A preliminary version of this paper appears in the the Proceedings of Eurocrypt 2012. This is the full version.

Standard Security Does Not Imply Security Against Selective-Opening

Mihir Bellare[1] Rafael Dowsley[2] Brent Waters[3] Scott Yilek[4]

October 2011

Abstract

We show that no commitment scheme that is hiding and binding according to the standard definition is semantically-secure under selective opening attack (SOA), resolving a long-standing and fundamental open question about the power of SOAs. We also obtain the first examples of IND-CPA encryption schemes that are not secure under SOA, both for sender corruptions where encryption coins are revealed and receiver corruptions where decryption keys are revealed. These results assume only the existence of collision-resistant hash functions.

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~mihir`. Supported in part by NSF grants CNS-0627779 and CCF-0915675.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `rdowsley@cs.ucsd.edu`. URL: `http://cseweb.ucsd.edu/~rdowsley`.

[3] Department of Computer Science, University of Texas at Austin, 1616 Guadalupe, Suite 2.408, Austin, TX 78701, USA. Email: `bwaters@cs.utexas.edu`. URL: `http://www.cs.utexas.edu/~bwaters`.

[4] 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`.

Contents

1 Introduction

A *commitment scheme* $\mathcal{E}$ can be applied to a message m and coins r to (deterministically) produce a commitment $c \leftarrow \mathcal{E}(m;r)$ that is sent to a receiver. The sender can later "open" the commitment by providing m, r and the receiver checks that $\mathcal{E}(m;r) = c$. The first security requirement, often called hiding, is formalized as IND-CPA, namely an adversary knowing m_0, m_1 and $\mathcal{E}(m_b;r)$ for random b, r has negligible advantage in computing challenge bit b. The second requirement, binding, asks that it be hard for an adversary to produce r_0, r_1 and *distinct* m_0, m_1 such that $\mathcal{E}(m_0;r_0) = \mathcal{E}(m_1;r_1) \neq \perp$. Let us refer to a commitment scheme as HB-secure (Hiding and Binding) if it satisfies both these properties. HB-security is the standard requirement and HB-secure commitment schemes are a fundamental tool in cryptography in general and in protocol design in particular. HB-secure commitment implies PRGs [31], PRFs [21] and ZK proofs for **NP** [24].

Suppose there are n committers, the i-th computing its commitment $\mathbf{c}[i] \leftarrow \mathcal{E}(\mathbf{m}[i];\mathbf{r}[i])$ to its message $\mathbf{m}[i]$ using coins $\mathbf{r}[i]$, the coins of different committers being of course not only random but also independent of each other. An adversary computes, as a function of the vector $\mathbf{c}$ of commitments, a subset $I \subseteq \{1, \ldots, n\}$ of the senders, and obtains the corresponding openings, namely $\langle \mathbf{m}[i] : i \in I \rangle$ and $\langle \mathbf{r}[i] : i \in I \rangle$. This is called a selective opening attack (SOA). We say that $\mathcal{E}$ is SOA-secure if privacy of the un-opened messages is preserved, meaning the adversary, after its SOA, cannot learn anything about $\langle \mathbf{m}[i] : i \notin I \rangle$ other than it would from possession of $\langle \mathbf{m}[i] : i \in I \rangle$. (That is, the coins are unhelpful.) SOAs arise quite naturally in multi-party cryptographic protocols and SOA-security is desirable in many such settings.

A fundamental question that was posed in this area is whether (standard) HB-security implies SOA-security, meaning, is a HB-secure commitment scheme also SOA-secure? So far, the question has received neither a positive nor a negative answer. Intuitively, the answer would appear to be "yes," for how could the coins accompanying the opened messages help, beyond the opened messages themselves, in revealing something about the un-opened messages? Yet attempts to prove SOA-security of a commitment scheme based on its HB-security have failed. But attempts to find a counter-example have also failed. We do not have a single example, even artificial, of a HB-secure commitment scheme that is demonstrably not SOA-secure. This situation has vexed and intrigued cryptographers for many years and been the subject or inspiration for much work [11, 19, 12, 35, 2, 20, 29, 6, 28].

This paper answers this long-standing open question. We show that the answer is negative. We give an example of a HB-secure commitment scheme which we prove is not SOA-secure. In fact our result is much stronger. It shows that *no* HB-secure commitment scheme is SOA-secure. Given any HB-secure commitment scheme, we present an attack showing it is not SOA-secure. Before going on to our results on encryption let us expand on this result on commitment including its implications and its relation to previous work.

SOA-SECURE COMMITMENT. Dwork, Naor, Reingold and Stockmeyer (DNRS) [19] gave a definition of SOA-secure commitments, henceforth referred to as SS-SOA, that captures semantic security for relations via a simulation-based formalization. Suitable for applications and widely accepted as the right definition, SS-SOA is what we use in our results. We show that no HB-secure commitment scheme is SS-SOA-secure by presenting, for any given HB-secure commitment scheme $\mathcal{E}$, an adversary for which we prove that there is no successful simulator. We do *not* assume the simulation is blackbox. The only assumption made is the existence of a collision-resistant (CR) hash function.

This general result rules out SS-SOA security for particular schemes. For example, a widely employed way to commit to $m \in \mathbb{Z}_p$ is by picking $r \in \mathbb{Z}_p$ at random and returning $\mathcal{E}(m;r) = g^m h^r \in \mathbb{G}$ where g, h are generators of a group $\mathbb{G}$ of prime order p [36]. This scheme is binding if the DL problem is hard in $\mathbb{G}$ and it is unconditionally hiding. Our results imply that it is not SS-SOA secure. They yield a specific attack, in the form of an adversary for which there is no simulator. Since CR hash functions exist if DL is hard, one does not even need extra assumptions. We stress that this is just an example; our result rules out SS-SOA security for *all* HB-secure schemes.

Implications for IND-SOA-CRS. An indistinguishability-based definition of SOA-secure commitment is given in [2, 29]. It only applies when the message vector **m** is drawn from what's called a "conditionally re-samplable (CRS) distribution," and accordingly we denote it IND-SOA-CRS. This definition is of limited use in applications because message distributions there are often not CRS, but for CRS distributions the definition is intuitively compelling and sound.

Letting SS-SOA-CRS denote the restriction of SS-SOA to CRS distributions, [2, 29] had noted that SS-SOA-CRS implies IND-SOA-CRS and asked whether the converse was true. We settle this question in the negative, showing that SS-SOA-CRS is strictly stronger. We arrive at this separation by combining two facts. First, the message distribution underlying our negative result is CRS, meaning we say that there does not exist a HB-secure commitment scheme that is SS-SOA-CRS, not just SS-SOA. Second, it is known that there does exist a HB-secure commitment scheme that is IND-SOA-CRS [2, 29].

Hofheinz [2, 29] shows that any commitment scheme that is *statistically* hiding and binding is IND-SOA-CRS. This positive result does not contradict our result, because, as we have just seen (indeed, invoking this positive result to do so), IND-SOA-CRS is a strictly weaker requirement than SS-SOA or SS-SOA-CRS. A question that still remains open is whether HB-security implies IND-SOA-CRS security.

Message distribution. It has been suggested that the difficulty in showing that HB-security implies SS-SOA is that the messages in the vector **m** may be related to each other. Our results imply that although showing HB-security implies SS-SOA-security is not just hard but impossible, it is not for this reason. We have already noted that our negative result holds for a message distribution that is CRS. In fact, the message distribution is uniform, meaning the messages in the vector are uniformly and independently distributed strings. Even for this uniform distribution, no HB-secure commitment scheme is SS-SOA secure. This may at first glance appear to contradict known results, for DNRS [19] showed that HB-security implied SOA-security for independently distributed messages. The difference is that they only showed this for what they called semantic security for functions, a notion implied by, but not known to imply their main notion of semantic security for relations that we call SS-SOA. Thus, not only is there no contradiction, but our results settle an open question from [19]. Namely we show that their result does not extend to SS-SOA and also that SS-SOA is strictly stronger than semantic security for functions.

Random oracles. Our result holds in the standard model and in the non-programmable random oracle (RO) model [32]. (In the latter the simulator is given oracle access to the RO and cannot define it.) In the standard (programmable) RO model [4], where the simulator can define the RO, our result is not true: there *do* exist HB-secure schemes that are SS-SOA secure. As an example, commitment scheme $\mathcal{E}^H(m; r) = H(m; r)$, where H is the RO, is HB-secure in the non-programmable RO. Our results show it is not SS-SOA in this model. However, it can be easily shown SS-SOA in the programmable RO model. Consequently, our results yield another separation between the programmable and non-programmable RO models complementing that of [32].

Previous negative results. Hofheinz [2, 29] shows that no HB-secure scheme can be proven SS-SOA secure via blackbox reduction to "standard" assumptions. (A "standard" assumption as defined in [17, 2, 29] is one specified by a certain type of game.) However, it might still be possible to prove that a particular HB-secure scheme was SS-SOA in some ad hoc and non-blackbox way. The blackbox separation does not yield a single example of an HB-secure scheme that is not SS-SOA secure, let alone show, as we do, that all HB-secure schemes fail to be SS-SOA secure.

Interaction. Our result applies to non-interactive commitment schemes. When commitment involves an interactive protocol between sender and receiver the corresponding claim is not true. There *does* exist an interactive HB and SS-SOA secure commitment scheme. Specifically, Hofheinz [2, 29] presents a particular construction of such a scheme based on one-way permutations. Further results on interactive SOA-secure commitment are [39, 34].

SOA-secure encryption for sender corruptions. Turning now to encryption, consider a setting with n senders and one receiver, the latter having public encryption key ek. Sender i picks random coins $\mathbf{r}[i]$, encrypts its message $\mathbf{m}[i]$ via $\mathbf{c}[i] \leftarrow \mathcal{E}(ek, \mathbf{m}[i]; \mathbf{r}[i])$, and sends ciphertext $\mathbf{c}[i]$ to the receiver. The adversary selects, as a function of $\mathbf{c}$, a set $I \subseteq \{1, \ldots, n\}$ of the senders and corrupts them, obtaining their messages $\langle \mathbf{m}[i] : i \in I \rangle$ and coins $\langle \mathbf{r}[i] : i \in I \rangle$. As before, we say that $\mathcal{E}$ is SOA-secure if privacy of the un-opened messages is preserved. An SS-SOA definition analogous to the one for commitment was given in [2, 7].

The standard and accepted security condition for encryption since [26] is of course IND-CPA. SOA-security was identified upon realizing that it is necessary to implement the assumed-secure channels in multi-party secure computation protocols like those of [8, 13]. The central open question was whether or not IND-CPA implies SS-SOA. Neither a proof showing the implication is true, nor a counter-example showing it is false, had been given. We show that IND-CPA does not imply SS-SOA by exhibiting a large class of IND-CPA encryption schemes that we prove are not SS-SOA. The class includes many natural and existing schemes.

DNRS [19] had pointed out that the obstacle to proving that IND-CPA implies SS-SOA is that most encryption schemes are "committing." Our results provide formal support for this intuition. We formalize a notion of binding-security for encryption. Our result is that no binding encryption scheme is SS-SOA secure. As with commitment, it holds when the distribution on messages is uniform.

The existence of a decryption algorithm corresponding to the encryption algorithm means that for any ek created by honest key-generation, there do not exist r_0, r_1 and distinct m_0, m_1 such that $\mathcal{E}(ek, m_0; r_0) = \mathcal{E}(ek, m_1; r_1)$. Binding strengthens this condition to also hold when ek is adversarially chosen, while also relaxing it from unconditional to computational. It is thus a quite natural condition and is met by many schemes.

Inability to show that IND-CPA implies SS-SOA led to the search for specific SS-SOA secure encryption schemes. Non-commiting encryption [11] yields a solution when the number of bits encrypted is bounded by the length of the public key. The first full solution was based on lossy encryption [2, 7]. Deniable encryption [10] was used to obtain further solutions [20, 6]. More lossy-encryption based solutions appear in [28]. In all these solutions, the encryption scheme is *not* binding. Our results show that this is necessary to achieve SS-SOA security.

SOA-security has so far been viewed as a theoretical rather than practical issue because even if there was no proof that IND-CPA implies SS-SOA, there were no attacks on standard, practical schemes such as ElGamal. Our results change this situation for they show that ElGamal and other practical schemes are not SS-SOA secure. Thus, the above-mentioned schemes that achieve SS-SOA in more involved ways are necessary if we want SS-SOA security.

IND-CCA doesn't help: The Cramer-Shoup scheme [14] meets our definition of binding and is thus not SS-SOA secure. As with commitment, our results imply that IND-SOA-CRS security is *strictly* weaker than SS-SOA-CRS security, answering an open question from [2, 7]. Subsequent to our work, the relations between different notions of SOA-security under sender corruptions were further clarified in [9] but whether there exist schemes that are IND-CPA but not IND-SOA-CRS secure remains open.

SOA-SECURE ENCRYPTION FOR RECEIVER CORRUPTIONS. In a dual of the above setting, there are n receivers and one sender, receiver i having public encryption key $\mathbf{ek}[i]$ and secret decryption key $\mathbf{dk}[i]$. For each i the sender picks random coins $\mathbf{r}[i]$, encrypts message $\mathbf{m}[i]$ via $\mathbf{c}[i] \leftarrow \mathcal{E}(\mathbf{ek}[i], \mathbf{m}[i]; \mathbf{r}[i])$, and sends ciphertext $\mathbf{c}[i]$ to receiver i. The adversary selects, as a function of $\mathbf{c}$, a set $I \subseteq \{1, \ldots, n\}$ of the receivers and corrupts them, obtaining not only the messages $\langle \mathbf{m}[i] : i \in I \rangle$ but also the decryption keys $\langle \mathbf{dk}[i] : i \in I \rangle$. As usual, we say that $\mathcal{E}$ is SOA-secure if privacy of the un-opened messages is preserved. An SS-SOA definition analogous to the ones for commitment and sender-corruptions in encryption is given in Section 5.

