Semantically-Secure Functional Encryption: Possibility Results, Impossibility Results and the Quest for a General Definition

Mihir Bellare[1] Adam O'Neill[2]

August 2012

Abstract

This paper explains that SS1-secure functional encryption (FE) as defined by Boneh, Sahai and Waters implicitly incorporates security under key-revealing selective opening attacks (SOA-K). This connection helps intuitively explain their impossibility results and also allows us to prove stronger ones. To fill this gap and move us closer to the (laudable) goal of a general and achievable notion of FE security, we seek and provide two "sans SOA-K" definitions of FE security that we call SS2 and SS3. We prove various possibility results about these definitions. We view our work as a first step towards the challenging goal of a general, meaningful and achievable notion of FE security.

[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://www.cs.ucsd.edu/users/mihir`. Supported in part by NSF grants CNS-0627779 and CCF-0915675.

[2] Department of Computer Computer Science, Boston University, 111 Cummington St., Boston, MA 02215. Email: `amoneill@bu.edu`. URL: `http://cs-people.bu.edu/amoneill`. Supported in part by NSF grants 0546614, 0831281, 1012910, and 1012798.

1 Introduction

Background. Functional encryption (FE) was introduced by Boneh, Sahai and Waters (BSW) [14]. A FE-scheme for a functionality $\mathcal{F}\colon \mathbb{N} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is a tuple of algorithms $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$. An authority lets $(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$, where λ is the security parameter, and publishes pk. Anyone may now encrypt an input x via $c \leftarrow\$ \mathsf{Enc}(pk, x)$. A user may provide the authority with a *functionality index* a and receive a secret key $sk_a \leftarrow\$ \mathsf{KDer}(sk, a)$. If the user now applies the decryption algorithm to sk_a and any encryption c of x, the result $\mathsf{Dec}(sk_a, c)$ will equal $\mathcal{F}(\lambda, a, x)$. Security requires that the user learns nothing more.

The intent was to generalize and unify many forms of encryption including IBE (Identity-based encryption) [32, 13], ABE (Attribute-based encryption) [31, 20] and PE (Predicate encryption) [22]. An existing form $\mathbf{E}$ of encryption would correspond to a functionality $\mathcal{F}_{\mathbf{e}}$. IBE for example corresponds to the functionality $\mathcal{F}_{\mathrm{ibe}}$ which regards a as an identity and parses x as a pair (a', m) consisting of another identity a' and a message m, returning m if $a = a'$ and $\perp$ otherwise. PE generalizes to functionalities $\mathcal{F}$ for which there is a relation $\mathcal{P}$ such that $\mathcal{F}$, given a and $x = (a', m)$, returns m if $\mathcal{P}(a, a')$ is true and $\perp$ otherwise, IBE being the case where $\mathcal{P}(a, a')$ is true iff $a = a'$. ABE schemes are a subclass of PE schemes.

BSW [14] sought a general definition of security that applied to an arbitrary functionality. They first provide an indistinguishability-based one (IND). It had the attractive feature of coinciding, for the IBE and PE functionalities, with the the existing definitions of these notions from the literature. But both BSW [14] and O'Neill [29] point to inherent deficiencies of IND when it comes to capturing security of general functionalities. The "main" definition of BSW was accordingly a simulation-based semantic-security one that we call SS1. We may now speak of the SS1-security of an FE scheme FE for *any* functionality $\mathcal{F}$.

The FE framework is elegant and the goals are laudable. A proliferating number of notions of encryption are now put under a single umbrella, seen as special cases of a single primitive. Ad hoc, notion-specific security definitions need not be given. One only has to specify the functionality and SS1 security would return a suitable definition.

Impossibility of SS1 in the NPROM. However, having introduced SS1, BSW [14] claim that it can't be achieved in the standard model, even for IBE, which is the most basic functionality in this area. This is a strong and disappointing claim. Before we delve into its implications, we take a closer look at it. We point out that BSW don't actually prove this. What they prove is that SS1-secure IBE cannot be achieved in the NPROM (Non-Programmable Random Oracle Model). At a first glance, this only sounds like a stronger claim. Every standard model scheme is a NPROM scheme and every standard-model adversary is a NPROM one, so if NPROM achievability is ruled out, isn't standard model achievability ruled out as well? The answer is no. BSW [14] establish their claim by providing an adversary for which they prove that there is no simulator. But their adversary makes calls to the RO, and this is exploited crucially in the proof of non-existence of a simulator. Their proof does not rule out the existence of a simulator for adversaries that do not call the RO, meaning for standard-model adversaries, and thus it does not rule out standard-model achievability of SS1, even for IBE.

This gives a ray of hope. Perhaps SS1-security can be achieved in the standard model after all. This would be interesting even for IBE and certainly beyond. This hope is fueled by a look at the technique underlying the negative result of BSW [14]. It is not a priori clear how to extend this technique to rule out simulators for standard-model adversaries.

A new impossibility result for SS1. We fill the gap by showing that SS1-secure IBE is not achievable even in the standard model. The result is actually more general, ruling out SS1-security for *any non-trivial functionality*, IBE being covered as a special case. Non-triviality essentially means the functionality is not a constant function. The only assumption made is the existence of collision-resistant hash functions.

Our result exploits the recent technique of Bellare, Dowsley, Waters and Yilek (BDWY) [5], used to prove the impossibility of SOA-secure commitment, in combination with techniques from Nielsen's proof of impossibility of non-committing encryption (NCE) [26]. We are able to present a *standard model* adversary for which we can prove that there is no simulator.

Taking a closer look, our result, as is the case with those of Nielsen and BSW, is actually a trade-off. It shows that SS1-security requires long keys, this meaning that the total number of bits in messages securely encrypted must be bounded by the length of a secret key. However, it does this in the standard model.

An explanation. This paper offers an explanation for this anamoly that seeds further contributions in a natural way. We contend that SS1 does not capture "plain" FE security. Instead, it captures FE security in the presence of key-revealing selective-opening attacks (SOA-Ks). These are attacks where the adversary may adaptively corrupt some users and obtain their decryption keys *without restrictions*.[1] The revealing fact is that, if we were to write down a definition of SOA-K-security for IBE, what emanates is *exactly* SS1-secure IBE. We now have a natural explanation of why SS1 is subject to such broad unachievability and also why SS1-secure IBE is not the same as the classical IND-secure IBE from [13]. Namely, the former incorporates SOA-K security and the latter does not.

Why is SOA-K-security part of SS1? BSW [14] did not throw it in "on purpose." (Their work has no explicit recognition of the fact that their definition incorporates security against SOA-K. They do however comment on the relation to NCE and [26], which is only a step removed.) Rather, the natural approach to defining semantic security for a general functionality, which is the one followed by BSW [14], leads to the inadvertent incorporation of SOA-K security.

While it is usually easier to define "plain" security than security against SOA-K, with FE, it seems to be the opposite. It is not clear how to define semantically-secure FE in a way that "decouples" basic and SOA-K security. This, in our view, is rather interesting.

SS2 and SS3. As indicated above, we believe that unifying different existing forms of encryption under a general definition for FE is a highly worthwhile goal. SS1 has not achieved this, capturing instead the SOA-K-secure versions of these goals and thence being subject to strong impossibility results. We move towards the just-stated goal with two new notions that we call SS2 and SS3. Defining "sans SOA-K" FE security in forms of varying strength, they are able to meet many of the broad goals in this domain and open the door to further efforts.

Our main result about SS2 is that it is equivalent to IND for *all* functionalities. This equivalence has its plusses and its minuses. Let us begin with the former. IND-secure IBE as per [13] is a well established definition, targeted in thousands of papers and proven to work for applications, and IND-secure PE as per [22] is also accepted. The SS2=IND equivalence provides a semantic-security based backing for this IND definition which has so far been absent. Conceptually, it mirrors in the FE setting the classic equivalence between semantic-security and indistinguishability in the PKE setting [18] that is a cornerstone of our understanding of, and faith in, these definitions. More pragmatically, it immediately yields possibility results for semantically-secure FE which were absent under SS1. This is because IND-secure IBE is well-known to be achievable in the standard model [11, 36, 34], and various possibility results for ABE and PE are known as well [20, 30, 22, 27, 33, 23, 2, 25, 24, 28].

We believe this is progress towards bringing semantically-secure FE closer. But, while the equivalence of SS2 with IND is a plus for common functionalities like ABE, PE and IBE, it is a minus when looking further, for we already know that IND is *not* a good definition of FE security in general [29]. Thus, we would like another definition to complement SS2. We suggest SS3, a strengthening of SS2. We believe SS3 is a good candidate for a general definition of FE for arbitrary functionalties. One reason is that it does not appear to have the drawbacks of IND for beyond-PE functionalities. (BSW [14] and O'Neill [29] present IND-secure FE schemes that are intuitively insecure. However, their schemes will correctly be

[1] In the standard formulation of IBE, the adversary has a key-derivation oracle via which it may obtain decryption keys for identities of its choice, but use of the oracle is restricted to identities not underlying challenge ciphertexts. An SOA-K results when there are many challenge ciphertexts and *this restriction is dropped*. This is exactly what happens in SS1-secure FE. The interesting thing is that in the context of semantic security for general FE it is not clear how to make appropriate restrictions to exclude the SOA-K. We will elaborate in a bit.

SS3-insecure.) Another reason is that our impossibility result for SS1 does not extend to SS3. (So in particular, SS3-secure IBE is not ruled out.)

In support of SS3 we show that it is equivalent to IND for "re-sampleable" functionalities. Unfortunately, re-sampleable functionalities does not seem to include common functionalities of interest such as IBE. Indeed, we have not been able to either prove or disprove the equivalence of SS3 with IND for PE functionalities. We suggest that IBE and PE schemes may be directly proven to meet SS3 and leave this as an interesting subject for future work. We note that recent independent and concurrent results of Agrawal *et al.* [3] (discussed further below) imply that there exists a functionality that cannot meet SS3. However, this situation is very different from that of SS1, for which we show that no non-trivial functionality can meet it. Indeed, we expect that there do exist IBE and PE schemes that meet SS3.

A CLOSER LOOK. Recall that in IBE, the adversary is given a key-derivation oracle, allowing it to obtain a secret key for any identity of its choice. This does *not* by itself constitute a SOA-K because the adversary is not allowed to call this oracle for the identities underlying challenge ciphertexts. In the SS1 definition, the adversary also gets a key-derivation oracle to obtain a secret key for any functionality index a of its choice. But there seems no simple or natural way to make a rule disallowing querying this oracle on "challenge" ciphertexts because there is no general way to "match" indexes with ciphertexts. Indeed, any key allows the adversary to learn, in principle, something from *all* challenge ciphertexts and we can hardly disallow all queries. Instead, SS1 allows unrestricted key-derivation queries and gives a compensating ability to the simulator. But now it incorporates SOA-K and is thus rarely achievable.

Roughly, the idea for SS2 is to run in parallel to the real game a "shadow" game where the inputs are independently generated as per the adversary-provided distributions. Key-derivation queries remain unrestricted. But at the end of the game, we check that the revealed keys don't "differentiate" the real and shadow games. We disallow adversaries who create such differentiation. In essence, this means that we require that the functionality take *predictable* values on the challenge messages when evaluated with the adversary's key derivation queries. One can compare this to the IND definition where the adversary is required to make key derivation queries that take the same value on the (known) challenge messages, so the adversary knows these values. Our definition may be written quite modularly relative to SS1, by adding appropriate boxed statements and checks in the games for the latter.

Our SS3 definition strengthens SS2 by dropping the restriction put by SS2 on key-derivation queries made by an adversary *before* seeing a challenge ciphertext. As such, we believe the SS3 definition is an essentially as-strong-as-possible security definition for FE subject to the constraint that it be achievable without any unnatural restrictions on the adversary or message space. To see why, note the definition of "unpredictable functionalities" used for our impossibility result in Section 4 and the fact that the latter crucially uses the adversary's ability to make "adaptive" key-derivation queries—i.e., depending on a challenge ciphertext. In essence, the SS3 definition demands that the functionality restricted to the adversary's adaptive key derivation queries be *predictable* wrt. the message space.

STANDARD-MODEL POSSIBILITY OF SS1. Returning to SS1, the negative results discussed above imply that we will need long keys, but we do not know that this is sufficient. There exists only one positive result, and this is in the PROM. Namely, BSW [14] provide a long-key, SS1-secure FE scheme for any functionality $\mathcal{F}$ where the space of functionality indexes on which $\mathcal{F}$ is non-trivial has polynomial size. We extend their result to the standard model. We do this by (again) exploiting the SOA-K connection. Namely we establish the same conclusion as BSW but assuming only the existence of a SOA-K-secure PKE scheme, which we know exists in the standard model because we are allowing keys to be long [15, 16].

SUMMARY OF CONTRIBUTIONS. We make a connection between selective-opening attacks (SOA-K) and FE by observing the implicit presence of the former in SS1, an observation that seeds all the further contributions of this paper, summarized as follows. (1) We show impossibility of SS1-secure FE in the standard model by exploiting techniques underlying negative results for SOA-K [5]. (2) We present the SS2 definition for sans-SOA-K FE and prove it equivalent to IND for all functionalties, thus obtaining a slew of possibility results for SS2 via known possibility results for IBE and PE. (3) We present the stronger

SS3 definition to function as a potential target for functionalites beyond PE and prove a possibility result for it. (4) We extend the only known positive result for a general functionality, namely one from BSW [14] for the case that the the set of indexes on which the functionality is non-trivial has polynomial size, from the PROM (Programmable Random Oracle Model) to the standard model, by using as starting point a SOA-K-secure PKE scheme with large keys, which exists in the standard model [15, 16].

Discussion and Related work. The observation underlying BSW's impossibility proof is that SS1-secure IBE must achieve something similar to NCE. O'Neill [29] had the same intuitive observation but did not take it to a result or proof. Our work can be viewed as taking this intuition further to say that SS1-secure IBE must be exactly SOA-K-secure IBE, and similarly for other functionalities.

The difference between NCE and SOA-K is subtle but important, and under-recognized by the community. For example, some works say (for the PKE case) that SOA-K security is impossible with short keys, citing [26]. But, in ruling out NCE, the latter does not rule out SOA-K-security because there are potentially non-NCE ways to achieve SOA-K-security. Our techniques, however, rule out SOA-K-secure PKE with short keys. Although we have known an impossibility result for NCE for a decade, one for SOA-K has only emerged now.

