A preliminary version of this paper appears as part of our Eurocrypt 2009 paper with Hofheinz [3]. This is an updated full version.

Encryption Schemes Secure under Selective Opening Attack

Mihir Bellare* Scott Yilek†

Abstract

We provide the first public key encryption schemes proven secure against selective opening attack (SOA). This means that if an adversary obtains a number of ciphertexts and then corrupts some fraction of the senders, obtaining not only the corresponding messages *but also the coins under which they were encrypted* then the security of the other messages is guaranteed. Whether or not schemes with this property exist has been open for many years. Our schemes are based on a primitive we call lossy encryption. Our schemes have short keys (public and secret keys of a fixed length suffice for encrypting an arbitrary number of messages), are stateless, are non-interactive, 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-SOA-C) and a simulation-style, semantic security (SS-SOA-C) 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://cseweb.ucsd.edu/~mihir Supported in part by NSF grants CNS 0524765, CNS 0627779, CCF 0915675, CNS 1116800, CNS 0904380 and a gift from Intel Corporation.

†Department of Computer and Information Sciences, University of St. Thomas, 2115 Summit Ave. Mail #OSS-402, Saint Paul, MN 55105, USA. Email: syilek@stthomas.edu. URL: http://personal.stthomas.edu/yile5901 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) [18, 12, 28, 17, 13, 10]. Let us provide some background on SOA and then discuss our results and related work.

1.1 Background

THE PROBLEM. Suppose a receiver with public encryption key ek receives a vector $\mathbf{c} = (\mathbf{c}[1], \ldots, \mathbf{c}[n])$ of ciphertexts, where sender i created ciphertext $\mathbf{c}[i] = \mathcal{E}(ek, \mathbf{m}[i]; \mathbf{r}[i])$ by encrypting a message $\mathbf{m}[i]$ under ek 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 subset $I \subseteq \{1, \ldots, n\}$ of senders, 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 SOA-C attack.[1] The security requirement is that the privacy of the unopened messages, namely $\mathbf{m}[i_1], \ldots, \mathbf{m}[i_{n-t}]$ where $\{i_1, \ldots, i_{n-t}\} = \{1, \ldots, n\} \setminus I$ and $t = |I|$, 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 soon.) The question is whether SOA-C-secure encryption schemes exist.

STATUS AND MOTIVATION. One's first impression may be that a simple hybrid argument would show that any IND-CPA scheme is SOA-C-secure. Nobody has yet been able to push such an argument through, and the question of whether IND-CPA implies SOA-C-security has remained open, neither a proof that it is true, nor a counter-example to show that it is false, appearing.[2] One might think that IND-CCA, at least, would imply SOA-C-security, but even this is not true [2]. This means that one has to look for new approaches in order to build SOA-C-secure schemes. To date, this has not been successful, and no such schemes exist. The difficulty of the problem is well understood and documented [18, 12, 13, 28, 17, 10].

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 [7, 14]. 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-C-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. Possession of the coins allows the adversary to verify that the opening is correct, since it can compute $\mathcal{E}(ek, \mathbf{m}[i]; \mathbf{r}[i])$ and check that this equals $\mathbf{c}[i]$ for all $i \in I$. This apparent commitment

[1] Here "SOA" stands for "selective opening attack" and the "C" indicates that coins are being revealed, to distinguish this from another version of SOA, denoted SOA-K. In the latter there are many receivers and one sender, and receivers are corrupted to reveal their decryption keys. Sometimes SOA-C is also referred to as SOA with sender corruptions while SOA-K is referred to as SOA with receiver corruptions.

[2] Results subsequent to ours show that IND-CPA does not imply SS-SOA-C [2]. We will discuss this and other subsequent work later.

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 [18]. 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.

1.2 Contributions

Definitions. We provide two definitions of SOA-C security. One is a simulation-based, semantic-security style definition that we call SS-SOA-C. It is based on a similar notion for commitment from [18]. We also provide an indistinguishability-based formalization that we denote IND-SOA-C. We show that SS-SOA-C implies IND-SOA-C, but whether the converse is true in general remains open. (Relations between these and other notions are discussed in more detail later.)

The formalization of SOA-secure commitment of [18] was in the above setting where the adversary gets a vector of ciphertexts all in one shot and then corrupts some fraction of the senders, again all in one shot. Our setting is more general. The adversary gets ciphertexts one by one, the distribution of the underlying message being adaptively chosen by the adversary depending on previous ciphertexts. It can make corruptions, adaptively, at any time. This extends the setting of our preliminary work and other previous work [3, 18] and better captures the needs of applications.

Main results. We provide the first public-key encryption schemes provably secure against SOA-C. Unlike non-committing encryption schemes [12, 28], ours 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 IND-SOA-C and SS-SOA-C notions of security, although, with a given assumption, we may pay in efficiency for the latter.

The main tool (that we define and employ) is lossy encryption, an encryption analogue of lossy trapdoor functions (LTDFs) [32] that is closely related to meaningful-meaningless encryption [26] and dual-mode encryption [31]. We provide lossy encryption schemes based on DDH and QR. We also show that any (sufficiently) lossy trapdoor function yields lossy encryption. Via [32, 9, 33] we thereby obtain alternative lossy encryption schemes based on DDH, as well as one's based on Paillier's DCR [29] and on LWE. We show that any lossy encryption scheme is IND-SOA-C-secure, thereby obtaining IND-SOA-C-secure schemes based on the assumptions just mentioned.

If the lossy encryption scheme has an additional property that we call efficient openability, we show that it is also SS-SOA-C-secure. We observe that the classical QR-based encryption scheme of Goldwasser and Micali [23] is lossy with efficient openability, thereby obtaining QR-based SS-SOA-C-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.) We also present a DDH-based lossy encryption scheme with efficient opening. (It less efficient than the above-mentioned DDH-based lossy encryption scheme which lacks efficient opening.) Thus we get SS-SOA-C-secure encryption from DDH as well.

Relations among notions. Having discussed our main results, namely the first SOA-C-secure encryption schemes, we back up a bit to put them into perspective and better understand the issues and definitions. Figure 1 summarizes a bigger picture of related notions. We consider SOA-M, where opening reveals messages but not coins, formulating IND-SOA-M and SS-SOA-M as analogues of IND-SOA-C and SS-SOA-C, respectively. We show that that IND-SOA-M and SS-SOA-M are equivalent to each other and to the classical IND-CPA notion. This result is expected but it provides a good pedagogic starting point for the understanding of SOA-C security, for we see that the difficulty comes from the

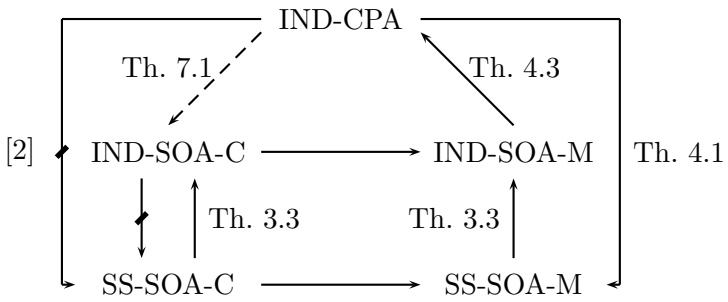

Figure 1: **Notions of security and their relations.** On the right are the two formalizations of selective opening of messages only, which are equivalent to the standard IND-CPA notion at the top. On the left are the two formalizations of selective opening of coins (and messages). The dotted arrow indicates that when encrypted messages are independently distributed, IND-CPA implies IND-SOA-C. Coupling this with the fact that the shown separation of [2] holds even for independently distributed messages means that IND-SOA-C does not imply SS-SOA-C. Unannotated arrows represent trivial implications. The question that remains open is whether IND-CPA (in general) implies IND-SOA-C.

---

coins and we also see the hybrid argument that we might expect works in the SOA-C case. In general it does not, but we use it to show that when the messages being encrypted are independently distributed, IND-CPA implies IND-SOA-C. Our proof does not extend to show that IND-CPA implies SS-SOA-C for independently-distributed messages. Indeed, [2] show that IND-CPA does not imply SS-SOA-C, even for (uniformly and) independently-distributed messages. Putting these two results together shows that IND-SOA-C does not imply SS-SOA-C, meaning the latter is strictly stronger. The major open question regarding relations is whether or not IND-CPA implies IND-SOA-C, meaning whether or not they are equivalent. Neither a proof nor a counter-example has yet emerged.

1.3 Related work

The problem of selective opening has been considered in a variety of forms in a variety of models and the precise relations between the variants have not in all cases been formally established. We attempt to overview here some of the work and likely relations.

History. Canetti, Feige, Goldreich and Naor [12] introduced non committing encryption (NCE) to achieve adaptively secure multi-party computation in the computational (as opposed to secure channels) setting without erasures. In their treatment, NCE is an interactive protocol, and their definition of security is in the MPC framework. The model allows corruption of both senders and receivers. They show how to achieve NCE but, viewed as a public-key system, they would have long keys, meaning keys longer than the total number of message bits that may be securely encrypted. More efficient NCE protocols followed [16, 15] but the restriction remained, and Nielsen [28] showed it was necessary. Canetti, Halevi and Katz [13] provide adaptively-secure encryption 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.)

Dwork, Naor, Reingold and Stockmeyer [18] extracted out a stand-alone notion of commitment secure against selective opening defined directly by a game rather than via the MPC framework. Corruptions allow the adversary to obtain the committer's coins along with its message. We denote it SEM-SO-COM. We have followed their lead and use stand-alone, game-based definitions. SS-SOA-C adapts SEM-SO-COM. Beyond the fact that it is for encryption, there are differences in details, such as the inputs to the relation. Also, in SEM-SO-COM, both the adversary and the simulator are one-shot, while we allow

the distribution of the next encrypted message to be adaptively chosen by the adversary depending on previous ciphertexts. This is more suitable for applications.

In the SOA-C version of the problem that we consider, there is one receiver and many senders. Senders may be corrupted, with the corruption exposing their coins and messages. 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. We call this SOA-K. A definition of SOA-K security analagous to our SS-SOA-C definition for SOA-C was given in [2]. We would imagine that security under the MPC definitions of the above works implies both SOA-K and SOA-C but are not aware of this claim having been formally established.

Our schemes do not suffer from any of the restrictions of the ones discussed above. We have short public and secret keys, do not rely on erasures, and achieve strong notions of security. However, we achieve only SOA-C security while the previous works target security under both sender and receiver corruptions, meaning achieve SOA-K as well.

In the symmetric setting, Panjwani [30] proves SOA-security against receiver corruptions against a limited class of attacks with an indistinguishability style definition.

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.

Canetti, Dwork, Naor and Ostrovsky [11] introduced deniable encryption, where a sender may open a ciphertext to an arbitrary message by providing coins produced by a faking algorithm. The authors explain that this is stronger than NCE because in the latter only a simulator can open in this way.

Commitment. Recall that SEM-SO-COM denotes the notion of SOA-security for commitment formalized by [18]. 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 [18] 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, [22] 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 [25] continued the investigation of SOA-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 [5].) He does not consider encryption.

The commitment results do show that the SOA-C-security of an encryption scheme cannot be proved using a black-box reduction, *but only if encryption constitutes a commitment.* But our SOA-C-secure encryption schemes do not give rise to commitment schemes. So there is no contradiction.

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 [3]. Full versions have, however, been written separately as the present paper and [25].

SUBSEQUENT WORK. Since the appearance of the preliminary version of our work [3] there has been quite a lot of activity in this area. Hemenway, Libert, Ostrovsky and Vergnaud [24] showed that re-randomizable encryption and statistically hiding, two-round oblivious transfer imply lossy encryption, yielding still more examples of SOA-C secure PKE schemes via our lossy-implies-SOA-C-secure connection. Further constructions of LTDFs were given in [20] and, via our results, yield more lossy encryption schemes. Fehr, Hofheinz, Kiltz, and Wee [19] use a deniable encryption [11] approach to achieve CC-SOA (Chosen-Ciphertext SOA-C) secure PKE. SOA-C-secure identity-based encryption was defined and achieved in [6].

A major open question in this area was whether IND-CPA implies SOA-C security. Bellare, Dowsley, Waters and Yilek [2] show that IND-CPA does not imply SS-SOA-C. They show this is true even for uniformly distributed, independent messages, and for natural and common schemes. They show a similar result for SOA-K, and they rule out non-interactive SEM-SO-COM-secure commitment altogether. Whether IND-CPA implies IND-SOA-C remains open.

2 Basic definitions

Here we recall basic notation and definitions.

NOTATION AND CONVENTIONS. If $n \in \mathbb{N}$ then we let 1^n denote the string of n ones (the unary representation of n) and $[n]$ the set $\{1, \ldots, n\}$. The empty string is denoted by ε and the length of a string x is denoted $|x|$. If a, b are strings then $a \,\|\, b$ denotes their concatenation and if $|a| = |b|$ then $a \oplus b$ denotes the bitwise xor of a and b. If a is tuple then $(a_1, \ldots, a_n) \leftarrow a$ means we parse a into its constituents. We use boldface letters for vectors. If $\mathbf{x}$ is a vector then we let $|\mathbf{x}|$ denote the number of components of $\mathbf{x}$ and for $1 \le i \le |\mathbf{x}|$ we let $\mathbf{x}[i]$ denote its i-th component. We say $\mathbf{x}$ is an n-vector if $|\mathbf{x}| = n$. If $\mathbf{x}$ is an n-vector and $I \subseteq [n]$ then $\mathbf{x}[I] = (\mathbf{x}[i_1], \mathbf{x}[i_2], \ldots, \mathbf{x}[i_l])$ where $I = \{i_1, \ldots, i_l\}$ and $i_1 < i_2 < \ldots < i_l$. We let Len() be the function that on input a vector $\mathbf{x}$ of strings returns the $|\mathbf{x}|$-vector $\mathbf{len}$ whose i-th component is $|\mathbf{x}[i]|$ (the length of the string $\mathbf{x}[i]$) for all $i \in [|\mathbf{x}|]$. If S is a (finite) set then $|S|$ denotes its size and $s \xleftarrow{\$} S$ denotes the operation of drawing s uniformly at random from S. We say a function $\mu\colon\ \mathbb{N} \to \mathsf{R}$ is negligible if $\mu \in o(n^{-\omega(1)})$.

All algorithms in this paper are randomized, unless otherwise specified as being deterministic. Let $y \leftarrow A(x_1, x_2, \ldots; r)$ denote that algorithm A is run on inputs $x_1, x_2, \ldots$ with coins r and the result is named y. Let $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ denote that we pick coins r at random and let $y \leftarrow A(x_1, x_2, \ldots; r)$. Coins are assumed drawn uniformly a random from a set that might depend on the inputs. Let $[A(x_1, x_2, \ldots)]$ denote the set of all y for which there exists r such that $y = A(x_1, x_2, \ldots; r)$. "PT" stands for "polynomial time." An algorithm is called unbounded if its running time is not necessarily polynomial. An adversary is an algorithm.