The status and issues are analogous to what we have seen above, namely that it has been open whether IND-CPA security implies SS-SOA for receiver corruptions, neither a proof nor a counter-example ever being given. We settle this with the first counter-examples. We define a notion of decryption verifiability for encryption that can be seen as a weak form of robustness [1]. It asks that there is an algorithm $\mathcal{W}$

such that it is hard to find ek, dk_0, dk_1 and distinct m_0, m_1 such that $\mathcal{W}(ek, dk_0, m_0)$ and $\mathcal{W}(ek, dk_1, m_1)$ both accept. We show that no IND-CPA and decryption-verifiable encryption scheme is SS-SOA secure. Standard encryption schemes like ElGamal are decryption verifiable (even though they are not robust) so our result continues to rule out SS-SOA security for many natural schemes.

Non-committing encryption [11] yields an SS-SOA scheme secure for receiver corruptions when the number of bits encrypted is bounded by the length of the secret key. Nielsen [32] showed that any non-committing encryption scheme has keys larger than the total number of message bits it can securely encrypt. This result is not known to extend to SS-SOA, meaning the existence of an SS-SOA scheme for receiver corruptions without this restriction is open. Our results do not rule out such a full solution but indicate that the scheme must not be decryption-verifiable.

2 Technical approach

We provide a high-level description of our approach, focusing for simplicity on commitment schemes and the claim that no HB-secure commitment scheme is SS-SOA secure. We then discuss extensions and variants of our results.

The definition. Let $\mathcal{E}$ be a commitment scheme. To compact notation, we extend it to vector inputs by letting $\mathcal{E}(\mathbf{m}; \mathbf{r})$ be the vector whose i-th component is $\mathcal{E}(\mathbf{m}[i]; \mathbf{r}[i])$. Let $\mathcal{M}$ be a message sampler that outputs a vector $\mathbf{m}$ of messages and let R be a relation. Adversary A, given ciphertext vector $\mathbf{c} = \mathcal{E}(\mathbf{m}; \mathbf{r})$ will corrupt a subset I of the senders, get their messages and coins, and output a value w. It is said to win if $\mathsf{R}(\mathbf{m}, I, w)$ is true. The simulator, given no ciphertexts, can also corrupt a subset I of senders but gets back only the corresponding messages, and outputs a value w. It too is said to win if $\mathsf{R}(\mathbf{m}, I, w)$ is true. Security requires that for every $\mathcal{M}, \mathsf{R}$ and adversary A there is a simulator S such that S wins with about the same probability as A. DNRS [19, Sec 7.1] require this to be true even for any auxiliary input a given initially to A and also to S. See Section 4 for a formal definition.

The attack. Let $\mathcal{E}$ be any, given HB-secure commitment scheme. We construct $\mathcal{M}, \mathsf{R}, A$ for which we prove there is no simulator. We let $\mathcal{M}$ output $n = 2h$ randomly and independently distributed messages, each of length ℓ. Our adversary A applies to the vector $\mathbf{c} = \mathcal{E}(\mathbf{m}; \mathbf{r})$ of commitments a hash function H to get back an h-bit string $b[1] \ldots b[h]$ and then corrupts the set of indices $I = \{2j - 1 + b[j] \,:\, 1 \le j \le h\}$ to get back $\langle \mathbf{m}[i] : i \in I \rangle$ and $\langle \mathbf{r}[i] : i \in I \rangle$. Its output w consists of $\mathbf{c}$ and $\langle \mathbf{r}[i] : i \in I \rangle$. We define R, on inputs $\mathbf{m}, I$ and w, to check two constraints. The *opening constraint* is that $\mathcal{E}(\mathbf{m}[i]; \mathbf{r}[i]) = \mathbf{c}[i]$ for all $i \in I$. The *hash constraint* is that $I = \{2j - 1 + b[j] \,:\, 1 \le j \le h\}$ for $b[1] \ldots b[h] = H(\mathbf{c})$. A detailed description of A and R is in Figure 3.

The simulator gets no ciphertexts. It must corrupt some set I of indices to get back $\langle \mathbf{m}[i] : i \in I \rangle$. Now it must create a ciphertext vector $\mathbf{c}$ and a list $\langle \mathbf{r}[i] : i \in I \rangle$ of coins to output as w to R, and to satisfy the latter it must satisfy both constraints. Intuitively, the simulator faces a Catch-22. It is helpful for the intuition to think of H as a random oracle. The simulator could first pick I in some way, get $\langle \mathbf{m}[i] : i \in I \rangle$ from its oracle, and compute $\mathbf{c}$ and $\langle \mathbf{r}[i] : i \in I \rangle$ to satisfy the opening constraint. But it is unlikely, given only poly$(\cdot)$ queries to H, to satisfy the hash constraint. On the other hand it could pick some $\mathbf{c}$, define I to satisfy the hash constraint, and get $\langle \mathbf{m}[i] : i \in I \rangle$ from its oracle. But now it would have a hard time satisfying the opening constraint *because the commitment scheme is binding.*

This intuition that the simulator's task is hard is, however, not a proof that a simulator does not exist. Furthermore, the intuition relies on the hash function being a random oracle and we only want to assume collision-resistance. Our proof takes an arbitrary simulator and proves that the probability that it makes the relation true is small unless it finds a hash collision or violates binding. The proof involves backing up the simulator, feeding it different, random responses to its corruption query, and applying a Reset Lemma analogous to that of [3]. We do not assume the simulation is blackbox. See Theorem 4.1.

Related work. The strategy of specifying challenges by a hash of commitments arose first in showing failure of parallel-repetition to preserve zero-knowledge [22, 23]. The model, goals and techniques are however quite different. Also in [23] the simulator is assumed to make only blackbox calls to the adversary (verifier) and we make no such assumption, and they use a pairwise independent hash rather than a CR one. We point out that although the seed of our technique can be traced back 20 years it was not noted until now that it could be of use in settling the long-standing open question of whether HB-secure commitments are SS-SOA-secure.

Adaptive security. Our definition of SS-SOA, following [19, 2, 6] is one-shot, meaning the adversary gets all the ciphertexts at once and performs all its corruptions in parallel. A definition where the adversary can make adaptive ciphertext-creation and corruption requests is more suitable for applications. But our result is negative so using a restricted adversary only makes it stronger. (We are saying there is an attack with a one-shot adversary so certainly there is an attack with an adaptive adversary.)

The flip side is that if the adversary is allowed to be adaptive, so is the simulator. Our theorems only consider (and rule out) one-shot simulators for simplicity, but the proofs can be extended to also rule out adaptive simulators. We discuss briefly how to do this following the proof of Theorem 4.1.

Auxiliary inputs. As indicated above, the definition of DNRS [19] that we use allows both the adversary and simulator to get an auxiliary input, denoted "z" in [19, Sec 7.1]. The simplest and most basic form of our result exploits the auxiliary input to store the key describing the CR hash function. (If the simulator can pick this key the function will not be CR.)

Auxiliary inputs model history. They were introduced in the context of zero-knowledge by Goldreich and Oren [25] who showed that in their presence ZK had natural and desirable composability properties absent under the original definition of [27]. They have since become standard in zero-knowledge and also in simulation-based definitions in other contexts [18, 19] to provide composability. Their inclusion in the SS-SOA definition of commitment by DNRS [19] was thus correct and justified and we put them to good use.

Later definitions [2, 29] however appear to have dropped the auxiliary inputs. Although this appears to be only for notational simplicity (modern works on ZK also often drop auxiliary inputs since it is well understood how to extend the definition to include them) it does raise an interesting technical question, namely what negative results can we prove without auxiliary inputs?

A simple solution is to use one of the messages as a key. The adversary would corrupt the corresponding party to get this key, thereby defining the hash function, and then proceed as above. This however makes the adversary adaptive, and while this is still a significant result, we ask whether anything can be shown for one-shot adversaries without using auxiliary inputs.

This turns out to be technically challenging. The difficulty is that the simulator can control the hash key. In Section 6 we present a construction relying on a new primitive we call an encrypted hash scheme (EHS). The idea is that there is an underlying core hash function whose keys are messages and an encrypted hash function whose keys are ciphertexts. We show how to build an EHS based on DDH.

We remark that from a practical perspective these distinctions are moot since hash functions like SHA-256 are keyless. Also, it is possible to work theoretically with keyless hash functions [38]. But in classical asymptotic theoretical cryptography, hash functions are keyed and we were interested in results in this setting.

3 Preliminaries

Notation and conventions. If $n \in \mathbb{N}$ then let 1^n denote the string of n ones and $[n]$ the set $\{1, \ldots, n\}$. The empty string is denoted by ε. By $a \,\|\, b$ we denote the concatenation of strings a, 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. For a set $I \subseteq [|\mathbf{x}|]$ we let $\mathbf{x}[I]$ be the $|\mathbf{x}|$-vector whose i-th component is $\mathbf{x}[i]$ if $i \in I$ and $\perp$ otherwise. We let $\perp_n$ denote the n-vector all of whose components are $\perp$. We define the Embedding subroutine Emb to take 1^n, $I \subseteq [n]$, a $|I|$-vector $\mathbf{x}^*$ and a n-vector $\overline{\mathbf{x}}$ and return the n-vector that consists

```
INITIALIZE($1^\lambda$)
$b \leftarrow_\$ \{0,1\}$ ; $\pi \leftarrow_\$ \mathcal{P}(1^\lambda)$
$(ek, dk) \leftarrow_\$ \mathcal{K}(\pi)$
Return $(\pi, ek)$

LR($m_0, m_1$)
$c \leftarrow_\$ \mathcal{E}(1^\lambda, \pi, ek, m_b)$
Return $c$

FINALIZE($b'$)
Return $(b' = b)$
```

```
INITIALIZE($1^\lambda$)
$\pi \leftarrow_\$ \mathcal{P}(1^\lambda)$
Return $\pi$

FINALIZE($ek, c, m_0, m_1, r_0, r_1$)
$d_0 \leftarrow \mathcal{V}(1^\lambda, \pi, ek, c, m_0, r_0)$
$d_1 \leftarrow \mathcal{V}(1^\lambda, \pi, ek, c, m_1, r_1)$
Return $(d_0 \wedge d_1 \wedge (m_0 \neq m_1))$
```

Figure 1: Game IND_Π (left) and game BIND_Π (right) defining, respectively, IND-CPA privacy and binding security of CE scheme $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$.

of $\overline{\mathbf{x}}$ with $\mathbf{x}^*$ embedded in the positions indexed by I. More precisely,

```
Subroutine Emb($1^n, I, \mathbf{x}^*, \overline{\mathbf{x}}$)
$j \leftarrow 0$ ; For $i = 1, \ldots, n$ do If $i \in I$ then $j \leftarrow j + 1$ ; $\overline{\mathbf{x}}[i] \leftarrow \mathbf{x}^*[j]$
Return $\overline{\mathbf{x}}$.
```

All algorithms are randomized, unless otherwise specified as being deterministic. We use the abbreviation PT for polynomial-time. If A is an algorithm then $y \leftarrow A(x_1, \ldots, x_n; r)$ represents the act of running the algorithm A with inputs $x_1, \ldots, x_n$ and coins r to get an output y and $y \leftarrow_\$ A(x_1, \ldots, x_n)$ represents the act of picking r at random and letting $y \leftarrow A(x_1, \ldots, x_n; r)$. By $[A(x_1, \ldots, x_n)]$ we denote the set of all y for which there exists r such that $y = A(x_1, \ldots, x_n; r)$.

GAMES. We use the language of code-based game-playing [5]. A game (see Figure 1 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, and we let "$\mathrm{G}^A(\lambda)$" denote the event that this game output takes value true.

CE SCHEMES. We introduce CE (Committing Encryption) schemes as a way to unify commitment and encryption schemes under a single syntax and avoid duplicating similar definitions and results for the two cases. A *CE scheme* $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ is specified by four PT algorithms. Via $\pi \leftarrow_\$ \mathcal{P}(1^\lambda)$ the parameter-generation algorithm $\mathcal{P}$ generates system parameters such as a description of a group. Via $(ek, dk) \leftarrow_\$ \mathcal{K}(\pi)$ the key-generation algorithm $\mathcal{K}$ generates an encryption key ek and decryption key dk. Via $c \leftarrow \mathcal{E}(1^\lambda, \pi, ek, m; r)$ the encryption algorithm deterministically maps a message m and coins $r \in \{0,1\}^{\rho(\lambda)}$ to a ciphertext $c \in \{0,1\}^* \cup \{\perp\}$ where ρ: $\mathbb{N} \to \mathbb{N}$ is the *randomness length* associated to Π and $c \neq \perp$ iff $|m| = \ell(\lambda)$ where ℓ: $\mathbb{N} \to \mathbb{N}$ is the *message length* associated to Π. Via $d \leftarrow \mathcal{V}(1^\lambda, \pi, ek, c, m, r)$, deterministic verification algorithm $\mathcal{V}$ returns true or false. We require that $\mathcal{V}(1^\lambda, \pi, ek, \mathcal{E}(1^\lambda, \pi, ek, m; r), m, r) = \mathsf{true}$ for all $\lambda \in \mathbb{N}$, all $\pi \in [\mathcal{P}(1^\lambda)]$, all $(ek, dk) \in [\mathcal{K}(\pi)]$, all $r \in \{0,1\}^{\rho(\lambda)}$ and all $m \in \{0,1\}^*$ such that $\mathcal{E}(1^\lambda, \pi, ek, m; r) \neq \perp$. We say that the verification algorithm $\mathcal{V}$ is *canonical* if $\mathcal{V}(1^\lambda, \pi, ek, c, m, r)$ returns the boolean $(\mathcal{E}(1^\lambda, \pi, ek, m; r) = c \neq \perp)$.

Game IND_Π of Figure 1 captures the standard notion of indistinguishability under chosen-plaintext attack (IND-CPA) [26] and serves to define privacy for CE schemes. The adversary is allowed only one LR query and the messages m_0, m_1 involved must be of the same length. Game BIND_Π captures binding security. For adversaries A, B we let

$$\mathbf{Adv}^{\mathrm{indcpa}}_{\Pi,A}(\lambda) = 2\Pr[\mathrm{IND}^A_\Pi(\lambda)] - 1 \quad \text{and} \quad \mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\lambda) = \Pr[\mathrm{BIND}^B_\Pi(\lambda)] .$$

We say that Π is IND-CPA secure if $\mathbf{Adv}^{\mathrm{indcpa}}_{\Pi,A}(\cdot)$ is negligible for all PT A, *binding* if $\mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\cdot)$ is negligible for all PT B and *perfectly binding* if $\mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\cdot) = 0$ for all (not necessarily PT) B.