SOAs have so far mainly been considered in the public-key setting. The adversary gets a number of challenge ciphertexts, "opens" a subset of them, and aims to discover something about the messages underlying the rest. There are two kinds of SOAs. In a coin-revealing SOA (SOA-C) the ciphertexts are encrypted under a single public key and opening reveals the coins. Achieving security is challenging but has been done [6, 17, 21]. SOA-C-security was also considered and achieved for IBE [9]. SOA-C is not relevant to our present concerns. In a key-revealing SOA (SOA-K) for PKE, the ciphertexts are encrypted under different public keys and opening reveals the corresponding decryption keys. But SOA-K has not been defined or considered for IBE, let alone for FE. We claim SS1 is, implicitly, defining SOA-K secure FE.

O'Neill [29] considers non-adaptive adversaries (meaning ones that don't make any key-derivation queries after seeing the challenge ciphertexts). He provides a non-adaptive version of SS1 and shows it equivalent to a non-adaptive version of IND for preimage sampleable functionalities. Most PE functionalities considered in the literature have this property.

Independently, a recent breakthrough work of Gorbunov, Vaikuntanathan and Wee [19] shows that FE for *all polynomial-time functionalities* can be achieved, subject to the caveat that the adversary in the security experiment make up to q key-derivation queries, where q is fixed ahead of time. Interestingly, they make crucial use of the brute force scheme in their construction, instantiating it with a specific non-committing encryption scheme. (Since the construction only needs to be non-committing wrt. the receiver's secret key and not the sender's randomness, they use a simple construction from [16].) Our result on the brute force scheme generalizes theirs as well as BSW's. Another recent breakthrough work of Waters [35] contructs public-index PE for regular languages. We believe this recent progress on FE constructions for broader classes of functionalities underscores the importance of our efforts on the definitional front.

Also, interesting works on definitions of FE have emerged on eprint concurrently and independently of our work, namely Agrawal, Gorbunov, Vaikuntanathan and Wee (AGVW) [3] and Barbosa and Farshim (BF) [4]. AGVW present impossibility results for a wPRF based functionality for notions weaker than SS1. As discussed above, their results apply to SS3 and show that SS3 will not be met by all functionalities. They also propose a variant of SS1 that allows an *unbounded simulator* they call USIM. Interestingly, while they claim BSW's proof also applies to rule out USIM, this impossibility result inherits the same weaknesses we point out above for the basic impossibility result of BSW. Moreover, our improved impossibility result for SS1 assumes collision-resistant hash functions so does not to rule out USIM (because an unbounded simulator can break collision-resistance). Thus, to the best of our knowledge, whether USIM can be achieved is still open. On the other hand BF point to weaknesses in the BSW definition having to do with "set-up" security. Our definition of SS1 and its variants do not appear to inherit these weaknesses because the simulator is not allowed to choose the auxilliary input. (See the body of the paper for further explanation.) BF also propose variants of the definition that seem similar in spirit to our SS3 and show various possiblity and impossibility results about them. We have not yet had a chance to do a detailed comparison to our results.

2 Notation and conventions

If A is an algorithm then $y \leftarrow A(x_1, \ldots, x_n; r)$ means we run A on inputs $x_1, \ldots, x_n$ and coins r and denote the output by y. By $y \leftarrow_{\$} A(x_1, \ldots, x_n)$ we denote the operation 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 that have positive probability of being output by A on inputs $x_1, \ldots, x_n$. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for "polynomial time." The security parameter is denoted $\lambda \in \mathbb{N}$ and whenever λ is input to an algorithm it is understood that it is encoded in unary.

If s is a string then $|s|$ denotes its length, $s[i]$ denotes its ith bit, and $s[i \ldots j]$ denotes the substring consisting of its ith through jth bits. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes the number of its components, $\mathbf{x}[i]$ denotes its ith component, and $\mathbf{x}[i \ldots j]$ denotes the subvector consisting of its ith through jth components. We write $\mathsf{El}(\mathbf{x})$ to mean $\{\mathbf{x}[i] : 1 \leq i \leq |\mathbf{x}|\}$. If f is a function and $\mathbf{x}$ is a vector then $f(x_1, \ldots, x_{i-1}, \mathbf{x}, x_{i+1}, \ldots, x_n)$ denotes the vector whose i-th component is $f(x_1, \ldots, x_{i-1}, \mathbf{x}[i], x_{i+1}, \ldots, x_n)$ for $1 \leq i \leq |\mathbf{x}|$. A predicate is a function with boolean output.

Games. We use the language of code-based game-playing [8]. A game 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 λ and can then call game procedures. Its first oracle query must be Initialize(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. Let $A^{\mathrm{G}}(\lambda)$ denote the output of the adversary and $T(\mathrm{G}, A, \lambda)$ denote $\Pr\left[\,\mathrm{G}^A(\lambda) \text{ outputs } \mathsf{true}\,\right]$.

Standard primitives. In Appendix A we recall the standard notions of public-key encryption and collision-resistant hashing.

3 Functional Encryption and its Security

Functionalities and FE schemes. A *functionality* $\mathcal{F}$: $\mathbb{N} \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is a deterministic PT algorithm. The first input is the security parameter. The second input is called the index and the third input is called the payload. A *functional encryption (FE) scheme* is a tuple of algorithms $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$. The setup algorithm Setup on input λ returns a key-pair (pk, sk), the master public and secret keys. The key-derivation algorithm KDer on inputs sk, a returns a secret key dk for a. The encryption algorithm Enc on inputs pk, x returns a ciphertext c. The deterministic decryption algorithm Dec on inputs dk, c returns a string y. We say that an FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathcal{F}$-correct, or simply an $\mathcal{F}$-FE scheme, if $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x; r)) = \mathcal{F}(\lambda, a, x)$ for all λ, a, x, r satisfying $\mathcal{F}(\lambda, a, x) \neq \perp$, all $(pk, sk) \in [\mathsf{Setup}(\lambda)]$ and all $dk \in [\mathsf{KDer}(sk, a)]$. We stress that correctness makes no requirements when $\mathcal{F}(\lambda, a, x) = \perp$. (We do not mandate that $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x; r)) = \mathcal{F}(\lambda, a, x)$ in this case, but we do not disallow it either.)

Syntax and correctness in BSW. The range of a functionality in the formal definition of BSW [14] does not include $\perp$, and correctness asks that $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x; r)) = \mathcal{F}(\lambda, a, x)$ for all λ, a, x, r, all $(pk, sk) \in [\mathsf{Setup}(\lambda)]$ and all $dk \in [\mathsf{KDer}(sk, a)]$. However, specific functionalites given in BSW (such as that for IBE, $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{P},\mathsf{p}}$ in our notation) do return $\perp$. So it would appear that the formal syntax ought to be amended to add $\perp$ to the range of $\mathcal{F}$. Once this is done, the correctness condition of BSW must be revisited. If left unchanged, it would be asking that $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x; r)) = \mathcal{F}(\lambda, a, x)$ even when $\mathcal{F}(\lambda, a, x) = \perp$. This, however, would be incorrect. Attacks from [1] show that BB-style IBE schemes [10], including the BB IBE scheme [10] and Waters's IBE scheme [36], fail to meet this correctness condition relative to $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{P},\mathsf{p}}$.[2] It was not clear to us exactly what BSW intended but we expect they did intend for existing IBE schemes to meet the correctness condition, and accordingly we have relaxed it to only hold when $\mathcal{F}(\lambda, a, x) \neq \perp$.

Particular functionalities. The most important special case of FE in the literature is predicate encryption (PE). We say that $\mathcal{F}$ is a *predicate encryption functionality* if there is a predicate $\mathcal{P}$ such that $\mathcal{F}$ is $\mathcal{P}$-induced. This means that for all λ, all $a \neq \varepsilon$ and all (a', m) we have $\mathcal{F}(\lambda, a, (a', m)) = m$ if $\mathcal{P}(\lambda, a, a') = \mathsf{true}$ and $\perp$ otherwise. (We also require that $\mathcal{F}(\lambda, a, x)$ returns $\perp$ if x is not a pair. Note that no requirement is made on $\mathcal{F}(\lambda, \varepsilon, (a', m))$, so a single predicate could induce many different functionalities which vary in what is revealed under $a = \varepsilon$.) We call m the message. The IBE predicate $\mathcal{P}_{\mathrm{ibe}}$ is defined by $\mathcal{P}_{\mathrm{ibe}}(\lambda, a, a') = (a = a')$, and we say that $\mathcal{F}$ is an IBE functionality if it is $\mathcal{P}_{\mathrm{ibe}}$-induced. (So, again, there may be many different IBE functionalities.) Within the class of PE functionalities, we distinguish whether the index, the message, or both are to be kept private, with corresponding IBE functionalities as canonical examples:

- Public index, private message: We say that $\mathcal{F}$ is a (P, p)-PE functionality if $\mathcal{F}(\lambda, \varepsilon, (a', m)) = (a', |m|)$. Called PE with public index in the literature. The canonical example is the IBE functionality $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{P},\mathsf{p}}$ which sets $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{P},\mathsf{p}}(\lambda, \varepsilon, (a', m)) = (a', |m|)$, corresponding to IBE that hides the message but not necessarily the identity.
- Private index, private message: We say that $\mathcal{F}$ is a (p, p)-PE functionality if $\mathcal{F}(\lambda, \varepsilon, (a', m)) = |m|$. Called PE with private index in the literature. The canonical example is the IBE functionality $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{p},\mathsf{p}}$ which sets $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{p},\mathsf{p}}(\lambda, \varepsilon, (a', m)) = |m|$, corresponding to IBE that hides both the message and the identity (i.e. is anonymous).
- Private index, public message: We say that $\mathcal{F}$ is a (p, P)-PE functionality if $\mathcal{F}(\lambda, \varepsilon, (a', m)) = m$. Called predicate-only PE in the literature. The canonical example is the IBE functionality $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{p},\mathsf{P}}$ which sets $\mathcal{F}_{\mathrm{ibe}}^{\mathsf{p},\mathsf{P}}(\lambda, \varepsilon, (a', m)) = m$, corresponding to IBE that hides the identity but not necessarily the message. PEKS [12] is a (p, P)-PE functionality that additionally satisfies robustness [1].

We don't discuss (P, P)-PE because it reveals everything and is uninteresting.

SS1 definition. The following definition is adapted from [14]. Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{F}$-FE scheme. The definition uses games $\mathrm{RSS1}_{\mathsf{FE},\mathcal{F},Z,D,R}$ and $\mathrm{ISS1}_{\mathcal{F},Z,D,R}$ of Figure 1. We provide some intuition for these games below. We say that FE is *SS1-secure* if for every auxiliary input generator Z, every PT message sampler D, every PT relation R and every PT adversary A, there is a PT simulator S such that

$$\mathbf{Adv}^{\mathrm{ss1}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;=\; T(\mathrm{RSS1}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) - T(\mathrm{ISS1}_{\mathcal{F},Z,D,R}, S, \cdot)$$

is negligible. We note that the auxiliary input will be used in our impossibility result in Section 4 (where it contains a key for a collision-resistant hash function). Although we omit to do this for simplicity because it does not affect our results, it can also be given as an additional argument to a functionality itself. For example, in the case of the inner-product functionality introduced in [22] it can then contain the modulus N of unknown factorization.

Intuitive overview of the definition. To gain some intuition for the games, let us first look at the "real world" game with the adversary. It has access to two main oracles, an encryption oracle Enc and key-derivation oracle Kd. The former takes input α, which describes a message-space from which to sample, and outputs the encryption of a sampled message x. The latter takes as input a functionality index a and returns a corresponding secret key. Note that the game records the queries made to these oracles, in order, and provides this as input to the relation R. Now let us look at the "ideal world" game

[2] The difficulty is that correctness is required for all x, a and thus when $x = (a', m)$ with $a' \neq a$, it is required that $\mathsf{Dec}(dk, \mathsf{Enc}(pk, (a', m); r)) = \perp$ when $dk \in [\mathsf{KDer}(sk, a)]$. This is a form of robustness as defined in [1] and, as indicated there, often useful, but it is not a standard requirement for IBE schemes and most don't meet it.

```
proc Initialize($\lambda$):
    $(pk, sk) \leftarrow_\$ \mathsf{Setup}(\lambda)$
    $z \leftarrow_\$ Z(\lambda)$
    $i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$
    Return $(pk, z)$
proc Enc($\alpha$):
    $i \leftarrow i + 1$
    $\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
    $(St, \mathbf{x}[i]) \leftarrow_\$ D(St, \alpha)$
    $\mathbf{c}[i] \leftarrow_\$ \mathsf{Enc}(pk, \mathbf{x}[i])$
    Return $(\mathbf{c}[i], \mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i]))$
proc Kd($a$):
    $i \leftarrow i + 1$
    $\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
    $dk \leftarrow_\$ \mathsf{KDer}(sk, a)$
    Return $dk$
proc Finalize($w$):
    Return $R(\lambda, z, \mathbf{x}, \mathbf{q}, \mathbf{t}, St, w)$
```

```
proc Initialize($\lambda$):
    $i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$
    $z \leftarrow_\$ Z(\lambda)$
    Return $z$
proc Msg($\alpha$):
    $i \leftarrow i + 1$
    $\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
    $(St, \mathbf{x}[i]) \leftarrow_\$ D(St, \alpha)$
    Return $\mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i])$
proc Op($a$):
    $i \leftarrow i + 1$
    $\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
    Return $\varepsilon$
proc F($a, s$):
    If $a \in \mathsf{El}(\mathbf{a})$ and $1 \leq s \leq i$ then
        Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
    Else return $\perp$
proc Finalize($w$):
    Return $R(\lambda, z, \mathbf{x}, \mathbf{q}, \mathbf{t}, St, w)$
```

```
proc Initialize($\lambda$):
    $(pk, sk) \leftarrow_\$ \mathsf{Setup}(\lambda)$ ; $i, j \leftarrow 0$
    Return $pk$
proc LR($x_0, x_1$)
    $i \leftarrow i + 1$ ; $(\mathbf{x}_0[i], \mathbf{x}_1[i]) \leftarrow (x_0, x_1)$
    $\mathbf{c}[i] \leftarrow_\$ \mathsf{Enc}(pk, x_b)$
    Return $\mathbf{c}[i]$
proc Kd($a$):
    $j \leftarrow j + 1$ ; $\mathbf{a}[j] \leftarrow a$
    $dk \leftarrow_\$ \mathsf{KDer}(sk, a)$
    Return $dk$
proc Finalize($b'$):
    $\mathbf{a}[j + 1] \leftarrow \varepsilon$
    For $j' = 1, \ldots, j + 1$ do
        If $\mathcal{F}(\lambda, \mathbf{a}[j'], \mathbf{x}_0) \neq \mathcal{F}(\lambda, \mathbf{a}[j'], \mathbf{x}_1)$ then
            return false
    Return $(b' = 1)$
```

Figure 1: Left: "Real world" game $\mathrm{RSS1}_{\mathsf{FE},\mathcal{F},Z,D,R}$ for the SS1 definition. Middle: "Ideal world" game $\mathrm{ISS1}_{\mathcal{F},Z,D,R}$ for the SS1 definition. Right: game $\mathrm{IND}_{\mathsf{FE},\mathcal{F},b}$ for the IND definition.

with the simulator. The simulator has access to not two but three main oracles, a message sampling oracle Msg, an operation oracle Op, and a functionality oracle F. The first on input α, which again describes a message-space from which to sample, samples a message x but simply returns $\mathcal{F}(\lambda, \varepsilon, x)$. (We follow BSW [14] in using the value under index ε to describe what information about the message is publicly computable from a ciphertext.) The second records that an input functionality index a is "legal to be used" by the last oracle. The last oracle takes such an index a and a position s to return $\mathcal{F}(\lambda, a, x_s)$ where x_s is the sth sampled message by Msg. Intuitively, Op queries of the simulator correspond to Kd queries of the adversary, and indeed they are input to the relation R in the analogous manner. F queries can always be made "for free" by the simulator (they are not input to R).

