On Access Control Encryption without Sanitization*

Cecilia Boschini[1], Ivan Damgård[2], Claudio Orlandi[2]

[1] Technion and Reichman University (IDC), Israel
cecilia.boschini@gmail.com
[2] Aarhus University, Aarhus, Denmark
{ivan, orlandi}@cs.au.dk

June 28, 2022

Abstract. Access Control Encryption (ACE) [12] allows to control information flow between parties by enforcing a policy that specifies which user can send messages to whom. The core of the scheme is a sanitizer, i.e., an entity that "sanitizes" all messages by essentially re-encrypting the ciphertexts under its key. In this work we investigate the natural question of whether it is still possible to achieve some meaningful security properties in scenarios when such a sanitization step is not possible. We answer positively by showing that it is possible to limit corrupted users to communicate only through insecure subliminal channels, under the necessary assumption that parties do not have pre-shared randomness. Moreover, we show that the bandwidth of such channels can be limited to be $O(\log(\lambda))$ by adding public ciphertext verifiability to the scheme under computational assumptions. In particular, we rely on a new security definition for obfuscation, Game Specific Obfuscation (GSO), which is a weaker definition than VBB, as it only requires the obfuscator to obfuscate programs in a specific family of programs, and limited to a fixed security game.

* An extended abstract appeared at SCN 2022. This is the full paper.

Table of Contents

1 Introduction

Designers of practical secure IT systems are often interested in controlling the flow of information in their system. For this purpose one sets up a security policy that contains rules on what operations the entities in the system are allowed to execute. Crucially, such rules must constrain both write and read operations as both types may lead to unwanted transfer of data. This was formalized in the classical Bell-Lapadula security policy as the "no read up" (entities with low security clearance cannot read top-secret data) and "no write-down" rules (entities with a high security clearance cannot write to public files). If entities are not assumed to be honest, such a security policy cannot be enforced unless we assume a trusted party, often known as a *sanitizer*, which will stop and/or modify unwanted communication. Of course, the sanitizer cannot do this unless we assume that parties can only communicate via the sanitizer. In practical systems one usually tries to ensure this by a combination of hardware security and software design, for instance in the kernel of the operating system.

In [12] Damgård et al. asked whether cryptography can be used to simplify the job of the sanitizer, and reduce the amount of trust we need to place in it. To this end, they introduced the notion of Access Control Encryption (ACE). Using an ACE scheme, the sanitizer does not need to know the security policy or the identities of any parties in the system. It just needs to process every message it receives and pass it on. The processing essentially amounts to re-randomize every message received. Instead of asking the sanitizer to enforce the security policy, an ACE scheme integrates the policy in the key generation algorithm, which gives an encryption key to each sender, a decryption key to each receiver and a sanitizer key for the sanitizer. The keys are designed such that, after sanitization, a receiver can decrypt a message, only if it was encrypted by a sender that is allowed to send to that receiver.

Observe that security requires the physical assumption that a corrupt sender cannot bypass the sanitizer and send directly to any receiver she wants. Indeed, it may seem that nothing non-trivial can be achieved if we drop this assumption. On the other hand, assuming such a communication bottleneck may be hard to justify in practice, and makes the system vulnerable to DDoS attacks (in case the sanitizer is offline). It is then natural to wonder:

Can we achieve any meaningful security without sanitization?

2 Our Results

In this paper, we answer affirmatively to the previous question analyzing two new models, both avoiding the need of preprocessing ciphertexts before delivery. We present formal definitions of ACE in these models, and we instantiate them under various computational assumptions. Along the way we obtain a standard ACE with sender anonymity from standard assumptions, which had been left as an open problem in [18] (deferred to Appendix C due to lack of space).

2.1 Modeling ACE without Sanitization

2.1.1 ACE without Sanitizer (ACEnoS) Removing the sanitization bottleneck implies that senders can now post to a bulletin board that receivers can read from. As in standard ACE, parties have no other communication channel available, and key generation assumes a trusted party. However, senders are now free to post whatever they want. What security properties can we hope to achieve in such a model? Clearly, we can do what cryptography "natively" allows us to do, namely what we call the *No Read Rule* (NRR): an honest sender can encrypt a message such that only the designated receiver can extract information about the plaintext from the ciphertext; furthermore we can guarantee that a ciphertext does not reveal the identity of the sender. What we can do about a corrupt sender is more subtle: clearly, we cannot stop a sender from simply posting any message she wants, thus broadcasting confidential data. But, on the other hand, this is often not what a corrupt sender wants to do. If, for instance, the data involved is extremely valuable, it may be more attractive to break the security policy by sending a *secret* message that can only be decrypted by a

specific (corrupted) receiver she is not allowed to send to. This attack we can actually hope to stop, through what we call the *No Secret Write Rule* (NSWR)[1]: parties cannot communicate secretly if the policy does not allow it. If parties manage to communicate against policy, then anyone can read their communication.