DISCUSSION. Commitment and encryption schemes can be recovered as special cases of CE schemes as follows. We say that Π is a *commitment scheme* if $\mathcal{K}$ always returns $(\varepsilon, \varepsilon)$. We see that our two security requirements capture the standard hiding and binding properties. In Section 1 we had simplified by assuming the verification algorithm is canonical and there were no parameters but here we are more general. We say that $\mathcal{D}$ is a decryption algorithm for CE scheme Π if $\mathcal{D}(1^\lambda, \pi, dk, \mathcal{E}(1^\lambda, \pi, ek, m; r)) = m$ for all $\lambda \in \mathbb{N}$, all $\pi \in [\mathcal{P}(1^\lambda)]$, all $(ek, dk) \in [\mathcal{K}(\pi)]$, all $r \in \{0,1\}^{\rho(\lambda)}$ and all $m \in \{0,1\}^*$ such that $\mathcal{E}(1^\lambda, \pi, ek, m; r) \neq \perp$. We say that Π *admits decryption* if it has a PT decryption algorithm and in that case we say Π is an *encryption scheme*. IND-CPA is then, of course, the standard privacy goal.

Typical encryption schemes are perfectly binding under canonical verification with some added checks. For example, the ElGamal encryption scheme over a order-p group $\mathbb{G}$ with generator g (these quantities in the parameters) is binding under a verification algorithm that performs the re-encryption check and then also checks that quantities that should be in $\mathbb{G}$ or $\mathbb{Z}_p$ really are. RSA-based schemes can be made binding by requiring the encryption exponent to be a prime larger than the modulus.

Lossy encryption schemes [2, 30, 37] are not binding because the adversary could provide a lossy encryption key and, under this, be able to generate encryption collisions. Non-commiting [11, 16] and deniable [10, 33] encryption schemes are intentionally not binding. These types of encryption schemes have been shown to have SOA security. Our results show that the lack of binding was necessary for their success at this task.

HASH FUNCTIONS. A hash function $\Gamma = (\mathcal{A}, \mathcal{H})$ with associated output length $h\colon \mathbb{N} \to \mathbb{N}$ is a tuple of PT algorithms. Via $a \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$ the key-generation algorithm $\mathcal{A}$ produces a key a. Via $y \leftarrow \mathcal{H}(a, x)$ the deterministic hashing algorithm $\mathcal{H}$ produces the $h(\lambda)$-bit hash of a string x under key a. Collision-resistance is defined via game CR_Γ whose INITIALIZE(1^λ) procedure returns $a \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$ and whose FINALIZE procedure on input (x, x') returns $(x \neq x') \wedge (\mathcal{H}(a, x) = \mathcal{H}(a, x'))$. There are no other procedures. The advantage of an adversary C is defined by $\mathbf{Adv}^{\mathrm{cr}}_{\Gamma,C}(\lambda) = \Pr\left[\mathrm{CR}^C_\Gamma(\lambda)\right]$. We say that Γ is collision-resistant (CR) if $\mathbf{Adv}^{\mathrm{cr}}_{\Gamma,C}(\cdot)$ is negligible for every PT C. The following says that CR hash functions must have super-logarithmic output length and will be useful later:

Proposition 3.1 *Let $\Gamma = (\mathcal{A}, \mathcal{H})$ be a hash function with associated output length $h\colon \mathbb{N} \to \mathbb{N}$. If Γ is collision-resistant then the function $2^{-h(\cdot)}$ is negligible.*

4 SOA-C insecurity of CE schemes

Here we show that no CE-scheme that is binding is SOA-C secure. This implies that no HB-secure commitment scheme is SOA-secure and that no binding IND-CPA encryption scheme is SOA-secure under sender corruptions. In Section 5 we establish similar results for SOA-K to show that no robust IND-CPA encryption scheme is SOA-secure for receiver corruptions.

SOA-C SECURITY. A *relation* is a PT algorithm with boolean output. A *message sampler* is a PT algorithm $\mathcal{M}$ taking input 1^λ and a string α and returning a vector over $\{0,1\}^*$. There must exist a function $n\colon \mathbb{N} \to \mathbb{N}$ (called the number of messages) and a function $\ell\colon \mathbb{N} \times \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ (called the message length) such that $|\mathbf{m}| = n(\lambda)$ and $|\mathbf{m}[i]| = \ell(\lambda, \alpha, i)$ for all $\mathbf{m} \in [\mathcal{M}(1^\lambda, \alpha)]$ and all $i \in [n]$. An *auxiliary-input generator* is a PT algorithm.

Let $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ be a CE-scheme, R a relation, $\mathcal{M}$ a message sampler and $\mathcal{A}$ an auxiliary-input generator. We define SOA-C security via the games of Figure 2. "Real" game $\mathrm{RSOAC}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ will be executed with an adversary A. An soa-c adversary's (mandatory, starting) INITIALIZE(1^λ) call results in its being returned an auxiliary input, parameters, and an encryption key, the latter corresponding to the single receiver modeled here. The adversary is then required to make exactly one ENC(α) call. This results in production of a message vector whose encryption is provided to the adversary. Now the adversary is required to make exactly one CORRUPT(I) call to get back the messages *and coins* corresponding to the senders named in the set $I \subseteq [n(\lambda)]$. It then calls FINALIZE with some value w of its choice and wins if the relation returns true on the inputs shown. A soa-c simulator S runs with the simulator game

```
INITIALIZE($1^\lambda$)
$\pi \leftarrow\$ \mathcal{P}(1^\lambda)$ ; $a \leftarrow \mathcal{A}(1^\lambda)$ ; $(ek, dk) \leftarrow\$ \mathcal{K}(\pi)$
Return $(a, \pi, ek)$

ENC($\alpha$)
$\mathbf{m} \leftarrow\$ \mathcal{M}(1^\lambda, \alpha)$
For $i = 1, \ldots, n(\lambda)$ do
    $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\rho(\lambda)}$ ; $\mathbf{c}[i] \leftarrow \mathcal{E}(1^\lambda, \pi, ek, \mathbf{m}[i]; \mathbf{r}[i])$
Return $\mathbf{c}$

CORRUPT($I$)
Return $\mathbf{m}[I], \mathbf{r}[I]$

FINALIZE($w$)
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
```

```
INITIALIZE($1^\lambda$)
$\pi \leftarrow\$ \mathcal{P}(1^\lambda)$ ; $a \leftarrow \mathcal{A}(1^\lambda)$
Return $(a, \pi)$

MSG($\alpha$)
$\mathbf{m} \leftarrow\$ \mathcal{M}(1^\lambda, \alpha)$

CORRUPT($I$)
Return $\mathbf{m}[I]$

FINALIZE($w$)
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
```

Figure 2: Game $\mathrm{RSOAC}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ capturing the real-world SOA-C attack to be mounted by an adversary (left) and game $\mathrm{SSOAC}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ capturing the simulated-world SOA-C attack to be mounted by a simulator (right).

$\mathrm{SSOAC}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ and gets back only auxiliary input and parameters from its INITIALIZE(1^λ) call, there being no encryption key in its world. It is then required to make exactly one MSG(α) call resulting in creation of a message vector but the simulator is returned nothing related to it. It must then make its CORRUPT(I) and FINALIZE(w) calls like the adversary and wins under the same conditions. The soa-c-advantage of an soa-c-adversary A with respect to CE-scheme Π, message sampler $\mathcal{M}$, relation R, auxiliary input generator $\mathcal{A}$ and soa-c simulator S is defined by

$$\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \;=\; \Pr\left[\mathrm{RSOAC}^{A}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\right] - \Pr\left[\mathrm{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\right] .$$

We say that Π is $(\mathcal{M},\mathcal{A})$-SOA-C-secure if for every PT R and every PT soa-c adversary A there exists a PT soa-c simulator S such that $\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\cdot)$ is negligible. We say that Π is SOA-C-secure if it is $(\mathcal{M},\mathcal{A})$-SOA-C-secure for every PT $\mathcal{M},\mathcal{A}$.

RESULT. The following implies that any binding CE-scheme is not SOA-C-secure.

Theorem 4.1 *Let $\Pi = (\mathcal{P},\mathcal{K},\mathcal{E},\mathcal{V})$ be a binding CE-scheme with message length ℓ: $\mathbb{N} \to \mathbb{N}$. Let $\Gamma = (\mathcal{A},\mathcal{H})$ be a collision-resistant hash function with associated output length h: $\mathbb{N} \to \mathbb{N}$. Let $n(\cdot) = 2h(\cdot)$ and let $\mathcal{M}$ be the message sampler that on input $1^\lambda, \alpha$ (ignores α and) returns a $n(\lambda)$-vector whose components are uniformly and independently distributed over $\{0,1\}^{\ell(\lambda)}$. Then there exists a PT soa-c adversary A and a PT relation R such that for all PT simulators S there is a negligible function ν such that $\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \geq 1 - \nu(\lambda)$ for all $\lambda \in \mathbb{N}$.* ▌

Thus, Π is not $(\mathcal{M},\mathcal{A})$-SOA-C-secure and hence cannot be SOA-C-secure. Moreover, this is true when the distribution on messages is uniform. These claims would only require $\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\cdot)$ in the theorem to be non-negligible, but we show more, namely that it is almost one. Note that ℓ is arbitrary and could even be $\ell(\cdot) = 1$, meaning we rule out SOA-C-security even for bit-commitment and encryption of 1-bit messages. The proof will make use of the following variant of the Reset Lemma of [3].

Lemma 4.2 *Let $V = \{V_\lambda\}_{\lambda\in\mathbb{N}}$ be a collection of non-empty sets. Let P_1, P_2 be algorithms, the second with boolean output. The single-execution acceptance probability $\mathbf{AP}_1(P_1, P_2, V, \lambda)$ is defined as the probability that $d = \mathsf{true}$ in the single execution experiment $\overline{St} \leftarrow\$ P_1(1^\lambda)$; $\mathbf{m}^* \leftarrow\$ V_\lambda$; $d \leftarrow\$ P_2(\overline{St}, \mathbf{m}^*)$. The double-execution acceptance probability $\mathbf{AP}_2(P_1, P_2, V, \lambda)$ is defined as the probability that $d_1 = d_2 = \mathsf{true}$ and $\mathbf{m}^*_0 \neq \mathbf{m}^*_1$ in the double execution experiment $\overline{St} \leftarrow\$ P_1(1^\lambda)$; $\mathbf{m}^*_0, \mathbf{m}^*_1 \leftarrow\$ V_\lambda$; $d_0 \leftarrow\$ P_2(\overline{St}, \mathbf{m}^*_0)$; $d_1 \leftarrow\$ P_2(\overline{St}, \mathbf{m}^*_1)$. Then $\mathbf{AP}_1(P_1, P_2, V, \lambda) \;\leq\; 1/|V_\lambda| + \sqrt{\mathbf{AP}_2(P_1, P_2, V, \lambda)}$ for all $\lambda \in \mathbb{N}$.* ▌

The two executions in the double-execution experiment are not independent because $\overline{St}$ is the same for both, which is why the lemma is not trivial.

```
Adversary $A(1^\lambda)$
$(a, \pi, ek) \leftarrow$ INITIALIZE$(1^\lambda)$
$\mathbf{c} \leftarrow$ ENC$(\varepsilon)$
$b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(a, ek \,\|\, \mathbf{c})$
$I \leftarrow \{2j - 1 + b[j] \,:\, 1 \le j \le h(\lambda)\}$
$(\overline{\mathbf{m}}, \overline{\mathbf{r}}) \leftarrow$ CORRUPT$(I)$
$w \leftarrow (ek, \mathbf{c}, \overline{\mathbf{r}})$
FINALIZE$(w)$
```

```
Relation $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
If $\alpha \neq \varepsilon$ then return false
$(ek, \mathbf{c}, \overline{\mathbf{r}}) \leftarrow w$ ; $b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(a, ek \,\|\, \mathbf{c})$
If $(I \neq \{2j - 1 + b[j] \,:\, 1 \le j \le h(\lambda)\})$ then return false
If $|\mathbf{c}| \neq n(\lambda)$ or $|\overline{\mathbf{r}}| \neq n(\lambda)$ then return false
For all $i \in I$ do
    If $\mathcal{V}(1^\lambda, \pi, ek, \mathbf{c}[i], \mathbf{m}[i], \overline{\mathbf{r}}[i]) =$ false then return false
Return true
```

Figure 3: Adversary A and relation R for the proof of Theorem 4.1.