Discussion of SS1. We have discussed SS1 as being the BSW [14] definition, which it is in spirit, but there are some differences in detail. BSW indicate that there are several dimensions of choice. They choose to formalize a non-adaptive version with blackbox simulators, saying that variants may be formalized similarly. We have chosen to formalize the variant with adaptive security and non-blackbox simulation. BSW give pk as input to the relation and we do not, but this choice does not matter. However, a novelty of our definition is the introduction of auxiliary inputs. Besides what is noted above in their regard, we note that our use of auxiliary inputs rescues our definitions from the weaknesses of the BSW definition pointed out in BF [4]. The issue raised by the latter arises with a functionality, such as inner-product PE [22], that depends on a parameter, such as a hard-to-factor modulus, that must be generated in a setup phase. Under BSW [14] and O'Neill [29], this would have to be done by the Setup algorithm of the FE scheme and the modulus would be part of pk. The problem raised by BF [4] then occurs because the simulator can pick pk. We, however, do not give pk as input to $\mathcal{F}$ and would capture setup-based functionalities by having the setup done by the auxiliary input generator algorithm Z, so that the modulus, in our example, would be part of the output z of this algorithm. However, the simulator is *not* allowed to pick z, and thus the attack of BF [4] would not appear to apply.

IND definition. Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{F}$-FE scheme. The definition uses game

```
proc Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$
$z \leftarrow\$ Z(\lambda)$
$i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$ ; $\boxed{St' \leftarrow \varepsilon}$
Return $(pk, z)$

proc Enc($\alpha$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
$(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
$\boxed{(St', \mathbf{x}'[i]) \leftarrow\$ D(St', \alpha)}$
$\mathbf{c}[i] \leftarrow\$ \mathsf{Enc}(pk, \mathbf{x}[i])$
Return $(\mathbf{c}[i], \mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i]))$

proc Kd($a$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
$dk \leftarrow\$ \mathsf{KDer}(sk, a)$
Return $dk$
```

```
proc Initialize($\lambda$):
$i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$ ; $\boxed{St' \leftarrow \varepsilon}$
$z \leftarrow\$ Z(\lambda)$
Return $z$

proc Msg($\alpha$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
$(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
$\boxed{(St', \mathbf{x}'[i]) \leftarrow\$ D(St', \alpha)}$
Return $\mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i])$

proc Op($a$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
Return $\varepsilon$

proc F($a, s$):
If $a \in \mathsf{El}(\mathbf{a})$ and $1 \le s \le i$ then
    Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
Else return $\perp$
```

```
proc Finalize($w$):
┌
$\mathbf{q}[j+1] \leftarrow \varepsilon$
For $j' = 1, \ldots, j+1$ do
    If $\mathbf{t}[j'] = \mathsf{kd}$ then
        If $\mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}) \neq \mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}')$ then
            $\mathsf{bad} \leftarrow \mathsf{true}$
┘
Return $R(\lambda, z, \mathbf{x}, \mathbf{q}, \mathbf{t}, St, w)$
```

Figure 2: Left: "Real world" game $\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}$ for the SS2 definition. Middle: "Ideal world" game $\mathrm{ISS2}_{\mathcal{F},Z,D,R}$ for the SS2 definition. Right: Finalize procedure, common to the two games.

$\mathrm{IND}_{\mathsf{FE},\mathcal{F},b}$ of Figure 1 for $b \in \{0,1\}$. We say that FE is *IND-secure* if for every adversary B,

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{F},B}(\cdot) \;=\; T(\mathrm{IND}_{\mathsf{FE},\mathcal{F},1}, B, \cdot) - T(\mathrm{IND}_{\mathsf{FE},\mathcal{F},0}, B, \cdot)$$

is negligible.

Robustness. Robustness, introduced for IBE and PKE in [1], seems important more generally for FE, particularly for predicate-only predicate encryption. To explain the issue, recall that correctness was mute in the case that $\mathcal{F}(\lambda, a, x) = \perp$, meaning in this case no requirement was put on the output of $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x))$ when $dk \in [\mathsf{KDer}(sk, a)]$. Roughly, robustness asks that $\mathsf{Dec}(dk, \mathsf{Enc}(pk, x)) = \perp$ in this case. In the case of PEKS this is important to avoid false positives in the testing.

The reason it is not quite so simple is that asking for the above condition globally and unconditionally seems to yield something that is hard to achieve. Instead, one can ask for various computational relaxations in the style of [1]. To exemplify, here is one that is very strong but attractive due to its simplicity: procedure Initialize(λ) of game $\mathrm{ROB}_{\mathsf{FE},\mathcal{F}}$ lets $(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$ and returns *both* keys, meaning the adversary gets sk. Finalize(a, x) returns $((\mathcal{F}(\lambda, a, x) = \perp) \wedge (\mathsf{Dec}(\mathsf{KDer}(sk, a), \mathsf{Enc}(pk, x)) \neq \perp)$.

4 Impossibility Results

We show that the SS1 notion is *impossible* to achieve in the *standard model*, so long as the functionality is reasonably likely to take more than one possible value on a challenge message. This result only assumes the existence of a collision-resistant hash function.

Following [14] we also consider a relaxation of the SS1 notion where vectors $\mathbf{a}, \boldsymbol{\alpha}$ are replaced by *unordered sets*, thus giving the simulator more power (since it can make its queries in a different order than the adversary). We obtain a similar but more restrictive impossibility result in this case. Here we present the ordered case. The unordered one is in Appendix B.

Unpredictable functionalities. In the ordered case our result applies to any *unpredictable functionality.* Let $\mathcal{F}$ be a functionality, $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functionality indices (strings), and $\mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of payload distributions. We say that $\mathcal{F}$ is $p(\cdot)$*-unpredictable wrt.* $\mathcal{A}, \mathcal{X}$ if for

```
proc Initialize($\lambda$):
$(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$
$z \leftarrow\$ Z(\lambda)$
$i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$ ; $St' \leftarrow \varepsilon$
Return $(pk, z)$

proc Enc($\alpha$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
$(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
$(St', \mathbf{x}'[i]) \leftarrow\$ D(St', \alpha)$
$\mathbf{c}[i] \leftarrow\$ \mathsf{Enc}(pk, \mathbf{x}[i])$
Return $(\mathbf{c}[i], \mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i]))$

proc Kd($a$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
$dk \leftarrow\$ \mathsf{KDer}(sk, a)$
Return $dk$
```

```
proc Initialize($\lambda$):
$i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$ ; $St' \leftarrow \varepsilon$
$z \leftarrow\$ Z(\lambda)$
Return $z$

proc Msg($\alpha$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
$(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
$(St', \mathbf{x}'[i]) \leftarrow\$ D(St', \alpha)$
Return $\mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i])$

proc Op($a$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
Return $\varepsilon$

proc F($a, s$):
If $a \in \mathsf{El}(\mathbf{a})$ and $1 \le s \le i$ then
    Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
Else return $\perp$
```

```
proc Finalize($w$):
$\mathbf{q}[i+1] \leftarrow \varepsilon$ ; $\mathbf{t}[i+1] \leftarrow \mathsf{kd}$
For $i' = 1, \ldots, i+1$ and $j' = i'+1, \ldots, i+1$ do
    If $\mathbf{t}[i'] = \mathsf{enc} \wedge \mathbf{t}[j'] = \mathsf{kd}$ then
        If $\mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}[i']) \ne \mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}'[i'])$ then
            $\mathsf{bad} \leftarrow \mathsf{true}$
Return $R(\lambda, z, \mathbf{x}, \mathbf{q}, \mathbf{t}, St, w)$
```

Figure 3: Left: "Real world" game $\mathrm{RSS3}_{\mathsf{FE},\mathcal{F},Z,D,R}$ for the SS3 definition. Middle: "Ideal world" game $\mathrm{ISS3}_{\mathcal{F},Z,D,R}$ for the SS3 definition. Right: FINALIZE procedure, common to the two games.

all $\lambda \in \mathbb{N}$ and all $y \in \{0,1\}^* \cup \{\perp\}$, $\Pr[x \leftarrow\$ X_\lambda \,:\, y = \mathcal{F}(\lambda, a_\lambda, x)] \le 1 - 1/p(\lambda)$.

For example, the functionality $\mathcal{F}^{\mathsf{P},\mathsf{p}}_{\text{bit-ibe}}$ for a one-bit IBE scheme, which parses x as (a', b), and returns b if $a = a'$ and $\perp$ otherwise, is a 2-unpredictable function wrt. $\mathcal{A}, \mathcal{X}$ where, for all $\lambda \in \mathbb{N}$, we let a_λ be a fixed but arbitrary identity and $\mathcal{X}_\lambda$ return (a_λ, d) where the message $d \in \{0,1\}$ is random. As another example, the functionality $\mathcal{F}^{\mathsf{p},\mathsf{P}}_{\text{peks}}$for a PEKS scheme, which returns 1 if $a = x$ and $\perp$ otherwise, is a 2-unpredictable function wrt. $\mathcal{A}, \mathcal{X}$ where for all $\lambda \in \mathbb{N}$, we again let a_λ be fixed but arbitrary keyword and $\mathcal{X}_\lambda$ return a random keyword $x \in \{a_\lambda, a'_\lambda\}$ for some also fixed but arbitrary $a'_\lambda \ne a_\lambda$. Indeed, unpredictability with respect to *some* family of input distributions and functionality indices is a minimal requirement for a functionality to be interesting; otherwise, it is trivial to build an FE scheme for it because anyone can decrypt correctly without even using the ciphertext. In this sense, our result below rules out an SS1-secure FE scheme for any non-trivial functionality.

SECRET-KEY LENGTH. we say that an FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ has *secret-key length* $\ell(\cdot)$ if $|dk| \le \ell(\lambda) = \mathcal{F}(\lambda, a, x)$ for all λ, a, x, r, all $(pk, sk) \in [\mathsf{Setup}(\lambda)]$, and all $dk \in [\mathsf{KDer}(sk, a)]$. Note that every FE scheme must have some polynomial $\ell(\cdot)$ secret-key length in order to be efficient.

Theorem 4.1 *Let $p(\cdot) > 1$ be a polynomial. Suppose $\mathcal{F}$ is a $p(\cdot)$-unpredictable functionality wrt. $\mathcal{A} = \{a_\lambda\}_{\lambda \in \mathbb{N}}, \mathcal{X} = \{X_\lambda\}_{\lambda \in \mathbb{N}},$. Furthermore, suppose that for every $\lambda \in N$, $\mathcal{F}(\lambda, \varepsilon, x)$ is the same for all $x \in [X_\lambda]$. Let $\mathcal{H} = (K, H)$ be a collision-resistant hash function. Then there does not exist an SS1-secure $\mathcal{F}$-FE scheme. More precisely, suppose FE is a $\mathcal{F}$-FE scheme with secret-key length $\ell(\cdot)$. Then for any function $\mu(\cdot)$ there exists a PT auxiliary input generator Z, message sampler D, PT adversary A, PT relation R, and CR-adversary C such that for every simulator S*

$$\mathbf{Adv}^{\mathrm{ss}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;\ge\; 1 - \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\cdot) + 1/\mu(\cdot)}\;.$$

Adversary A makes $p(\cdot)(\ell(\cdot) + \log \mu(\cdot))$ encryption queries and two key-derivation queries.