GAMES. We use the language of code-based game-playing [4]. A game (see Figure 2 for examples) has an INITIALIZE procedure, procedures to respond to adversary oracle queries, and a FINALIZE procedure. A game G is executed with an adversary A and security parameter λ as follows. A is given input 1^λ and can then call game procedures. Its first oracle query must be INITIALIZE(1^λ) and its last oracle query must be to FINALIZE, and it must make exactly one query to each of these oracles. In between it can query the other procedures as oracles as it wishes. The output of FINALIZE, denoted $\mathrm{G}^A(\lambda)$, is called the output of the game. We let "$\mathrm{G}^A(\lambda)$" denote the event that this output takes the boolean value true. The running time of an adversary is the worst case time for the execution of the adversary with the game

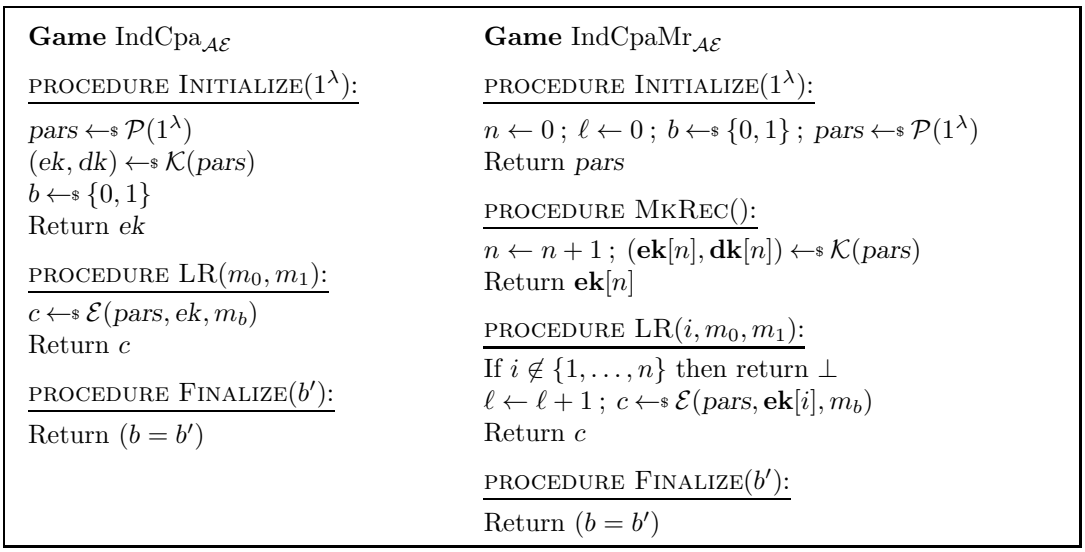

Figure 2: Games to define IND-CPA and IND-CPA-MR security of PKE scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$.

defining its security with the convention that the oracle calls count as one unit of time. The number of oracle queries of an adversary is an integer-valued function of the security parameter alone.

PKE SCHEMES. A public-key encryption scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is a 4-tuple of PT algorithms. The parameter generation algorithm $\mathcal{P}$ takes as input the security parameter 1^λ in unary and outputs a string *pars* called the parameters or public parameters. The key generation algorithm $\mathcal{K}$ takes as input *pars* and outputs a pair (ek, dk) consisting of an encryption key ek and matching decryption key dk. Associated to the scheme is a message space function $\mathsf{MsgSp}_{\mathcal{AE}}$ that, on input *pars*, returns a set of strings. The encryption algorithm $\mathcal{E}$ takes as input *pars*, an encryption key ek and a message $m \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$ and outputs a ciphertext c. The deterministic decryption algorithm takes as input *pars*, a decryption key dk and a ciphertext c and outputs either a message $m \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$ or the special non-string symbol $\perp$, denoting failure. We let $\mathsf{Coins}_{\mathcal{E}}(pars)$ be the set from which $\mathcal{E}$ draws its coins on inputs $pars, ek, m$. (The set does not depend on ek, m.) Scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is *correct* if $\mathcal{D}(pars, dk, \mathcal{E}(pars, ek, m; r)) = m$ for all $\lambda \in \mathbb{N}$, all $pars \in [\mathcal{P}(\lambda)]$, all $(ek, dk) \in [\mathcal{K}(pars)]$, all $m \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$ and all $r \in \mathsf{Coins}_{\mathcal{E}}(pars)$. Correctness is *not* a default assumption on an encryption scheme, meaning if it is not stated that a scheme is correct, it need not be. If the set $\mathsf{Coins}_{\mathcal{E}}(pars)$ is empty for all ek then we say that $\mathcal{AE}$ is a deterministic PKE scheme.

IND-CPA. Game IndCpa$_{\mathcal{AE}}$ of Figure 2 captures the basic notion of indistinguishability under chosen-plaintext attack (IND-CPA) [23]. We say that A is an ind-cpa-adversary if it makes only one LR query, this consisting of two messages in $\mathsf{MsgSp}_{\mathcal{AE}}(pars)$ that have the same length. The ind-cpa-advantage of such an adversary A is

$$\mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\lambda) = 2 \cdot \Pr\left[\, \text{IndCpa}^{A}_{\mathcal{AE}}(\lambda) \,\right] - 1 \ .$$

We say that $\mathcal{AE}$ is IND-CPA-secure if the function $\mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\cdot)$ is negligible for all PT ind-cpa-adversaries A. We also let IND-CPA denote the set of all encryption schemes $\mathcal{AE}$ that are IND-CPA-secure. This convention allows us to compactly express relations between notions as containments and non-containments between the corresponding sets. We say that $\mathcal{AE}$ is u-IND-CPA-secure if the function $\mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\cdot)$ is negligible for all (whether PT or not) ind-cpa-adversaries A, and we let u-IND-CPA $\subseteq$ IND-CPA ("u"

stands for "unbounded") denote the set of all encryption schemes $\mathcal{AE}$ that are u-IND-CPA-secure.

IND-CPA-MR. It will be useful for some of our proofs to consider the multi-user version of IND-CPA from [1], captured by game $\mathrm{IndCpaMr}_{\mathcal{AE}}$ of Figure 2. An ind-cpa-mr-adversary can create a new receiver by calling MkRec, receiving the encryption key in return. The adversary can obtain via LR an encryption of a challenge message under an encryption key of a receiver of its choice. It is required that in any such query, the two messages are in $\mathsf{MsgSp}_{\mathcal{AE}}(pars)$ and have the same length. Note that the same challenge bit is used for all receivers. The ind-cpa-mr-advantage of an adversary A is

$$\mathbf{Adv}^{\text{ind-cpa-mr}}_{A,\mathcal{AE}}(\lambda) = 2 \cdot \Pr\left[\, \mathrm{IndCpaMr}^{A}_{\mathcal{AE}}(\lambda) \,\right] - 1 \ .$$

We say that $\mathcal{AE}$ is IND-CPA-MR-secure if the function $\mathbf{Adv}^{\text{ind-cpa-mr}}_{A,\mathcal{AE}}(\cdot)$ is negligible for all PT ind-cpa-mr-adversaries A, and we let IND-CPA-MR denote the set of all encryption schemes $\mathcal{AE}$ that are IND-CPA-MR-secure. The following result of [1] says that IND-CPA = IND-CPA-MR.

Proposition 2.1 (IND-CPA-MR = IND-CPA) [1] Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let A be an ind-cpa-mr-adversary making at most $q_{\mathrm{mk}}(\cdot)$ queries to MkRec. Also assume that for each i it makes at most $q_{\mathrm{spr}}(\cdot)$ $\mathrm{LR}(i, \cdot, \cdot)$ queries. Then there exists an ind-cpa-adversary B such that for all $\lambda \in \mathbb{N}$ we have

$$\mathbf{Adv}^{\text{ind-cpa-mr}}_{A,\mathcal{AE}}(\lambda) \leq q_{\mathrm{mk}}(\lambda) \cdot q_{\mathrm{spr}}(\lambda) \cdot \mathbf{Adv}^{\text{ind-cpa}}_{B,\mathcal{AE}}(\lambda) \ .$$

The running time of B is around the same as that of A. ∎

3 Notions of SOA security

We provide a simulation-based semantic-security style formalizations of SOA security as well as an indistinguishabilty-style one. We consider both SOA-M, where only messages are opened, and SOA-C, where both messages and coins are opened. We will see that mere opening of messages, even adaptively, adds no power, the resulting notions being equivalent to IND-CPA, but opening of coins is different.

SS-SOA-X. We first give the formal definitions and then discuss them. Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. A *relation* is an algorithm $\mathcal{R}$ with boolean output. A *message sampler* is an algorithm $\mathcal{M}$ taking input *pars*, a string α and a current state to return a pair consisting of a message $m \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$ and an updated state. We consider the games of Figure 3, the first played by an adversary A and the second by a simulator S. For $(\mathrm{x}, \mathrm{X}) \in \{(\mathrm{m}, \mathrm{M}), (\mathrm{c}, \mathrm{C})\}$ we define the ss-soa-x-advantage of A relative to S via

$$\mathbf{Adv}^{\text{ss-soa-x}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) = \Pr\left[\, \mathrm{SsSoaXReal}^{A}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \,\right] - \Pr\left[\, \mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \,\right] \ .$$

We say that $\mathcal{AE}$ is SS-SOA-X-secure if for every PT adversary A, every PT message sampler $\mathcal{M}$ and every PT relation $\mathcal{R}$ there is a PT simulator S such that the function $\mathbf{Adv}^{\text{ss-soa-m}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\cdot)$ is negligible. We let SS-SOA-X denote the set of all encryption schemes $\mathcal{AE}$ that are SS-SOA-M-secure.

In games $\mathrm{SsSoaCReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ and $\mathrm{SsSoaMReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$, the adversary can create receivers via MkRec(), with receiver n getting an encryption key $\mathbf{ek}[n]$ and decryption key $\mathbf{dk}[n]$. By calling procedure Sample with a receiver index i and a string α, the adversary can get the game to generate the next message $\mathbf{m}[\ell]$ via $(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(pars, \alpha, s)$. In each invocation, $\mathcal{M}$ tosses coins, and, at its discretion, may include these in the updated state so that the state can potentially hold all the coins generated so far and the generated message may depend on these and thus in particular on previously generated messages. The procedure returns an encryption of $\mathbf{m}[\ell]$ under the encryption key of the i-th receiver. Via Corrupt(j) the adversary can obtain in game $\mathrm{SsSoaMReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ the j-th message and in game $\mathrm{SsSoaCReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ also the coins used to encrypt it. Eventually the adversary calls Finalize with some argument w of its

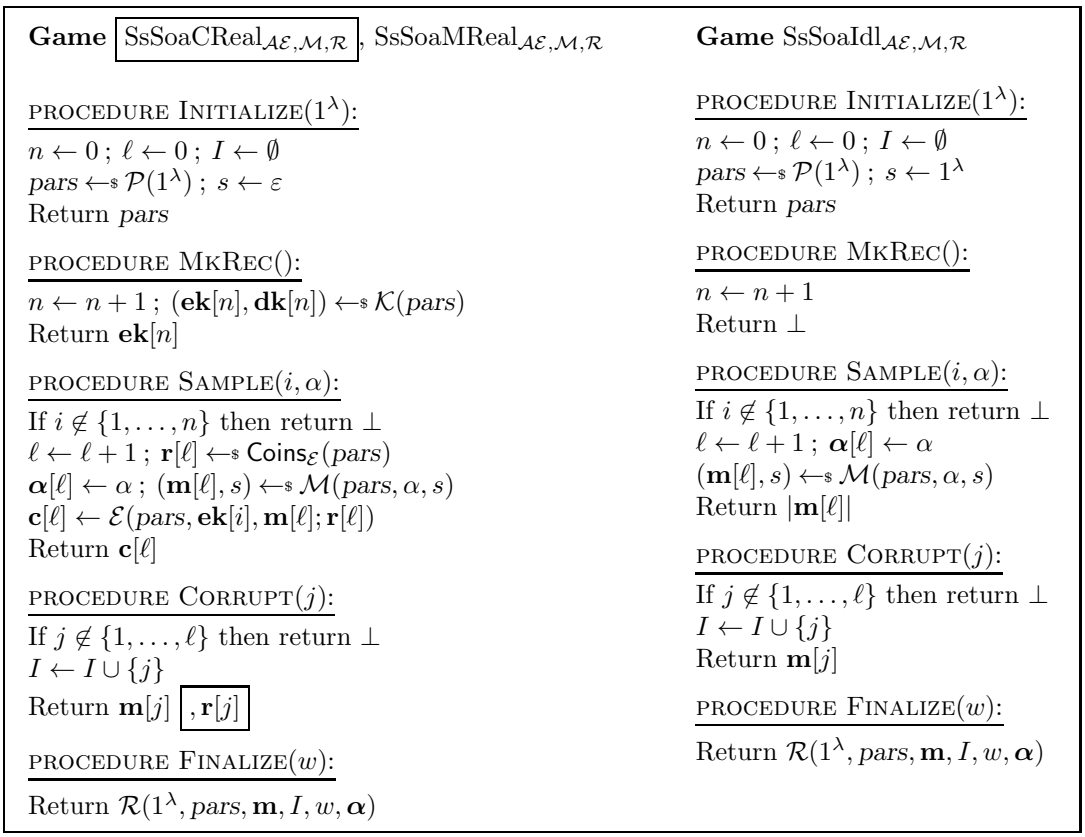

Figure 3: Games to define SS-SOA-C and SS-SOA-M security of PKE scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. Game SsSoaCReal$_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ includes the boxed code in the Corrupt(j) procedure while game SsSoaMReal$_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ does not.

---

choice and the game evaluates the relation $\mathcal{R}$ as shown. Game SsSoaIdl$_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ is the same whether we consider soa-m or soa-c. Initialize is unchanged. MkRec() simply records that another receiver exists but no keys are generated. Sample(i, α) generates a message as before but performs no encryption and returns only the length of the message. Corrupt(j) returns the j-th message, and Finalize is as before. The following says that any ss-soa-c-secure scheme is also ss-soa-m-secure, which is obvious.