2.1.2 Communication Restrictions For this goal to be meaningful, we can allow corrupted senders and receivers to share a common strategy, but not randomness. Without this constraint, any no secret write rule can trivially be broken just using one-time-pad encryption, for instance. Assuming that the parties' initial states are uncorrelated, the rough intuition is that if the key generation does not supply a corrupted sender and receiver with sufficiently correlated key material, the receiver's ability to decrypt a ciphertext cannot depend on the keys she has. But if it does not, then anyone should be able to extract the message the corrupted sender wants to leak, and so the message is effectively publicly available. Observe that the assumption that parties do not have pre-shared randomness is not new: in fact, Alwën et al. [3] already pointed out the need for such an assumption when building collusion-free protocols.

2.1.3 Verifiable ACE (VACE) Our solution above implies that whatever information a corrupt sender embeds in his message can in principle be accessed by anyone. But there is no limit on the *amount* of information she can leak in this way. Is there some way to plausibly limit such leakage? We answer affirmatively, by adding a way to publicly verify the posted ciphertexts. Intuitively, if a ciphertext verifies, it is correctly formed according to the encryption algorithm, not something the sender can choose as he likes (e.g., no unencrypted messages). However, a sender may still try to output a valid ciphertext that equals the encoding of an n-bit subliminal message. The hope is that now the sender's situation becomes somewhat similar to having to generate ciphertexts by calling a random (encryption) oracle. In this scenario embedding a random n-bits string requires a number of queries exponential in n, as the sender can only make a polynomial number of calls and cannot control the (somewhat) random outputs. This limits senders to leak up to a logarithmic number of bits, which is optimal[2].

Finally, as anyone can verify, senders are discouraged from posting invalid ciphertexts (e.g., unencrypted messages) – as in practice, content that does not verify would be taken down and there might be consequences for the sender. With this we obtain fast communication (no need of a sanitization bottleneck), while maintaining some accountability. Observe that public verification yields something different from a standard ACE, albeit very close. The difference is that not only the sanitization key is public (as in [14]), but the sanitization step (the verification in this case) can be performed by *any* party, after ciphertexts are posted. This was not the case in [14], where the sanitizer does more than just a routine check (in fact, it injects honestly generated randomness in ciphertexts, cf. [14, Section 3.1]).

2.2 Instantiating ACEnoS and VACE

2.2.1 Constructing ACEnoS Even assuming parties not to have shared randomness, it is not straightforward to obtain a ACEnoS by simply "removing" the sanitization step from pre-existing ACE constructions: the security of existing ACE schemes strongly relies on some transformation to be applied on a ciphertext *before* its delivery. In our work, we give several constructions under various standard assumptions that match in efficiency the existing ACE constructions (e.g., [12, 14]). One of these requires a new primitive, key-indistinguishable predicate encryption. The definition is rather natural, and very useful, as it immediately yields a solution of a problem left open by Kim and Wu [18] (see Appendix C).

[1] This is closely related to the notion of subliminal channels [25], where the information sent is hidden in messages that are seemingly created for a different purpose. In that language, NSWR says that, while a corrupt sender may be able to establish a subliminal channel to a receiver he should not send to, any such channel is non-secret.

[2] This reasoning yields a clear lower bound: no ACE scheme can prevent a sender to embed a logarithmic number of bits in a ciphertext (by generating ciphertexts until, say, the first few bits of the string are equal to the message bits she wants to embed) without sanitization.

2.2.2 Constructing VACE We give a construction of an ACE scheme with verification and minimal leakage based on a new definition of obfuscation. The need of a new assumption arises from the fact that building a VACE is highly non-trivial. To see why, we can consider what seems at first a promising solution: assume the sender is committed to a PRF key K and is supposed to compute the ciphertext c she posts using randomness generated from K and the encrypted message m, via the PRF, i.e., $c = E_{pk}(m, \mathsf{PRF}_K(m))$. In addition, the sender adds a non-interactive zero-knowledge proof that c was correctly computed. This allows verification. Moreover, it also seems to imply that a malicious sender cannot manipulate the randomness to embed a subliminal message m' in c. However a closer look shows that this is not clear at all: the intuition assumes that the sender chooses a message m to encrypt and the subliminal message m' first, then generates randomness using the PRF key and hopes that the resulting ciphertext will be an encoding m'. In fact, the sender does not have to do this: she might be able to instead compute simultaneously c and m from the subliminal message m', in a way that depends on K, such that $c = E_{pk}(m, \mathsf{PRF}_K(m))$ holds. The security properties of the PRF and the encryption function do not imply that this is infeasible: the PRF is only secure if the key is not known, and the encryption function is only hard to invert on a random ciphertext, and this does not prevent the adversary from generating c and m simultaneously from K. With this approach, it is completely unclear that c could not be an encoding of a subliminal message m' that the adversary wants. One might be able to make these problems go away if one is willing to model the PRF as a random oracle. But now the problem is that the zero-knowledge proof requires access to the code of the instantiation of the oracle. This code is no longer available once we pass to the random oracle model, so it is not clear how to prove security.