To compare, BSW [14] ruled out SS1-secure IBE against adversaries with access to a non-programmable random oracle, so our result improves theirs in two respects: to applies to any non-trivial functionality and standard-model adversaries. It also reveals a trade-off between secret-key length and the total number of bits encrypted. Namely, when the difference is even one bit (i.e., the total number of bits encrypted is one

```
Alg $A(pk, hk)$:
For $i = 1, \ldots, n(\lambda)$ do:
    $\mathbf{c}[i] \leftarrow\$ \textsc{Enc}(\lambda)$
$h \leftarrow H(hk, pk\|\mathbf{c})$
$sk_h \leftarrow\$ \textsc{Kd}(h)$
$sk_a \leftarrow\$ \textsc{Kd}(a_\lambda)$
$w \leftarrow (pk, \mathbf{c}, h, f, sk_h, sk_a)$
Return $w$
```

```
Alg $R(\lambda, \mathbf{x}, \mathbf{a}, \boldsymbol{\alpha}, St, w)$:
$(pk, \mathbf{c}, h, f, sk_h, sk_a) \leftarrow w$
If $h \neq H(hk, pk\|\mathbf{c})$ then return false
If $|sk_h| \neq \ell$ or $|sk_a| \neq \ell$ then return false
If $|\boldsymbol{\alpha}| \neq n \ \vee\ \boldsymbol{\alpha} \neq (\lambda, \ldots, \lambda)$ then return false
If $|\mathbf{a}| \neq 2 \ \vee\ \mathbf{a}[1] \neq h \ \vee\ \mathbf{a}[2] \neq a_\lambda$ then return false
If $\mathsf{Dec}(sk_a, \mathbf{c}) \neq \mathcal{F}(\lambda, a_\lambda, \mathbf{x})$ then return false
Return true
```

Figure 4: Algorithms A and R for proof of Theorem 4.1.

```
Alg $S_1(\lambda)$:
$i \leftarrow 0$
$z \leftarrow\$ \textsc{Initialize}(\lambda)$
Run $S(z)$:
On message-query $\alpha$ do:
    $i \leftarrow i + 1$ ; Return $\overline{\varepsilon}$
On op-query $a$ do:
    Halt computation of $S_1$ with state $St$
$\overline{St} \leftarrow St\|i$ ; Return $\overline{St}$
```

```
Alg $S_2(\overline{St})$:
$St\|i \leftarrow \overline{St}$ ; $A \leftarrow \emptyset$
For $i = 1, \ldots. n(\lambda)$ do $\mathbf{x}[i] \leftarrow\$ X_\lambda$
Run $S$ at state $St$:
On message-query $\alpha$ do:
    $i \leftarrow i + 1$ ; Return $\overline{\varepsilon}$
On op-query $a$ do:
    $A \leftarrow A \cup \{a\}$ ; Return $\varepsilon$
On $F$-query $(a, s)$ do:
    If $1 \leq s \leq i$ and $a \in A$ do:
    Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
    Else return $\perp$
Let $w$ be the output of $S$ ; Return $w$
```

```
Adversary $C(\lambda)$:
$hk \leftarrow\$ \textsc{Initialize}(\lambda)$
Run $S_1$ on $\lambda$,
    replying to \textsc{Initialize} with $hk$
Let $\overline{St}$ be the output of $S_1$
$w^1, w^2 \leftarrow\$ S_2(\overline{St})$
$(pk^1, \mathbf{c}^1, h^1, f^1, sk_h^1, sk_a^1) \leftarrow w^1$
$(pk^2, \mathbf{c}^2, h^2, f^2, sk_h^2, sk_a^2) \leftarrow w^2$
Return $(pk^1\|\mathbf{c}^1, pk^2\|\mathbf{c}^2)$
```

Figure 5: Algorithms S_1, S_2 and C for proof of Theorem 4.1.

more than the secret-key length) our adversary's advantage is non-negligible. We also note that, while for technical reasons we require $\mathcal{F}(\lambda, \varepsilon, x)$ to take the same value on every possible challenge payload x, this is not a major restriction in practice since typically $\mathcal{F}(\lambda, \varepsilon, x) = |x|$; then we are just requiring as usual that possible challenge messages have the same length.

The proof combines and extends ideas of [14] and [5]. As in [5] will make use of a version of the Reset Lemma of [7].

Lemma 4.2 *Let P_1, P_2 be algorithms, the second with boolean output. The single-execution acceptance probability $\mathbf{AP}_1(P_1, P_2, \lambda)$ is defined as the probability that $d =$ true in the single execution experiment $\overline{St} \leftarrow\$ P_1(\lambda)$; $d \leftarrow\$ P_2(\overline{St})$. The double-execution acceptance probability $\mathbf{AP}_2(P_1, P_2, \lambda)$ is defined as the probability that $d_1 = d_2 =$ true in the double execution experiment $\overline{St} \leftarrow\$ P_1(\lambda)$; $d_0, d_1 \leftarrow\$ P_2(\overline{St})$. Then $\mathbf{AP}_1(P_1, P_2, \lambda) \ \leq\ \sqrt{\mathbf{AP}_2(P_1, P_2, \lambda)}$ for all $\lambda \in \mathbb{N}$.* ▌

Proof of Theorem 4.1: For $\lambda \in \mathbb{N}$ denote by $\overline{\varepsilon}_\lambda$ the value such that $\mathcal{F}(\lambda, \varepsilon, x) = \overline{\varepsilon}_\lambda$ for all $x \in [X_\lambda]$. Let $n(\cdot) = p(\cdot)(\ell(\cdot) + \log \mu(\cdot))$. Define Z on input λ to return $hk \leftarrow\$ K(\lambda)$. Define message sampler D on inputs St, α to (ignore St and) return $x \leftarrow\$ X(\alpha)$. Define adversary A and relation R as in Figure 4.

By construction $\Pr\left[\mathrm{RSS}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right] = 1$. Let S be any simulator. Wlog assume that it makes no F-query preceding its first Op-query (the response to such a query would be $\perp$). Furthermore, parsing the output of S as $(pk, \mathbf{c}, h, f, sk_h, sk_a, hk) \leftarrow w$, we assume it holds that $h = H(hk, pk\|\mathbf{c})$, $|sk_h| = |sk_a| = \ell$, $|\boldsymbol{\alpha}| = n$, $\boldsymbol{\alpha} = (\lambda, \ldots, \lambda)$, $|\mathbf{a}| = 2$, $\mathbf{a}[1] = h$, and $\mathbf{a}[2] = a_\lambda$, since otherwise the relation R returns false. Towards applying Lemma 4.2, we write execution of S in Game ISS1 as a composition of two algorithms S_1, S_2 as in Figure 5.

Namely, S_1 runs S up to the point that it makes its first Op query, and S_2 runs S following this Op query. It also samples $\mathbf{x}$ all at once instead of defining its individual components when responding to each message query, but this is equivalent and makes the analysis more transparent. Below we justify the following sequence of inequalities, for C, Y defined below:

$$\begin{aligned}
\Pr\left[\,\mathrm{ISS1}^{S}_{\mathcal{F},Z,D,R}(\cdot)\,\right] &= \mathbf{AP}_1(S_1,S_2,\lambda) \;\le\; \sqrt{\mathbf{AP}_2(S_1,S_2,\lambda)} \;\le\; \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\lambda) + \Pr\left[\,\mathcal{F}(\lambda,a_\lambda,\mathbf{x}) \in Y\,\right]} \\
&\le \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\lambda) + 2^{\ell}(1-1/p)^n} < \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\lambda) + 2^{\ell}2^{-(\ell+\log\mu(\lambda))}} \\
&= \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\lambda) + 1/\mu(\lambda)}\;.
\end{aligned}$$

Above, the first inequality is by construction and the second is by Lemma 4.2. For the third, consider the CR-adversary C against $\mathcal{H}$ also defined in Figure 5. Note that C simply mimics the double execution experiment with S_1, S_2. Let us also denote the two outputs of S_2 in that experiment as $w^1 = (pk^1, \mathbf{c}^1, h^1, f^1, sk^1_h, sk^1_a)$ and $w^2 = (pk^2, \mathbf{c}^2, h^2, f^2, sk^2_h, sk^2_a)$. Now, unless C finds a collision we have $\mathbf{c}^1 = \mathbf{c}^2$, denote by $\mathbf{c}^*$ this common value. Then define the set $Y := \{y \in \{0,1\}^* \;:\; \exists s \in \{0,1\}^{\ell} \text{ s.t. } \mathsf{Dec}(s,\mathbf{c}^*) = y\}$. If it is not the case that $\mathcal{F}(\lambda,a_\lambda,\mathbf{x}) \in Y$ when $\mathbf{x}$ is sampled by S_2 then the relation R must reject, as no sk^2_a will satisfy $\mathsf{Dec}(sk^2_a,\mathbf{c}) = \mathcal{F}(\lambda,a_\lambda,\mathbf{x})$ (recall we assume $|sk^2_a| = \ell$ here as otherwise R rejects), justifying the third inequality above. The fourth equality above uses the unpredictability of $\mathcal{F}$ and a union bound. For the fifth we substitute $n(\lambda) = p(\lambda)(\ell(\lambda) + \log(\mu(\lambda)))$ and use the inequality $(1-1/x)^x \le 1/e < 1/2$ for any real number $x \ge 1$ (here e is Euler's constant). ▌

5 Equivalence for Restricted Definitions of Semantic Security

Motivated by the impossibility results of Section 4, take up the direction of providing variants of the SS definition that are *achievable.*

5.1 SS2 and its Equivalence to IND for All Functionalities

We start by providing our SS2 definition that we show is *equivalent* to IND, thus demonstrating that the IND definition targeted in the literature is indeed equivalent to some form of semantic security.

SS2 DEFINITION. Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{F}$-FE scheme. The definition uses games $\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}$ and $\mathrm{ISS2}_{\mathcal{F},Z,D,R}$ of Figure 2, with the boxed code indicating the differences from the corresponding games in Figure 1, i.e., removing the boxed code recovers the SS1 games. Let D be a message sampler and A an adversary. We say that (D, A) is *SS2-valid with failure probability* $\nu(\cdot)$ if $\Pr\left[\,\mathrm{RSS2}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}\,\right] \le \nu(\cdot)$. When $\nu(\cdot)$ is negligible we say (D, A) is *SS2-valid.* (Note that this probability does not depend on the relation R, so it can be anything.) We say that FE is *SS2-secure* if for every PT auxiliary input generator Z, every message sampler D and PT adversary A such that (D, A) is SS2-valid, and every PT relation R, there is a PT simulator S such that

$$\mathbf{Adv}^{\mathrm{ss2}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;=\; T(\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) - T(\mathrm{ISS2}_{\mathcal{F},Z,D,R}, S, \cdot)$$

is negligible. *Intuitively, the definition mandates that an adversary only make key-derivation queries for functions under which any possible challenge message takes the same value.* For example, for IBE this corresponds to mandating that any key-derivation query made by the adversary never makes a key-derivation query for a key that decrypts a challenge ciphertext (or decrypts it to a predictable value), which is a natural restriction. The following theorem follows from the claims below.

Theorem 5.1 *Let $\mathcal{F}$ be a functionality and let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ *be an $\mathcal{F}$-FE scheme. Then* FE *is SS2-secure if and only if it is IND-secure.*

Claim 5.2 (SS2 ⇒ IND) Let B be an IND-adversary. Then there is a message sampler D, relation R, and adversary A such that for every simulator S

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{F},B}(\cdot) \;\le\; 2 \cdot \mathbf{Adv}^{\mathrm{ss2}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot)\;. \tag{1}$$

```
Alg $D(St, \alpha)$:
If $St = \varepsilon$ then $St \leftarrow_\$ \{0,1\}$
$x_0 \| x_1 \leftarrow \alpha$
Return $(St, x_{St})$
```

```
Alg $A(pk)$:
Run $B(pk)$:
    On left-or-right query $(x_0, x_1)$:
        $c \leftarrow_\$ \textsc{Enc}(x_0 \| x_1)$
        Return $c$
    On query key-derivation query $a$:
        $dk \leftarrow_\$ \textsc{Kd}(a)$
        Return $dk$
Let $b'$ be the output of $A$
Return $b'$
```

```
Alg $R(1^k, \mathbf{x}, \mathbf{a}, \boldsymbol{\alpha}, St, w)$:
$\mathbf{x}_0 \| \mathbf{x}_1 \leftarrow \boldsymbol{\alpha}$
$\mathbf{a}[|\mathbf{a}|+1] \leftarrow \varepsilon$
For $j = 1, \ldots, |\mathbf{a}|+1$ do
    If $\mathcal{F}(1^k, \mathbf{a}[j], \mathbf{x}_0) \neq \mathcal{F}(1^k, \mathbf{a}[j], \mathbf{x}_1)$
        then return $\mathsf{false}$
Return $(\mathbf{x} = \mathbf{x}_w)$
```

Figure 6: Algorithms for proof of Claim 5.2.

Furthermore, (D, A) is SS2-valid and the running-time of D, R, A is that of B.

Proof: Define the message sampler D, adversary A, and relation R as in Figure 6. Then by construction $T(\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) = T(\mathrm{IND}_{\mathsf{FE},\mathcal{F}}, B, \cdot)$. Moreover, we claim that for every simulator S we have $T(\mathrm{ISS2}_{\mathcal{F},Z,D,R,S}, \cdot) \leq 1/2$. This is because we can assume wlog that S's queries are such that $\mathcal{F}(1^k, a, \mathbf{x}_0) = \mathcal{F}(1^k, a, \mathbf{x}_1)$ for all $\mathsf{El}(a) \in \mathbf{a} \cup \varepsilon$ (since otherwise R returns false), so S gets no information about the bit b. Subtracting, we get

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ss2}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) &= T(\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) - T(\mathrm{ISS2}_{\mathcal{F},Z,D,R}, S, \cdot) \geq T(\mathrm{IND}_{\mathsf{FE},\mathcal{F}}, B, \cdot) - 1/2 \\ &= 1/2 \cdot \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{F},B}(\cdot)\end{aligned}$$

which implies Equation (1). To complete the proof we note that (D, A) is SS2-valid because we may assume wlog that B's queries are such that $\mathcal{F}(1^k, a, \mathbf{x}_0) = \mathcal{F}(1^k, a, \mathbf{x}_1)$ for all $a \in \mathsf{El}(\mathbf{a}) \cup \varepsilon$ (otherwise its advantage can only go down). ▌

Claim 5.3 (IND $\Rightarrow$ SS2) Let D be a message sampler, A be an SS2-adversary, and R be a relation such that (D, A) is SS2-valid with failure probability $\nu(\cdot)$. Then there is a simulator S and an IND-adversary B such that

$$\mathbf{Adv}^{\mathrm{ss2}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \leq \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{F},B}(\cdot) + 2\nu(\cdot)\ .$$

The running-time of S is that of D, R, A and the running-time of B is at most twice that of R, A plus twice that of D.