Proof of Lemma 4.2: Let $\delta = 1/|V_\lambda|$. Let $\mathsf{X}(\omega) = \Pr[d = \mathsf{true}]$ in the experiment $\overline{St} \leftarrow P_1(1^\lambda; \omega)$; $\mathbf{m}^* \leftarrow\!\!\$\, V_\lambda$; $d \leftarrow\!\!\$\, P_2(\overline{St}, \mathbf{m}^*)$. So $\mathbf{E}[\mathsf{X}] = \mathbf{AP}_1(P_1, P_2, V, \lambda)$ where the expectation is over the coins ω of P_1. Let $a_1 = \mathbf{AP}_1(P_1, P_2, V, \lambda)$ and $a_2 = \mathbf{AP}_2(P_1, P_2, V, \lambda)$. Then

$$a_2 \;\ge\; \mathbf{E}[\mathsf{X}(\mathsf{X} - \delta)] \;=\; \mathbf{E}[\mathsf{X}^2] - \delta \cdot \mathbf{E}[\mathsf{X}] \;\ge\; \mathbf{E}[\mathsf{X}]^2 - \delta \cdot \mathbf{E}[\mathsf{X}] \;=\; a_1^2 - \delta \cdot a_1$$

where the third step above is by Jensen's inequality. Now $a_1^2 - \delta \cdot a_1 = (a_1 - \delta/2)^2 - \delta^2/4$ so

$$a_1 \;\le\; \delta/2 + \sqrt{a_2 + \delta^2/4} \;\le\; \delta/2 + \sqrt{a_2} + \sqrt{\delta^2/4} \;=\; \delta + \sqrt{a_2}$$

which yields the lemma. ▮

Proof of Theorem 4.1: The adversary A and relation R are depicted in Figure 3. Let S be any PT soa-c simulator. In the real game the adversary always makes the relation return true hence

$$\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \;=\; 1 - \Pr\left[\,\text{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] .$$

We will construct a binding-adversary B and cr-adversary C such that

$$\Pr\left[\,\text{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] \;\le\; 2^{-h(\lambda)\ell(\lambda)} + \sqrt{\mathbf{Adv}^{\text{cr}}_{\Gamma,C}(\lambda) + \mathbf{Adv}^{\text{bind}}_{\Pi,B}(\lambda)}\;. \tag{1}$$

The assumptions that Γ is collision-resistant, Π is binding, together with Proposition 3.1, imply that the RHS of Eq. (1) is negligible, which proves the theorem. It remains to construct B and C. Given S we can define sub-algorithms S_1, S_2 such that S can be written in terms of S_1, S_2 as follows:

```
Simulator $S(1^\lambda)$
$(a, \pi) \leftarrow$ INITIALIZE$(1^\lambda)$ ; MSG$(1^\lambda, \varepsilon)$ ; $(St, I) \leftarrow\!\!\$\, S_1(a, \pi)$
$\overline{\mathbf{m}} \leftarrow$ CORRUPT$(I)$ ; $w \leftarrow S_2(St, \overline{\mathbf{m}})$ ; FINALIZE$(w)$
```

We clarify that we are not defining S; the latter is given and arbitrary. Rather, *any* S has the form above for *some* S_1, S_2 that can be determined given S. Specifically, S_1 runs S until S makes its CORRUPT(I) query, returning I along with the current state St of S. Then S_2, given the response $\overline{\mathbf{m}}$ to the query, feeds it back to S and continues executing S from St. By having S_1 put all S's coins in St we can assume S_2 is deterministic. We may assume wlog that $|I|$ is always $h(\lambda)$ and that the argument α in S's MSG call is ε since otherwise R rejects. We now define adversary B. The embedding subroutine Emb it calls and the notation $\perp_{n(\lambda)}$ were defined in Section 3:

```
Adversary $B(1^\lambda)$
$\pi \leftarrow$ INITIALIZE$(1^\lambda)$ ; $a \leftarrow\!\!\$\, \mathcal{A}(1^\lambda)$ ; $(St, I) \leftarrow\!\!\$\, S_1(a, \pi)$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow\!\!\$\, (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$ ; $t \leftarrow\!\!\$\, I$
For all $i \in I$ do If $\overline{\mathbf{m}}_0[i] \neq \overline{\mathbf{m}}_1[i]$ then $t \leftarrow i$
FINALIZE$(ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{r}}_0[t], \overline{\mathbf{r}}_1[t])$
```

Adversary B is running S to get its CORRUPT query I and then, by backing it up, providing two different responses. Adversary C has a similar strategy, only deviating in how the final values are used:

```
Adversary $C(1^\lambda)$
$a \leftarrow \textsc{Initialize}(1^\lambda)$ ; $\pi \leftarrow_{\$} \mathcal{P}(1^\lambda)$ ; $(St, I) \leftarrow_{\$} S_1(a, \pi)$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$
$\textsc{Finalize}((ek_0 \,\|\, \mathbf{c}_0, ek_1 \,\|\, \mathbf{c}_1))$.
```

The analysis will use Lemma 4.2. Let $V_\lambda = (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$ and $V = \{V_\lambda\}_{\lambda \in \mathbb{N}}$. Define P_1, P_2 via:

```
Algorithm $P_1(1^\lambda)$
$\pi \leftarrow \mathcal{P}(1^\lambda)$ ; $a \leftarrow_{\$} \mathcal{A}(1^\lambda)$ ; $(St, I) \leftarrow_{\$} S_1(a, \pi)$
$\mathbf{m} \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{n(\lambda)}$ ; $\overline{St} \leftarrow (1^\lambda, a, \pi, \mathbf{m}, I, St)$
Return $\overline{St}$
```

```
Algorithm $P_2(\overline{St}, \mathbf{m}^*)$
$(1^\lambda, a, \pi, \mathbf{m}, I, St) \leftarrow \overline{St}$
$\overline{\mathbf{m}} \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}^*, \perp_{n(\lambda)})$ ; $w \leftarrow S_2(St, \overline{\mathbf{m}})$
$\mathbf{m} \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}^*, \mathbf{m})$
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \varepsilon, I, w)$
```

Above the argument $\mathbf{m}^*$ to P_2 is drawn from V_λ. Now

$$\Pr\left[\,\mathrm{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] \;=\; \mathbf{AP}_1(P_1, P_2, V, \lambda) \;\le\; 2^{-h(\lambda)\ell(\lambda)} + \sqrt{\mathbf{AP}_2(P_1, P_2, V, \lambda)} \tag{2}$$

Above the equality is from the definitions and the inequality is by Lemma 4.2. Finally we claim that

$$\mathbf{AP}_2(P_1, P_2, V, \lambda) \;\le\; \mathbf{Adv}^{\mathrm{cr}}_{\Gamma,C}(\lambda) + \mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\lambda)\;. \tag{3}$$

Eqs. (2) and (3) imply Eq. (1) and conclude the proof. We now justify Eq. (3). To do so it is helpful to write down the double-execution experiment underlying $\mathbf{AP}_2(P_1, P_2, V, \lambda)$:

```
$\pi \leftarrow \mathcal{P}(1^\lambda)$ ; $a \leftarrow_{\$} \mathcal{A}(1^\lambda)$ ; $(St, I) \leftarrow_{\$} S_1(a, \pi)$ ; $\mathbf{m} \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{n(\lambda)}$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$
$\mathbf{m}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \mathbf{m})$ ; $\mathbf{m}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \mathbf{m})$
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_0, \varepsilon, I, w_0) \wedge \mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_1, \varepsilon, I, w_1) \wedge (\mathbf{m}_0^* \neq \mathbf{m}_1^*)$.
```

Assume this experiment returns true. By definition of R it must be that $I = \{2j - 1 + b_0[j] \;:\; 1 \le j \le h(\lambda)\}$ where $b_0[1] \ldots b_0[h(\lambda)] = \mathcal{H}(a, ek_0 \,\|\, \mathbf{c}_0)$ and also $I = \{2j - 1 + b_1[j] \;:\; 1 \le j \le h(\lambda)\}$ where $b_1[1] \ldots b_1[h(\lambda)] = \mathcal{H}(a, ek_1 \,\|\, \mathbf{c}_1)$. However, I is the same in both cases, so we must have $\mathcal{H}(a, ek_0 \,\|\, \mathbf{c}_0) = \mathcal{H}(a, ek_1 \,\|\, \mathbf{c}_1)$, meaning we have a hash collision. This means that C succeeds unless $ek_0 \,\|\, \mathbf{c}_0 = ek_1 \,\|\, \mathbf{c}_1$. But we now argue that in the latter case, B succeeds. We know $\mathbf{m}_0^* \neq \mathbf{m}_1^*$ so there is some $t \in I$ such that $\overline{\mathbf{m}}_0[t] \neq \overline{\mathbf{m}}_1[t]$. The definition of R implies that $\mathcal{V}(1^\lambda, \pi, ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{r}}_0[t]) = \mathsf{true}$ and also $\mathcal{V}(1^\lambda, \pi, ek_1, \mathbf{c}_1[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{r}}_1[t]) = \mathsf{true}$. But since $ek_0 \,\|\, \mathbf{c}_0 = ek_1 \,\|\, \mathbf{c}_1$ we have $\mathcal{V}(1^\lambda, \pi, ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{r}}_0[t]) = \mathsf{true}$ and also $\mathcal{V}(1^\lambda, \pi, ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{r}}_1[t]) = \mathsf{true}$ with $\overline{\mathbf{m}}_0[t] \neq \overline{\mathbf{m}}_1[t]$ so B wins. ▌

Extensions, applications and remarks. The SOA-C definition could be weakened by allowing the simulator's corruptions to be adaptive, meaning S is allowed multiple queries to procedure Corrupt that now would take input $i \in [n(\lambda)]$ and return $\mathbf{m}[i]$. The proof strategy of Theorem 4.1 no longer works but can be extended to also rule out adaptive simulators. We would back S up to its last Corrupt query and give a new response only to this query. We would now require $\ell(\cdot)$ to be super-logarithmic so that collisions are rare on single messages. We omit the details.

Theorem 4.1 applies to all commitment schemes since they are binding by definition. Not all encryption schemes are binding, but many popular ones are. For example, the ElGamal scheme is binding. The Cramer-Shoup scheme [14] is also binding, showing that IND-CCA is not a panacea against SOAs.

Our model allows a scheme to have system parameters π that effectively function as auxiliary input. This means the simulator cannot modify them. This is not necessary but merely makes the results more general. If one wishes to view commitment, as in DNRS [19], as having no parameters, just restrict attention to schemes where π is always 1^λ. Our result applies to these as a special case.

5 SOA-K insecurity of encryption schemes

Here we show that no decryption-verifiable IND-CPA encryption scheme is SOA-secure for receiver corruptions. The techniques are very similar to those of Section 4.

SOA-K SECURITY. This is the dual of SOA-C where there are multiple receivers and a single sender rather than a single receiver and multiple senders, and corruptions reveal decryption keys rather than coins. The definition uses games $\mathrm{RSOAK}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ and $\mathrm{SSOAK}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ of Figure 4. The soa-k-advantage of an soa-k-adversary A with respect to the encryption scheme Π, message sampler $\mathcal{M}$, relation R, auxiliary input generator $\mathcal{A}$ and soa-k simulator S is defined by

$$\mathbf{Adv}^{\text{soa-k}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \;=\; \Pr\left[\mathrm{RSOAK}^{A}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}\right] - \Pr\left[\mathrm{SSOAK}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}\right] .$$

We say that Π is $(\mathcal{M},\mathcal{A})$-SOA-K-secure if for every PT R and every PT soa-k adversary A there exists a PT soa-k simulator S such that $\mathbf{Adv}^{\text{soa-k}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\cdot)$ is negligible. We say that Π is SOA-K-secure if it is $(\mathcal{M},\mathcal{A})$-SOA-K-secure for every PT $\mathcal{M},\mathcal{A}$.

DECRYPTION VERIFIABILITY. Let $\Pi = (\mathcal{P},\mathcal{K},\mathcal{E},\mathcal{V})$ be an encryption scheme. A decryption verifier for Π is a deterministic PT algorithm $\mathcal{W}$ which on inputs $1^\lambda,\pi,ek,dk,c,m$ returns true or false. We require that $\mathcal{W}(1^\lambda,\pi,ek,dk,\mathcal{E}(1^\lambda,\pi,ek,m;r),m) = \mathsf{true}$ for all $\lambda\in\mathbb{N}$, all $\pi\in[\mathcal{P}(1^\lambda)]$, all $(ek,dk)\in[\mathcal{K}(\pi)]$, all $r\in\{0,1\}^{\rho(\lambda)}$ and all $m\in\{0,1\}^*$ such that $\mathcal{E}(1^\lambda,\pi,ek,m;r)\neq\bot$. We say that the decryption verifier $\mathcal{W}$ is *canonical* if $\mathcal{W}(1^\lambda,\pi,ek,dk,c,m)$ returns $(\mathcal{D}(1^\lambda,\pi,dk,c)=m)$ where $\mathcal{D}$ is the decryption algorithm associated to Π. Game $\mathrm{DV}_{\Pi,\mathcal{W}}$ of Figure 5 captures decryption verifiability. We let $\mathbf{Adv}^{\text{dv}}_{\Pi,\mathcal{W},W}(\lambda) = \Pr[\mathrm{DV}^{W}_{\Pi,\mathcal{W}}(\lambda)]$. We say that Π is *decryption-verifiable* if $\mathbf{Adv}^{\text{dv}}_{\Pi,\mathcal{W},W}(\cdot)$ is negligible for all PT W and *perfectly decryption verifiable* if $\mathbf{Adv}^{\text{dv}}_{\Pi,\mathcal{W},W}(\cdot)=0$ for all (not necessarily PT) W.

When the decryption verifier is the canonical one, decryption verifiability is a form of robustness as defined in [1] but our formulation is more general, allowing verification to do more than merely decrypt. This results in a weaker requirement than robustness, which means that our result applies to a wider class of schemes. For example the ElGamal encryption scheme is not robust [1] but it is (perfectly) decryption verifiable, and thus our result below applies to it.

RESULT. The following implies that any decryption-verifiable encryption scheme is not SOA-K-secure.

Theorem 5.1 *Let $\Pi = (\mathcal{P},\mathcal{K},\mathcal{E},\mathcal{V})$ be a decryption-verifiable encryption scheme with decryption verifier $\mathcal{W}$ and message length $\ell\colon\mathbb{N}\to\mathbb{N}$. Let $\Gamma=(\mathcal{A},\mathcal{H})$ be a collision-resistant hash function with associated output length $h\colon\mathbb{N}\to\mathbb{N}$. Let $n(\cdot)=2h(\cdot)$ and let $\mathcal{M}$ be the message sampler that on input $1^\lambda,\alpha$ (ignores α and) returns a $n(\lambda)$-vector whose components are uniformly and independently distributed over $\{0,1\}^{\ell(\lambda)}$. Then there exists a PT soa-k adversary A and a PT relation R such that for all PT soa-k simulators S there is a negligible function ν such that $\mathbf{Adv}^{\text{soa-k}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda)\geq 1-\nu(\lambda)$ for all $\lambda\in\mathbb{N}$.* ▌

Thus, Π is not $(\mathcal{M},\mathcal{A})$-SOA-K-secure and hence cannot be SOA-K-secure.

Proof of Theorem 5.1: The adversary A and relation R are depicted in Figure 6. Let S be any PT soa-k simulator. In the real game the adversary always makes the relation return true hence

$$\mathbf{Adv}^{\text{soa-k}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \;=\; 1-\Pr\left[\mathrm{SSOAK}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\right] .$$

We will construct a dv-adversary W and cr-adversary C such that

$$\Pr\left[\mathrm{SSOAK}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\right] \;\leq\; 2^{-h(\lambda)\ell(\lambda)} + \sqrt{\mathbf{Adv}^{\text{cr}}_{\Gamma,C}(\lambda)+\mathbf{Adv}^{\text{dv}}_{\Pi,\mathcal{W},W}(\lambda)} \,. \tag{4}$$

But since we assumed that Γ is collision-resistant and that Π is decryption-verifiable, we conclude that the right hand side of Eq. (4) is negligible when we also take Proposition 3.1 into account, and this implies the verity of the theorem. We present the adversaries W and C below.