Proposition 3.1 SS-SOA-C $\subseteq$ SS-SOA-M. ∎

Conditional resampling. Our ind-style definition will rely on conditional resampling from the message sampler $\mathcal{M}$. Consider the first two algorithms on the right side of Figure 4. Here $\boldsymbol{\alpha}$ is a vectors of strings; $\mathbf{len}$ is a vector over $\mathbb{N}$; $|\boldsymbol{\alpha}| = |\mathbf{len}|$; $I \subseteq [|\boldsymbol{\alpha}|]$; $\mathbf{x}$ is a $|I|$-vector of strings; $\boldsymbol{\omega}$ is a $|\boldsymbol{\alpha}|$-vector whose components are coins. Resampling, given $pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len}$, aims to produce a vector $\mathbf{m}_1$ distributed according to the output of $\mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha})$ subject to the constraint that $\mathbf{m}_1[I] = \mathbf{x}$ and $\mathrm{Len}(\mathbf{m}_1) = \mathbf{len}$. To define this more formally we let

$$\mathrm{CS}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len}) \;=\; \{\, \boldsymbol{\omega} \;:\; \mathrm{Test}(\mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha}; \boldsymbol{\omega}), I, \mathbf{x}, \mathbf{len}) = \mathsf{true} \,\} \,.$$

This is the set of vectors $\boldsymbol{\omega}$ of coins such that an execution of $\mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha})$ with coins from $\boldsymbol{\omega}$ results in a vector $\mathbf{m}_1$ for which the test returns true, meaning that $\mathbf{m}_1[I] = \mathbf{x}$ and $\mathrm{Len}(\mathbf{m}_1) = \mathbf{len}$. The third algorithm on the right side of Figure 4 now performs the resampling, which consists of picking $\boldsymbol{\omega}$ at

Game $\mathrm{Rsmp}_{\mathcal{M}}$

PROCEDURE INITIALIZE(1^λ):

$pars \xleftarrow{\$} \mathcal{P}(1^\lambda)$; $b \xleftarrow{\$} \{0,1\}$
Return $pars$

PROCEDURE CHALLENGE($\boldsymbol{\alpha}, I, \mathbf{x}$):

$\mathbf{len} \leftarrow \mathrm{Len}(\boldsymbol{\alpha})$
If $b = 1$ then $\mathbf{m}_1 \xleftarrow{\$} \mathcal{M}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$
Else $\mathbf{m}_1 \xleftarrow{\$} \mathrm{Resamp}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$
Return $\mathbf{m}_1$

PROCEDURE FINALIZE(b'):

Return $(b = b')$

$\mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha}; \boldsymbol{\omega})$

$s \leftarrow \varepsilon$; $n \leftarrow |\boldsymbol{\alpha}|$
For $i = 1, \ldots, n$ do
  $(\mathbf{m}_1[i], s) \leftarrow \mathcal{M}(pars, \boldsymbol{\alpha}[i], s; \boldsymbol{\omega}[i])$
Return $\mathbf{m}_1$

$\mathrm{Test}(\mathbf{m}_1, I, \mathbf{x}, \mathbf{len})$

Return $(\mathbf{m}_1[I] = \mathbf{x}$ and $\mathrm{Len}(\mathbf{m}_1) = \mathbf{len})$

$\mathrm{Resamp}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$

If $\mathrm{CS}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len}) = \emptyset$ then return $\perp$
$\boldsymbol{\omega} \xleftarrow{\$} \mathrm{CS}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$
$\mathbf{m}_1 \leftarrow \mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha}; \boldsymbol{\omega})$
Return $\mathbf{m}_1$

Figure 4: On the left is the game to define conditional resampling error for message sampler $\mathcal{M}$. On the right are algorithms called in this and later games. The set $\mathrm{CS}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$ is defined in the text.

random from this set and then returning $\mathbf{m}_1 = \mathrm{Run}_{\mathcal{M}}(pars, \boldsymbol{\alpha}; \boldsymbol{\omega})$.

Note that this resampling process need not in general be PT. Our ind definition will require an extra property of the message sampler $\mathcal{M}$, namely that a distribution statistically close to the resampled one can be sampled from in PT. We will require that the sampler has two modes. Its "sample" mode is the usual one above, where it takes $pars, \alpha, s$ to return (m, s). In "resample" mode it takes $pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len}$, as above. It returns a $|\boldsymbol{\alpha}|$-vector of strings, in PT. Then the requirement is that the output of $\mathcal{M}$ on input $pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len}$ has a distribution that is statistically close to the distribution of the output $\mathbf{m}_1$ of $\mathrm{Resamp}_{\mathcal{M}}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$.

To more easily use this condition in our proofs, and also to be more precise, we now define a resampling error based on the game on the left side of Figure 4. A resamp-adversary A makes exactly one CHALLENGE query and outputs a bit. For any $\lambda \in \mathbb{N}$ we let

$$\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A}(\lambda) = 2 \cdot \Pr\left[\, \mathrm{Rsmp}^{A}_{\mathcal{M}}(\lambda) \,\right] - 1 \quad \text{and} \quad \mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M}}(\lambda) = \max_{A} \left\{ \mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A}(\lambda) \right\} .$$

The maximum is over all adversaries, meaning A is unbounded rather than PT. The function $\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M}}(\cdot)$ is the resampling error of $\mathcal{M}$. We say that $\mathcal{M}$ permits efficient re-sampling if $\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M}}(\cdot)$ is negligible and $\mathcal{M}$ runs in PT in both sample and resample modes. Note that the game procedures may not run in PT even if A does because they run $\mathrm{Resamp}_{\mathcal{M}}(\cdot)$.

An important case in which efficient resampling is possible is when successive messages generated by the sampler are independently distributed, and there are other cases as well. But not every message sampler permits efficient resampling, and indeed the assumption that it is possible is quite strong and translates to a restriction in the ind definition that follows.

IND-SOA-X. Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let $\mathcal{M}$ be a message sampler that permits efficient re-sampling as discussed above. We consider the game of Figure 5. We say that A is an ind-soa-adversary if it makes exactly one CHALLENGE query and following this it makes no oracle queries except to FINALIZE. The ind-soa-m advantage of A is

$$\mathbf{Adv}^{\text{ind-soa-m}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) = 2 \cdot \Pr\left[\, \mathrm{IndSoaM}^{A}_{\mathcal{AE},\mathcal{M}}(\lambda) \,\right] - 1 .$$

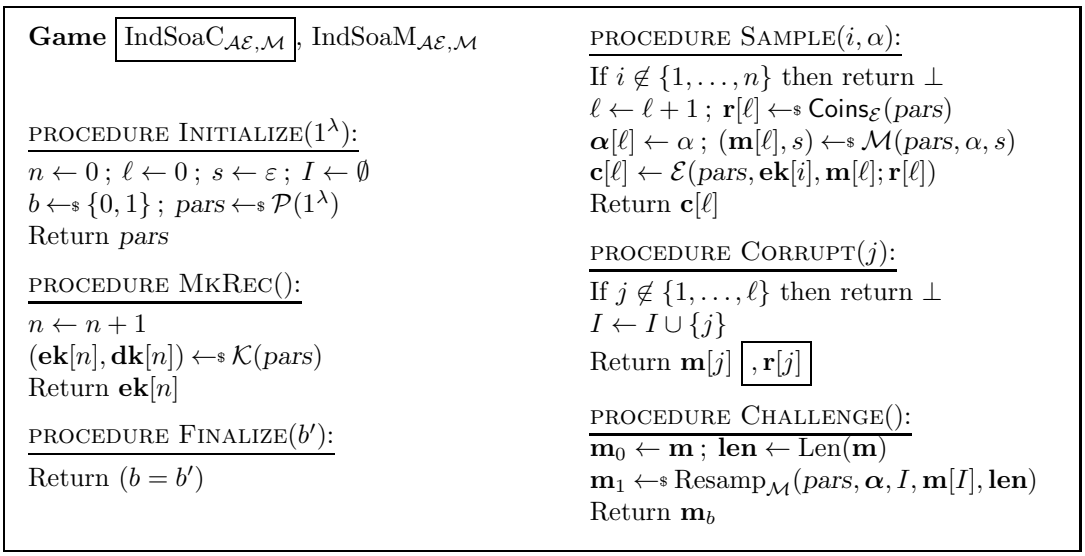

Figure 5: Games to define IND-SOA-C and IND-SOA-M security of PKE scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$. Game IndSoaC$_{\mathcal{AE},\mathcal{M}}$ includes the boxed code in the Corrupt(j) procedure while game IndSoaM$_{\mathcal{AE},\mathcal{M}}$ does not.

We let IND-SOA-M denote the set of all encryption schemes $\mathcal{AE}$ with the property that $\mathbf{Adv}^{\text{ind-soa-m}}_{A,\mathcal{AE},\mathcal{M}}(\cdot)$ is negligible for every PT ind-soa-adversary A and every PT message sampler $\mathcal{M}$ that permits efficient resampling. The ind-soa-c-advantage of A is

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) = 2 \cdot \Pr\left[\, \text{IndSoaC}^{A}_{\mathcal{AE},\mathcal{M}}(\lambda) \,\right] - 1 \ .$$

We let IND-SOA-C denote the set of all encryption schemes $\mathcal{AE}$ with the property that $\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\cdot)$ is negligible for every PT ind-soa-adversary A and every PT message sampler $\mathcal{M}$ that permits efficient resampling.

In the game, variable n will keep track of the number of active receivers; ℓ will be the number of messages sampled; s will be the current state of the message sampling algorithm; I will be the set of corrupted indices. The adversary can call MkRec at any time to create a *receiver*, a party for whom the game creates keys so that it can receive encrypted messages. The adversary can then query Sample with input a receiver index i and a string $\alpha \in \{0,1\}^*$. After checking that i corresponds to an active receiver, incrementing the number of sampled messages ℓ, and storing α in the vector $\boldsymbol{\alpha}$, the Sample procedure will run the message sampler $\mathcal{M}$ on α and the current message sampler state s. The sampler $\mathcal{M}$ will return a message which is stored in the message vector $\mathbf{m}$ at location ℓ, as well as an updated message sampling state string s. Procedure Sample then creates an encryption of the sampled message using independent and uniform coins, returning the ciphertext to the adversary. The adversary can also query Corrupt on a message index j, and as a result will learn $\mathbf{m}[j]$, the jth message sampled by Sample, as well as the coins used by Sample to encrypt the message in the SOA-C case. The set I of corrupted indices is updated with j. Once the adversary is finished making MkRec, Sample, and Corrupt queries, it can make one query to Challenge with no input. Depending on the challenge bit chosen in Initialize, procedure Challenge will either return the actual messages sampled by Sample or a resampled vector. Note that due to the invocation of $\mathrm{Resamp}_{\mathcal{M}}(\cdot)$ the game need not run in PT, but the game is used only for a definition. The following says that any ind-soa-c-secure scheme is also ind-soa-m-secure, which is obvious.

Proposition 3.2 IND-SOA-C $\subseteq$ IND-SOA-M. ▌

As the above indicates, security is only required for message samplers permitting efficient re-sampling. The reason is that otherwise the definition is difficult or impossible to use in applications. Yet, the restriction itself also inhibits applicability, for not all message samplers that arise in applications permit efficient re-sampling. The version of the definition that drops the efficiency requirement on the re-sampling is considered and related to this one and other definitions in [8].

SS-SOA-X IMPLIES IND-SOA-X. We prove that security under our simulation-based notion of security (SS-SOA-X) implies security under our indistinguishability-based notion (IND-SOA-X). This is true both for X=C and X=M.

Theorem 3.3 [SS-SOA-C $\subseteq$ IND-SOA-C and SS-SOA-M $\subseteq$ IND-SOA-M] Let $\mathcal{AE}$ be a PKE scheme and $\mathcal{M}$ a PT message sampler permitting efficient resampling. Let $\mathrm{x} \in \{\mathrm{m}, \mathrm{c}\}$. Let A be a PT ind-soa adversary. Then there exists a PT adversary B and a PT relation $\mathcal{R}$ such that for any simulator S and all $\lambda \in \mathbb{N}$ we have

$$\mathbf{Adv}^{\text{ind-soa-x}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{ss-soa-x}}_{B,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) + 4 \cdot \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda)\ . \ \blacksquare \tag{1}$$

A potentially confusing aspect of the above statement is that it is for all simulators. So let us see why it implies that SS-SOA-X $\subseteq$ IND-SOA-X. Assume $\mathcal{AE} \in$ SS-SOA-X. We want to show that $\mathcal{AE} \in$ IND-SOA-X. Let $\mathcal{M}$ be a PT message sampler permitting efficient resampling and let A be a PT ind-soa adversary. Let $B, \mathcal{R}$ be as per the theorem. The assumption $\mathcal{AE} \in$ SS-SOA-X means that there is a PT simulator S such that $\mathbf{Adv}^{\text{ss-soa-x}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\cdot)$ is negligible. The assumption that $\mathcal{M}$ permits efficient resampling means that $\mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\cdot)$ is negligible. Equation (1) now implies that $\mathbf{Adv}^{\text{ind-soa-x}}_{A,\mathcal{AE},\mathcal{M}}(\cdot)$ is negligible. This means that $\mathcal{AE} \in$ IND-SOA-X as desired.

Proof of Theorem 3.3: We can break the operation of A into two algorithms A_1, A_2 as

$$w \leftarrow\!\!\$\ A_1^{\text{Initialize,MkRec,Sample,Corrupt}}(1^\lambda)\ ;\ b' \leftarrow A_2(w, \mathbf{m})$$

Algorithm A_1 takes 1^λ and runs A on these inputs until it makes its CHALLENGE query, returning the state w of the execution at this point. A_2 takes the state w and the response $\mathbf{m}_b$ of the CHALLENGE oracle to return a bit b' that is the input to FINALIZE, and we may assume A_2 is deterministic since A_1 can put all coins in w. Adversary B, on input 1^λ, simply runs A_1 on input 1^λ, replying to its oracle queries via its own oracles INITIALIZE, MKREC, SAMPLE, CORRUPT, to get w. It then queries FINALIZE(w) and halts. We define relations $\mathcal{R}, \overline{\mathcal{R}}$ via

Relation $\mathcal{R}(1^\lambda, \mathit{pars}, \mathbf{m}, I, w, \boldsymbol{\alpha})$	Relation $\overline{\mathcal{R}}(1^\lambda, \mathit{pars}, \mathbf{m}, I, w, \boldsymbol{\alpha})$
$b \leftarrow\!\!\$\ \{0,1\}\ ;\ \mathbf{m}_0 \leftarrow \mathbf{m}\ ;\ \mathbf{len} \leftarrow \mathrm{Len}(\mathbf{m})$	$b \leftarrow\!\!\$\ \{0,1\}\ ;\ \mathbf{m}_0 \leftarrow \mathbf{m}\ ;\ \mathbf{len} \leftarrow \mathrm{Len}(\mathbf{m})$
$\mathbf{m}_1 \leftarrow\!\!\$\ \mathcal{M}(\mathit{pars}, \boldsymbol{\alpha}, I, \mathbf{m}[I], \mathbf{len})$	$\mathbf{m}_1 \leftarrow\!\!\$\ \mathrm{Resamp}_{\mathcal{M}}(\mathit{pars}, \boldsymbol{\alpha}, I, \mathbf{m}[I], \mathbf{len})$
$b' \leftarrow A_2(w, \mathbf{m}_b)$	$b' \leftarrow A_2(w, \mathbf{m}_b)$
Return $(b = b')$	Return $(b = b')$

Relation $\mathcal{R}$ is PT since $\mathcal{M}$ is assumed to support efficient resampling. Relation $\overline{\mathcal{R}}$ need not be PT but will be used only in the analysis. Let X=C if x=c and X=M if x=m. We claim that for all (even unbounded)

simulators S we can build adversaries A_1, A_2 such that for all $\lambda \in \mathbb{N}$ we have:

$$\begin{aligned}
\Pr\left[\,\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] &= \Pr\left[\,\mathrm{IndSoaX}^{A}_{\mathcal{AE},\mathcal{M}}(\lambda)\,\right] && (2)\\
\Pr\left[\,\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] &\le \frac{1}{2} && (3)\\
\Pr\left[\,\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] - \Pr\left[\,\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda)\,\right] &\le \mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A_1}(\lambda) && (4)\\
\Pr\left[\,\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda)\,\right] - \Pr\left[\,\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] &\le \mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A_2}(\lambda)\,. && (5)
\end{aligned}$$