Proof: Define S and B as in Figure 7. Without affecting the output of the game, we may have the Finalize procedure of Game $\mathrm{IND}^{B}_{\mathsf{FE},\mathcal{F},0}(\cdot)$ set a flag bad when the "return false" statement is executed. Then, viewing Games $\mathrm{IND}^{B}_{\mathsf{FE},\mathcal{F},0}(\cdot)$ and $\mathrm{RSS2}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)$ as executed over a common finite space of coins, we have that in the language of [8] they are identical-until-bad. Therefore, by the Fundamental Lemma of [8]

$$T(\mathrm{RSS2}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) \leq T(\mathrm{IND}_{\mathsf{FE},\mathcal{F},1}, B, \cdot) + \Pr\left[\mathrm{RSS2}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}\right].$$

By an analogous argument

$$T(\mathrm{ISS2}_{\mathcal{F},Z,D,R}, S, \cdot) \geq T(\mathrm{IND}_{\mathsf{FE},\mathcal{F},0}, B, \cdot) - \Pr\left[\mathrm{ISS2}^{S}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}\right].$$

Subtracting yields Equation (5.3). ▌

It is worth pointing out in the proof of the second claim above that the constructed simulator S needs not ever query its oracles. This is because we are guaranteed that, since (D, A) is SS2-valid, the "dummy" messages sampled independently from the challenge ones have the same value under any function queried by A to its key-derivation oracle.

```
Alg $S(1^k)$:
$St \leftarrow \varepsilon$
$(pk, sk) \leftarrow_{\$} \mathsf{Setup}(1^k)$
Run $A(pk)$:
    On encryption query $\alpha$:
        $y \leftarrow_{\$} \mathrm{MSG}(\alpha)$
        $(St, x) \leftarrow_{\$} D(St, \alpha)$
        $c \leftarrow_{\$} \mathsf{Enc}(pk, x)$
        Return $(c, y)$
    On key-derivation query $a$:
        $dk \leftarrow_{\$} \mathsf{KDer}(sk, a)$
        Return $dk$
Let $w$ be the output of $A$
Return $w$
```

```
Alg $B(pk)$:
$i, j \leftarrow 0$
$St_0, St_1 \leftarrow \varepsilon$
Run $A(pk)$:
    On encryption query $\alpha$:
        $i \leftarrow i + 1$ ; $\boldsymbol{\alpha}[i] \leftarrow \alpha$
        $(St_0, \mathbf{x}_0[i]) \leftarrow_{\$} D(St_0, \alpha)$
        $(St_1, \mathbf{x}_1[i]) \leftarrow_{\$} D(St_1, \alpha)$
        Return $\mathrm{LR}(\mathbf{x}_0[i], \mathbf{x}_1[i])$
    On key-derivation query $a$:
        $j \leftarrow j + 1$ ; $\mathbf{a}[j] \leftarrow a$
        $dk \leftarrow_{\$} \mathrm{KD}(a)$
        Return $dk$
Let $w$ be the output of $A$
Return $R(1^k, \mathbf{x}_1, \mathbf{a}, \boldsymbol{\alpha}, St_1, w)$
```

Figure 7: Algorithms for proof of Claim 5.3.

5.2 SS3 and its Equivalence to IND for Resampleable Functionalities

The SS2 definition is not as strong as one would like because for some functionalities IND (which we showed equivalent to SS2) is a "bad" definition. To address this we now introduce the SS3 definition, which strengthens SS2 by dropping the restriction put by SS2 on key-derivation queries made by an adversary *before* seeing a challenge ciphertext. Indeed, we believe the SS3 definition is an essentially as-strong-as-possible security definition for FE subject to the constraint that it be achievable without any unnatural restrictions on the adversary or message space. To see why, note the definition of "unpredictable functionalities" used for our impossibility result in Section 4 and the fact that the latter crucially uses the adversary's ability to make "adaptive" key-derivation queries—i.e., depending on a challenge ciphertext. In some sense, the SS3 definition demands that the functionality restricted to the adversary's adaptive key derivation queries be *predictable* wrt. the message space.

SS3 DEFINITION. Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{F}$-FE scheme. The definition uses games $\mathrm{RSS3}_{\mathsf{FE},\mathcal{F},Z,D,R}$ and $\mathrm{ISS3}_{\mathcal{F},Z,D,R}$ of Figure 3, with the boxed code indicating the differences from the corresponding games in Figure 2, i.e., the games differ from SS2 only in the Finalize procedure. Let D be a message sampler and A an adversary. We say that (D, A) is *SS3-valid with failure probability* $\nu(\cdot)$ if $\Pr\left[\mathrm{RSS3}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}\right] \leq \nu(\cdot)$. When $\nu(\cdot)$ is negligible we say (D, A) is *SS3-valid.* (Note that this probability does not depend on the relation R, so it can be arbitrary.) We say that FE is *SS3-secure* if for every PT auxiliary input generator Z, every message sampler D and PT adversary A such that (D, A) is valid, and every PT relation R, there is a PT simulator S such that

$$\mathbf{Adv}^{\mathrm{ss3}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;=\; T(\mathrm{RSS3}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) - T(\mathrm{ISS3}_{\mathcal{F},Z,D,R}, S, \cdot)$$

is negligible.

RESAMPLEABILITY. Let $\mathcal{F}$ be a functionality, $\overline{D}, D$ be algorithms. For an adversary B we let

$$\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F},B}(\cdot) \;=\; T(\mathrm{Rsmp}_{\overline{D},D,\mathcal{F},1}, B, \cdot) - T(\mathrm{Rsmp}_{\overline{D},D,\mathcal{F},0}, B, \cdot)$$

where the game is in Figure 10. Furthermore, let $\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F}}(\cdot) \;=\; \max_B\{\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F},B}(\cdot)\}$ where the maximum is over all PT B making one challenge query.

We say that an algorithm $\overline{D}$ is a $\mu(\cdot)$-*accurate resampler* for $\mathcal{F}$ relative to algorithm D if $\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F}}(\cdot) \leq \mu(\cdot)$. We say that $\mathcal{F}$ is $\mu(\cdot)$-*accurately reampleable* if for all PPT D there exists a PPT $\overline{D}$ such that $\overline{D}$ is a $\mu(\cdot)$-accurate resampler for $\mathcal{F}$ relative to D. When $\mu(\cdot)$ is negligible we say that $\mathcal{F}$ is *accurately resampleable.*

THE EQUIVALENCE. The following says that the SS3 notion is equivalent to IND for any accurately resampleable functionality $\mathcal{F}$.

Theorem 5.4 *Let $\mathcal{F}$ be an accurately resampleable functionality and let* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ *be an $\mathcal{F}$-FE scheme. Then* FE *is SS3-secure if and only if it is IND-secure.*

Since SS3 is clearly stronger than SS2, it is immediate from Claim 5.2 that SS3 implies IND. Below we show the converse, which proves the theorem.

Claim 5.5 (IND $\Rightarrow$ SS3) Let D be a message sampler, A be an SS2-adversary making at most q_e encryption queries, and R be a relation such that (D, A) is SS3-valid with failure probability $\nu(\cdot)$. Suppose $\mathcal{F}$ is $\mu(\cdot)$-accurately resampleable, and let $\overline{D}$ denote the corresponding $\mu(\cdot)$-accurate resampler. Then there is a simulator S and an IND-adversary B such that

$$\mathbf{Adv}^{\mathrm{ss3}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;\leq\; \mathbf{Adv}^{\mathrm{ind}}_{\mathsf{FE},\mathcal{F},B}(\cdot) + 2\nu(\cdot) + 2q_e\mu(\cdot)\;.$$

The running-time of S is that of D, R, A and the running-time of B is at most twice that of R, A plus the time for q_e executions of $\overline{D}$.

Proof:

Define S and B as in Figure 9. Furthermore, define hybrid games $\text{RSS3-H1}_{\mathsf{FE},\mathcal{F},Z,D,R}$ and $\text{RSS3-H2}_{\mathsf{FE},\mathcal{F},Z,D,R}$ as in Figure 8. Below we justify the following sequence of inequalities:

$$\begin{aligned}
T(\text{RSS3}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) \quad &\leq \quad T(\text{RSS3-H1}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) + \Pr\left[\,\text{RSS3}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}_1\,\right]\\
&\leq \quad T(\text{RSS3-H2}_{\mathsf{FE},\mathcal{F},Z,D,R}, A, \cdot) + \Pr\left[\,\text{RSS3-H1}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}_2\,\right]\\
&\qquad + \Pr\left[\,\text{RSS3}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}_1\,\right]\\
&= \quad T(\text{IND}_{\mathsf{FE},\mathcal{F},1}, B, \cdot) + \Pr\left[\,\text{RSS3-H1}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}\,\right]\\
&\qquad + \Pr\left[\,\text{RSS3}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}_1\,\right]\\
&\leq \quad T(\text{IND}_{\mathsf{FE},\mathcal{F},1}, B, \cdot) + \Pr\left[\,\text{RSS3-H1}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot) \text{ sets } \mathsf{bad}_2\,\right] + \nu(\cdot)\\
&\leq \quad T(\text{IND}_{\mathsf{FE},\mathcal{F},1}, B, \cdot) + q_e(\cdot)\mu(\cdot) + \nu(\cdot)\;.
\end{aligned}$$

Above, the first two inequalities are by the Fundamental Lemma of [8] (by a slight abuse of notation we identify bad in $\text{RSS3}_{\mathsf{FE},\mathcal{F},Z,D,R}$ with bad_1 in $\text{RSS3-H1}_{\mathsf{FE},\mathcal{F},Z,D,R}$, and third equality is by construction. The fourth uses the assumption that (D, A) is SS3-valid with failure probability $\nu(\cdot)$.

Finally, the last inequality follows by considering a run of Games $\text{Rsmp}^{B^*}_{\overline{D},D,\mathcal{F},b}(\lambda)$ for $b \in \{0, 1\}$ and B^* given in Figure 9, and of Game $\text{RSS3-H1}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\lambda)$ over some fixed coin sequence drawn from a common finite set of coins, but not including the coins used to draw i^* in $\text{RSS3-H1}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\lambda)$. Suppose bad is set by the latter for the *first* time (i.e., changed from false to true) on the q-th query to $\text{E}{\small NC}$ made by A when executed by $\text{RSS3-H1}_{\mathsf{FE},\mathcal{F},Z,D,R}$ using these coins. Then with probability $1/q_e$ in the execution of $\text{Rsmp}^{B^*}_{\overline{D},D,\mathcal{F},b}(\lambda)$ it will be the case that $i^* = q$. In this case $\text{Rsmp}^{B^*}_{\overline{D},D,\mathcal{F},b}(\cdot)$ outputs 1 just when $b = 1$. Using the assumption that $\mathcal{F}$ is $\mu(\cdot)$-accurately resampleable and re-arranging yields the last inequality above.

A symmetric sequence of inequalities to the above yields

$$T(\text{ISS2}_{\mathcal{F},Z,D,R}, S, \cdot) \quad \geq \quad T(\text{IND}_{\mathsf{FE},\mathcal{F},0}, B, \cdot) - \nu(\cdot) - q_e\mu(\cdot)\;.$$

Re-arranging and subtracting yields Equation (5.5). ▌

```
PROC INITIALIZE($\lambda$):
$(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$
$z \leftarrow\$ Z(\lambda)$
$i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$ ; Return $(pk, z)$

PROC ENC($\alpha$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow \alpha$ ; $\mathbf{t}[i] \leftarrow \mathsf{enc}$
$(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
For $j' = 1$ to $j$ do:
    $\mathbf{f}[j'] \leftarrow \mathrm{F}(\mathbf{a}[j'], i)$
    $(St, \mathbf{x}[i]) \leftarrow\$ \overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
If $\mathsf{Test}(\lambda, \mathbf{x}[i], \mathbf{a}, \mathbf{f}) = 0$ then $\mathsf{bad}_2 \leftarrow \mathsf{true}$
$\mathbf{c}[i] \leftarrow\$ \mathsf{Enc}(pk, \mathbf{x}[i])$
Return $(\mathbf{c}[i], \mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i]))$

PROC KD($a$):
$i \leftarrow i + 1$
$\mathbf{q}[i] \leftarrow a$ ; $\mathbf{t}[i] \leftarrow \mathsf{kd}$
$dk \leftarrow\$ \mathsf{KDer}(sk, a)$
Return $dk$

PROC FINALIZE($w$):
$\mathbf{q}[i+1] \leftarrow \varepsilon$ ; $\mathbf{t}[i+1] \leftarrow \mathsf{kd}$
For $i' = 1, \ldots, i+1$ and $j' = i'+1, \ldots, i+1$ do
    If $\mathbf{t}[i'] = \mathsf{enc} \wedge \mathbf{t}[j'] = \mathsf{kd}$ then
        If $\mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}[i']) \neq \mathcal{F}(\lambda, \mathbf{q}[j'], \mathbf{x}'[i'])$ then
            $\mathsf{bad}_1 \leftarrow \mathsf{true}$ ; Return false
If $\mathsf{bad}_2 = \mathsf{true}$ then
        $\boxed{\text{Return false}}$
Else return $R(\lambda, z, \mathbf{x}, \mathbf{q}, \mathbf{t}, St, w)$
```

Figure 8: Hybrid games RSS3-H2$_{\mathsf{FE},\mathcal{F},Z,D,R}$ and RSS3-H2$_{\mathsf{FE},\mathcal{F},Z,D,R}$ for the proof of Claim 5.5. The latter includes the boxed code while in the former it is removed.

5.3 Resampleability of Some Functionalities

We show resampleability of the following type of functionality.

FUNCTIONALITIES WITH POLYNOMIAL-SIZE RANGE. Let $\mathcal{F}$ be a functionality. We say that $\mathcal{F}$ *has polynomial-size range* if there is a polynomial $w(\cdot)$ such that $\mathcal{F}(\lambda, \mathbf{a}, x) \in F_\lambda$ for all $\lambda \in \mathbb{N}$, $x \in \{0,1\}^*$ and $\mathbf{a} \in (\{0,1\}^*)^*$ where $F_\lambda = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{w(\cdot)}\}$.

EXAMPLES AND DISCUSSION. An example of a functionality with polynomial-size range is PEKS [12] with a polynomial number of keywords. Although as shown by [12] this can be constructed from any IND-CPA public-key encryption scheme, the construction is inefficient for large polynomials, and more efficient constructions may be of interest to practitioners. These would be SS3 secure by our results.

We compare our notion of resampleability to that of *preimage sampleability* defined by O'Neill [29]. Our notion of resampleability appears to be much more restrictive, in the sense that showing it requires sampling a preimage that *lies in the message space of the adversary* rather than an arbitrary preimage. Since in general we cannot hope to "reverse engineer" the adversary's message space, it is unclear what we can do with it besides sample it in a black-box way. Indeed, the proof of our result does this and does not exploit any properties of the functionality itself as in [29], who showed that most functionalities considered in the literature have preimage sampleability. On the other hand, we show how to achieve a much stronger security definition than [29] who considered only non-adaptive security, meaning the adversary makes *no* key-derivation queries after seeing the challenge ciphertext.

Proposition 5.6 *Let $\mathcal{F}$ be a functionality with polynomial-size range. Then $\mathcal{F}$ is accurately resampleable. More precisely, for any PPT D there is a PPT $\overline{D}$ such that for every $\lambda \in \mathbb{N}$*

$$\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F}}(\lambda) \leq \frac{1}{2^\lambda} .$$

Furthermore, the running-time of $\overline{D}$ is that for at most $\lambda w(\lambda)$ executions of D.