We break the simulator S into sub-algorithms S_1,S_2 in the same way that was done in Section 4. We now define the adversaries W and C that for the most part are equal to the respective ones of Section 4.

Adversary $W(1^\lambda)$

```
Initialize(1^λ)
    $\pi \leftarrow\!\!\$\, \mathcal{P}(1^\lambda)$ ; $a \leftarrow \mathcal{A}(1^\lambda)$
    For $i = 1, \ldots, n(\lambda)$ do $(\mathbf{ek}[i], \mathbf{dk}[i]) \leftarrow\!\!\$\, \mathcal{K}(\pi)$
    Return $(a, \pi, \mathbf{ek})$
Enc(α)
    $\mathbf{m} \leftarrow\!\!\$\, \mathcal{M}(1^\lambda, \alpha)$
    For $i = 1, \ldots, n(\lambda)$ do
        $\mathbf{r}[i] \leftarrow\!\!\$\, \{0,1\}^{\rho(\lambda)}$ ; $\mathbf{c}[i] \leftarrow \mathcal{E}(1^\lambda, \pi, \mathbf{ek}[i], \mathbf{m}[i]; \mathbf{r}[i])$
    Return $\mathbf{c}$
Corrupt(I)
    Return $\mathbf{m}[I], \mathbf{dk}[I]$
Finalize(w)
    Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
```

```
Initialize(1^λ)
    $\pi \leftarrow\!\!\$\, \mathcal{P}(1^\lambda)$ ; $a \leftarrow \mathcal{A}(1^\lambda)$
    Return $(a, \pi)$
Msg(α)
    $\mathbf{m} \leftarrow\!\!\$\, \mathcal{M}(1^\lambda, \alpha)$
Corrupt(I)
    Return $\mathbf{m}[I]$
Finalize(w)
    Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
```

Figure 4: Game $\mathrm{RSOAK}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ capturing the real-world SOA-K attack to be mounted by an adversary (left) and game $\mathrm{SSOAK}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}$ capturing the simulated-world SOA-K attack to be mounted by a simulator (right).

```
Initialize(1^λ)
    $\pi \leftarrow\!\!\$\, \mathcal{P}(1^\lambda)$
    Return $\pi$
```

```
Finalize(ek, c, m_0, m_1, dk_0, dk_1)
    $d_0 \leftarrow \mathcal{W}(1^\lambda, \pi, ek, dk_0, c, m_0)$
    $d_1 \leftarrow \mathcal{W}(1^\lambda, \pi, ek, dk_1, c, m_1)$
    Return $(d_0 \wedge d_1 \wedge (m_0 \neq m_1))$
```

Figure 5: Game $\mathrm{DV}_{\Pi,\mathcal{W}}$ defining the decryption verifiability of encryption scheme Π with decryption verifier $\mathcal{W}$.

```
$\pi \leftarrow$ Initialize$(1^\lambda)$ ; $a \leftarrow\!\!\$\, \mathcal{A}(1^\lambda)$ ; $(St, I) \leftarrow\!\!\$\, S_1(a, \pi)$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow\!\!\$\, (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(\mathbf{ek}_0, \mathbf{c}_0, \overline{\mathbf{dk}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(\mathbf{ek}_1, \mathbf{c}_1, \overline{\mathbf{dk}}_1) \leftarrow w_1$ ; $t \leftarrow\!\!\$\, I$
For all $i \in I$ do If $\overline{\mathbf{m}}_0[i] \neq \overline{\mathbf{m}}_1[i]$ then $t \leftarrow i$
Finalize$(\mathbf{ek}_0[t], \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{dk}}_0[t], \overline{\mathbf{dk}}_1[t])$
```

```
Adversary C(1^λ)
    $a \leftarrow$ Initialize$(1^\lambda)$ ; $\pi \leftarrow\!\!\$\, \mathcal{P}(1^\lambda)$
    $(St, I) \leftarrow\!\!\$\, S_1(a, \pi)$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow\!\!\$\, (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
    $\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
    $w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(\mathbf{ek}_0, \mathbf{c}_0, \overline{\mathbf{dk}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(\mathbf{ek}_1, \mathbf{c}_1, \overline{\mathbf{dk}}_1) \leftarrow w_1$
    Finalize$((\mathbf{ek}_0 \,\|\, \mathbf{c}_0, \mathbf{ek}_1 \,\|\, \mathbf{c}_1))$.
```

Let $V_\lambda = (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$, let $V = \{V_\lambda\}_{\lambda \in \mathbb{N}}$ and define the algorithms P_1, P_2 as in Section 4. Then

$$\Pr\left[\mathrm{SSOAK}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\right] \;=\; \mathbf{AP}_1(P_1, P_2, V, \lambda) \;\leq\; 2^{-h(\lambda)\ell(\lambda)} + \sqrt{\mathbf{AP}_2(P_1, P_2, V, \lambda)} \tag{5}$$

Finally we will prove that

$$\mathbf{AP}_2(P_1, P_2, V, \lambda) \;\leq\; \mathbf{Adv}^{\mathrm{cr}}_{\Gamma,C}(\lambda) + \mathbf{Adv}^{\mathrm{dv}}_{\Pi,\mathcal{W},W}(\lambda)\;. \tag{6}$$

Writing down the double-execution experiment underlying $\mathbf{AP}_2(P_1, P_2, V, \lambda)$:

```
$\pi \leftarrow \mathcal{P}(1^\lambda)$ ; $a \leftarrow\!\!\$\, \mathcal{A}(1^\lambda)$ ; $(St, I) \leftarrow\!\!\$\, S_1(a, \pi)$ ; $\mathbf{m} \leftarrow\!\!\$\, (\{0,1\}^{\ell(\lambda)})^{n(\lambda)}$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow\!\!\$\, (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \perp_{n(\lambda)})$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \perp_{n(\lambda)})$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(\mathbf{ek}_0, \mathbf{c}_0, \overline{\mathbf{dk}}_0) \leftarrow w_0$ ; $(\mathbf{ek}_1, \mathbf{c}_1, \overline{\mathbf{dk}}_1) \leftarrow w_1$
```

```
Adversary $A(1^\lambda)$
$(a, \pi, \mathbf{ek}) \leftarrow \textsc{Initialize}(1^\lambda)$
$\mathbf{c} \leftarrow \textsc{Enc}(\varepsilon)$
$b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(a, \mathbf{ek} \,\|\, \mathbf{c})$
$I \leftarrow \{2j - 1 + b[j] \;:\; 1 \le j \le h(\lambda)\}$
$(\overline{\mathbf{m}}, \overline{\mathbf{dk}}) \leftarrow \textsc{Corrupt}(I)$
$w \leftarrow (\mathbf{ek}, \mathbf{c}, \overline{\mathbf{dk}})$
$\textsc{Finalize}(w)$
```

```
Relation $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
If $\alpha \ne \varepsilon$ then return false
$(\mathbf{ek}, \mathbf{c}, \overline{\mathbf{dk}}) \leftarrow w$
$b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(a, \mathbf{ek} \,\|\, \mathbf{c})$
If $(I \ne \{2j - 1 + b[j] \;:\; 1 \le j \le h(\lambda)\})$ then return false
If $|\mathbf{ek}| \ne n(\lambda)$ or $|\mathbf{c}| \ne n(\lambda)$ or $|\overline{\mathbf{dk}}| \ne n(\lambda)$ then return false
For all $i \in I$ do
    If $\mathcal{W}(1^\lambda, \pi, \mathbf{ek}[i], \overline{\mathbf{dk}}[i], \mathbf{c}[i], \mathbf{m}[i]) = \mathsf{false}$ then return false
Return true
```

Figure 6: Adversary A and relation R for the proof of Theorem 5.1.

$\mathbf{m}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_0^*, \mathbf{m})$; $\mathbf{m}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I, \mathbf{m}_1^*, \mathbf{m})$
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_0, \varepsilon, I, w_0) \wedge \mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_1, \varepsilon, I, w_1) \wedge (\mathbf{m}_0^* \ne \mathbf{m}_1^*)$.

As in Section 4, if this experiment returns true, then $\mathcal{H}(a, \mathbf{ek}_0 \,\|\, \mathbf{c}_0) = \mathcal{H}(a, \mathbf{ek}_1 \,\|\, \mathbf{c}_1)$. Hence C succeeds unless $\mathbf{ek}_0 \,\|\, \mathbf{c}_0 = \mathbf{ek}_1 \,\|\, \mathbf{c}_1$. But if $\mathbf{ek}_0 \,\|\, \mathbf{c}_0 = \mathbf{ek}_1 \,\|\, \mathbf{c}_1$, then $\mathbf{c}_0[t] = \mathbf{c}_1[t]$ and $\mathbf{ek}_0[t] = \mathbf{ek}_1[t]$ and due to the tests performed by R we have that $\mathcal{W}(1^\lambda, \pi, \mathbf{ek}_0[t], \overline{\mathbf{dk}}_0[t], \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t]) = \mathsf{true}$ and $\mathcal{W}(1^\lambda, \pi, \mathbf{ek}_0[t], \overline{\mathbf{dk}}_1[t], \mathbf{c}_0[t], \overline{\mathbf{m}}_1[t]) = \mathsf{true}$. Therefore W wins the decryption verifiability game.

Eqs. (5) and (6) together prove Eq. (4). ▌

6 SOA insecurity without Auxiliary Input

The model of Section 4, following [19], allows the adversary and simulator a common auxiliary input a. The model is well-justified because a represents history that is not under the simulator's control and the auxiliary input model has through an extensive history come to be viewed as the "right" one for simulation-based definitions. However it is technically interesting to ask what can be shown when the simulator has the extra power of not being constrained to a given auxiliary input. In this section, we show that there are encryption schemes that fail to be SOA-C secure even in this case.

We could "cheat" by having the encryption scheme put the auxiliary input in the system parameters which in Section 4 are also a common input to adversary and simulator. Thus, we also consider only encryption schemes without system parameters. Formally, let $\mathcal{A}$ return ε on any input and let $\mathcal{P}$ return 1^λ on input 1^λ. We will show that there is a CE scheme $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ and a message sampler $\mathcal{M}$ such that Π is not $(\mathcal{M}, \mathcal{A})$-SOA-C secure.

We would like to use the same idea as in the previous section, but now the adversary no longer has access to a hash function key in the auxiliary input. We cannot let the adversary pick the key and put it in w for then the simulator could pick a key under which the function is not CR. Instead, the adversary will use part of the ciphertext vector as the hash key. However the simulator gets to choose the ciphertext vector, and thus the hash key. Thus, any proof that tried to embed a hash key runs into a problem.

To solve this, we introduce a primitive we call an encrypted hash scheme (EHS) that combines a CE-scheme with a hash function. The best way to think about such a scheme is that there is an underlying core hash function with keys coming from the message space of the CE scheme, and an encrypted hash function that takes encrypted messages as hash keys. With such schemes, we need a new notion of collision resistance that we call special collision resistance. Here an adversary is given a core hash function key (i.e., a random message from the CE-scheme's message space) and must produce two encryptions of that core key (which are themselves hash keys for the encrypted hash function) and two inputs that lead to a collision in the encrypted hash function. Since the underlying core hash function key is a message in the CE scheme, and messages are out of the simulator's control in the selective opening game, this special collision resistance notion allows us to prove the counterexample. A challenge will be to find an EHS that

```
$\underline{\text{INITIALIZE}(1^\lambda)}$
Return $1^\lambda$

$\underline{\text{MSG}(ek)}$
$m \leftarrow\!\$\ \{0,1\}^\ell$
Return $m$

$\underline{\text{FINALIZE}(c_0, c_1, r_0, r_1, v_0, v_1)}$
$d_0 \leftarrow \mathcal{V}(1^\lambda, 1^\lambda, ek, c_0, m, r_0)$
$d_1 \leftarrow \mathcal{V}(1^\lambda, 1^\lambda, ek, c_1, m, r_1)$
$d_2 \leftarrow (\mathcal{H}(1^\lambda, c_0, v_0) = \mathcal{H}(1^\lambda, c_1, v_1))$
$d_3 \leftarrow (v_0 \neq v_1)$
Return $(d_0 \wedge d_1 \wedge d_2 \wedge d_3)$
```

```
**Algorithm** $\mathcal{H}_c(1^\lambda, K, x)$
$(m_1, m_2, m_3, m_4, m_5) \leftarrow K$
$(x_1, x_2, x_3, x_4, x_5) \leftarrow x$
Return $m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5}$

**Algorithm** $\mathcal{H}^+(1^\lambda, K, T)$
$(t_1, \ldots, t_{n+1}) \leftarrow \mathsf{pad}(T)$
$y_0 \leftarrow 0^k$
For $i = 1$ to $n+1$ do
    $y_i = \mathcal{H}(K, y_{i-1} \,\|\, t_i)$
Return $y_{n+1}$
```

```
**Algorithm** $\mathcal{E}(1^\lambda, \pi, ek, m)$
$(m_1, m_2, m_3, m_4, m_5) \leftarrow m$
$g \leftarrow\!\$\ \mathbb{G}_\lambda^* \ ;\ s \leftarrow\!\$\ \mathbb{Z}_p \ ;\ h \leftarrow g^s$
$u_1, u_2, u_3, u_4, u_5 \leftarrow\!\$\ \mathbb{G}_\lambda$
$w_1 \leftarrow m_1 \cdot u_1^s \ ;\ w_2 \leftarrow m_2 \cdot u_2^s$
$w_3 \leftarrow m_3 \cdot u_3^s \ ;\ w_4 \leftarrow m_4 \cdot u_4^s$
$w_5 \leftarrow m_5 \cdot u_5^s$
Return $(g, h, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3, w_4, w_5)$

**Algorithm** $\mathcal{H}(1^\lambda, K, x)$
$(g, h, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3, w_4, w_5) \leftarrow K$
$(x_1, x_2, x_3, x_4, x_5) \leftarrow x$
$Y_1 \leftarrow u_1^{x_1} u_2^{x_2} u_3^{x_3} u_4^{x_4} u_5^{x_5}$
$Y_2 \leftarrow w_1^{x_1} w_2^{x_2} w_3^{x_3} w_4^{x_4} w_5^{x_5}$
Return $(g, h, Y_1, Y_2)$
```

Figure 7: Game SCR_Δ (left) and algorithms $\mathcal{E}$, $\mathcal{H}$, $\mathcal{H}^+$, and $\mathcal{H}_c$ used in our construction.

can be proven special collision-resistant. We now formally define encrypted hash schemes and describe our construction.