We first show how to obtain Equation (1) and then return to briefly justify the above claims. From the above we have

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) &= 2\cdot\Pr\left[\,\mathrm{IndSoaX}^{A}_{\mathcal{AE},\mathcal{M}}(\lambda)\,\right] - 1\\
&= 2\cdot\left(\Pr\left[\,\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] - \frac{1}{2}\right)\\
&\le 2\cdot\left(\Pr\left[\,\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right] - \Pr\left[\,\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}(\lambda)\,\right]\right)\\
&\le 2\cdot\mathbf{Adv}^{\text{ss-soa-x}}_{B,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) + 2\cdot\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A_1}(\lambda) + 2\cdot\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M},A_2}(\lambda)\\
&\le 2\cdot\mathbf{Adv}^{\text{ss-soa-x}}_{B,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) + 4\cdot\mathbf{Adv}^{\mathrm{resamp}}_{\mathcal{M}}(\lambda)\,.
\end{aligned}$$

This yields Equation (1). The construction and the definitions of the games justify Equation (2). Equation (3) is true because the real and resampled message vectors are identically distributed given the view of the simulator. Adversary A_1 making Equation (4) true runs B in such a way that B is executing with game $\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}$ if the challenge bit of A_1 is 0 and B is executing with game $\mathrm{SsSoaXReal}^{B}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ if the challenge bit of A_1 is 1. A_1 obtains *pars* via its INITIALIZE query and then directly simulates oracles MKREC, SAMPLE, CORRUPT. When B calls FINALIZE(w), adversary A_1 simulates the relation, obtaining $\mathbf{m}_1$ via its CHALLENGE oracle, so that the relation is $\mathcal{R}$ when the challenge bit of A_1 is 1 and $\overline{\mathcal{R}}$ otherwise. If the simulated relation returns true, it calls FINALIZE(0) else it calls FINALIZE(1). Adversary A_2 making Equation (5) true runs S in such a way that S is executing with game $\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\overline{\mathcal{R}}}$ if the challenge bit of A_2 is 0 and S is executing with $\mathrm{SsSoaIdl}^{S}_{\mathcal{AE},\mathcal{M},\mathcal{R}}$ if the challenge bit of A_1 is 1. A_2 obtains *pars* via its INITIALIZE query and then directly simulates oracles MKREC, SAMPLE, CORRUPT. When S calls FINALIZE(w), adversary A_2 simulates the relation, obtaining $\mathbf{m}_1$ via its CHALLENGE oracle, so that the relation is $\mathcal{R}$ when the challenge bit of A_2 is 1 and $\overline{\mathcal{R}}$ otherwise. If the simulated relation returns true, it calls FINALIZE(1) else it calls FINALIZE(0). ∎

4 Equivalence of IND-CPA, SS-SOA-M and IND-SOA-M

In this section we show that when only messages (and not coins) are revealed, both notions of SOA-security are equivalent to each other and to the standard IND-CPA. This confirms an expected result but we feel it is worth seeing in perspective to the later difficulties with opening of coins.

IND-CPA IMPLIES SS-SOA-M. We show that any encryption scheme that is IND-CPA secure is also SS-SOA-M secure. While interesting in its own right, the proof of this result also serves as a good warm-up for our main SOA-C results.

Theorem 4.1 [IND-CPA ⊆ SS-SOA-M] Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, $\mathcal{M}$ a PT message sampler, $\mathcal{R}$ a PT relation, and A a PT adversary making q_{mk} queries to MKREC, q_{s} queries to SAMPLE

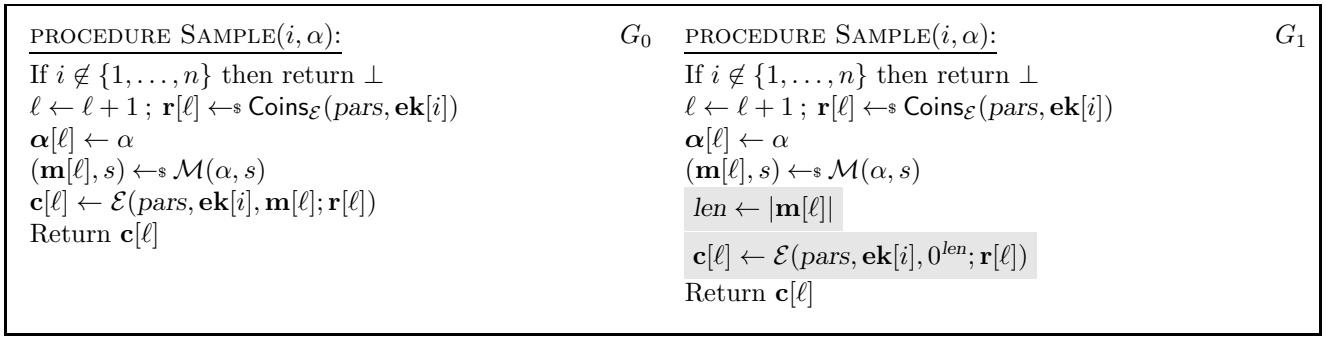

Figure 6: The SAMPLE procedures in games G_0 and G_1 used in the proof of Theorem 4.1.

---

of which at most q_{spr} are for any particular receiver, and q_{c} queries to CORRUPT. Then, there exists a PT simulator S and a PT indcpa-adversary C such that for every $\lambda \in \mathbb{N}$ we have

$$\mathbf{Adv}^{\text{ss-soa-m}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \leq q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{C,\mathcal{AE}}(\lambda) \ . \ \blacksquare$$

Proof: We will prove the theorem using a sequence of game transitions, starting with a game G_0, which is the same as game SsSoaMReal.

First, recall that

$$\mathbf{Adv}^{\text{ss-soa-m}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) = \Pr\left[\ \text{SsSoaMReal}^{A}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1\ \right] - \Pr\left[\ \text{SsSoaIdl}^{S}_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1\ \right] \ . \tag{6}$$

We let game G_0 be identical to game $\text{SsSoaMReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda)$. Our game transition from G_0 to G_1 makes only one change, shown in Figure 6: in SAMPLE, instead of encrypting the actual message sampled by $\mathcal{M}$, encrypt a dummy message of the appropriate length (represented in the figure by 0^{len}). We claim that there is an efficient adversary B such that

$$\Pr\left[\ G^{A}_{0} \Rightarrow 1\ \right] - \Pr\left[\ G^{A}_{1} \Rightarrow 1\ \right] \leq \mathbf{Adv}^{\text{ind-cpa-mr}}_{B,\mathcal{AE}}(\lambda) \ , \tag{7}$$

where adversary B, shown in Figure 7, makes q_{mk} queries to MKREC and at most q_{spr} queries to LR for any receiver index. Adversary B runs A and answers SAMPLE queries using its own LR oracle queried on a sampled message and a dummy message of the same length. Thus, depending on the bit used by LR, B either perfectly simulates game G_0 or G_1 for A, leading to the claim.

Applying Proposition 2.1, there is an efficient adversary C such that

$$\Pr\left[\ G^{A}_{0} \Rightarrow 1\ \right] - \Pr\left[\ G^{A}_{1} \Rightarrow 1\ \right] \leq q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{C,\mathcal{AE}}(\lambda) \ , \tag{8}$$

where C makes one LR query.

Finally, we note that running game G_1 with adversary A is the same as running game SsSoaIdl with simulator S, shown in Figure 7:

$$\Pr\left[\ G^{A}_{1} \Rightarrow 1\ \right] = \Pr\left[\ \text{SsSoaIdl}^{S}_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1\ \right] \ . \tag{9}$$

This is because S only needs the message length in SAMPLE (which it can get from its own SAMPLE oracle) and actual sampled messages are not needed until CORRUPT, where they can be learned by S by querying its own CORRUPT oracle.

Adversary $B(pars)$:

$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$
Run $A(pars)$.

On query MkRec():
$n \leftarrow n+1$
$\mathbf{ek}[n] \leftarrow \text{MkRec}_B()$
Return $\mathbf{ek}[n]$

On query Sample(i, α):
If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$
$(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$len \leftarrow |\mathbf{m}[\ell]|$
$\mathbf{c}[\ell] \leftarrow \text{LR}_B(i, \mathbf{m}[\ell], 0^{len})$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):
If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
Return $\mathbf{m}[j]$

When A halts with output w,
halt and output $\mathcal{R}(1^\lambda, \mathbf{m}, I, w, \boldsymbol{\alpha})$

Simulator $S(1^\lambda)$:

$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$
$pars \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
Run $A(pars)$.

On query MkRec():
$n \leftarrow n+1$
$\text{MkRec}_S()$
$(\mathbf{ek}[n], \mathbf{dk}[n]) \leftarrow \mathcal{K}(pars)$
Return $\mathbf{ek}[n]$

On query Sample(i, α):
If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$; $\mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars, \mathbf{ek}[i])$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$
$len \leftarrow \text{Sample}_S(i, \alpha)$
$\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], 0^{len}; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):
If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{m}[j] \leftarrow \text{Corrupt}_S(j)$
Return $\mathbf{m}[j]$

When A halts with output w,
halt and output w

Figure 7: Adversary and simulator used in the proof of Theorem 4.1

Combining the above equations, we can see that

$$\begin{aligned}
\mathbf{Adv}^{\text{ss-soa-c}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) &= \Pr\left[\,\text{SsSoaCReal}^A_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1\,\right] - \Pr\left[\,\text{SsSoaIdl}^S_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1\,\right] \\
&= \Pr\left[\,G^A_0 \Rightarrow 1\,\right] - \Pr\left[\,G^A_1 \Rightarrow 1\,\right] \\
&\leq q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{C,\mathcal{AE}}(\lambda)\ ,
\end{aligned}$$

which proves the theorem. ∎

Combining Theorem 4.1 with Theorem 3.3 gives us that IND-CPA also implies IND-SOA-M.

Corollary 4.2 IND-CPA $\subseteq$ IND-SOA-M. ∎

4.1 IND-SOA-M implies IND-CPA

Next, we will show that IND-SOA-M implies IND-CPA.

Theorem 4.3 [IND-SOA-M $\subseteq$ IND-CPA] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, and A a PT ind-cpa adversary against $\mathcal{AE}$. Then there exists a PT adversary B and PT message sampler $\mathcal{M}$ permitting efficient resampling such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\lambda) \leq 2 \cdot \mathbf{Adv}^{\text{ind-soa-m}}_{B,\mathcal{AE},\mathcal{M}}(\lambda)\ .$$

Adversary $B(pars)$:
$ek \leftarrow$ MkRec()
Run $A(ek)$.

On query LR(m_0, m_1):
$\alpha \leftarrow (m_0, m_1)$
$c \leftarrow$ Sample$(1, \alpha)$
Return c

When A halts with output b':
$m \leftarrow$ Challenge()
If $m = m_{b'}$ halt with output 0, else halt with output 1

Figure 8: Adversary used in proof of Theorem 4.3.

Adversary B makes 1 MkRec query, 1 Sample query, and 0 Corrupt queries. □

Proof: We first define $\mathcal{M}(\alpha, s)$ to parse α as (m_0, m_1), flip a bit d, and return (m_d, s). It is easy to see $\mathcal{M}$ runs in polynomial time and also permits efficient resampling. We define adversary B in Figure 8. The adversary B runs A and uses Sample to simulate the LR oracle.

Recall that

$$\mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\lambda) = 2 \cdot \Pr\left[\, \text{IndCpa}^{A}_{\mathcal{AE}}(\lambda) \Rightarrow \mathsf{true} \,\right] - 1 \,, \tag{10}$$

and

$$\mathbf{Adv}^{\text{ind-soa-m}}_{B,\mathcal{AE},\mathcal{M}}(\lambda) = 2 \cdot \Pr\left[\, \text{IndSoaM}^{B}_{\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] - 1 \,. \tag{11}$$

Now, let's look at $\Pr\left[\, \text{IndSoaM}^{B}_{\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right]$ in more detail. Let $\text{IndSoaM}_{b,\mathcal{AE},\mathcal{M}}(\lambda)$ be the same game, but parameterized by the challenge bit b. (In other words, the game sets b to that value instead of choosing it uniformly at random.) Then

$$\begin{aligned}&\Pr\left[\, \text{IndSoaM}^{B}_{\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] = \\ &\quad \frac{1}{2} \cdot \Pr\left[\, \text{IndSoaM}_{0,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] + \frac{1}{2} \cdot \Pr\left[\, \text{IndSoaM}_{1,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] \,.\end{aligned}$$

We first claim that

$$\Pr\left[\, \text{IndSoaM}^{B}_{0,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] = \Pr\left[\, \text{IndCpa}^{A}_{\mathcal{AE}}(\lambda) \Rightarrow \mathsf{true} \,\right] \,. \tag{12}$$

This is true since if the challenge bit in IndSoaM is 0 then the actual encrypted message is returned by Challenge. Adversary B will correctly guess 0 only if adversary A guessed the bit of the encrypted message correctly, meaning it won the IND-CPA game.

We next claim

$$\Pr\left[\, \text{IndSoaM}^{B}_{1,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \,\right] = \frac{1}{2} \,. \tag{13}$$

This is true since if the challenge bit in IndSoaM is 1 the resampled message is returned, and this value will just be uniformly chosen between the two options. This uniform choice will be independent of anything given to A, so A has a 1/2 probability of guessing correctly.

Combining the above equations, we see that

$$\begin{aligned}
&\mathbf{Adv}^{\text{ind-soa-m}}_{B,\mathcal{AE},\mathcal{M}}(\lambda) \\
&= 2 \cdot \left(\frac{1}{2} \cdot \Pr\left[\, \text{IndSoaM}_{0,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \, \right] + \frac{1}{2} \cdot \Pr\left[\, \text{IndSoaM}_{1,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow \mathsf{true} \, \right] \right) - 1 \\
&= \Pr\left[\, \text{IndCpa}^{A}_{\mathcal{AE}}(\lambda) \Rightarrow \mathsf{true} \, \right] + \frac{1}{2} - 1 \\
&= \frac{1}{2} \mathbf{Adv}^{\text{ind-cpa}}_{A,\mathcal{AE}}(\lambda) \, ,
\end{aligned}$$

which proves the theorem. ∎

To summarize the results of this section, we have that

$$\text{IND-CPA} \subseteq \text{SS-SOA-M} \subseteq \text{IND-SOA-M} \subseteq \text{IND-CPA} \, .$$

5 Lossy Encryption

Our main results on SOA-C rely on what we call a *Lossy Encryption Scheme*. Informally, a public-key encryption scheme is lossy if there is an efficient alternate key generation algorithm (called the "lossy key generation algorithm") that produces "lossy" public keys that are indistinguishable from "real" public keys; however, encryptions under lossy public keys statistically contain little or no information about the encrypted message. Peikert, Vaikuntanathan, and Waters [31] 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 [26], formalized by Kol and Naor.