The proof will use the following mathematical lemma.

```
Alg $S(1^k)$:
$i, j \leftarrow 0$
$St \leftarrow \varepsilon$
$(pk, sk) \leftarrow_\$ \mathsf{Setup}(1^k)$
Run $A(pk)$:
    On encryption query $\alpha$:
        $i \leftarrow i + 1$ ; $\boldsymbol{\alpha}[i] \leftarrow \alpha$
        For $j' = 1$ to $j$ do:
            $\mathbf{f}[j'] \leftarrow \mathrm{F}(\mathbf{a}[j'], i)$
        $(St, x) \leftarrow_\$ \overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
        $c \leftarrow_\$ \mathsf{Enc}(pk, x)$
        Return $c, y$
    On key-derivation query $a$:
        $j \leftarrow j + 1$ ; $\mathbf{a}[j] \leftarrow a$
        $\mathrm{OP}(a)$
        Return $\mathsf{KDer}(sk, a)$
Let $w$ be the output of $A$
Return $R(1^k, \mathbf{x}, \mathbf{a}, \boldsymbol{\alpha}, St, w)$
```

```
Alg $B(pk)$:
$i, j \leftarrow 0$
$St_0, St_1 \leftarrow \varepsilon$
Run $A(pk)$:
    On encryption query $\alpha$:
        $i \leftarrow i + 1$ ; $\boldsymbol{\alpha}[i] \leftarrow \alpha$
        $(St_0, \mathbf{x}_0[i]) \leftarrow_\$ D(St_0, \alpha)$
        For $j' = 1$ to $j$ do:
            $\mathbf{f}[j'] \leftarrow \mathcal{F}(\lambda, \mathbf{a}[j'], \mathbf{x}_0[i])$
        $(St_1, \mathbf{x}_1[i]) \leftarrow_\$ \overline{D}(\lambda, St_1, \alpha, \mathbf{a}, \mathbf{f})$
        Return $\mathrm{LR}(\mathbf{x}_0[i], \mathbf{x}_1[i])$
    On key-derivation query $a$:
        $j \leftarrow j + 1$ ; $\mathbf{a}[j] \leftarrow a$
        Return $\mathrm{KD}(a)$
Let $w$ be the output of $A$
Return $R(1^k, \mathbf{x}_1, \mathbf{a}, \boldsymbol{\alpha}, St, w)$
```

```
Alg $B^*(\lambda)$:
$i, j \leftarrow 0$
$St \leftarrow \varepsilon$
$(pk, sk) \leftarrow_\$ \mathsf{Setup}(\lambda)$
Run $A(pk)$:
    On encryption query $\alpha$:
        $i \leftarrow i + 1$ ; $\boldsymbol{\alpha}[i] \leftarrow \alpha$
        For $j' = 1$ to $j$ do:
            $\mathbf{f}[j'] \leftarrow \mathrm{F}(\mathbf{a}[j'], i)$
        If $i = i^*$ then
            $(St, x) \leftarrow_\$ \mathrm{CHALLENGE}(St, \alpha, \mathbf{a})$
            Halt execution of $A$
        $(St, x) \leftarrow_\$ \overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
        $c \leftarrow_\$ \mathsf{Enc}(pk, x)$
        Return $c, y$
    On key-derivation query $a$:
        $j \leftarrow j + 1$ ; $\mathbf{a}[j] \leftarrow a$
        Return $\mathsf{KDer}(sk, a)$
Return $\mathsf{Test}(\lambda, x, \mathbf{a}, \mathbf{f})$
```

Figure 9: Algorithms for proof of Claim 5.5.

Lemma 5.7 *Fix integers $n, k \geq 1$. For $\mathbf{x} \in \mathbb{R}^n$ define the function $f(\mathbf{x}) = \sum_{i=1}^n x_i(1-x_i)^k$. Subject to constraints $\sum_{i=1}^n x_i = 1$ and $0 \leq x_i \leq 1$ for all $1 \leq i \leq n$, $f(\cdot)$ is maximum at $x_1 = \ldots = x_n = 1/n$.*

Proof: (Of Lemma 5.7) We use a change of variables: let $y_i = (1 - x_i)$ for all $1 \leq i \leq n$. Then we equivalently seek to maximize

$$f'(\mathbf{y}) = \sum_{i=1}^{n} (1 - y_i) y_i^k$$

subject to $\sum_{i=1}^n y_i = n - 1$ and $0 \leq y_i \leq 1$ for all $1 \leq i \leq n$. Now use Lagrange multipliers: we seek a constant λ such that

$$\nabla f' \;=\; \lambda \nabla g \tag{2}$$

where $g(\mathbf{x}) = \sum_{i=1}^n y_i$. Fix any $1 \leq i \leq n$. Since

$$\frac{\partial f'}{\partial y_i} \;=\; 1 - (k+1) y_i^k \quad \text{and} \quad \frac{\partial g}{\partial y_i} = 1$$

by Equation (2) we have

$$1 - (k+1) y_i^k = \lambda \implies y_i = \left(\frac{1-\lambda}{k+1}\right)^{1/k} . \tag{3}$$

Using the constraint $\sum_{i=1}^n y_i = n - 1$ we have

$$n \left(\frac{1-\lambda}{k+1}\right)^{1/k} = n - 1 \implies \lambda = 1 - (k+1)\left(1 - \frac{1}{n}\right)^k .$$

Substituting this into Equation (3) we get $y_i = 1 - 1/n$ and so $x_i = -(y_i - 1) = 1/n$ as desired. Note that this must be a maximum (rather than minimum) since f vanishes when $x_i = 1$ for some i. ▌

We now prove the proposition.

Proof: (Of Proposition 5.6) Given any D, define the corresponding resampler $\overline{D}$ as follows:

```
Alg $\overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$:
For $i = 1$ to $\lambda w(\lambda)$ do:
    $x' \leftarrow\$ D(St, \alpha)$
    If $\mathsf{Test}(\lambda, x', \mathbf{a}, \mathbf{f}) = 1$ then return $x'$
Return $\perp$
```

We need to show that for any B

$$\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F},B}(\cdot) \;\leq\; \frac{1}{2^\lambda}\,.$$

To this end define game $\mathrm{Rsmp}_{\overline{D},D,\mathcal{F}}$ to pick a bit b at random and run the first experiment if $b = 1$ and the second if $b = 0$; also, have it set bad to true if $\overline{D}$ ever returns $\perp$. Define $\mathrm{Rsmp\text{-}H}_{\overline{D},D,\mathcal{F}}$ to be like the former but return $\mathsf{Resample}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$ if bad is set. Then using a standard conditioning argument and the Fundamental Lemma [8]:

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\mathrm{rs}}_{\overline{D},D,\mathcal{F},B}(\cdot) &= T(\mathrm{Rsmp}_{\overline{D},D,\mathcal{F}}, B, \cdot) \\
&\leq T(\mathrm{Rsmp\text{-}H}_{\overline{D},D,\mathcal{F}}, B, \cdot) + \Pr\left[\mathrm{Rsmp}^{B}_{\overline{D},D,\mathcal{F}}(\cdot) \text{ sets } \mathsf{bad}\right] \\
&= \frac{1}{2} + \Pr\left[\mathrm{Rsmp}^{B}_{\overline{D},D,\mathcal{F}}(\cdot) \text{ sets } \mathsf{bad}\right]\,.
\end{aligned}$$

Finally, we claim

$$\Pr\left[\mathrm{Rsmp}^{B}_{\overline{D},D,\mathcal{F}}(\cdot) \text{ sets } \mathsf{bad}\right] \;\leq\; \frac{1}{2^\lambda}\,.$$

To see this, fix any inputs $\lambda, St, \alpha, \mathbf{a}, \mathbf{f}$ and for all $1 \leq i \leq w(\lambda)$ let

$$p_i \;=\; \Pr[\mathcal{F}(\lambda, \mathbf{a}, x) = \mathbf{f}_i \;:\; x \leftarrow\$ D(St, \alpha)]\,.$$

Then

$$\begin{aligned}
\Pr\left[\,\overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f}) \text{ outputs } \perp\,\right] &= \sum_i p_i \cdot (1 - p_i)^{\lambda w(\lambda)} \\
&\leq \sum_i \frac{1}{w(\lambda)} \cdot \left(1 - \frac{1}{w(\lambda)}\right)^{\lambda \cdot w(\lambda)} \\
&\leq \frac{1}{2^\lambda}\,.
\end{aligned}$$

The second line above is justified by Lemma 5.7. The third line uses the inequality $(1-1/x)^x \leq 1/e < 1/2$ for any real number $x \geq 1$ (here e is Euler's constant). ▌

6 Brute-Force Construction Revisited

We now revisit the "brute-force" scheme defined by BSW [14], which provides a way to construct FE for any functionality with a polynomially-sized index space.

Let $\mathcal{F}$ be a functionality. We say that $\mathcal{F}$ has *polynomially-sized index space* if $\mathcal{F}(\lambda, a, x) = \perp$ if $a \notin \mathcal{A}_\lambda$ where $\mathcal{A}_\lambda = \{\varepsilon, a_1, a_2, \ldots, a_{p(\lambda)}\}$ for a polynomial $p(\cdot)$. Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a PKE scheme. Then we define a the *brute-force FE scheme* for $\mathcal{F}$ as follows:

```
proc Initialize($\lambda$):
    Return $\lambda$
proc Challenge($St, \alpha, \mathbf{a}$):
    $x \leftarrow_\$ D(St, \alpha)$
    $\mathbf{f} \leftarrow (\mathcal{F}(\lambda, \mathbf{a}, x), \mathcal{F}(\lambda, \varepsilon, x))$
    If $b = 1$ then $x' \leftarrow_\$ \overline{D}(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
    Else $x' \leftarrow_\$ \mathsf{Resample}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
    Return $x'$
proc Finalize($b'$):
    Return $(b' = 1)$

Alg $\mathsf{Resample}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$:
    If $\mathsf{CS}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f}) = \emptyset$ then return $\perp$
    $w \leftarrow_\$ \mathsf{CS}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f})$
    $x' \leftarrow D(St, \alpha; w)$
    Return $x'$
Alg $\mathsf{Test}(\lambda, x, \mathbf{a}, \mathbf{f})$:
    $\mathbf{f}' \leftarrow (\mathcal{F}(\lambda, \mathbf{a}, x), \mathcal{F}(\lambda, \varepsilon, x))$
    Return $(\mathbf{f} = \mathbf{f}')$
Set $\mathsf{CS}_D(\lambda, St, \alpha, \mathbf{a}, \mathbf{f}) := \{w : \mathsf{Test}(\lambda, D(St, \alpha; w), \mathbf{a}, \mathbf{f}) = \mathsf{true}\}$
```

Figure 10: Left: Game $\text{Resamp}_{\overline{D},D,\mathcal{F},b}$ for the resampleability definition. Right: Associated algorithms and definitions used by the game's procedures.

```
Setup($\lambda$)
    For $i = 1, \ldots p(\lambda)$ do
        $(\mathbf{pk}[i], \mathbf{sk}[i]) \leftarrow \mathcal{G}(\lambda)$
    Return $(\mathbf{pk}, \mathbf{sk})$
KDer($sk, a$)
    For $i = 1, \ldots, p(\lambda)$ do
        If $a_i = a$ then return $(i, \mathbf{sk}[i])$

Enc($\mathbf{pk}, m$)
    For $i = 1, \ldots, p(\lambda)$ do
        $\mathbf{c}[i] \leftarrow_\$ \mathcal{E}(\mathbf{pk}[i], \mathcal{F}(\lambda, a_i, m))$
    Return $\mathbf{c}$
Dec($(i, \mathbf{sk}[i]), \mathbf{c})$)
    If $(\mathbf{sk}[i] = \varepsilon)$ then return $|m|$
    Return $\mathcal{D}(\mathbf{sk}[i], \mathbf{c}[i])$
```

BSW [14] show that this construction is IND-secure (and hence, by our results in Section 5, SS2-secure) provided that the underlying PKE scheme is semantically secure. Moreover, they show a slightly decorated construction which is SS1-secure in the random oracle model. What we show is that it suffices for the underlying PKE scheme to be secure against *key-revealing SOAs* (SOA-K) for this FE scheme to be SS1-secure. In fact, for their result BSW implicitly use the non-committing (which implies SOA-K) PKE scheme of Nielsen [26] in the random oracle model as the underlying PKE scheme, so our result is a generalization of theirs. In particular, it allows us to obtain instantiations in the standard model by (necessarily) allowing long keys, meaning longer than the total number of bits encrypted; SOA-K secure PKE is known to exist in this setting [15, 16].

The intuition is that opening a particular $\mathcal{F}(\lambda, a_i, m)$ is equivalent to giving away $\mathbf{sk}[i]$. Therefore, it should be the case that if some $\mathcal{F}$ values for a particular set $I = \{a_{j_1}, a_{j_2}, \ldots, a_{j_{|I|}}\} \subseteq \mathcal{A}$ of indices are revealed, it is equivalent to opening up $\mathbf{sk}[j_1], \mathbf{sk}[j_2], \ldots, \mathbf{sk}[j_{|I|}]$ and should therefore, by SOA security, not make it any easier to obtain decryptions under any other secret key, i.e. to evaluate $\mathcal{F}$ on any other index.

SOA-K definition.To formalize our result we will need an explicit notion of SOA-K security, as described in BDWY [5]. Here we need to extend the BDWY definition to allow the adversary and simulator to ask for encryptions and secret keys adaptively (as in our SS notions for FE) rather than asking for an initial vector of ciphertexts and then afterwards a subset of the secret keys in one shot.

Let Π be a PKE scheme. The definition uses games $\text{RSOAK}_{\Pi,Z,D,R,n}$ and $\text{SSOAK}_{\Pi,Z,D,R,n}$ of Figure 11. We say that Π is *SOAK-secure* if for every auxiliary input generator Z, every PT message sampler D, every PT relation R, every PT adversary A, and every polynomial $n(\cdot)$ there is a PT simulator S such that

$$\mathbf{Adv}^{\text{soa-k}}_{\Pi,A,S,Z,D,R,n}(\cdot) \;=\; \Pr\left[\text{RSOAK}^{A}_{\Pi,Z,D,R,n}(\cdot)\right] - \Pr\left[\text{SSOAK}^{S}_{\Pi,Z,D,R,n}(\cdot)\right]$$

is negligible.