ENCRYPTED HASH SCHEMES. Let $\mathcal{P}$ be the identity function. Let $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ be a CE-scheme with message length ℓ and randomness length ρ. Let $\mathcal{H}$ be a deterministic algorithm which on input $1^\lambda, K, x$ returns an $h(\lambda)$-bit string. We call $h\colon \mathbb{N} \to \mathbb{N}$ the output length. We refer to $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ as an encrypted hash scheme.

Adversary C must make exactly one MSG query in game SCR_Δ of Figure 7 and its scr-advantage is $\mathbf{Adv}^{\text{scr}}_{\Delta,C}(\lambda) = \Pr\left[\text{SCR}^C_\Delta(\lambda)\right]$. We say that Δ is *special collision resistant* if $\mathbf{Adv}^{\text{scr}}_{\Delta,C}(\cdot)$ is negligible for all PT C. In this game, the keys c_0, c_1 for the hash function are ciphertexts under partial adversary control. They must encrypt a random message chosen by the game, but the public key and coins are chosen by the adversary.

The proof that an EHS scheme is not SOA-C secure is very similar to the proof of Theorem 4.1 and can be found in Appendix A.1. The difficulty is building an EHS scheme that is special collision resistant, and it is this we now turn to.

CONSTRUCTION. We let $\mathcal{G}$ be a deterministic PT algorithm called the *fixed group generator* that on input 1^λ returns a cyclic group $\mathbb{G}_\lambda$ of prime order p_λ, which is possible under conjectures on the distribution of prime numbers in intervals. We remark that if it were possible to deterministically also produce generators g, h of $\mathbb{G}_\lambda$ such that $\log_g(h)$ is hard to compute our problem would trivialize because we could define a keyless CR hash, but this appears to be a much stronger assumption. Under our assumption there is no known way to get a keyless CR hash function. But we will get an EHS. We assume there is an injective, PT encoding of a $\ell(\lambda)$-bit string as an element of $\mathbb{G}_\lambda^5$ and implicity assume this when writing statements such as $(m_1, m_2, m_3, m_4, m_5) \leftarrow m$ to indicate that bitstring m is parsed into its encoding. We will assume hardness of DL and DDH relative to $\mathcal{G}$ as defined in Figure 9.

Now we can describe our encrypted hash scheme. Let $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ be a CE scheme with $\mathcal{P}$ the identity function, $\mathcal{K}$ that outputs $(\varepsilon, \varepsilon)$, $\mathcal{E}$ as shown in Figure 7, and the canonical $\mathcal{V}$. Our EHS is $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$ where $\mathcal{H}^+$ is defined in Figure 7. Here $k < 5 \log p_\lambda$ is the size in bits to represent four group elements and pad is a padding function from [15] which we fully specify in Appendix A.3. One way to think about Δ^+ is that it is an encrypted and extended version of the core hash function $\Gamma = (\mathcal{A}, \mathcal{H}_c)$ where $\mathcal{H}_c$ is shown in Figure 7 and $\mathcal{A}$ simply returns a bitstring encoding five random group elements. The messages of Π can be used as keys in the core hash function, while ciphertexts will be used as keys in the EHS's hash algorithm $\mathcal{H}^+$. In Appendix A.2 we show that $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ is hiding and binding under DDH. In Appendix A.3 we show that Δ^+ is SCR-secure under DL based on two lemmas. The first lemma establishes SCR security of $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ with fixed input length hash algorithm

$\mathcal{H}$. The second lemma shows us that we can apply Merkle-Damgård to $\mathcal{H}$ and the resulting encrypted hash scheme still has special collision resistance.

References

[1] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, Feb. 2010. 3, 11

[2] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Apr. 2009. 1, 2, 3, 5, 7

[3] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 162–177. Springer, Aug. 2002. 4, 8

[4] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 2

[5] 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. 6

[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. 1, 3, 5

[7] M. Bellare and S. Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive, Report 2009/101, 2009. `http://eprint.iacr.org/`. 3

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. 3

[9] F. Böhl, D. Hofheinz, and D. Kraschewski. On definitions of selective opening security. Cryptology ePrint Archive, Report 2011/678, 2011. `http://eprint.iacr.org/`. 3

[10] 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. 3, 7

[11] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. 1, 3, 4, 7

[12] R. Canetti, S. Halevi, and J. Katz. Adaptively-secure, non-interactive public-key encryption. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168. Springer, Feb. 2005. 1

[13] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. 3

[14] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 3, 10

[15] I. Damgård. A design principle for hash functions. In G. Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 416–427. Springer, Aug. 1990. 14, 23

[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. 7

[17] Y. Dodis, R. Oliveira, and K. Pietrzak. On the generic insecurity of the full domain hash. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 449–466. Springer, Aug. 2005. 2

[18] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. 5

[19] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003. 1, 2, 3, 4, 5, 10, 13

[20] 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. 1, 3

[21] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986. 1

[22] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. In M. Paterson, editor, *ICALP 90*, volume 443 of *LNCS*, pages 268–282. Springer, July 1990. 4

[23] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996. 4, 5

[24] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. 1

[25] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994. 5

[26] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 3, 6

[27] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 5

[28] 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*, LNCS, page ? Springer, Dec. 2011. 1, 3

[29] D. Hofheinz. Possibility and impossibility results for selective decommitments. *Journal of Cryptology*, 24(3):470–516, July 2011. 1, 2, 5

[30] G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 320–339. Springer, Mar. 2008. 7

[31] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. 1

[32] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Aug. 2002. 2, 4

[33] A. O'Neill, C. Peikert, and B. Waters. Bi-deniable public-key encryption. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 525–542. Springer, Aug. 2011. 7

[34] R. Ostrovsky, V. Rao, A. Scafuro, and I. Visconti. Revisiting lower and upper bounds for selective decommitments. Cryptology ePrint Archive, Report 2011/536, 2011. `http://eprint.iacr.org/`. 2

[35] S. Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 21–40. Springer, Feb. 2007. 1

[36] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Aug. 1992. 1

[37] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Aug. 2008. 7

[38] P. Rogaway. Formalizing human ignorance. In P. Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 211–228. Springer, Sept. 2006. 5

[39] D. Xiao. (nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 541–558. Springer, Mar. 2011. 2

A Details for Section 6

Here we provide the results and proofs omitted from Section 6.

```
Adversary $A(1^\lambda)$
$(a, \pi, ek) \leftarrow \textsc{Initialize}(1^\lambda)$
$\mathbf{c} \leftarrow \textsc{Enc}(\varepsilon)$
$b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(\mathbf{c}[n(\lambda)], \mathbf{c})$
$I \leftarrow \{2j - 1 + b[j] \,:\, 1 \le j \le h(\lambda)\}$
$b[1] \ldots b[z(\lambda)] \leftarrow ek$
For $j = 1 \ldots z(\lambda)$ do
    If $b[j] = 1$ then $I \leftarrow I \cup \{2h(\lambda) + j\}$
$I \leftarrow I \cup \{n(\lambda)\}$
$(\overline{\mathbf{m}}, \overline{\mathbf{r}}) \leftarrow \textsc{Corrupt}(I)$
$w \leftarrow (ek, \mathbf{c}, \overline{\mathbf{r}})$
$\textsc{Finalize}(w)$
```

```
Relation $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}, \alpha, I, w)$
If $\alpha \neq \varepsilon$ then return $\mathsf{false}$
$(ek, \mathbf{c}, \overline{\mathbf{r}}) \leftarrow w$
$b[1] \ldots b[h(\lambda)] \leftarrow \mathcal{H}(\mathbf{c}[n(\lambda)], \mathbf{c})$
$I' \leftarrow \{2j - 1 + b[j] \,:\, 1 \le j \le h(\lambda)\}$
$b[1] \ldots b[z(\lambda)] \leftarrow ek$
For $j = 1 \ldots z(\lambda)$ do
    If $b[j] = 1$ then $I' \leftarrow I' \cup \{2h(\lambda) + j\}$
$I' \leftarrow I' \cup \{n(\lambda)\}$
If $(I \neq I')$ then return $\mathsf{false}$
If $|\mathbf{c}| \neq n(\lambda)$ or $|\overline{\mathbf{r}}| \neq n(\lambda)$ then return $\mathsf{false}$
For all $i \in I$ do
    If $\mathcal{V}(1^\lambda, \pi, ek, \mathbf{c}[i], \mathbf{m}[i], \overline{\mathbf{r}}[i]) = \mathsf{false}$ then return $\mathsf{false}$
Return $\mathsf{true}$
```

Figure 8: Adversary A and relation R for the proof of Theorem A.1.

A.1 SOA-C insecurity of EHS schemes

We say that $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ has public-key length $z\colon \mathbb{N} \to \mathbb{N}$ if $|ek| = z(\lambda)$ for all $(ek, dk) \in [\mathcal{K}(1^\lambda)]$ and all $\lambda \in \mathbb{N}$.

Theorem A.1 *Let $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ be a special collision resistant EHS scheme where $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$ is a binding CE-scheme with message length $l\colon \mathbb{N} \to \mathbb{N}$ and public key length $z\colon \mathbb{N} \to \mathbb{N}$. Let $h\colon \mathbb{N} \to \mathbb{N}$ be the output length of $\mathcal{H}$. Let $n(\cdot) = 2h(\cdot) + z(\cdot) + 1$ and let $\mathcal{M}$ be the message sampler that on input $1^\lambda, \alpha$ (ignores α and) returns a $n(\lambda)$-vector whose components are uniformly and independently distributed over $\{0,1\}^{\ell(\lambda)}$. Let $\mathcal{A}$ be the auxiliary-input generator that returns ε on any input. Then there exists a PT soa-c adversary A and a PT relation R such that for all PT soa-c simulators S there is a negligible function ν such that $\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \ge 1 - \nu(\lambda)$ for all $\lambda \in \mathbb{N}$.* ▌

The proof will construct PT adversary C against the special collision-resistance of Δ and PT adversary B against the binding security of the Π such that for all λ we have

$$\mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda) \;\ge\; 1 - 2^{-\ell(\lambda)h(\lambda)} - \sqrt{\mathbf{Adv}^{\text{scr}}_{\Delta,C}(\lambda) + \mathbf{Adv}^{\text{bind}}_{\Pi,B}(\lambda)}.$$

The proof is very similar to the proof of Theorem 4.1, there are basically only two minor differences. First, the adversary C against the special collision resistance of Δ should give a public key ek to its challenge, so we add $z(\cdot)$ senders and we corrupt (or not) them accordingly to the value of ek, thus it is possible to recover the value of ek from the set of corrupt senders I. In addition, there should be one particular sender that is always corrupted and whose ciphertext is used as the key of the encrypted hash.

Proof of Theorem A.1: The adversary A and relation R are shown in Figure 8. Consider any PT simulator S. Note that for the specified adversary A, the output of the relation R is always true, therefore

$$\Pr\left[\,\mathrm{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] = 1 - \mathbf{Adv}^{\text{soa-c}}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A},A,S}(\lambda)$$

Now we construct a binding-adversary B and a scr-adversary C such that

$$\Pr\left[\,\mathrm{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] \;\le\; 2^{-\ell(\lambda)h(\lambda)} + \sqrt{\mathbf{Adv}^{\text{scr}}_{\Delta,C}(\lambda) + \mathbf{Adv}^{\text{bind}}_{\Pi,B}(\lambda)}\,. \tag{7}$$

Then Proposition 3.1, Eq. (7) and the hypotheses together imply that the theorem is true.

We divide the simulator S into sub-algorithms S_1, S_2 as in Section 4. We now define the adversaries B and C.

```
Adversary $B(1^\lambda)$
$\pi \leftarrow \textsc{Initialize}(1^\lambda)$ ; $(St, I) \leftarrow\!\!\$\; S_1(\varepsilon, \pi)$
$I_H \leftarrow I \cap \{1, \ldots, 2h(\lambda)\}$ ; $I_{PK} \leftarrow I \cap \{2h(\lambda)+1, \ldots, 2h(\lambda)+z(\lambda)\}$
```

```
$\mathbf{m}^* \leftarrow (\{0,1\}^{\ell(\lambda)})^{|I_{PK}|+1}$ ; $\overline{\mathbf{m}}_0 \leftarrow \overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_{PK} \cup \{n(\lambda)\}, \mathbf{m}^*, \bot_{n(\lambda)})$
$\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$ ; $\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_0^*, \overline{\mathbf{m}}_0)$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_1^*, \overline{\mathbf{m}}_1)$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$ ; $t \leftarrow_{\$} I$
For all $i \in I$ do If $\overline{\mathbf{m}}_0[i] \neq \overline{\mathbf{m}}_1[i]$ then $t \leftarrow i$
$\textsc{Finalize}(ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{r}}_0[t], \overline{\mathbf{r}}_1[t])$
```

```
Adversary $C(1^\lambda)$
$\pi \leftarrow \textsc{Initialize}(1^\lambda)$ ; $(St, I) \leftarrow_{\$} S_1(\varepsilon, \pi)$
$I_H \leftarrow I \cap \{1, \ldots, 2h(\lambda)\}$ ; $I_{PK} \leftarrow I \cap \{2h(\lambda)+1, \ldots, 2h(\lambda)+z(\lambda)\}$
For $j = 1 \ldots z(\lambda)$ do If $(2h(\lambda)+j \in I_P)$ then $b[j] = 1$ ; $ek \leftarrow b[1] \ldots b[z(\lambda)]$
$\mathbf{m}^* \leftarrow (\{0,1\}^{\ell(\lambda)})^{|I_{PK}|}$ ; $m' \leftarrow \textsc{Message}(ek)$
$\overline{\mathbf{m}}_0 \leftarrow \overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_{PK} \cup \{n(\lambda)\}, \mathbf{m}^* \parallel m', \bot_{n(\lambda)})$
$\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$ ; $\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_0^*, \overline{\mathbf{m}}_0)$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_1^*, \overline{\mathbf{m}}_1)$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$
$\textsc{Finalize}(\mathbf{c}_0[n(\lambda)], \mathbf{c}_1[n(\lambda)], \mathbf{r}_0[n(\lambda)], \mathbf{c}_1[n(\lambda)], \mathbf{c}_0, \mathbf{c}_1)$.
```

