[\,\mathrm{SEMSOREAL}^{A}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1\,\right] = \Pr\left[\,G_0^A \Rightarrow 1\,\right] .$$

Our first claim is that there is an efficient adversary B such that

$$\Pr\left[\,G_0^A \Rightarrow 1\,\right] - \Pr\left[\,G_1^A \Rightarrow 1\,\right] \le \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) .$$

This is easy to see because the adversary B simply runs A exactly as in games G_0 and G_1, but instead of generating its own key pair, B substitutes its challenge key. Now, we claim that

$$\Pr\left[\,G_1^A \Rightarrow 1\,\right] = \Pr\left[\,H_0^A \Rightarrow 1\,\right] . \tag{5}$$

The claim follows for the same reasons given in the proof of Theorem 5.1 above. Also similar to that proof, we can use a standard hybrid argument to show there is an *unbounded* adversary C such that

$$\Pr\left[\,H_0^A \Rightarrow 1\,\right] - \Pr\left[\,H_n^A \Rightarrow 1\,\right] \le n \cdot \mathbf{Adv}^{\text{los-ind2}}_{C,\mathcal{AE}}(\lambda) . \tag{6}$$

The only difference from the above proof is that now we need to bound the difference by the los-ind2 advantage of C, since C will have access to the lossy secret key.

We can now rewrite game H_n and move the choice of the message vector $\mathbf{m}$ to the **Corrupt** procedure. This is because **Initialize** only uses the dummy message vector in H_n. We call the resulting game $G_\star$. It is clear that

$$\Pr\left[\,H_n^A \Rightarrow 1\,\right] = \Pr\left[\,G_\star^A \Rightarrow 1\,\right] .$$

Lastly, we claim that there exists a semso-simulator S such that

$$\Pr\left[\,\mathrm{SEMSOIDEAL}^{S}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1\,\right] = \Pr\left[\,G_\star^A \Rightarrow 1\,\right] .$$

```
procedure Initialize($\lambda$):                         procedure Corrupt($I$):        $G_1$
$(pk, sk) \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$                        Return $(\mathbf{r}[I], \mathbf{m}[I])$
$\mathbf{m} \leftarrow_\$ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, n(\lambda)$ do                        procedure Finalize($w$):
    $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}[i])$        Return $\mathsf{R}(\mathbf{m}, w)$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$
```

```
procedure Initialize($\lambda$):                         procedure Corrupt($I$):        $H_0$
$(pk, sk) \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$                        For $i \in I$ do
$\mathbf{m} \leftarrow_\$ \mathcal{M}(1^\lambda)$                               $\mathbf{r}'[i] \leftarrow_\$ \mathsf{Opener}(sk, pk, \mathbf{m}[i], \mathbf{c}[i])$
For $i = 1, \ldots, n(\lambda)$ do                        Return $(\mathbf{r}'[I], \mathbf{m}[I])$
    $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$        procedure Finalize($w$):
Return $(pk, \mathbf{c})$                                 Return $\mathsf{R}(\mathbf{m}, w)$
```

```
procedure Initialize($\lambda$):                         procedure Corrupt($I$):        $H_j$
$(pk, sk) \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$                        For $i \in I$ do
$\mathbf{m} \leftarrow_\$ \mathcal{M}(1^\lambda)$                               $\mathbf{r}'[i] \leftarrow_\$ \mathsf{Opener}(sk, pk, \mathbf{m}[i], \mathbf{c}[i])$
For $i = 1, \ldots, n(\lambda)$ do                        Return $(\mathbf{r}'[I], \mathbf{m}[I])$
    If $i \leq j$ then
        $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}_{dum}[i])$    procedure Finalize($w$):
        $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$    Return $\mathsf{R}(\mathbf{m}, w)$
    Else
        $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}[i])$
        $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}[i]; \mathbf{r}[i])$
Return $(pk, \mathbf{c})$
```

```
procedure Initialize($\lambda$):                         procedure Corrupt($I$):        $H_n$
$(pk, sk) \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$                        For $i \in I$ do
$\mathbf{m} \leftarrow_\$ \mathcal{M}(1^\lambda)$                               $\mathbf{r}'[i] \leftarrow_\$ \mathsf{Opener}(sk, pk, \mathbf{m}[i], \mathbf{c}[i])$
For $i = 1, \ldots, n(\lambda)$ do                        Return $(\mathbf{r}'[I], \mathbf{m}[I])$
    $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}_{dum}[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$    procedure Finalize($w$):
Return $(pk, \mathbf{c})$                                 Return $\mathsf{R}(\mathbf{m}, w)$
```

```
procedure Initialize($\lambda$):                         procedure Corrupt($I$):        $G_\star$
$(pk, sk) \leftarrow_\$ \mathcal{K}_\ell(1^\lambda)$                        $\mathbf{m} \leftarrow_\$ \mathcal{M}(1^\lambda)$
For $i = 1, \ldots, n(\lambda)$ do                        For $i \in I$ do
    $\mathbf{r}[i] \leftarrow_\$ \mathsf{Coins}_\mathcal{E}(pk, \mathbf{m}_{dum}[i])$        $\mathbf{r}'[i] \leftarrow_\$ \mathsf{Opener}(sk, pk, \mathbf{m}[i], \mathbf{c}[i])$
    $\mathbf{c}[i] \leftarrow \mathcal{E}(pk, \mathbf{m}_{dum}[i]; \mathbf{r}[i])$    Return $(\mathbf{r}'[I], \mathbf{m}[I])$
Return $(pk, \mathbf{c})$
                                                   procedure Finalize($w$):
                                                   Return $\mathsf{R}(\mathbf{m}, w)$
```