Theorem 6.1 *Let $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a SOA-K secure PKE scheme. Then the corresponding brute-force FE scheme for any functionality $\mathcal{F}$ with a polynomially-sized index space is SS1-secure.* ▌

```
$\underline{\text{INITIALIZE}(\lambda)}$
$i', j \leftarrow 0$
$St \leftarrow \varepsilon$
For $j' = 1, \ldots, n(\lambda)$ do
    $(\mathbf{pk}[j'], \mathbf{sk}[j']) \leftarrow_{\$} \mathcal{G}(\lambda)$
$z \leftarrow_{\$} Z(\lambda)$
Return $(\mathbf{pk}, z)$

$\underline{\text{ENC}(\alpha)}$
$(St, \mathbf{m}) \leftarrow_{\$} D(St, \alpha)$
$i' \leftarrow i' + 1$
$\boldsymbol{\alpha}[i'] \leftarrow \alpha$ ; $\mathbf{x}[i'] \leftarrow \mathbf{m}$
For $j' = 1, \ldots, n(\lambda)$ do
    $\mathbf{c}[j'] \leftarrow_{\$} \mathcal{E}(\mathbf{pk}[j'], \mathbf{m}[j'])$
Return $\mathbf{c}$

$\underline{\text{CORRUPT}(i)}$
$j \leftarrow j + 1$
$\mathbf{i}[j] \leftarrow i$
For $j' = 1, \ldots, i'$ do
    $\mathbf{m}[j'] \leftarrow \mathbf{x}[j'][i]$
Return $(\mathbf{m}, \mathbf{sk}[i])$

$\underline{\text{FINALIZE}(w)}$
Return $R(\lambda, z, \mathbf{x}, \mathbf{i}, \boldsymbol{\alpha}, St, w)$
```

```
$\underline{\text{INITIALIZE}(\lambda)}$
$i', j \leftarrow 0$
$z \leftarrow_{\$} Z(\lambda)$
$St \leftarrow \varepsilon$
Return $z$

$\underline{\text{MSG}(\alpha)}$
$(St, \mathbf{m}) \leftarrow_{\$} D(St, \alpha)$
$i' \leftarrow i' + 1$
$\boldsymbol{\alpha}[i'] \leftarrow \alpha$ ; $\mathbf{x}[i'] \leftarrow \mathbf{m}$
Return $\varepsilon$

$\underline{\text{CORRUPT}(i)}$
$j \leftarrow j + 1$
$\mathbf{i}[j] \leftarrow i$
For $j' = 1, \ldots, i'$ do
    $\mathbf{m}[j'] \leftarrow \mathbf{x}[j'][i]$
Return $\mathbf{m}$

$\underline{\text{FINALIZE}(w)}$
Return $R(\lambda, z, \mathbf{x}, \mathbf{i}, \boldsymbol{\alpha}, St, w)$
```

Figure 11: Left: "Real world" game $\text{RSOAK}_{\Pi,Z,D,R,n}$ for the SOA-K definition. Right: "Ideal world" game $\text{SSOAK}_{\Pi,Z,D,R,n}$ for the SOA-K definition.

```
**Alg** $R'(\lambda, z, \mathbf{x}', \mathbf{i}, \boldsymbol{\alpha}, St, w)$:
For $j' = 1, \ldots, |\mathbf{x}'|$
    $\mathbf{x}[j'] \leftarrow \mathbf{x}'[j'][p(\lambda) + 1]$
For $j' = 1, \ldots, |\mathbf{i}|$
    Let $i$ be such that $a_i = \mathbf{i}[j']$
    $\mathbf{a}[j'] \leftarrow a_i$
Return $R(\lambda, z, \mathbf{x}, \mathbf{a}, \boldsymbol{\alpha}, St, w)$
```

```
**Alg** $D'(St, \alpha)$:
$x \leftarrow D(St, \alpha)$
For $i = 1, \ldots, p(\lambda)$ do
    $\mathbf{m}[i] \leftarrow \mathcal{F}(\lambda, a_i, x)$
$\mathbf{m}[p(\lambda) + 1] \leftarrow x$
Return $\mathbf{m}$
```

```
**Alg** $A'(\lambda)$:
$(\mathbf{pk}, z) \leftarrow_{\$} \text{INITIALIZE}(\lambda)$
Run $A$ on input $\lambda$:
    On initialize query $\lambda$ return $(\mathbf{pk}, z)$
    On encryption query $\alpha$
        $\mathbf{c} \leftarrow_{\$} \text{ENC}(\alpha)$
        Return $(\mathbf{c}[1 \ldots p(\lambda)], \mathsf{msglen}(\mathbf{c}[p(\lambda) + 1]))$
    On key-derivation query $a$
        Let $i$ be such that $a_i = a$
        Return $\text{CORRUPT}(i)$
Let $w$ be the output of $A$
Return $w$
```

Figure 12: Algorithms for proof of security of brute-force scheme.

Proof of Theorem 6.1: Denote by $\mathcal{A}_\lambda = \{\varepsilon, a_1, a_2, \ldots, a_{p(\lambda)}\}$ the family of sets satisfying the definition of polynomially-sized index space for $\mathcal{F}$. Let Z be an auxiliary input generator, D be a message sampler, R be a relation, and A be a SS1 adversary. Consider the auxiliary input generator $Z' = Z$, and then the relation R', message sampler D' and SOA-K adversary A' defined as in Figure 12.

Above, for a ciphertext c, $\mathsf{msglen}(c)$ returns the length of the decryption of c (which we assume is efficiently computable from a ciphertext, or else that D always outputs messages of some known length). Also note that we set $n(\cdot) = p(\cdot) + 1$ in the SOAK games; this allows A' to pass the length of the "actual" payload x to A and similarly R' to pass the "actual" payload vector $\mathbf{x}$ to R. Since Π is SOA-K secure, we know that there exists a PT simulator S' such that

$$\mathbf{Adv}^{\text{soa-k}}_{\Pi,A',S',Z',D',R'}(\cdot) = \Pr\left[\text{RSOAK}^{A'}_{\Pi,Z',D',R'}(\cdot)\right] - \Pr\left[\text{SSOAK}^{S'}_{\Pi,Z',D',R'}(\cdot)\right]$$

is negligible. Now we use S' to construct S, a simulator for the ISS game, in a similar way we adapted A to get A':

```
Alg $S(\lambda)$:
$z \leftarrow\!\!\$\ \textsc{Initialize}(\lambda)$
Run $S'$ on input $\lambda$:
    $i' \leftarrow 0$
    On initialize query $\lambda$ return $z$
    On message query $\alpha$:
        $i' \leftarrow i' + 1$ ; $\textsc{Msg}(\alpha)$
        Return $\varepsilon$
    On corrupt query $i$:
        $\textsc{Op}(a_i)$
        For $j = 1, \ldots, i'$
            $\mathbf{x}[j] \leftarrow \mathrm{F}(a_i, j)$
        Return $\mathbf{x}$
Let $w$ be the output of $S'$
Return $w$
```

By construction, we now have

$$\Pr\left[\mathrm{RSOAK}^{A'}_{\Pi,Z',D',R',p+1}(\cdot)\right] = \Pr\left[\mathrm{RSS}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right]$$

$$\Pr\left[\mathrm{SSOAK}^{S'}_{\Pi,Z',D',R',p+1}(\cdot)\right] = \Pr\left[\mathrm{ISS}^{S}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right]$$

Therefore,

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ss1}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) &= \Pr\left[\mathrm{RSS}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right] - \Pr\left[\mathrm{ISS}^{S}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right] \\ &= \Pr\left[\mathrm{RSOAK}^{A'}_{\Pi,Z',D',R'}(\cdot)\right] - \Pr\left[\mathrm{SSOAK}^{S'}_{\Pi,Z',D',R',p+1}(\cdot)\right] \\ &= \mathbf{Adv}^{\mathrm{soa\text{-}k}}_{\Pi,A',S',Z',D',R',p+1}(\cdot)\end{aligned}$$

which is negligible by assumption. ▌

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. 5, 6, 8

[2] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Dec. 2011. 2

[3] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bound. Cryptology ePrint Archive, Report 2012/468, 2012. `http://eprint.iacr.org/`. 3, 4

[4] M. Barbosa and P. Farshim. Semantically secure functional encryption revisited. Cryptology ePrint Archive, Report 2012/474, 2012. `http://eprint.iacr.org/`. 4, 7

[5] M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In D. Pointcheval, editor, *EUROCRYPT 2012*, LNCS. Springer, 2012. 1, 3, 10, 18

[6] 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. 4

[7] 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. 10

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. 5, 12, 14, 17

[9] 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. 4

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004. 5, 6

[11] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Aug. 2004. 2

[12] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, May 2004. 6, 15

[13] D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 1, 2

[14] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Mar. 2011. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 17, 18

[15] 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. 3, 4, 18

[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. 3, 4, 18

[17] 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. 4

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

[19] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012. 4

[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309. 1, 2

[21] B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88. Springer, Dec. 2011. 4

[22] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Apr. 2008. 1, 2, 6, 7, 23

[23] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, May 2010. 2

[24] A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, May 2011. 2

[25] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, May 2011. 2

[26] 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, 18

[27] T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Dec. 2009. 2

[28] T. Okamoto and K. Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 35–52. Springer, Mar. 2011. 2

[29] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`. 1, 2, 4, 7, 15

[30] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM CCS 07*, pages 195–203. ACM Press, Oct. 2007. 2

[31] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, May 2005. 1

[32] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Aug. 1985. 1

[33] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473. Springer, Mar. 2009. 2

[34] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Aug. 2009. 2

[35] B. Waters. Functional encryption for regular languages. In *CRYPTO*, pages 218–235, 2012. 4

[36] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005. 2, 6

A Standard Primitives

Public-key encryption schemes. An *public-key encryption scheme* $\Pi = (\mathcal{G}, \mathcal{E}, \mathcal{D})$ is specified by three PT algorithms. Via $(pk, sk) \leftarrow\!\!\$\, \mathcal{G}(\lambda)$ the key-generation algorithm $\mathcal{G}$ generates a public key and matching secret key. Via $c \leftarrow\!\!\$\, \mathcal{E}(pk, m)$ the encryption algorithm $\mathcal{E}$ takes pk and message m and returns a ciphertext $c \in \{0,1\}^* \cup \{\bot\}$. Via $m \leftarrow \mathcal{D}(sk, c)$, the deterministic decryption algorithm $\mathcal{V}$ returns a message m. We require that $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$ for all $\lambda \in \mathbb{N}$, all $(pk, sk) \in [\mathcal{G}(\lambda)]$, and all $m \in \{0,1\}^*$

Hash Functions. A hash function $\mathcal{H} = (K, H)$ is a tuple of PT algorithms. Via $hk \leftarrow\!\!\$\, K(\lambda)$ the key-generation algorithm K produces a key hk. Via $y \leftarrow H(hk, x)$ the deterministic hashing algorithm H produces the hash of a string x under key hk. Collision-resistance is defined via game CR_Γ whose Initialize(λ) procedure returns $hk \leftarrow\!\!\$\, K(\lambda)$ and whose Finalize procedure on input (x, x') returns $(x \neq x') \wedge (H(hk, x) = H(hk, x'))$. There are no other procedures. The advantage of an adversary C is defined by $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\lambda) = \Pr\left[\mathrm{CR}^C_{\mathcal{H}}(\lambda)\right]$. We say that $\mathcal{H}$ is collision-resistant (CR) if $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\cdot)$ is negligible for every PT C.

B The Unordered Case

Unordered SS1. We start by formalizing the definition. Let $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KDer}, \mathsf{Enc}, \mathsf{Dec})$ be a $\mathcal{F}$-FE scheme. The definition uses games $\mathrm{RSS1\text{-}U}_{\mathsf{FE},\mathcal{F},Z,D,R}$ and $\mathrm{ISS1\text{-}U}_{\mathcal{F},Z,D,R}$ of Figure 13. ("U" for unordered.) We say that FE is *SS1-U-secure* if for every auxiliary input generator Z, every PT message sampler D, every PT relation R and every PT adversary A there is a PT simulator S such that

$$\mathbf{Adv}^{\text{ss-u}}_{\mathsf{FE},\mathcal{F},A,S,Z,D,R}(\cdot) \;=\; \Pr\left[\mathrm{RSS1\text{-}U}^A_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right] - \Pr\left[\mathrm{ISS1\text{-}U}^S_{\mathcal{F},Z,D,R}(\cdot)\right]$$

is negligible.

```
PROC INITIALIZE($\lambda$):
    $(pk, sk) \leftarrow\$ \mathsf{Setup}(\lambda)$
    $z \leftarrow\$ Z(\lambda)$
    $i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$
    Return $(pk, z)$

PROC ENC($\alpha$):
    $Q_\alpha \leftarrow Q_\alpha \cup \{\alpha\}$
    $(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
    $\mathbf{c}[i] \leftarrow\$ \mathsf{Enc}(pk, \mathbf{x}[i])$
    Return $(\mathbf{c}[i], \mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i]))$

PROC KD($a$):
    $Q_a \leftarrow Q_a \cup \{a\}$
    $dk \leftarrow\$ \mathsf{KDer}(sk, a)$
    Return $dk$

PROC FINALIZE($w$):
    Return $R(\lambda, z, \mathbf{x}, Q_a, Q_\alpha, St, w)$
```

```
PROC INITIALIZE($\lambda$):
    $i, j \leftarrow 0$ ; $St \leftarrow \varepsilon$
    $z \leftarrow\$ Z(\lambda)$ ; $A \leftarrow \emptyset$
    Return $z$

PROC MSG($\alpha$):
    $Q_\alpha \leftarrow Q_\alpha \cup \{\alpha\}$
    $(St, \mathbf{x}[i]) \leftarrow\$ D(St, \alpha)$
    Return $\mathcal{F}(\lambda, \varepsilon, \mathbf{x}[i])$

PROC OP($a$):
    $Q_a \leftarrow Q_a \cup \{a\}$
    Return $\varepsilon$

PROC F($a, s$):
    If $a \in \mathsf{El}(\mathbf{a})$ and $1 \le s \le i$ then
        Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
    Else return $\perp$

PROC FINALIZE($w$):
    Return $R(\lambda, z, \mathbf{x}, Q_a, Q_\alpha, St, w)$
```

Figure 13: Left: "Real world" game RSS1-U$_{\mathsf{FE},\mathcal{F},Z,D,R}$ for the SS1-U definition. Right: "Ideal world" game ISS1-U$_{\mathcal{F},Z,D,R}$ for the SS1-U definition.