Let $V_\lambda = (2^{-\ell(\lambda)})^{h(\lambda)}$, let $V = \{V_\lambda\}_{\lambda\in\mathbb{N}}$ and define the algorithms P_1, P_2 as in Section 4. Then

$$\Pr\left[\,\mathrm{SSOAC}^{S}_{\Pi,\mathcal{M},\mathsf{R},\mathcal{A}}(\lambda)\,\right] \;=\; \mathbf{AP}_1(P_1, P_2, V, \lambda) \;\leq\; 2^{-\ell(\lambda)h(\lambda)} + \sqrt{\mathbf{AP}_2(P_1, P_2, V, \lambda)} \tag{8}$$

Finally we will show that

$$\mathbf{AP}_2(P_1, P_2, V, \lambda) \;\leq\; \mathbf{Adv}^{\mathrm{scr}}_{\Delta,C}(\lambda) + \mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\lambda). \tag{9}$$

Eqs. (8) and (9) together imply the truth of Eq. (7). Writing down the double-execution experiment underlying $\mathbf{AP}_2(P_1, P_2, V, \lambda)$:

```
$\pi \leftarrow \mathcal{P}(1^\lambda)$ ; $(St, I) \leftarrow_{\$} S_1(\varepsilon, \pi)$ ; $\mathbf{m} \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{n(\lambda)}$ ; $\mathbf{m}_0^*, \mathbf{m}_1^* \leftarrow_{\$} (\{0,1\}^{\ell(\lambda)})^{h(\lambda)}$
$I_H \leftarrow I \cap \{1, \ldots, 2h(\lambda)\}$ ; $I_{PK} \leftarrow I \cap \{2h(\lambda)+1, \ldots, 2h(\lambda)+z(\lambda)\}$
$\overline{\mathbf{m}}_0 \leftarrow \overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_{PK} \cup \{n(\lambda)\}, \mathbf{m}, \bot_{n(\lambda)})$
$\overline{\mathbf{m}}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_0^*, \overline{\mathbf{m}}_0)$ ; $\overline{\mathbf{m}}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_1^*, \overline{\mathbf{m}}_1)$
$w_0 \leftarrow S_2(St, \overline{\mathbf{m}}_0)$ ; $w_1 \leftarrow S_2(St, \overline{\mathbf{m}}_1)$ ; $(ek_0, \mathbf{c}_0, \overline{\mathbf{r}}_0) \leftarrow w_0$ ; $(ek_1, \mathbf{c}_1, \overline{\mathbf{r}}_1) \leftarrow w_1$
$\mathbf{m}_0 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_0^*, \mathbf{m})$ ; $\mathbf{m}_1 \leftarrow \mathrm{Emb}(1^{n(\lambda)}, I_H, \mathbf{m}_1^*, \mathbf{m})$
Return $\mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_0, \varepsilon, I, w_0) \wedge \mathsf{R}(1^\lambda, a, \pi, \mathbf{m}_1, \varepsilon, I, w_1) \wedge (\mathbf{m}_0^* \neq \mathbf{m}_1^*)$.
```

As in Section 4, if this experiment returns true, then $\mathcal{H}(\mathbf{c}_0[n(\lambda)], \mathbf{c}_0) = \mathcal{H}(\mathbf{c}_1[n(\lambda)], \mathbf{c}_1)$. If the experiments returns true we also have $ek_0 = ek_1$, since the set of corrupted senders is the same in both executions and the public keys are recoverable from this set. These facts together with the tests perform by R, implies that C succeeds unless $\mathbf{c}_0 = \mathbf{c}_1$. But if $\mathbf{c}_0 = \mathbf{c}_1$, then $\mathbf{c}_0[t] = \mathbf{c}_1[t]$ and by the definition of R we have $\mathcal{V}(1^\lambda, \pi, ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_0[t], \overline{\mathbf{r}}_0[t]) = \mathsf{true}$ and $\mathcal{V}(1^\lambda, \pi, ek_0, \mathbf{c}_0[t], \overline{\mathbf{m}}_1[t], \overline{\mathbf{r}}_1[t]) = \mathsf{true}$ with $\overline{\mathbf{m}}_0[t] \neq \overline{\mathbf{m}}_1[t]$ and therefore B wins. ▌

A.2 IND-CPA and Binding of the CE underlying Δ^+

We say the discrete logarithm advantage of an adversary A against fixed group generator $\mathcal{G}$ is

$$\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G},A}(\lambda) = \Pr\left[\,\mathrm{DL}^{A}_{\mathcal{G}}(\lambda)\,\right] ,$$

where game $\mathrm{DL}_{\mathcal{G}}$ is show in Figure 9. We say the DDH advantage of an adversary B against fixed group generator $\mathcal{G}$ is

$$\mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},B}(\lambda) = 2 \cdot \Pr\left[\,\mathrm{DDH}^{B}_{\mathcal{G}}(\lambda)\,\right] - 1 ,$$

where game $\mathrm{DDH}_{\mathcal{G}}$ is also shown in Figure 9.

Consider EHS Δ^+ defined in Section 6 with underlying CE Π. We show that Π is IND-CPA and binding.

```
$\underline{\textsc{Initialize}(1^\lambda)}$
$(\mathbb{G}, p) \leftarrow \mathcal{G}(1^\lambda)$ ; $b \leftarrow\$ \{0,1\}$
$g \leftarrow\$ \mathbb{G}^*$ ; $x, y \leftarrow\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$ ; $Y \leftarrow g^y$
If $b = 0$ then $Z \leftarrow g^{xy}$ else $Z \leftarrow\$ \mathbb{G}$
Return $(g, X, Y, Z)$
$\underline{\textsc{Finalize}(b')}$
Return $(b = b')$
```

```
$\underline{\textsc{Initialize}(1^\lambda)}$
$(\mathbb{G}, p) \leftarrow \mathcal{G}(1^\lambda)$
$g \leftarrow\$ \mathbb{G}^*$ ; $y \leftarrow\$ \mathbb{Z}_p$ ; $h \leftarrow g^y$
Return $(g, h)$
$\underline{\textsc{Finalize}(y')}$
Return $(y = y')$
```

Figure 9: Game $\mathrm{DDH}_{\mathcal{G}}$ (left) and $\mathrm{DL}_{\mathcal{G}}$ (right).

```
**Algorithm** $C(g, X, Y, Z)$
$b \leftarrow\$ \{0,1\}$ ; $i \leftarrow\$ \{1,2,3,4,5\}$
Run $A(1^\lambda, \varepsilon)$
On query $\mathrm{LR}(M_0, M_1)$:
    $(m_1, m_2, m_3, m_4, m_5) \leftarrow M_b$
    $h \leftarrow X$
    $u_i \leftarrow Y$ ; $w_i \leftarrow m_i \cdot Z$
    For $1 \le j < i$ do
        $a_k \leftarrow\$ \mathbb{Z}_p$ ; $u_k \leftarrow g^{a_k}$ ; $w_k \leftarrow\$ \mathbb{G}$
    For $i < k \le 4$ do
        $a_j \leftarrow\$ \mathbb{Z}_p$ ; $u_j \leftarrow g^{a_j}$ ; $w_j \leftarrow X^{a_j} \cdot m_j$
    Return $(g, h, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3, w_4, w_5)$
On $\textsc{Finalize}(b')$:
    Output $(b = b')$
```

```
$\underline{\textsc{Initialize}(1^\lambda)}$
$b \leftarrow\$ \{0,1\}$
Return $(1^\lambda, \emptyset)$
$\underline{\mathrm{LR}(M_0, M_1)}$
$(m_1, m_2, m_3, m_4, m_5) \leftarrow M_b$
$u_1, u_2, u_3, u_4, u_5 \leftarrow\$ \mathbb{G}$
$g \leftarrow\$ \mathbb{G}^*$ ; $s \leftarrow\$ \mathbb{Z}_p$ ; $h \leftarrow g^s$
For $j = 1, 2, 3, 4, 5$ do
    If $j \le i$ then
        $w_j \leftarrow\$ \mathbb{G}$
    If $j > i$ then
        $w_j \leftarrow m_j \cdot u_j^s$
Return $(g, h, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3, w_4, w_5)$
$\underline{\textsc{Finalize}(b')}$
Return $(b' = b)$
```

Figure 10: Algorithm C and hybrid game H_i used in the proof of Theorem A.2.

Theorem A.2 (The CE scheme underlying Δ^+ is IND-CPA and binding) *Let λ be the security parameter, $\mathcal{G}$ be a fixed group generator, $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$ be the encrypted hash scheme defined in Section 6 and built from CE scheme $\Pi = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V})$. Let A be an IND-CPA adversary against Π and let B be a binding adversary against Π. Then there exists DDH adversary C against $\mathcal{G}$ such that for all λ*

$$\mathbf{Adv}^{\mathrm{indcpa}}_{\Pi,A}(\lambda) \le 10 \cdot \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},C}(\lambda)$$

and

$$\mathbf{Adv}^{\mathrm{bind}}_{\Pi,B}(\lambda) = 0\ . \quad \blacksquare$$

Binding is immediate since $h, u_1, u_2, u_3, u_4, u_5$ uniquely determine m_1 through m_5. To show IND-CPA, we can do a hybrid argument. Consider DDH adversary C and hybrid game H_i shown in Figure 10. Hybrid game H_0 is the same as the standard IND game. We see that

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2} \cdot \mathbf{Adv}^{\mathrm{ind-cpa}}_{\Pi,A}(\lambda) &= \Pr\left[\mathrm{H}_0^A(\lambda) \Rightarrow 1\right] \\
&\le \Pr\left[\mathrm{H}_5^A(\lambda) \Rightarrow 1\right] + 5 \cdot \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},C}(\lambda) \\
&\le \frac{1}{2} + 5 \cdot \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},C}(\lambda)
\end{aligned}$$

The last equation is true since in game H_5 the messages are not used, so nothing depending on the bit is given to the adversary and its probability of success is exactly 1/2. (In fact, to make this more explicit, we could transition to another game in which the choice of b is moved to $\textsc{Finalize}$.) $\blacksquare$

A.3 SCR-security of Δ^+

Let $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$ be the EHS specified in Section 6 with arbitrary input length hash algorithm $\mathcal{H}^+$ built from $\mathcal{H}$ using Merkle-Damgård. Before proceeding, we give the details of pad used in the construction of $\mathcal{H}^+$ from $\mathcal{H}$.

Recall that we let $k < 5 \log p_\lambda$ be the size in bits to represent four group elements and let $b = 5\lfloor \log p \rfloor - k$ be the number of bits each call to $\mathcal{H}$ compresses by. For simplicity, assume the groups we are using are such that are hash function compresses by at least 2 bits, i.e., $b \geq 2$. Let pad, on input T, be as follows:

(1) Parse input T into $b-1$ bit blocks $z_1, \ldots, z_n$. If $|T|$ is not a multiple of $b-1$, pad the last block with 0s. Let d be the number of bits of padding used. Set z_{n+1} to the $b-1$ bit binary encoding of d.

(2) Set $t_1 = 0 \,\|\, z_1$, and $t_i = 1 \,\|\, z_i$ for $i > 1$. Output $(t_1, \ldots, t_{n+1})$.

We now claim the following.

Theorem A.3 (Δ^+ is SCR if DL is hard) *Let λ be the security parameter, $\mathcal{G}$ be a fixed group generator, $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$ be the encrypted hash scheme described above with arbitrary input length hash algorithm $\mathcal{H}^+$. Let C be an adversary attacking the special collision resistance of Δ^+. Then there exists a discrete log adversary B against $\mathcal{G}$ such that for all λ*

$$\mathbf{Adv}^{\mathrm{scr}}_{\Delta^+,C}(\lambda) \leq 5 \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G},A}(\lambda) \quad \blacksquare$$

As we stated in the body, to prove the theorem, we prove two lemmas. The first lemma establishes the special collision resistance of the encrypted hash scheme Δ with fixed-length hash algorithm $\mathcal{H}$.

Lemma A.4 (Δ is SCR if DL is hard) *Let λ be the security parameter, $\mathcal{G}$ be a fixed group generator, $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ be the encrypted hash scheme in Figure 7 with fixed input length hash algorithm $\mathcal{H}$. Let C be an adversary attacking the special collision resistance of Δ. Then there exists a discrete log adversary B against $\mathcal{G}$ such that for all λ*

$$\mathbf{Adv}^{\mathrm{scr}}_{\Delta,C}(\lambda) \leq 5 \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G},B}(\lambda) \ . \quad \blacksquare$$

The proof of Lemma A.4 can be broken down into the following two claims:

Claim A.5 [Δ is SCR if $\mathcal{H}_c$ is CR] Let λ be the security parameter, let $\mathcal{G}$ be a fixed group generator, let $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ be the encrypted hash scheme shown in Figure 7 with fixed-length hash function $\mathcal{H}$, and let $\Gamma = (\mathcal{A}, \mathcal{H}_c)$ be the core hash function described above. Let C be an adversary against the special collision resistance of Δ. Then there exists an adversary B against the collision resistance of Γ such that for all λ

$$\mathbf{Adv}^{\mathrm{scr}}_{\Delta,C}(\lambda) \leq \mathbf{Adv}^{\mathrm{cr}}_{\Gamma,B}(\lambda) \ . \quad \blacksquare$$

Claim A.6 [$\mathcal{H}_c$ is CR assuming DL is hard] Let λ be the security parameter. Let $\mathcal{G}$ be a fixed group generator. Let $\Gamma = (\mathcal{A}, \mathcal{H}_c)$ be the hash function described above. Let B be an adversary against the (standard) collision resistance of Γ. Then there exists a discrete log adversary A against $\mathcal{G}$ such that for all λ

$$\mathbf{Adv}^{\mathrm{cr}}_{\Gamma,B}(\lambda) \leq 5 \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G},A}(\lambda) \ . \quad \blacksquare$$

The first claim states that the special collision resistance of Δ is tied to the (standard) collision resistance of the underlying core hash function. The second claim establishes the collision resistance of the core hash function based on the hardness of discrete log in our fixed group setting.