Before we give a formal definition of lossy encryption schemes, we define the key-ind advantage of an adversary A with respect to a PKE scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and an alternate key generation algorithm $\mathcal{K}_\ell$ as

$$\mathbf{Adv}^{\text{key-ind}}_{A,\mathcal{AE},\mathcal{K}_\ell}(\lambda) = 2 \cdot \Pr\left[\, \text{KeyInd}^{A}_{\mathcal{AE},\mathcal{K}_\ell}(\lambda) \Rightarrow \mathsf{true} \, \right] - 1 \, ,$$

where the security game KeyInd is defined as follows. The INITIALIZE procedure flips a bit b, runs parameter generation $\mathcal{P}(1^\lambda)$ to get parameters $pars$, and then runs key generation algorithm $\mathcal{K}$ to get keypair (ek_0, dk_0) and runs alternate key generation algorithm $\mathcal{K}_\ell$ to get keypair (ek_1, dk_1). It then returns $(pars, ek_b)$ to the adversary. The adversary outputs a guess bit b' and the output of the game is true if $b = b'$ and false otherwise.

Now we can formally define lossy encryption. We say a PKE scheme $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is *lossy* if there is a PT alternate key generation algorithm $\mathcal{K}_\ell$, called the lossy key generation algorithm, such that

1. *Indistinguishability of Real and Lossy Keys.* For all PT A, the key-ind advantage $\mathbf{Adv}^{\text{key-ind}}_{A,\mathcal{AE},\mathcal{K}_\ell}(\lambda)$ is negligible.

2. *Lossiness of encryption with lossy keys.* For PKE scheme scheme $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ and for all (even unbounded) adversaries B, the indcpa-advantage $\mathbf{Adv}^{\text{ind-cpa}}_{B,\mathcal{AE}_\ell}(\lambda)$ is negligible.

We will sometimes call the encryption keys output by $\mathcal{K}$ "real" public keys, and the encryption keys output by $\mathcal{K}_\ell$ "lossy" public keys.

Let $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a lossy encryption scheme with lossy key generation algorithm $\mathcal{K}_\ell$. Then, let $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme composed of the same algorithms as $\mathcal{AE}$ except with $\mathcal{K}$ replaced by the lossy key generation algorithm $\mathcal{K}_\ell$. We say that an algorithm Open_∞ is an opening algorithm for $\mathcal{AE}_\ell$ if for all λ, all $pars \in [\mathcal{P}(1^\lambda)]$, all $(ek_\ell, dk_\ell) \in [\mathcal{K}_\ell(pars)]$, all $m' \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$, and all ciphertexts

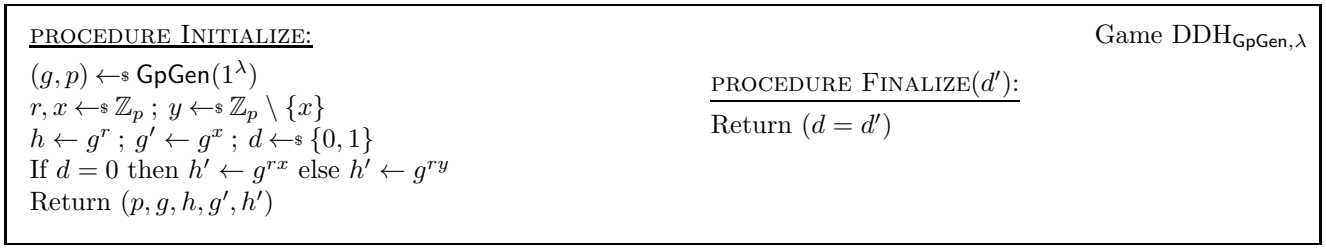

Figure 9: Decisional Diffie-Hellman (DDH) security game.

c, $\mathsf{Open}_\infty(ek_\ell, m', c)$ is distributed uniformly in the set $\{r' : r' \in \mathsf{Coins}_\mathcal{E}(pars) \wedge \mathcal{E}(ek_\ell, m'; r') = c\}$. This set may be empty, in which case Open_∞ should output $\perp$. Note that an opening algorithm always exists, but may not be polynomial time. For example, Open_∞ can find all $r' \in \mathsf{Coins}_\mathcal{E}(pars)$ such that $\mathcal{E}(ek_\ell, m'; r') = c$ and then output a uniformly chosen one of the r'.

We also consider efficient opening algorithms that take additional inputs. Specifically, we say that Open is an efficient opening algorithm for $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$ if for all λ, all $pars \in [\mathcal{P}(1^\lambda)]$, all $(ek_\ell, dk_\ell) \in [\mathcal{K}_\ell(pars)]$, and all messages $m, m' \in \mathsf{MsgSp}_{\mathcal{AE}}(pars)$, $\mathsf{Open}(ek_\ell, dk_\ell, m', m, r, \mathcal{E}(ek_\ell, m; r))$ is distributed identically to $\mathsf{Open}_\infty(ek_\ell, m', \mathcal{E}(ek_\ell, m; r))$ over the choice of $r \leftarrow\!\!\$\ \mathsf{Coins}_\mathcal{E}(pars, ek_\ell)$ and the coins of the opening algorithms.

We note that it immediately follows that any lossy encryption scheme is IND-CPA secure. We next provide numerous examples of lossy encryption schemes.

5.1 Lossy Encryption from DDH

We first describe a lossy public-key encryption scheme based on the DDH assumption. A group generator is an algorithm GpGen that, on input a security parameter 1^λ in unary, selects a cyclic group $\mathbb{G}$ of order a prime p and a generator g, and outputs a description of the group $\mathbb{G}$ as well as g and p. Now consider a game DDH (shown in Figure 9) played with an adversary. We define the ddh-advantage of an adversary A against a group generator GpGen as

$$\mathbf{Adv}^{\mathrm{ddh}}_{A,\mathsf{GpGen}}(\lambda) = 2 \cdot \Pr\left[\, \mathrm{DDH}^A_{\mathsf{GpGen},\lambda} \Rightarrow \mathsf{true} \,\right] - 1\ .$$

The DDH assumption for a group generator GpGen states that for all PT adversaries A the ddh-advantage of A against GpGen is a negligible function of the security parameter.

We can now describe our scheme. The scheme is originally from [27], yet some of our notation is taken from [31]. The scheme has structure similar to El Gamal encryption [21].

The scheme $\mathcal{AE}_{\mathrm{ddh1}} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and the associated lossy key generation algorithm $\mathcal{K}_\ell$ constructed from a group generation algorithm GpGen are as follows. Parameter generation algorithm $\mathcal{P}$, on input 1^λ, runs GpGen on input 1^λ and outputs the description of the group $\mathbb{G}$, prime p, and generator g. The message space of the scheme is the group, i.e., $\mathsf{MsgSp}_{\mathcal{AE}_{\mathrm{ddh1}}}(pars) = \mathbb{G}$, for $pars = (\mathbb{G}, p, g)$. (Note that we are abusing notation here and sometimes using $\mathbb{G}$ to denote the description of the group and sometimes using it to indicate the set of group elements.)

The rest of the algorithms:

Algorithm $\mathcal{K}(pars)$
$(\mathbb{G}, g, p) \leftarrow pars$; $x, r \leftarrow_\$ \mathbb{Z}_p$
$ek \leftarrow (g, g^r, g^x, g^{rx})$
$dk \leftarrow x$
Return (ek, dk)

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

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

Algorithm $\mathcal{K}_\ell(pars)$
$(\mathbb{G}, g, p) \leftarrow pars$
$r, x \leftarrow_\$ \mathbb{Z}_p$; $y \leftarrow_\$ \mathbb{Z}_p \setminus \{x\}$
$ek \leftarrow (g, g^r, g^x, g^{ry})$
$dk \leftarrow (r, x, y)$
Return (ek, dk)

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

To see the correctness property is satisfied, consider a (real) public key $ek = (g, g^r, g^x, g^{rx})$ and corresponding secret key $dk = x$. Then, for a message $m \in \mathbb{G}$

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

Now, we claim that $\mathcal{AE}_{\mathrm{ddh1}}$ is a lossy encryption scheme.

1. *Indistinguishability of Real Keys and Lossy Keys.* This follows from the assumption that DDH is hard for GpGen, 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 $y \neq x$, which exactly matches the case in game DDH.

2. *Lossiness of encryption with lossy keys.* Since a lossy key (g, g^r, g^x, g^y) is such that $rx \neq y$, then $s + rt$ and $sx + yt$ are linearly independent combinations of s and t, so an encryption under the lossy key results in two uniformly random group elements.

5.2 Lossy Encryption from Lossy TDFs

For our next scheme, we make use of *lossy trapdoor functions*, a primitive recently introduced by Peikert and Waters [32]. A family of injective trapdoor functions is syntactically the same as a deterministic PKE scheme, however we will use different letters to denote the different algorithms to avoid confusion. Specifically, a family of injective trapdoor functions with message length $n(\cdot)$ is a tuple of PT algorithms $\mathcal{F} = (\mathsf{P}, \mathsf{K}, \mathsf{F}, \mathsf{F}^{-1})$ with the following properties. The randomized parameter generation algorithm P takes as input security parameter 1^λ and outputs parameter string π. The randomized key generation algorithm K takes as input parameters π and outputs a function index σ and trapdoor τ. The deterministic function evaluation algorithm F takes as input π, a function index σ and $x \in \{0,1\}^{n(\lambda)}$ and outputs a point y. The function inverse algorithm F^{-1} takes as input π, a trapdoor τ and a point y and outputs a point x. We require the correctness condition that for all 1^λ, all $\pi \in [\mathsf{P}(1^\lambda)]$, all $(\sigma, \tau) \in [\mathsf{K}(\pi)]$, and all $x \in \{0,1\}^{n(\lambda)}$, it is the case that $\mathsf{F}^{-1}(\pi, \tau, \mathsf{F}(\pi, \sigma, x)) = x$.

We will also need a family of pairwise-independent hash functions. A family of hash functions is a tuple of PT algorithms $\mathcal{H} = (\mathsf{P}_\mathrm{h}, \mathsf{K}_\mathrm{h}, \mathsf{H})$ with message length $n(\cdot)$ and with the following properties. The randomized parameter generation algorithm P_h takes as input a security parameter 1^λ and outputs a parameter string π_h. The randomized key generation algorithm K_h takes as input a parameter string π_h and outputs a hash key κ. The deterministic hash evaluation algorithm H takes as input the parameters π_h, hash key κ and input $x \in \{0,1\}^{n(\lambda)}$ and outputs y. Let $R(\pi_\mathrm{h}) = \{\mathsf{H}(\pi_\mathrm{h}, \kappa, x) \ : \ \kappa \in [\mathsf{K}_\mathrm{h}(\pi_\mathrm{h})] \wedge$

$x \in \{0,1\}^n\}$. We say that $\mathcal{H}$ is pairwise-independent if for all λ, all $\pi_{\mathrm{h}} \in [\mathsf{P}_{\mathrm{h}}(1^\lambda)]$, all distinct inputs $x, x' \in \{0,1\}^{n(\lambda)}$, and all $y, y' \in R(\pi_{\mathrm{h}})$, it is the case that

$$\Pr\left[\, \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x) = y \wedge \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x') = y' \,\right] \leq \frac{1}{|R(\pi_{\mathrm{h}})|}\ ,$$

where the probability is taken over $\kappa \leftarrow\!\!\$\ \mathsf{K}_{\mathrm{h}}(\pi_{\mathrm{h}})$. We say $\mathcal{H}$ has output length ℓ if for all λ and all $\pi_{\mathrm{h}} \in [\mathsf{K}_{\mathrm{h}}(1^\lambda)]$, the range $R(pars) = \{0,1\}^{\ell(\lambda)}$.

Now, we say a family $\mathcal{F}$ of injective trapdoor functions with message length $n(\cdot)$ is (n,k)-lossy if there exists a PT alternative key generation algorithm K_ℓ (called the lossy key generation algorithm) such that

- For all λ, all $\pi \in [\mathsf{P}(1^\lambda)]$, and all $(\sigma_\ell, \tau_\ell) \in [\mathsf{K}_\ell(\pi)]$ the map $\mathsf{F}(\pi, \sigma_\ell, \cdot)$ has image size at most $2^{n(\lambda)-k(\lambda)}$.
- For all PT adversaries A, the key-ind advantage $\mathbf{Adv}^{\text{key-ind}}_{A,\mathcal{F},\mathsf{K}_\ell}(\lambda)$ is negligible in λ.

We now describe an instantiation of lossy (randomized) encryption based on lossy trapdoor functions. The following scheme was given by Peikert and Waters in [32] Let λ be a security parameter and let $\mathcal{F} = (\mathsf{P}, \mathsf{K}, \mathsf{F}, \mathsf{F}^{-1})$ define a collection of injective trapdoor functions that are (n,k)-lossy with associated lossy key generation algorithm K_ℓ. Also let $\mathcal{H} = (\mathsf{P}_{\mathrm{h}}, \mathsf{K}_{\mathrm{h}}, \mathsf{H})$ be a collection of pair-wise independent hash functions with message length $n(\lambda)$ and output length $\ell(\lambda)$; the message space of the cryptosystem will 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 $\mathcal{AE}_{\mathrm{ltdf}} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is then defined as follows. The parameter generation algorithm $\mathcal{P}$, on input security parameter 1^λ, simply runs $\pi \leftarrow\!\!\$\ \mathsf{P}(1^\lambda)$ and $\pi_{\mathrm{h}} \leftarrow\!\!\$\ \mathsf{P}_{\mathrm{h}}(1^\lambda)$ and outputs $pars = (\pi, \pi_{\mathrm{h}})$. The rest of the algorithms are as follows:

Algorithm $\mathcal{K}(pars)$
$(\pi, \pi_{\mathrm{h}}) \leftarrow pars$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{K}(\pi)$
$\kappa \leftarrow\!\!\$\ \mathsf{K}_{\mathrm{h}}(\pi_{\mathrm{h}})$
$ek \leftarrow (\sigma, \kappa)$; $dk \leftarrow (\tau, \kappa)$
Return (ek, dk)

Algorithm $\mathcal{E}(pars, ek, m)$
$(\pi, \pi_{\mathrm{h}}) \leftarrow pars$
$(\sigma, \kappa) \leftarrow ek$
$x \leftarrow\!\!\$\ \{0,1\}^n$
$c_1 \leftarrow \mathsf{F}(\pi, \sigma, x)$
$c_2 \leftarrow m \oplus \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x)$
Return (c_1, c_2)

Algorithm $\mathcal{D}(pars, dk, c)$
$(\pi, \pi_{\mathrm{h}}) \leftarrow pars)$
$(\tau, \kappa) \leftarrow dk$
$(c_1, c_2) \leftarrow c$
$x \leftarrow \mathsf{F}^{-1}(\pi, \tau, c_1)$
Return $\mathsf{H}(\pi_{\mathrm{h}}, \kappa, x) \oplus c_2$

The associated lossy key generation algorithm $\mathcal{K}_\ell$ is simply the same as $\mathcal{K}$, but using K_ℓ instead of K. The correctness of the scheme follows since when $ek = (\sigma, \kappa)$ was generated by $\mathcal{K}$,

$$\begin{aligned}
\mathcal{D}(pars, dk, \mathcal{E}(pars, ek, m)) &= \mathsf{H}(\pi_{\mathrm{h}}, \kappa, \mathsf{F}^{-1}(\pi, \tau, \mathsf{F}(\pi, \sigma, x)) \oplus (m \oplus \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x)) \\
&= \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x) \oplus m \oplus \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x) \\
&= m
\end{aligned}$$

We can then verify that the encryption scheme is lossy:

1. *Indistinguishability of real keys and lossy keys.* We need to show that any efficient adversary A, the key-ind advantage $\mathbf{Adv}^{\text{key-ind}}_{A,\mathcal{AE}_{\mathrm{ltdf}},\mathcal{K}_\ell}(\lambda)$ is negligible. This follows since $\mathcal{K}$ uses K while $\mathcal{K}_\ell$ uses K_ℓ, and since $\mathcal{F}$ is lossy, then by definition we know that for all efficient adversaries B, the key-ind advantage $\mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{F},\mathsf{K}_\ell}(\lambda)$ is negligible.