IDENTITY-EMBEDDABLE PE. In the unordered case our result applies to what we call *identity-embeddable* predicate encryption schemes. Intuitively, these are predicate encryption schemes whose decryption policy embeds an identity matrix. Let $\mathcal{F}$ be a predicate encryption functionality for predicate $\mathcal{P}$. We say that $\mathcal{F}$ is *$n(\cdot)$-identity embeddable* if there are $\{(a_{1,\lambda}, \dots, a_{n(\lambda),\lambda})\}_{\lambda\in\mathbb{N}}$ and $\{(w_{1,\lambda}, \dots, w_{n(\lambda),\lambda})\}_{\lambda\in\mathbb{N}}$ such that for all $\lambda \in \mathbb{N}$ and all $1 \le i, j \le n(\lambda)$, $\mathcal{P}(\lambda, a_i, w_j) = 1$ if $i = j$ and $\perp$ otherwise.

Note that the decryption policy described by an identity matrix is exactly that of IBE. Thus, any PE functionality that "contains" the functionality for an IBE scheme with n identities is n-identity embeddable. PE functionalities considered in the literature typically satisfy this requirement. For example, the inner-product functionality over $\mathbb{Z}_N$ introduced by [22] was shown to implement IBE by restricting a to the form $(\mathsf{id}, 1)$ and w to the form $(-1, \mathsf{id}')$ for $\mathsf{id}, \mathsf{id}' \in \mathbb{Z}_N$. Thus any inner-product functionality with vectors of dimension 2 is $|N|$-identity embeddable. Note that in [22] $|N|$ is exponential in λ, whereas for our application we just need it to be polynomial in λ (namely, the output length of a collision-resistant hash function).

Theorem B.1 *Let $\mathcal{F}$ be a predicate encryption functionality for predicate $\mathcal{P}$. Let $\mathcal{H} = (K, H)$ be a collision-resistant hash function with output length $\ell_{\mathcal{H}}(\cdot)$. Suppose that for every $\lambda \in N$, $\mathcal{F}(\lambda, \varepsilon, x)$ is the same for all $x \in [X_\lambda]$. Furthermore, suppose that $\mathcal{F}$ is $2\ell_{\mathcal{H}}(\cdot)$-identity embeddable. Then there does not exist an SS1-U-secure $\mathcal{F}$-FE scheme. More precisely, suppose* FE *is a $\mathcal{F}$-FE scheme with secret-key length $\ell_{sk}(\cdot)$. Then for any function $\mu(\cdot)$ there exists a PT auxiliary input generator Z, message sampler D, PT adversary A, PT relation R, and CR-adversary C such that for every simulator S*

$$\mathbf{Adv}^{\mathrm{ss}}_{\mathsf{FE},\mathcal{F},A,S,D,R}(\cdot) \;\le\; 1 - \sqrt{\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H},C}(\cdot) + 1/\mu(\cdot)}\;.$$

Adversary A makes $\ell_{sk}(\cdot) + \log \mu(\cdot)$ encryption queries and $\ell_{\mathcal{H}}(\cdot)$ key-derivation queries.

We remark that our theorem and proof treat the case of (P, p), (p, p), and (p, P) PE functionalities in a unified way. However, in the case of (P, p)-PE the condition in the theorem that $\mathcal{F}$ be $2\ell_{\mathcal{H}}(\cdot)$-identity embeddable can be improved to $\ell_{\mathcal{H}}(\cdot)$-identity embeddable. We also stress that, unlike Theorem 4.1, our result here does not apply to any "non-trivial" functionality but only those of a certain form. Better understanding the nature of the gap between the two results or closing it remains an interesting open problem. It is also interesting to note that our result here uses an adversary that makes $\ell_{\mathcal{H}}(\cdot)$ key-derivation queries as opposed to the constant 2 of Theorem 4.1. It would be interesting to know if this is tight, meaning whether SS1-U can be achieved against adversaries making some small bounded number of key derivation queries.

Proof of Theorem B.1: Denote by $\{(a_{1,\lambda}, \ldots, a_{2\ell_{\mathcal{H}}(\lambda),\lambda})\}_{\lambda\in\mathbb{N}}$ and $\{(w_{1,\lambda}, \ldots, w_{2\ell_{\mathcal{H}}(\lambda),\lambda})\}_{\lambda\in\mathbb{N}}$ the values that satisfy $2\ell_{\mathcal{H}}(\lambda)$-identity embeddability for $\mathcal{F}$. For $\lambda \in \mathbb{N}$ denote by $\overline{\varepsilon}_\lambda$ the value such that $\mathcal{F}(\lambda, \varepsilon, x) = \overline{\varepsilon}_\lambda$ for all $x \in [X_\lambda]$. Let $n(\cdot) := \ell(\cdot) + \log \mu(\cdot)$. Define auxiliary input generator Z on input λ to return $hk \leftarrow\$ K(\lambda)$. For $\lambda \in \mathbb{N}$ denote by M_λ a distribution on messages for FE so that if FE is private message then $M_\lambda = \{0, 1\}$ and otherwise is trivial. Then define message sampler D on input St, α to set $St \leftarrow 1$ if $St = \varepsilon$ and $St \leftarrow St+1$ otherwise, and to return $(St, (w_{2(\lfloor St/n\rfloor+1)-b,\lambda}, m))$ where $b \leftarrow\$ \{0, 1\}$ and $m \leftarrow\$ M$. Define adversary A and relation R as follows:

```
Alg $A(pk, hk)$:
For $i = 1, \ldots, \ell_{\mathcal{H}}(\lambda)n(\lambda)$ do:
    $\mathbf{c}[i] \leftarrow\$ \textsc{Enc}(\lambda)$
$h \leftarrow H(hk, pk\|\mathbf{c})$
For $i = 1$ to $\ell_{\mathcal{H}}(\lambda)$ do:
    $\mathbf{sk}[i] \leftarrow\$ \textsc{Kd}(a_{2i-h[i]})$
$w \leftarrow (pk, \mathbf{c}, h, \mathbf{sk})$
Return $w$
```

```
Alg $R(\lambda, \mathbf{x}, Q_a, Q_\alpha, St, w)$:
$(pk, hk, \mathbf{c}, h, \mathbf{sk}) \leftarrow w$
If $h \neq H(hk, pk\|\mathbf{c})$ then return false
If $|\mathbf{sk}| \neq \ell_{\mathcal{H}}(\lambda)$ then return false
For $i = 1, \ldots, \ell_{\mathcal{H}}(\lambda)$ do:
    If $|\mathbf{sk}[i]| \neq \ell_{sk}(\lambda)$ then return false
If $Q_a \neq \{a_{2i-h[i],\lambda} \ : \ 1 \leq i \leq \ell_{\mathcal{H}}(\lambda)\}$ then return false
If $\mathbf{x}| \neq \ell_{\mathcal{H}}(\lambda)n(\lambda) \ \vee \ Q_\alpha \neq \{\lambda\}$ then return false
For $i = 1, \ldots, \ell_{\mathcal{H}}(\lambda)$ do:
    If $\mathsf{Dec}(\mathbf{sk}[i], \mathbf{c}[i \ldots i+n-1]) \neq \mathcal{F}(\lambda, a_{2i-h[i]}, \mathbf{x}[i \ldots i+n-1])$
        Then return false
Return true
```

By construction $\Pr\left[\text{RSS1-U}^{A}_{\mathsf{FE},\mathcal{F},Z,D,R}(\cdot)\right] = 1$. Let S be any simulator. Furthermore, parsing the output of S as $(pk, \mathbf{c}, h, \mathbf{sk}) \leftarrow w$, we assume it holds that $h = H(hk, pk\|\mathbf{c})$, $|\mathbf{sk}| = \ell_{\mathcal{H}}(\lambda)$ and $|\mathbf{sk}[i]| = \ell_{sk}(\lambda)$ for $1 \leq i \leq \ell_{\mathcal{H}}(\lambda)$, $Q_a = \{a_{2n-h[i],\lambda} \ : \ 1 \leq i \leq \ell_{\mathcal{H}}\}$, and $|\boldsymbol{\alpha}| = \ell_{\mathcal{H}}(\lambda)n(\lambda)$ and $Q_\alpha = \{\lambda\}$, since otherwise the relation R returns false. Towards applying Lemma 4.2, we write the execution of S in the ISS-U game as a composition of two algorithms S_1, S_2 as follows:

```
Alg $S_1(\lambda)$:
$i \leftarrow 0$ ; $z \leftarrow\$ \textsc{Initialize}(\lambda)$
Run $S(z)$:
On message-query $\alpha$ do:
    $i \leftarrow i + 1$
    $b \leftarrow\$ \{0, 1\}$ ; $m \leftarrow\$ M_\lambda$
    $\mathbf{x}[i] \leftarrow (w_{2(\lfloor i/n\rfloor+1)-b,\lambda}, m)$
    Return $\overline{\varepsilon}$
On op-query $a$ do:
    $j \leftarrow j + 1$ ; $A \leftarrow A \cup \{a\}$
    If $j = \ell_{\mathcal{H}}(\lambda)$ then
        $a^* \leftarrow a$
        Halt computation of $S_1$ with state $St$
    Else return $\varepsilon$
On $F$-query $(a, s)$ do:
    If $1 \leq s \leq i$ and $a \in A$ do:
    Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
    Else return $\perp$
$\overline{St} \leftarrow St\|i\|j\|\mathbf{x}\|A\|a^*$
Return $\overline{St}$
```

```
Alg $S_2(\overline{St})$:
$St\|i\|j\|\mathbf{x}\|\|A\|a^* \leftarrow \overline{St}$
Let $i^*$ be such that $a^* \in \{a_{2i^*,\lambda}, a_{2i^*-1,\lambda}\}$
For $q = 1, \ldots, n$ do:
    $b \leftarrow\$ \{0, 1\}$ ; $m \leftarrow\$ M_\lambda$
    $\mathbf{x}[i^* \cdot n + q - 1] \leftarrow (w_{2i^*-b,\lambda}, m)$
Run $S$ at state $St$:
On message-query $\alpha$ do:
    $i \leftarrow i + 1$
    If $i \notin \{i^* \cdot n, \ldots, i^* \cdot n + n - 1\}$ do:
        $b \leftarrow\$ \{0, 1\}$ ; $m \leftarrow\$ M_\lambda$
        $\mathbf{x}[i] \leftarrow (w_{2\lfloor\alpha/n\rfloor-b,\lambda}, m)$
    Return $\overline{\varepsilon}$
On $F$-query $(a, s)$ do:
    If $1 \leq s \leq i$ and $a \in \mathsf{El}(\mathbf{a})$ do:
    Return $\mathcal{F}(\lambda, a, \mathbf{x}[s])$
    Else return $\perp$
Let $w$ be the output of $S$
Return $w$
```

Namely, S_1 runs S up to the point that it makes its *last* Op query (namely the $\ell_{\mathcal{H}}(\lambda)$-th one), and S_2 runs S following this Op query. Below we justify the following sequence of inequalities:

$$\begin{aligned}
\Pr\left[\,\text{ISS1-U}^{S}_{\mathcal{F},Z,D,R}(\cdot)\,\right] &= \mathbf{AP}_1(P_1,P_2,\lambda) \\
&\le \sqrt{\mathbf{AP}_2(S_1,S_2,\lambda)} \\
&\le \sqrt{\mathbf{Adv}^{\text{col}}_{\mathcal{H},C}(\lambda)+\Pr\left[\,\mathcal{F}(\lambda,a_\lambda,\mathbf{x})\in Y\,\right]} \\
&\le \sqrt{\mathbf{Adv}^{\text{col}}_{\mathcal{H},C}(\lambda)+2^{\ell}(1/2)^{n}} \\
&= \sqrt{\mathbf{Adv}^{\text{col}}_{\mathcal{H},C}(\lambda)+1/\mu(\lambda)}
\end{aligned}$$

Above, the first line is by construction; note that while S_2 chooses $\mathbf{x}^*$ and hence some components of $\mathbf{x}$ differently than S, this is equivalent due to the $2n$-identity embeddability of $\mathcal{F}$ and is only used to make the analysis more transparent. The second is by Lemma 4.2. For the third, we can define C analogously to the proof of Theorem 4.1 (namely, C simply mimics the double execution experiment with S_1, S_2). Let us denote the two outputs of S_2 in that experiment as $w^1 = (pk^1, \mathbf{c}^1, h^1, \mathbf{sk}^1)$ and $w^2 = (pk^2, \mathbf{c}^2, h^2, \mathbf{sk}^2)$. Now, unless C finds a collision we have $\mathbf{c}^1 = \mathbf{c}^2$, denote by $\mathbf{c}^*$ this common value. Then define the set $Y := \{y \in \{0,1\}^* \ : \ \exists s \in \{0,1\}^{\ell} \text{ s.t. } \mathsf{Dec}(s, \mathbf{c}^*[i^*n \dots i^*n+n-1]) = y\}$. If it is not the case that $\mathcal{F}(\lambda, a_{i^*,\lambda}, \mathbf{x}[i^*n \dots i^*n+n-1]) \in Y$ when $\mathbf{x}[i^*n \dots i^*n+n-1]$ is (re-)sampled by S_2 then the relation R must reject, as no $\mathbf{sk}^2$ will satisfy $\mathsf{Dec}(\mathbf{sk}^2[i^*], \mathbf{c}^*[i^*n \dots i^*\cdot n+n-1]) = \mathcal{F}(\lambda, a_{i^*n-h[i^*]}, \mathbf{x}[i^*n \dots i^*n+n-1])$, justifying the third line above. To see the fourth line, fix $s \in \{0,1\}^{\ell}$ and denote $\mathsf{Dec}(s, \mathbf{c}^*[i^*n \dots i^*n+n-1])$ by $\mathbf{x}^*$. As $\mathbf{x}[i^* \cdot n]$ is defined as $(w_{2\lfloor\alpha/n\rfloor-b,\lambda}, m)$ for random $b \in \{0,1\}$ and $m \in M_\lambda$, $\mathbf{x}[i^* \cdot n] = \mathbf{x}^*[1]$ with probability at most $1/2$, and similarly for the remaining components of $\mathbf{x}[i^* \cdot n \dots i^* \cdot n + n - 1]$, which are sampled independently. Taking a union bound over all possible $s \in \{0,1\}^{\ell}$ yields the fourth line. Finally, for the fifth line above we just substitute $n(\lambda) = \ell(\lambda) + \log(\mu(\lambda))$.

▮