Proof of Claim A.5: Let C be an adversary against the special collision resistance of $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$, the encrypted hash scheme with fixed input length hash algorithm $\mathcal{H}$. We will build an adversary B against the (standard) collision resistance of the core hash function $\Gamma = (\mathcal{A}, \mathcal{H}_c)$. Adversary B, on input core hash function key m that can be parsed into $(m_1, m_2, m_3, m_4, m_5)$, runs C and answers its single

Message query with this core hash key. (In other words, the hash function key is used as the message.) When C halts with output (C, C', r, r', X, X'), B outputs (X, X') as a collision on core hash function Γ.

We will now argue that if C succeeds in finding a special collision against Δ, then B succeeds in finding a collision for core hash function Γ. Suppose C is successful in the SCR game with its output (C, C', r, r', X, X'). About this output we know

(1) $C = (g, h, u_1, u_2, u_3, u_4, u_5, w_1, w_2, w_3, w_4, w_5)$ and $C' = (g', h', u'_1, u'_2, u'_3, u'_4, u'_5, w'_1, w'_2, w'_3, w'_4, w'_5)$

(2) r, the coins leading to ciphertext C, contains s such that $g^s = h$ and $w_i = u_i^s m_i$ for $i = 1, 2, 3, 4, 5$. Similarly, r' contains s' such that ${g'}^{s'} = h'$ and $w'_i = {u'_i}^{s'} m'_i$ for $i = 1, 2, 3, 4, 5$.

(3) $X = (x_1, x_2, x_3, x_4, x_5)$ and $X' = (x'_1, x'_2, x'_3, x'_4, x'_5)$ such that $(x_1, x_2, x_3, x_4, x_5) \neq (x'_1, x'_2, x'_3, x'_4, x'_5)$, meaning at least one of the $x_i \neq x'_i$.

(4) $\mathcal{H}(1^\lambda, C, X) = \mathcal{H}(1^\lambda, C', X') = (Y_1, Y_2, Y_3, Y_4)$.

Now, we first claim that $g = g'$ and $h = h'$, and therefore $s = s'$. To see this, note that the first two components of the output of the hash function, Y_1 and Y_2, are simply the first two elements of its key. Therefore, if $\mathcal{H}(1^\lambda, C, X) = \mathcal{H}(1^\lambda, C', X')$, then the first two elements of their outputs must be the same, meaning the first two elements of their keys are the same. This means g and g' are equal, h and h' are equal and, as a result, s and s' as well.

Next, we claim that $m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5} = m_1^{x'_1} m_2^{x'_2} m_3^{x'_3} m_4^{x'_4} m_5^{x'_5}$, meaning that X and X' give a collision for the core hash function. To see this, note that

$$\begin{aligned}
\mathcal{H}(1^\lambda, C, X) &= (Y_1, Y_2, Y_3, Y_4) \\
&= (g, h, u_1^{x_1} u_2^{x_2} u_3^{x_3} u_4^{x_4} u_5^{x_5}, w_1^{x_1} w_2^{x_2} w_3^{x_3} w_4^{x_4} w_5^{x_5}) \\
&= (g, h, u_1^{x_1} u_2^{x_2} u_3^{x_3} u_4^{x_4} u_5^{x_5}, (u_1^{x_1} u_2^{x_2} u_3^{x_3} u_4^{x_4} u_5^{x_5})^s m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5})
\end{aligned}$$

From this we can see that

$$Y_4 / Y_3^s = m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5}$$

and

$$Y_4 / Y_3^{s'} = m_1^{x'_1} m_2^{x'_2} m_3^{x'_3} m_4^{x'_4} m_5^{x'_5} \ .$$

But, as we just saw, $s = s'$, so X and X' actually lead to a collision in $\mathcal{H}_c$. ▌

Proof of Claim A.6: This is actually a simpler version of an exercise in the Katz-Lindell book. We include a full proof for completeness. Consider the sequence of games in Figure 11 to be played with a collision resistance adversary B on the core hash function $\Gamma = (\mathcal{A}, \mathcal{H}_c)$.

Now, consider a discrete logarithm adversary A against $\mathcal{G}$ that, on input (g, h) runs adversary B exactly as in game G_0, but using h from its input instead of generating it as in the game. In other words, A guesses which of the four parts of the key it should embed the DL challenge h in. (This guess is variable i in the game.) When B outputs its collision guess x, x', adversary A fails and outputs $\perp$ if $x_i = x'_i$, but if $x_i \neq x'_i$ then A lets j, k, ℓ, n be the indices other than i and outputs

$$y = \frac{a_j(x'_j - x_j) + a_k(x'_k - x_k) + a_\ell(x'_\ell - x_\ell) + a_n(x'_n - x_n)}{(x_i - x'_i)} \ .$$

It is easy to see that if B succeeds in finding a collision and $x_i \neq x'_i$, then A will succeed in computing the discrete logarithm of h. To exactly bound A's success, we go through the game transitions found in Figure 11. In game G_2, the key sampling is done without the use of i, but is equivalent. In game G_3,

```
INITIALIZE($1^\lambda$)
$(\mathbb{G}, p) \leftarrow \mathcal{G}(1^\lambda)$
$g \leftarrow_{\$} \mathbb{G}^*$
$i \leftarrow_{\$} \{1,2,3,4,5\}$
$a_1, a_2, a_3, a_4, a_5 \leftarrow_{\$} \mathbb{Z}_p$ ; $h \leftarrow g^{a_i}$
For $j \in \{1,2,3,4,5\} \setminus \{i\}$ do $m_j \leftarrow g^{a_j}$
$m_i \leftarrow h$
Return $(m_1, m_2, m_3, m_4, m_5)$

FINALIZE($(x, x')$)
$(x_1, x_2, x_3, x_4, x_5) \leftarrow x$
$(x'_1, x'_2, x'_3, x'_4, x'_5) \leftarrow x'$
If $m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5} = m_1^{x'_1} m_2^{x'_2} m_3^{x'_3} m_4^{x'_4} m_5^{x'_5} \wedge x \neq x'$ then
    $\mathsf{event}_1 \leftarrow \mathsf{true}$
If $x_i \neq x'_i$ then $\mathsf{event}_2 \leftarrow \mathsf{true}$
Return $\mathsf{event}_1 \wedge \mathsf{event}_2$
```

```
INITIALIZE($1^\lambda$)
$(\mathbb{G}, p) \leftarrow \mathcal{G}(1^\lambda)$
$i \leftarrow_{\$} \{1,2,3,4,5\}$
$m_1, m_2, m_3, m_4, m_5 \leftarrow_{\$} \mathbb{G}$
Return $(m_1, m_2, m_3, m_4, m_5)$

FINALIZE($(x, x')$)
$(x_1, x_2, x_3, x_4, x_5) \leftarrow x$
$(x'_1, x'_2, x'_3, x'_4, x'_5) \leftarrow x'$
If $m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5} = m_1^{x'_1} m_2^{x'_2} m_3^{x'_3} m_4^{x'_4} m_5^{x'_5} \wedge x \neq x'$ then
    $\mathsf{event}_1 \leftarrow \mathsf{true}$
If $x_i \neq x'_i$ then $\mathsf{event}_2 \leftarrow \mathsf{true}$
Return $\mathsf{event}_1 \wedge \mathsf{event}_2$
```

```
INITIALIZE($1^\lambda$)
$(\mathbb{G}, p) \leftarrow \mathcal{G}(1^\lambda)$
$m_1, m_2, m_3, m_4, m_5 \leftarrow_{\$} \mathbb{G}$
Return $(m_1, m_2, m_3, m_4, m_5)$

FINALIZE($(x, x')$)
$(x_1, x_2, x_3, x_4, x_5) \leftarrow x$
$(x'_1, x'_2, x'_3, x'_4, x'_5) \leftarrow x'$
If $m_1^{x_1} m_2^{x_2} m_3^{x_3} m_4^{x_4} m_5^{x_5} = m_1^{x'_1} m_2^{x'_2} m_3^{x'_3} m_4^{x'_4} m_5^{x'_5} \wedge x \neq x'$ then
    $\mathsf{event}_1 \leftarrow \mathsf{true}$
$i \leftarrow_{\$} \{1,2,3,4,5\}$
If $x_i \neq x'_i$ then $\mathsf{event}_2 \leftarrow \mathsf{true}$
Return $\mathsf{event}_1 \wedge \mathsf{event}_2$
```

Figure 11: Games G_0, G_1, G_2 (top to bottom) used in the proof of Claim A.6

since i is no longer used in INITIALIZE, we move it to FINALIZE. We can see that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G},A}(\lambda) &= \Pr\left[\,G_0^B(\lambda)\,\right] = \Pr\left[\,G_1^B(\lambda)\,\right] = \Pr\left[\,G_2^B(\lambda)\,\right] && (10)\\
&= \Pr\left[\,\mathsf{event}_2 \wedge \mathsf{event}_1\,\right] && (11)\\
&= \Pr\left[\,\mathsf{event}_2 | \mathsf{event}_1\,\right] \cdot \Pr\left[\,\mathsf{event}_1\,\right] && (12)\\
&\geq \frac{1}{5} \cdot \Pr\left[\,\mathsf{event}_1\,\right] && (13)\\
&= \frac{1}{5} \cdot \mathbf{Adv}^{\mathrm{cr}}_{\Gamma,B}(\lambda) && (14)
\end{aligned}$$

where event_2 and event_1 above refer to those events within G_2. Eq. (13) is true since if $x \neq x'$ there must be at least 1 of the 5 components that differs. Finally, the last equation follows since running B in G_2 exactly matches the collision resistance game, and event_1 exactly matches the output of FINALIZE in that game. ▌

The next lemma shows that we can use Merkle-Damgård with our fixed-length hash function and get special collision resistance for the encrypted hash scheme Δ^+.

Lemma A.7 (Δ^+ is SCR if Δ is SCR) *Let λ be the security parameter. Let $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$ be the encrypted hash scheme shown in Figure 7 with fixed input length hash function $\mathcal{H}$. Let $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$ be the encrypted hash scheme built from the same underlying CE scheme, but with hash function $\mathcal{H}^+$ constructed from $\mathcal{H}$ using the MD transform and padding described above. Let C be an adversary attacking the special collision resistance of Δ^+. Then there exists adversary A attacking the* *special collision resistance of* Δ *such that for all* λ

$$\mathbf{Adv}^{\mathrm{scr}}_{\Delta^+,C}(\lambda) \leq \mathbf{Adv}^{\mathrm{scr}}_{\Delta,A}(\lambda) \ . \ \blacksquare$$

Proof of Lemma A.7: We closely follow the proof given by Damgård in his original paper [15]. Of course, we are trying to prove special collision resistance is preserved instead of collision resistance but, as we shall see, the proofs are essentially the same.

Let C be an adversary attacking the special collision resistance of $\Delta^+ = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H}^+)$, the encrypted hash scheme with arbitrary input length hash algorithm $\mathcal{H}^+$. We build an adversary A attacking the special collision resistance of $\Delta = (\mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{V}, \mathcal{H})$. Adversary A runs C and forwards its Message query to its own Message oracle, returning the same answer. Then, at some point C halts with output (C, C', r, r', T, T').

Suppose C wins the SCR game after receiving Message answer m and outputting (C, C', r, r', T, T') as a special collision against encrypted hash scheme Δ^+. This means that $T \neq T'$, $\mathcal{H}^+(1^\lambda, C, T) = \mathcal{H}^+(1^\lambda, C', T')$, and both C and C' are valid encryptions of message m (under coins r and r', respectively). Let $(t_1, \ldots, t_{n+1}) \leftarrow \mathsf{pad}(T)$ and $(t'_1, \ldots, t'_{n'+1}) \leftarrow \mathsf{pad}(T')$ be the $b-1$ bit blocks of T and T', respectively. Let $y_1, \ldots, y_{n+1}$ and $y'_1, \ldots, y'_{n'+1}$ be the corresponding intermediate values in the computation of $\mathcal{H}^+$.

First, consider the case where T and T' require different amounts of padding. Since the last block is the binary encoding of the amount of padding, this means that $t_{n+1} \neq t'_{n'+1}$ and yet $\mathcal{H}(1^\lambda, C, y_n \,\|\, t_{n+1}) = \mathcal{H}(1^\lambda, C', y'_{n'} \,\|\, t'_{n'+1})$. Thus, $(C, C', r, r', y_n \,\|\, t_{n+1}, y'_{n'} \,\|\, t'_{n'+1})$ is a valid special collision against Δ and message m.

Next, consider the case where T and T' are exactly the same length (before and after padding). Since $T \neq T'$, there must be some i such that $t_i \neq t'_i$. Let i^* be the highest such i. It follows that $\mathcal{H}(1^\lambda, C, y_{i^*-1} \,\|\, t_{i^*}) = \mathcal{H}(1^\lambda, C', y'_{i^*-1} \,\|\, t'_{i^*})$, so $(C, C', r, r', y_{i^*-1} \,\|\, t_{i^*}, y'_{i^*-1} \,\|\, t'_{i^*})$ is a valid special collision against Δ and message m.

Finally, consider the case where $|T| \neq |T'|$, yet their padding lengths are the same. This means one must contain more blocks than the other. Without loss, let $|T'| > |T|$, meaning $n' > n$. Now we know that $\mathcal{H}(1^\lambda, C, y_n \,\|\, t_{n+1}) = \mathcal{H}(1^\lambda, C', y'_{n'} \,\|\, t'_{n'+1})$. Now let $j = n$ and $j' = n'$. If $(y_j \,\|\, t_{j+1}) \neq (y'_{j'} \,\|\, t'_{j'+1})$, we have our collision $(C, C', r, r', y_j \,\|\, t_{j+1}, y'_{j'} \,\|\, t'_{j'+1})$. Otherwise, we subtract one from j and j' and check again if we have a collision. Eventually, j will be 1 and $j' > j$ since T' had more blocks than T. Because of our padding scheme, this means that $t_j = t_1$ which starts with a 0, while $t_{j'}$ with $j' > 1$ will start with a 1. We will necessarily have a special collision at this point.

We can see that A succeed whenever C succeeds, proving the bound. $\blacksquare$

Combining Lemmas A.4 and A.7 completes the proof of Theorem A.3.