2. *Lossiness of encryption with lossy keys.* This was shown by Peikert and Waters in [32]. Reviewing their argument, the average min-entropy $\tilde{\mathrm{H}}_\infty(x|(c_1, ek_\ell))$ of the random variable x, given $\mathsf{F}(\pi, \sigma_\ell, x)$ and $ek_\ell = (\sigma_\ell, \kappa)$ is at least k, and since $\ell \leq k - 2\log(1/\delta)$ for negligible δ, it follows that $\mathsf{H}(\pi_{\mathrm{h}}, \kappa, x)$

PROCEDURE INITIALIZE:	Game $\mathrm{DQR}_{\mathsf{Par}}$
$(N,p,q) \leftarrow\!\!\$\ \mathsf{Par}(1^\lambda)$ $d \leftarrow\!\!\$\ \{0,1\}\ ;\ x_0 \leftarrow\!\!\$\ \mathsf{QR}_N\ ;\ x_1 \leftarrow\!\!\$\ \mathsf{QNR}_N^{+1}$ Return (N, x_d)	PROCEDURE FINALIZE(d'): Return $(d = d')$

Figure 10: Quadratic Residuosity Game.

will be statistically close to uniform and $m_b \oplus \mathsf{H}(\pi_{\mathrm{h}}, \kappa, x)$ will also be statistically close to uniform for either bit b. Thus, even unbounded adversaries have negligible ind-cpa advantage.

5.3 The GM Probabilistic Encryption Scheme is Lossy with Efficient Opening

The Goldwasser-Micali Probabilistic encryption scheme [23] is an example of a lossy encryption scheme with efficient opening. We briefly recall the GM scheme. Let Par be an algorithm that on input 1^λ efficiently chooses two large distinct random primes p and q congruent to 3 (mod 4) and outputs them along with their product N (a Blum integer). 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}. This is captured by game DQR, shown in Figure 10. We say that the qr-advantage of an adversary A with respect to Par is

$$\mathbf{Adv}^{\mathrm{dqr}}_{A,\mathsf{Par}}(\lambda) = 2 \cdot \Pr\left[\, \mathrm{DQR}^A_{\mathsf{Par}}(\lambda) \Rightarrow \mathsf{true} \,\right] - 1\ .$$

The quadratic residuosity assumption for Par is that for all efficient adversaries, the qr-advantage of A with respect to Par is negligible in λ.

The scheme $\mathcal{AE}_{\mathrm{GM}} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is then defined as follows. The parameter generation algorithm $\mathcal{P}$, on input 1^λ, simply outputs 1^λ. The other algorithms:

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

Algorithm $\mathcal{E}(1^\lambda, ek, m)$
$(N, x) \leftarrow ek$
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 $\mathcal{D}(1^\lambda, dk, \mathbf{c})$
$(p, q) \leftarrow dk$
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 associated lossy key generation 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}; the lossy secret key is still the factorization of N.

The correctness of the above scheme was shown in [23], while the indistinguishability of real keys from lossy keys follows directly from the quadratic residuosity assumption. It is also clear that encryptions under lossy keys are (perfectly) indistinguishable, since lossy ciphertexts are just sequences of random quadratic residues.

Now, we claim that $\mathcal{AE}_{\mathrm{GM}}$ is also efficiently openable. To see this consider the efficient algorithm Open that takes as input lossy secret key $dk = (p, q)$, lossy public key $ek = (N, x)$, plaintext m', ciphertext $\mathbf{c}$, and m and $\mathbf{r}$ such that $\mathcal{E}(ek, m; \mathbf{r}) = \mathbf{c}$. Say m' has length n bits. For each $i \in [n]$, Open 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 Open is the sequence $\mathbf{r}'$. Notice this is exactly what the inefficient Open_∞ would do (find the four square roots and output a uniformly chosen one of the four); the efficiency is gained by knowing the factorization of N.

5.4 A Scheme with Efficient Opening from DDH

We point out that if we modify the scheme in Section 5.1 to only encrypt one bit at a time, we can get an encryption scheme $\mathcal{AE}_{\text{ddh2}}$ that is lossy with efficient opening. More formally, we modify the encryption algorithm to be as follows:

Algorithm $\mathcal{E}(pars, ek, m)$
$(g, h, g', h') \leftarrow ek$
For $i = 1$ to $|m|$
  $(u, v) \leftarrow\!\!\$\ \mathsf{Rand}(pars, g, h, g', h')$
  $(\mathbf{c}_1[i], \mathbf{c}_2[i]) \leftarrow (u, v \cdot g^{m_i})$
Return $(\mathbf{c}_1, \mathbf{c}_2)$

In the above, $|m|$ is the bit length of m and m_i is the ith bit. The decryption algorithm is modified in the obvious way, while parameter generation, key generation, Rand, and the corresponding lossy key generation algorithms are unmodified from Section 5.1.

We will describe the efficient opening algorithm for a 1-bit message; the algorithm can be repeated on each individual component of the ciphertext in the multibit case. The efficient opening algorithm, on input

$$\begin{aligned} ek_\ell &= (g, g^r, g^x, g^{ry}) \\ dk_\ell &= (r, x, y) \\ (c_1, c_2) &= \left(g^s \cdot (g^r)^t \ ,\ (g^x)^s \cdot (g^{ry})^t \cdot g^m \right) \\ m &\in \{0,1\} \\ s, t &\in \mathbb{Z}_p \\ m' &\in \{0,1\} \ , \end{aligned}$$

needs to find $s', t' \in \mathbb{Z}_p$ such that encrypting m' with s' and t' as coins will result in ciphertext (c_1, c_2). To do so, the opening algorithm solves the equations

$$\begin{aligned} s + rt &= s' + rt' \\ xs + ryt + m &= xs' + ryt' + m' \ , \end{aligned}$$

which is possible in our scheme since $x \neq y$. Note that there is only one such pair (s', t'), so the output of the efficient opener will be the same as the output would have been from the inefficient opener Open_∞.

6 Lossy Encryption implies SOA-C Security

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

Theorem 6.1 [Lossy Encryption implies IND-SOA-C] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a lossy public-key encryption scheme with associated lossy key generation algorithm $\mathcal{K}_\ell$; let $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$. Let $\mathcal{M}$ be PT message sampler permitting efficient resampling. Let A be a PT ind-soa adversary making q_{mk} queries to MkRec, q_{s} queries to Sample of which at most q_{spr} are for any

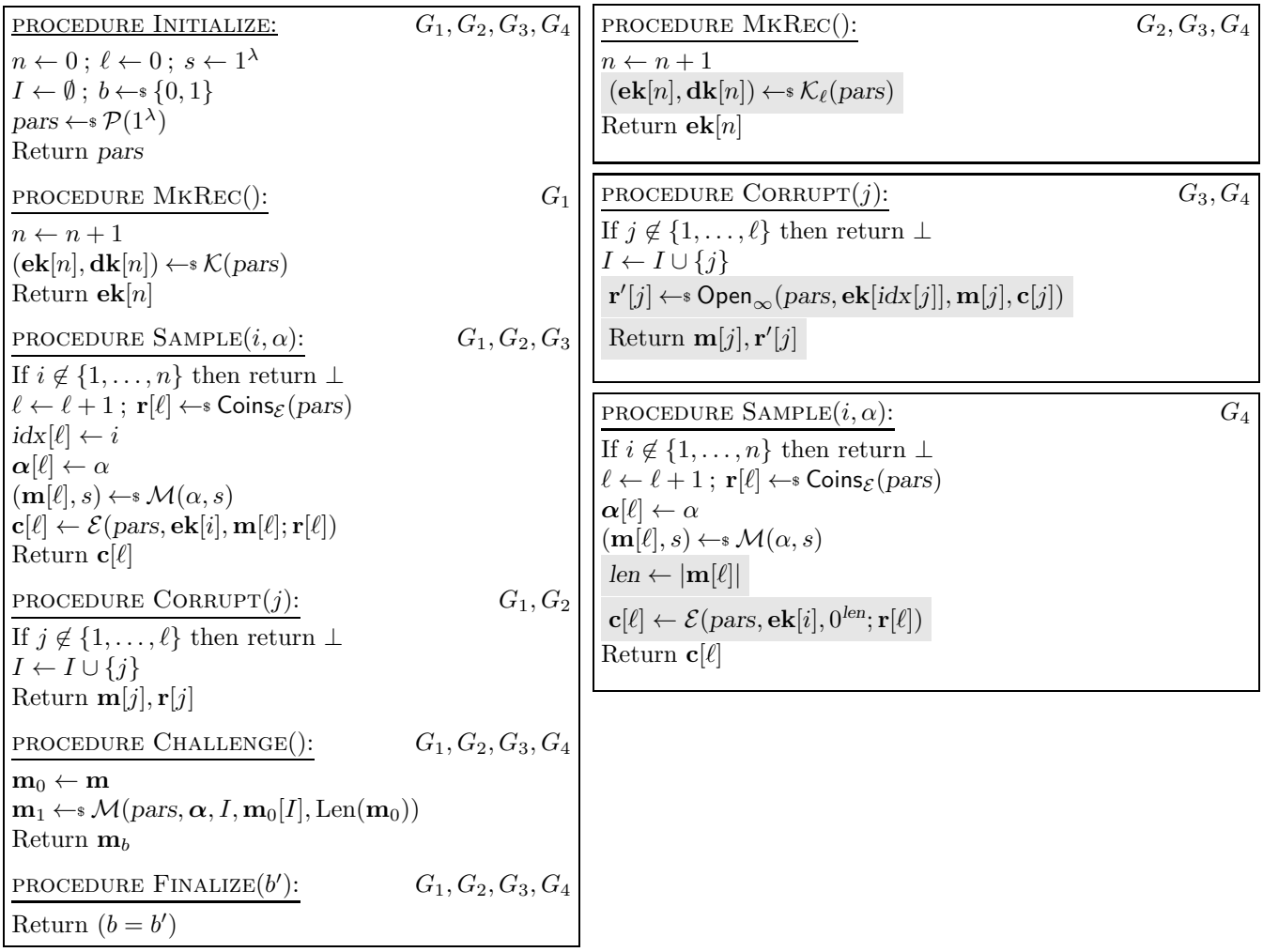

Figure 11: Games G_1 through G_4 for the proof of Theorem 6.1.

particular receiver, and q_c queries to CORRUPT. Then, there exists an efficient key-ind adversary B, and an unbounded ind-cpa adversary D such that

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) \leq 2 \cdot q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + 2 \cdot q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + 4 \cdot \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) \ . \ \blacksquare$$

Theorem 6.2 [Lossy Encryption with efficient opening implies SS-SOA-C] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be any lossy public-key encryption scheme with associated lossy key generation algorithm $\mathcal{K}_\ell$ and efficient opening algorithm Open; let $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$. Let $\mathcal{M}$ be a PT message sampler, $\mathcal{R}$ a PT relation, and A a PT adversary making q_{mk} queries to MKREC, q_s queries to SAMPLE of which at most q_{spr} are for any particular receiver, and q_c queries to CORRUPT. Then, there exists a PT simulator S, a PT key-ind adversary B, and an unbounded ind-cpa adversary D such that

$$\mathbf{Adv}^{\text{ss-soa-c}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \leq q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) \ . \ \blacksquare$$

Proof of Theorem 6.1: We will prove the theorem using a sequence of game transitions, starting with a game G_0, which is the same as game $\text{IndSoaC}_{\mathcal{AE},\mathcal{M}}(\lambda)$.

Adversary $C(pars)$:

$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$; $b \leftarrow\!\!\$\ \{0,1\}$
Run $A(pars)$.

On query MkRec():

$n \leftarrow n+1$
$\mathbf{ek}[n] \leftarrow \text{MkRec}_B()$
Return $\mathbf{ek}[n]$

On query Sample(i, α):

If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$; $idx[\ell] \leftarrow i$
$(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$len \leftarrow |\mathbf{m}[\ell]|$
$\mathbf{c}[\ell] \leftarrow \text{LR}_B(i, \mathbf{m}[\ell], 0^{len})$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):

If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{r}'[j] \leftarrow\!\!\$\ \mathsf{Open}_\infty(pars, \mathbf{ek}[idx[j]], \mathbf{m}[j], \mathbf{c}[j])$
Return $\mathbf{m}[j], \mathbf{r}'[j]$

On query Challenge():

$\mathbf{m}_0 \leftarrow \mathbf{m}$
$\mathbf{m}_1 \leftarrow\!\!\$\ \mathcal{M}(pars, \boldsymbol{\alpha}, I, \mathbf{m}_0[I], \text{Len}(\mathbf{m}))$
Return $\mathbf{m}_b$

When A halts with output b',
halt and output 1 if $(b = b')$ and 0 otherwise.

Figure 12: Adversary used in the proof of Theorem 6.1.

Thus, by definition

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) = 2 \cdot \Pr\left[\, G_0^A \Rightarrow \mathsf{true} \,\right] - 1 \,. \tag{14}$$

We will bound this probability by gradually changing game G_0 until we end up in a game in which A has no advantage. The first game transition replaces the call to the possible unbounded Resamp in the Challenge procedure with a call to the PT resample mode of $\mathcal{M}$. This gives us

$$\Pr\left[\, G_0^A \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_1^A \Rightarrow \mathsf{true} \,\right] \leq \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) \,. \tag{15}$$

The next game, game G_2, (see Figure 11 for games G_1 through G_4) replaces all of the public keys returned by MkRec with lossy public keys. A standard hybrid argument shows that there exists a PT key-ind adversary B such that

$$\Pr\left[\, G_1 \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_2 \Rightarrow \mathsf{true} \,\right] \leq q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) \,. \tag{16}$$

Adversary B, given a challenge public-key ek that is either real or lossy, randomly chooses a value $t \in [q_{\mathrm{mk}}]$ and runs adversary A as in game G_0, but replacing $\mathbf{ek}[t]$ with the challenge ek, and using the lossy key generation algorithm $\mathcal{K}_\ell$ to generate all keys $\mathbf{ek}[t']$ for $t' > t$. Notice that since adversary B needs to be efficient, we need the resampling procedure to also be efficient; this is the only place in our proof where we need this requirement.