Figure 4: Game transitions for proof of Theorem 5.2

The simulator S executes A internally exactly as in $G_\star$. In particular, S chooses a lossy key pair and gives A the lossy public key and dummy encryptions. When A makes a **Corrupt** query I, S makes the same **Corrupt** query to learn $\mathbf{m}[I]$ and uses the efficient Opener algorithm to open the dummy ciphertexts to $\mathbf{m}[I]$. Finally, when A outputs w, S will output the same. Since A and Opener are efficient, S will also be efficient.

Combining all of the above equations, we see that

$$\begin{aligned}\mathbf{Adv}^{\text{sem-so-enc}}_{A,S,\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) &= \Pr\left[\,\text{SEMSOREAL}^{A}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1\,\right] \\ &\quad - \Pr\left[\,\text{SEMSOIDEAL}^{S}_{\mathcal{AE},\mathcal{M},\mathsf{R},n,t}(\lambda) \Rightarrow 1\,\right] \\ &\le \mathbf{Adv}^{\text{los-key}}_{B,\mathcal{AE}}(\lambda) + n \cdot \mathbf{Adv}^{\text{los-ind2}}_{C,\mathcal{AE}}(\lambda)\ .\end{aligned}$$

which proves the theorem. ▮

Acknowledgements

The authors would like to thank Saurabh Panjwani for participating in early stages of this work which in particular involved the development of the indistinguishability-based definition IND-SO-ENC. The authors would also like to thank Dennis Hofheinz for pointing out that the Goldwasser-Micali scheme is a lossy encryption scheme with efficient opening, and for other useful feedback.

References

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *Advances in Cryptology, Proceedings of CRYPTO 2008*, number 5157 in Lecture Notes in Computer Science, pages 335–359. Springer, 2008.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *Advances in Cryptology, Proceedings of EUROCRYPT '09*, number 5479 in Lecture Notes in Computer Science, pages 1–35. Springer, 2009.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *20th ACM Symposium on Theory of Computing, Proceedings of STOC 1988*, pages 1–10. ACM, 1988.

[BR07] Mihir Bellare and Phillip Rogaway. Robust computational secrete sharing and a unified account of classical secret-sharing goals. In *14th ACM Conference on Computer and Communications Security, Proceedings of CCS 2007*, pages 172–184. ACM Press, 2007.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *20th ACM Symposium on Theory of Computing, Proceedings of STOC 1988*, pages 11–19. ACM, 1988.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In Burton S. Kaliski Jr., editor, *Advances in Cryptology, Proceedings of CRYPTO '97*, number 1294 in Lecture Notes in Computer Science, pages 90–104. Springer-Verlag, 1997.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Twenty-Eighth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 1995*, pages 639–648. ACM Press, 1996.

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *Theory of Cryptography, Proceedings of TCC 2005*, number 3378 in Lecture Notes in Computer Science, pages 150–168. Springer-Verlag, 2005.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on general complexity assumptions. In Mihir Bellare, editor, *Advances in Cryptology, Proceedings of CRYPTO 2000*, number 1880 in Lecture Notes in Computer Science, pages 432–450. Springer-Verlag, 2000.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2), 1984.

[GM06] Rosiario Gennaro and Silvio Micali. Independent zero-knowledge sets. In Michele Bugliese, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33th International Colloquium, Proceedings of ICALP 2006*, number 4052 in Lecture Notes in Computer Science, pages 34–45. Springer-Verlag, 2006.

[Hof08] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. IACR ePrint Archive, April 2008.

[KN08] Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In Ran Canetti, editor, *Theory of Cryptography, Proceedings of TCC 2008*, number 4948 in Lecture Notes in Computer Science, pages 320–339. Springer, 2008.

[Nie02] Jesper B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology, Proceedings of CRYPTO 2002*, number 2442 in Lecture Notes in Computer Science, pages 111–126. Springer-Verlag, 2002.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Twelfth Annual Symposium on Discrete Algorithms, Proceedings of SODA 2001*, pages 448–457. ACM/SIAM, 2001.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*, number 1592 in Lecture Notes in Computer Science, pages 223–238. Springer, 1999.

[Pan07] Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In Salil Vadhan, editor, *Theory of Cryptography, Proceedings of TCC 2007*, number 4392 in Lecture Notes in Computer Science, pages 21–40. Springer, 2007.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology, Proceedings of CRYPTO 2008*, number 5157 in Lecture Notes in Computer Science, pages 554–571. Springer, 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Fortieth Annual ACM Symposium on Theory of Computing, Proceedings of STOC 2008*, pages 187–196. ACM Press, 2008.

[RS08] Alon Rosen and Gil Segev. Efficient lossy trapdoor functions based on the composite residuosity assumption. IACR ePrint Archive, March 2008.