In our next game, G_3, we modify the CORRUPT procedure so that instead of opening with the actual coins $\mathbf{r}[j]$ used to encrypt in SAMPLE, it uses an unbounded opening algorithm Open_∞ to find just-as-likely coins $\mathbf{r}'[j]$ and returns those to the adversary. From the adversary's point of view, these coins are equally-likely by the definition of an opening algorithm, so

$$\Pr[\, G_2 \Rightarrow \mathsf{true} \,] = \Pr[\, G_3 \Rightarrow \mathsf{true} \,] \ . \tag{17}$$

Next, for game G_4 we modify the SAMPLE procedure so that after sampling a message $\mathbf{m}[\ell]$ it only uses the length of the message and instead encrypts a dummy message of the appropriate length. We emphasize that in the CORRUPT procedure we still open to the actual message that was sampled.

We can bound the gap between games G_3 and G_4 using the ind-cpa advantage of an unbounded adversary against the PKE scheme $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$. To see this, first consider adversary C, playing game IndCpaMr, and shown in Figure 12. If the challenge bit is 0, C perfectly simulates game G_3 for A, while if the challenge bit is 1 C perfectly simulates game G_4. Thus, it follows that

$$\Pr[\, G_3 \Rightarrow \mathsf{true} \,] - \Pr[\, G_4 \Rightarrow \mathsf{true} \,] \leq \mathbf{Adv}^{\text{ind-cpa-mr}}_{C,\mathcal{AE}_\ell}(\lambda) \ , \tag{18}$$

where adversary C makes q_{mk} queries to MKREC and at most q_{spr} queries to LR on any one index. Applying Proposition 2.1, there is an unbounded adversary D such that

$$\Pr\left[\, G_3^A \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_4^A \Rightarrow \mathsf{true} \,\right] \leq q_{\mathrm{mk}} \cdot q_{\mathrm{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) \ , \tag{19}$$

where D makes one LR query in game IndCpa.

Next, we modify game G_4 so that instead of using the resample mode of the message sampler $\mathcal{M}$ it instead uses the Resamp procedure, which resamples perfectly but may not be PT. We then see that

$$\Pr\left[\, G_4^A \Rightarrow \mathsf{true} \,\right] - \Pr\left[\, G_5^A \Rightarrow \mathsf{true} \,\right] \leq \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) \ . \tag{20}$$

Finally, we claim that

$$\Pr\left[\, G_5^A \Rightarrow \mathsf{true} \,\right] = \frac{1}{2} \ . \tag{21}$$

This is true since A is given no information about $\mathbf{m}_0$ other than the lengths of all of the messages and the messages at corrupted indices I, since dummy messages of the appropriate lengths are encrypted. Thus, the resampled message $\mathbf{m}_1$, which has the same lengths and same messages at the corrupted indices I, is by definition exactly equally likely.

Combining the above equations, we can see that

$$
\begin{aligned}
&\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) \\
&\quad = \; 2 \cdot \Pr\left[\, G_0^A \Rightarrow \mathsf{true} \,\right] - 1 \\
&\quad \le \; 2 \cdot \left(\mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + \Pr\left[\, G_1^A \Rightarrow \mathsf{true} \,\right]\right) - 1 \\
&\quad \le \; 2 \cdot \left(\mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + \Pr\left[\, G_2^A \Rightarrow \mathsf{true} \,\right]\right) - 1 \\
&\quad \le \; 2 \cdot \left(\mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + \Pr\left[\, G_3^A \Rightarrow \mathsf{true} \,\right]\right) - 1 \\
&\quad \le \; 2 \cdot \left(\mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + \Pr\left[\, G_4^A \Rightarrow \mathsf{true} \,\right]\right) - 1 \\
&\quad \le \; 2 \cdot \left(2 \cdot \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + \Pr\left[\, G_5^A \Rightarrow \mathsf{true} \,\right]\right) - 1 \\
&\quad \le \; 2 \cdot \left(2 \cdot \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda) + \frac{1}{2}\right) - 1 \\
&\quad \le \; 2 \cdot q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + 2 \cdot q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + 4 \cdot \mathbf{Adv}^{\text{resamp}}_{\mathcal{M}}(\lambda)
\end{aligned}
$$

which proves the theorem. ∎

Proof of Theorem 6.2: We will prove the theorem using a sequence of game transitions, starting with a game G_0, which is the same as game SsSoaCReal. All of the game transitions are shown in Figure 13.

First, recall that

$$\mathbf{Adv}^{\text{ss-soa-c}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) = \Pr\left[\, \text{SsSoaCReal}^A_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1 \,\right] - \Pr\left[\, \text{SsSoaIdl}^S_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1 \,\right] \; . \tag{22}$$

We let game G_0 be identical to game $\text{SsSoaCReal}_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda)$. Game G_1 differs from G_0 only in that MkRec uses the lossy key generation algorithm $\mathcal{K}_\ell$ instead of $\mathcal{K}$. A standard hybrid argument shows that there exists an efficient key-ind adversary B such that

$$\Pr\left[\, G_0 \Rightarrow 1 \,\right] - \Pr\left[\, G_1 \Rightarrow 1 \,\right] \le q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{B,\mathcal{AE},\mathcal{K}_\ell}(\lambda) \; . \tag{23}$$

Adversary B, given a challenge public-key ek that is either real or lossy, randomly chooses a value $t \in [q_{\text{mk}}]$ and runs adversary A as in game G_0, but replacing $\mathbf{ek}[t]$ with the challenge ek, and using the lossy key generation algorithm $\mathcal{K}_\ell$ to generate all keys $\mathbf{ek}[t']$ for $t' > t$.

Next, we change G_1 into G_2 by changing the Corrupt procedure so that on a query j, instead of returning the actual coins $\mathbf{r}[j]$ used in Sample, it finds equally-likely coins $\mathbf{r}'[j]$ that still open the jth ciphertext to $\mathbf{m}[j]$ by running a (possibly unbounded) opening algorithm Open_∞. Since these coins are just as likely given the view of the adversary, it follows that

$$\Pr\left[\, G_1 \Rightarrow 1 \,\right] = \Pr\left[\, G_2 \Rightarrow 1 \,\right] \; . \tag{24}$$

The next game, G_3, modifies the Sample procedure to encrypt dummy messages of the appropriate length instead of the messages sampled by $\mathcal{M}$. However, importantly, Corrupt still opens the ciphertexts to the actual messages sampled by $\mathcal{M}$. We can bound the gap between games G_2 and G_3 using the ind-cpa advantage of an unbounded adversary against $\mathcal{AE}_\ell = (\mathcal{P}, \mathcal{K}_\ell, \mathcal{E}, \mathcal{D})$, the PKE scheme with the lossy key generation algorithm. To see this, first consider adversary C, playing game IndCpaMr, and shown in Figure 14. If the challenge bit is 0, C perfectly simulates game G_2 for A, while if the challenge bit is 1

PROCEDURE INITIALIZE: All Games

$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$
$pars \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
Return $pars$

PROCEDURE MKREC(): G_0

$n \leftarrow n + 1$
$(\mathbf{ek}[n], \mathbf{dk}[n]) \leftarrow\!\!\$\ \mathcal{K}(pars)$
Return $\mathbf{ek}[n]$

PROCEDURE SAMPLE(i, α): G_0, G_1

If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$; $\mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars)$
$idx[\ell] \leftarrow i$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$
$(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], \mathbf{m}[\ell]; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

PROCEDURE CORRUPT(j): G_0, G_1

If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
Return $\mathbf{m}[j], \mathbf{r}[j]$

PROCEDURE FINALIZE(w): All Games

Return $\mathcal{R}(1^\lambda, \mathbf{m}, I, w, \boldsymbol{\alpha})$

PROCEDURE MKREC(): G_1, G_2, G_3, G_4

$n \leftarrow n + 1$
$(\mathbf{ek}[n], \mathbf{dk}[n]) \leftarrow\!\!\$\ \mathcal{K}_\ell(pars)$
Return $\mathbf{ek}[n]$

PROCEDURE CORRUPT(j): G_2, G_3

If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{r}'[j] \leftarrow\!\!\$\ \mathsf{Open}_\infty(pars, \mathbf{ek}[idx[j]], \mathbf{m}[j], \mathbf{c}[j])$
Return $\mathbf{m}[j], \mathbf{r}'[j]$

PROCEDURE SAMPLE(i, α): G_3, G_4

If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$; $\mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars)$
$idx[\ell] \leftarrow i$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$
$(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$len \leftarrow |\mathbf{m}[\ell]|$
$\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], 0^{len}; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

PROCEDURE CORRUPT(j): G_4

If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{r}'[j] \leftarrow\!\!\$\ \mathsf{Open}(pars, \mathbf{ek}[idx[j]], \mathbf{dk}[idx[j]], \mathbf{m}[j], 0^{|\mathbf{m}[j]|}, \mathbf{r}[j]\mathbf{c}[j])$
Return $\mathbf{m}[j], \mathbf{r}'[j]$

Figure 13: Games for proof of Theorem 6.2.

C perfectly simulates game G_3. Thus, it follows that

$$\Pr\left[\, G_2 \Rightarrow 1 \,\right] - \Pr\left[\, G_3 \Rightarrow 1 \,\right] \leq \mathbf{Adv}^{\text{ind-cpa-mr}}_{C,\mathcal{AE}_\ell}(\lambda)\ , \tag{25}$$

where adversary C makes q_{mk} queries to MKREC and at most q_{spr} queries to LR on any one index. Applying Proposition 2.1, there is an unbounded adversary D such that

$$\Pr\left[\, G_2^A \Rightarrow 1 \,\right] - \Pr\left[\, G_3^A \Rightarrow 1 \,\right] \leq q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda)\ , \tag{26}$$

where D makes one LR query in game IndCpa.

The final game transition is to game G_4, where the CORRUPT procedure (shown in Figure 13) uses the efficient opening algorithm Open instead of the potentially unbounded opening algorithm. Algorithm Open takes a few extra inputs, which the game chooses. Since by definition algorithms Open_∞ and Open have the same output distribution, the view of an adversary playing the game is identical and we see that

$$\Pr\left[\, G_3^A \Rightarrow 1 \,\right] = \Pr\left[\, G_4^A \Rightarrow 1 \,\right]\ . \tag{27}$$

Adversary $C(pars)$:

$n \leftarrow 0\,;\ \ell \leftarrow 0\,;\ s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$
Run $A(pars)$.

On query MkRec():
$n \leftarrow n+1$
$\mathbf{ek}[n] \leftarrow \text{MkRec}_B()$
Return $\mathbf{ek}[n]$

On query Sample(i,α):
If $i \notin \{1,\ldots,n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha\,;\ idx[\ell] \leftarrow i$
$(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$len \leftarrow |\mathbf{m}[\ell]|$
$\mathbf{c}[\ell] \leftarrow \text{LR}_B(i, \mathbf{m}[\ell], 0^{len})$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):
If $j \notin \{1,\ldots,\ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{r}'[j] \leftarrow\!\!\$\ \mathsf{Open}_\infty(pars, \mathbf{ek}[idx[j]], \mathbf{m}[j], \mathbf{c}[j])$
Return $\mathbf{m}[j], \mathbf{r}'[j]$

When A halts with output w,
halt and output $\mathcal{R}(1^\lambda, \mathbf{m}, I, w, \boldsymbol{\alpha})$

Simulator $S(1^\lambda)$:

$n \leftarrow 0\,;\ \ell \leftarrow 0\,;\ s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$
$pars \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
Run $A(pars)$.

On query MkRec():
$n \leftarrow n+1$
$\text{MkRec}_S()$
$(\mathbf{ek}[n], \mathbf{dk}[n]) \leftarrow \mathcal{K}_\ell(pars)$
Return $\mathbf{ek}[n]$

On query Sample(i,α):
If $i \notin \{1,\ldots,n\}$ then return $\perp$
$\ell \leftarrow \ell + 1\,;\ \mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars)$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha\,;\ idx[\ell] \leftarrow i$
$len \leftarrow \text{Sample}_S(i, \alpha)$
$\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], 0^{len}; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):
If $j \notin \{1,\ldots,\ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
$\mathbf{m}[j] \leftarrow \text{Corrupt}_S(j)$
$\mathbf{r}'[j] \leftarrow\!\!\$\ \mathsf{Open}(pars, \mathbf{ek}[idx[j]], \mathbf{dk}[idx[j]], \mathbf{m}[j], 0^{|\mathbf{m}[j]|}, \mathbf{r}[\ell], \mathbf{c}[j])$
Return $\mathbf{m}[j], \mathbf{r}'[j]$

When A halts with output w,
halt and output w

Figure 14: Adversary and simulator used in the proof of Theorem 6.2.

Finally, we note that running game G_4 with adversary A is the same as running game SsSoaIdl with the simulator S shown in Figure 14, meaning that

$$\Pr\left[\, G_4^A \Rightarrow 1 \,\right] = \Pr\left[\, \text{SsSoaIdl}^S_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1 \,\right] \ . \tag{28}$$

This is because to encrypt dummy messages, S only needs the message length in the procedure Sample (which it can get from its own Sample oracle) and actual sampled messages are not needed until Corrupt, where they can be learned by S by querying its own Corrupt oracle. Also in Corrupt, S can run the efficient opening algorithm Open since it chooses all of the inputs itself.

Combining the above equations, we can see that

$$\begin{aligned}
\mathbf{Adv}^{\text{ss-soa-c}}_{A,S,\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) &= \Pr\left[\, \text{SsSoaCReal}^A_{\mathcal{AE},\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1 \,\right] - \Pr\left[\, \text{SsSoaIdl}^S_{\mathcal{M},\mathcal{R}}(\lambda) \Rightarrow 1 \,\right] \\
&= \Pr\left[\, G_0^A \Rightarrow 1 \,\right] - \Pr\left[\, G_4^A \Rightarrow 1 \,\right] \\
&= (\Pr\left[\, G_0^A \Rightarrow 1 \,\right] - \Pr\left[\, G_1^A \Rightarrow 1 \,\right]) + (\Pr\left[\, G_1^A \Rightarrow 1 \,\right] - \Pr\left[\, G_2^A \Rightarrow 1 \,\right]) + \\
&\quad (\Pr\left[\, G_2^A \Rightarrow 1 \,\right] - \Pr\left[\, G_3^A \Rightarrow 1 \,\right]) + (\Pr\left[\, G_3^A \Rightarrow 1 \,\right] - \Pr\left[\, G_4^A \Rightarrow 1 \,\right]) \\
&\leq q_{\text{mk}} \cdot \mathbf{Adv}^{\text{key-ind}}_{A,\mathcal{AE},\mathcal{K}_\ell}(\lambda) + 0 + q_{\text{mk}} \cdot q_{\text{spr}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{D,\mathcal{AE}_\ell}(\lambda) + 0
\end{aligned}$$

which proves the theorem. ∎

7 IND-SOA-C for Special Message Distributions

While it is an open question whether IND-CPA $\subseteq$ IND-SOA-C in general, we can prove it is the case when we restrict to certain message distributions.

ILC Message Samplers. We will prove that IND-CPA implies IND-SOA-C when the message sampler is what we call independent and length-consistent (ILC). More formally, we say a message sampler $\mathcal{M}$ is *independent* if for all *pars* and for all α, the output of $\mathcal{M}(pars, \alpha, \varepsilon)$ is a pair (m, ε). In other words, $\mathcal{M}$ does not output any saved state s. We say a message sampler $\mathcal{M}$ is *length-consistent* if for all *pars* and for all α there exists a $k \in \mathbb{N}$ such that the output of $\mathcal{M}(pars, \alpha, \varepsilon)$ is a pair (m, s) such that $|m| = k$. In other words, if $\mathcal{M}$ is continually run on the same inputs *pars*, α, and $s = \varepsilon$ (but with different coins) it always outputs messages of the same length. If $\mathcal{M}$ is independent and length-consistent, we say it is ILC.

If a PT message sampler $\mathcal{M}$ is ILC, then we can see that it is also efficiently resamplable with resampling error exactly 0: for each $i \notin I$, algorithm $\mathcal{M}(pars, \boldsymbol{\alpha}, I, \mathbf{x}, \mathbf{len})$ simply runs $\mathcal{M}(pars, \boldsymbol{\alpha}[i], \varepsilon)$ for each $i \in [|\boldsymbol{\alpha}|] \setminus I$. (Notice that if $\mathcal{M}$ is not length-consistent, then this resampling will not necessarily be PT.)

Let IND-SOA-C-I denote the set of all encryption schemes $\mathcal{AE}$ with the property that $\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\cdot)$ is negligible for every PT ind-soa-adversary A and every PT ILC message sampler $\mathcal{M}$. We prove the following theorem.

Theorem 7.1 [IND-CPA $\subseteq$ IND-SOA-C-I] Let λ be a security parameter, $\mathcal{AE} = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme, $\mathcal{M}$ be a PT ILC message sampler, and A be a PT ind-soa adversary making q_{mk} queries to MkRec, q_{s} queries to Sample of which at most q_{spr} are for any particular receiver, and q_{c} queries to Corrupt. Then, there exists a PT ind-cpa adversary C such that

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) \leq q_{\text{mk}} \cdot q_{\text{s}} \cdot \mathbf{Adv}^{\text{ind-cpa}}_{C,\mathcal{AE}}(\lambda) \ . \ \blacksquare$$

Proof of Theorem 7.1: Let game $\text{IndSoaC}_{0,\mathcal{AE},\mathcal{M}}(\lambda)$ (resp. $\text{IndSoaC}_{1,\mathcal{AE},\mathcal{M}}(\lambda)$) be the same as IndSoaC except the challenge bit is hardcoded to 0 (resp. 1) and Finalize returns b', the output of the adversary, instead of $(b = b')$. It is easy to see that

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) = \Pr\left[\ \text{IndSoaC}^{A}_{1,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow 1\ \right] - \Pr\left[\ \text{IndSoaC}^{A}_{0,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow 1\ \right] \ . \tag{29}$$

Now, we consider hybrid game H_z for $z \in \{0, \ldots, q_{\text{s}}\}$; the game is shown in Figure 15. In the first hybrid game, H_0, Challenge returns all resampled messages. At the other extreme, hybrid game $H_{q_{\text{s}}}$ has Challenge return all original messages. More generally, hybrid H_z has Challenge return original messages through index z, and resampled messages after that. We emphasize that Challenge *always* returns the original messages for indices in the set I of corrupted messages, regardless of the value of z. Considering these hybrid games, we see that

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) &= \Pr\left[\ \text{IndSoaC}^{A}_{1,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow 1\ \right] - \Pr\left[\ \text{IndSoaC}^{A}_{0,\mathcal{AE},\mathcal{M}}(\lambda) \Rightarrow 1\ \right] \\
&= \Pr\left[\ H^{A}_{0} \Rightarrow 1\ \right] - \Pr\left[\ H^{A}_{q_{\text{s}}} \Rightarrow 1\ \right] \\
&= \sum_{z=1}^{q_{\text{s}}} \Pr\left[\ H^{A}_{z-1} \Rightarrow 1\ \right] - \Pr\left[\ H^{A}_{z} \Rightarrow 1\ \right] \ .
\end{aligned}$$

Game H_z

PROCEDURE INITIALIZE(1^λ):
$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow \varepsilon$; $I \leftarrow \emptyset$
$pars \leftarrow\!\!\$\ \mathcal{P}(1^\lambda)$
Return $pars$

PROCEDURE MKREC():
$n \leftarrow n + 1$
$(\mathbf{ek}[n], \mathbf{dk}[n]) \leftarrow\!\!\$\ \mathcal{K}(pars)$
Return $\mathbf{ek}[n]$

PROCEDURE FINALIZE(b'):
Return b'

PROCEDURE SAMPLE(i, α):
If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$; $\mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars)$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$; $(\mathbf{m}[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(pars, \alpha, s)$
$\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], \mathbf{m}[\ell]; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

PROCEDURE CORRUPT(j):
If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
Return $\mathbf{m}[j], \mathbf{r}[j]$

PROCEDURE CHALLENGE():
For $t = 1$ to $|\boldsymbol{\alpha}|$ do:
  If $t \in I$ then
    $\mathbf{m}^*[t] \leftarrow \mathbf{m}[t]$
  Else
    If $t \leq z$ then $\mathbf{m}^*[t] \leftarrow \mathbf{m}[t]$
    Else $(\mathbf{m}^*[t], s) \leftarrow \mathcal{M}(pars, \boldsymbol{\alpha}[t], \varepsilon)$
Return $\mathbf{m}^*$

Figure 15: Hybrid game H_z used in the Proof of Theorem 7.1

Now, let corr_v^z be the event that in the execution of game H_z with adversary A, the vth index is corrupted. Since the hybrid games only differ in what happens in CHALLENGE, which comes after all CORRUPT queries, it is the case that for all v and for all z, z'

$$\Pr\left[\,\mathsf{corr}_v^z\,\right] = \Pr\left[\,\mathsf{corr}_v^{z'}\,\right] .$$

Now, we claim that for all z

$$\Pr\left[\,H_{z-1}^A \Rightarrow 1 \wedge \mathsf{corr}_z^{z-1}\,\right] = \Pr\left[\,H_z^A \Rightarrow 1 \wedge \mathsf{corr}_z^z\,\right] . \tag{30}$$

Recall that in CHALLENGE game H_{z-1} returns actual messages through index $z-1$ and resampled messages after that, while H_z returns actual messages through index z. However, in the event that index z is corrupted, the zth message returned by CHALLENGE will be the same in the real and resampled case, meaning there is no difference between the games.

Adversary $B(pars)$:

$n \leftarrow 0$; $\ell \leftarrow 0$; $s \leftarrow 1^\lambda$
$I \leftarrow \emptyset$; $z \leftarrow\!\!\$\ [q_s]$
Run $A(pars)$.

On query MkRec():

$n \leftarrow n + 1$
$\mathbf{ek}[n] \leftarrow \text{MkRec}_B()$
Return $\mathbf{ek}[n]$

On query Sample(i, α):

If $i \notin \{1, \ldots, n\}$ then return $\perp$
$\ell \leftarrow \ell + 1$; $\mathbf{r}[\ell] \leftarrow\!\!\$\ \mathsf{Coins}_{\mathcal{E}}(pars, \mathbf{ek}[i])$
$\boldsymbol{\alpha}[\ell] \leftarrow \alpha$
$(\mathbf{m}_0[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
$(\mathbf{m}_1[\ell], s) \leftarrow\!\!\$\ \mathcal{M}(\alpha, s)$
If $\ell = z$ then
  $\mathbf{c}[\ell] \leftarrow \text{LR}_B(i, \mathbf{m}_0[\ell], \mathbf{m}_1[\ell])$
Else $\mathbf{c}[\ell] \leftarrow \mathcal{E}(pars, \mathbf{ek}[i], \mathbf{m}_0[\ell]; \mathbf{r}[\ell])$
Return $\mathbf{c}[\ell]$

On query Corrupt(j):

If $j \notin \{1, \ldots, \ell\}$ then return $\perp$
$I \leftarrow I \cup \{j\}$
If $j = z$ then halt and output $b' \leftarrow\!\!\$\ \{0, 1\}$
Return $\mathbf{m}_0[j], \mathbf{r}[j]$

On query Challenge():

For $t = 1$ to $|\boldsymbol{\alpha}|$ do:
  If $t \in I$ then
    $\mathbf{m}^*[t] \leftarrow \mathbf{m}_0[t]$
  Else
    If $t \leq z$ then $\mathbf{m}^*[t] \leftarrow \mathbf{m}_0[t]$
    Else $\mathbf{m}^*[t] \leftarrow \mathbf{m}_1[t]$
Return $\mathbf{m}^*$

When A halts with output b', halt with the same output b'.

Figure 16: Adversary B in the Proof of Theorem 7.1

Combining this fact with the equation above gives

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) &= \sum_{z=1}^{q_s} \left(\Pr\left[\, H^A_{z-1} \Rightarrow 1 \,\right] - \Pr\left[\, H^A_z \Rightarrow 1 \,\right]\right) \\
&= \sum_{z=1}^{q_s} \left(\Pr\left[\, H^A_{z-1} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z-1}_z \,\right] - \Pr\left[\, H^A_z \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z}_z \,\right]\right) \\
&\quad + \sum_{z=1}^{q_s} \left(\Pr\left[\, H^A_{z-1} \Rightarrow 1 \wedge \mathsf{corr}^{z-1}_z \,\right] - \Pr\left[\, H^A_z \Rightarrow 1 \wedge \mathsf{corr}^{z}_z \,\right]\right) \\
&= \sum_{z=1}^{q_s} \left(\Pr\left[\, H^A_{z-1} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z-1}_z \,\right] - \Pr\left[\, H^A_z \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z}_z \,\right]\right)
\end{aligned}$$

We can construct an ind-cpa adversary B, shown in Figure 16. Adversary B chooses a random z and then runs adversary A, answering its oracle calls similar to as in game H_z, but using its own LR oracle to answer the zth CORRUPT query from A. One key point is that if A then makes a call to CORRUPT(z), then B will be unable to answer and must halt with a random guess bit as output.

Let $\mathrm{IndCpaMr}_0$ (resp. $\mathrm{IndCpaMr}_1$) be the same as IndCpaMr but with the challenge bit set to 0 (resp. 1) and FINALIZE simply forwarding the adversary's output. Then, the IND-CPA advantage of B against $\mathcal{AE}$ is

$$\mathbf{Adv}^{\text{ind-cpa-mr}}_{B,\mathcal{AE}}(\lambda) = \Pr\left[\,\mathrm{IndCpaMr}^{B}_{1,\mathcal{AE}}(\lambda) \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{IndCpaMr}^{B}_{0,\mathcal{AE}}(\lambda) \Rightarrow 1\,\right] . \tag{31}$$

Notice that running B with $\mathrm{IndCpaMr}_1$ is the same as running game H_{z-1} with adversary A as long as index z is not corrupted; if z is corrupted B terminates with a random guess. Similarly, running B with $\mathrm{IndCpaMr}_0$ is the same as running game H_z with A as long as index z is not corrupted. Thus we can see that

$$\begin{aligned}
&\mathbf{Adv}^{\text{ind-cpa-mr}}_{B,\mathcal{AE}}(\lambda) \\
&\quad= \Pr\left[\,\mathrm{IndCpaMr}^{B}_{1,\mathcal{AE}}(\lambda) \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{IndCpaMr}^{B}_{0,\mathcal{AE}}(\lambda) \Rightarrow 1\,\right] \\
&\quad= \frac{1}{q_{\mathrm{s}}}\sum_{z=1}^{q_{\mathrm{s}}}\left(\Pr\left[\,H^{A}_{z-1} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z-1}_{z}\,\right] + \frac{1}{2}\cdot\Pr\left[\,\mathsf{corr}^{z-1}_{z}\,\right] - \Pr\left[\,H^{A}_{z} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z}_{z}\,\right] - \frac{1}{2}\cdot\Pr\left[\,\mathsf{corr}^{z}_{z}\,\right]\right) \\
&\quad= \frac{1}{q_{\mathrm{s}}}\sum_{z=1}^{q_{\mathrm{s}}}\left(\Pr\left[\,H^{A}_{z-1} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z-1}_{z}\,\right] - \Pr\left[\,H^{A}_{z} \Rightarrow 1 \wedge \overline{\mathsf{corr}}^{z}_{z}\,\right]\right) \\
&\quad= \frac{1}{q_{\mathrm{s}}}\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda)
\end{aligned}$$

Applying Proposition 2.1 we then get that there there is an efficient IND-CPA adversary C such that

$$\mathbf{Adv}^{\text{ind-soa-c}}_{A,\mathcal{AE},\mathcal{M}}(\lambda) \le q_{\mathrm{mk}}\cdot q_{\mathrm{s}}\mathbf{Adv}^{\text{ind-cpa}}_{C,\mathcal{AE}}(\lambda) . \tag{32}$$

This completes the proof. ∎

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-SOA-C. 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

[1] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT 2000*, volume 1807 of *LNCS*. Springer, 2000.

[2] M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662. Springer, Apr. 2012.

[3] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, number 5479 in Lecture Notes in Computer Science, pages 1–35. Springer, 2009.

[4] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006.

[5] M. Bellare and P. 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.

[6] M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 235–252. Springer, Mar. 2011.

[7] M. Ben-Or, S. Goldwasser, and A. 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.

[8] F. Böhl, D. Hofheinz, and D. Kraschewski. On definitions of selective opening security. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *Public Key Cryptography – PKC 2012*, volume 7293 of *LNCS*, pages 522–539. Springer, 2012.

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

[10] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. 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.

[11] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Aug. 1997.

[12] R. Canetti, U. Feige, O. Goldreich, and M. 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.

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

[14] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *20th ACM Symposium on Theory of Computing, Proceedings of STOC 1988*, pages 11–19. ACM, 1988.

[15] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Improved non-committing encryption with applications to adaptively secure protocols. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 287–302. Springer, Dec. 2009.

[16] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, Aug. 2000.

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

[18] C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[19] S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 381–402. Springer, May 2010.

[20] D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295. Springer, May 2010.

[21] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology – CRYPTO 1984*, pages 10–18. Springer, 1985.

[22] R. Gennaro and S. Micali. Independent zero-knowledge sets. In M. Bugliese, B. Preneel, V. Sassone, and I. 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.

[23] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[24] B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, Dec. 2011.

[25] D. Hofheinz. Possibility and impossibility results for selective decommitments. IACR ePrint Archive, Apr. 2008.

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

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

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

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

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

[31] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *Advances in Cryptology, Proceedings of CRYPTO 2008*, number 5157 in Lecture Notes in Computer Science, pages 554–571. Springer, 2008.

[32] C. Peikert and B. 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.

[33] A. Rosen and G. Segev. Efficient lossy trapdoor functions based on the composite residuosity assumption. IACR ePrint Archive, Mar. 2008.