In the absence of a solution based on standard assumptions, we rely on a new model for security of obfuscation that we call Game-Specific Obfuscation (GSO). As the name suggests, GSO only requires the obfuscator to obfuscate programs in a specific family of programs $\mathcal{F}$ used in a fixed security game G. Roughly speaking, the security requirement is that the obfuscated program does not help an adversary to win the specific game any more than oracle access to the program would have allowed. Note that while implied by VBB, GSO makes a much weaker demand than VBB: we assume that the obfuscation gives nothing more than oracle access, only as far as winning G is concerned, and the obfuscator only needs to obfuscate programs in $\mathcal{F}$. In particular, the impossibility result for VBB [6] does not apply to GSO. At the same time, GSO and iO are somewhat incomparable: GSO has no specific requirement on the family of programs, while iO needs them to compute the same function; on the other hand, iO still guarantees indistinguishability for every game, while GSO targets a specific one. Nevertheless, assuming GSO is a strong assumption, and our result mainly serves to rule out impossibility results for VACE with minimal leakage.

2.3 Concurrent work

Recently, Lu et al. [21] explore an analogous question in the context of collusion-preserving MPC [3]: could one get rid of mediation? At a high level, their solution is similar to our VACE construction: parties' messages are encrypted, signed, and sent on an authenticated broadcast channel by a trusted hardware, which thus performs the same task as the obfuscated program in our construction. However, to completely prevent subliminal channels they have to assume that senders cannot run the trusted hardware multiple times and choose which ciphertext to send, which is a stronger assumption than our communication model[3].

2.4 Future Directions

We believe that the question we study here is a fundamental one that is of interest, also outside the scope of ACE, as it can be phrased in a much more general context: assume a polynomial-time sender who is limited to sending messages that satisfy some verification predicate. The question is to what extent we can use the verification to limit the bandwidth of any subliminal channel that the sender may be able to embed? Given our results, it seems that a logarithmic number of bits per message can be achieved. However, we leave a solution based on standard assumptions as an open problem.

[3] In a sense, this is akin to reverse firewalls [23].

3 Access Control Encryption without Sanitization

In this section we define Access Control Encryption without a sanitization step and show how to instantiate it. In addition to the protocol algorithms and security definitions, the model includes assumptions on parties, communication policy and communication channels. We remark that, while the restriction on communication channels is new, the assumptions on parties and communication policy are usually implicitly included in models for standard ACE. We explicitly state them to avoid possible misunderstandings of the model setup.

Let $[n] = 0, 1, \ldots, n$ for an integer $n \in \mathbb{N}$, and λ be the security parameter. Denote by $|s|$ the length of a bit string s.

Parties. The protocol is run by n parties P_i. Each party can be *either* a sender *or* a receiver. We denote by n_S (resp., n_R) the number of senders (resp., receivers); thus $n = n_S + n_R$.

Policy. A policy is a function $\mathcal{P} : [n_S] \times [n_R] \to \{0, 1\}$ defined as follows:

- $\mathcal{P}(i, j) = 1$ means that the i-th sender can send messages to the j-th receiver (i.e., R_j can decrypt ciphertexts generated by S_i);
- $\mathcal{P}(i, j) = 0$ means that the i-th sender cannot send messages to the j-th receiver (i.e., R_j cannot decrypt ciphertexts generated by S_i);

Finally, the party identity $i = 0$ represents a sender or receiver with no rights, i.e., for all $j \in [n_R]$, $k \in [n_S]$ it holds $\mathcal{P}(0, j) = \mathcal{P}(k, 0) = 0$.

Communication Model. We assume only one-way channels between parties:

- parties cannot share any randomness nor other key setup, and
- parties only communicate through a bulletin board, and do not have private channels, or, in general, communication channels outside the protocol (analogously to ACE). Senders are the only ones allowed to write on the bulletin board, while receivers have read-only access to it.

An Access Control Encryption scheme without sanitizer (denoted by ACEnoS in this work) is composed by four algorithms:

Setup: $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$
Takes as input the security parameter λ and the policy $\mathcal{P}$, and outputs the public parameters of the scheme (that include the message space $\mathcal{M}$) and the master secret key.

Key Generation: $k_i \leftarrow \mathsf{KGen}(pp, msk, i, t)$
Takes as input the public parameters of the scheme, the master secret key, the identity of the party, and a type $t \in \{\mathsf{sen}, \mathsf{rec}\}$, and outputs a key k_i, generated depending on t and i as follows:

- $ek_i \leftarrow \mathsf{KGen}(pp, msk, i, \mathsf{sen})$ is the encryption key for $i \in [n_S]$;
- $dk_i \leftarrow \mathsf{KGen}(pp, msk, i, \mathsf{rec})$ is the decryption key for $i \in [n_R]$;
- $ek_0 = dk_0 = pp$.

Encryption: $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$
On input the secret key of S_i and a message $m \in \mathcal{M}$, outputs the ciphertext.

Decryption: $m' \leftarrow \mathsf{Dec}(pp, dk_i, c)$
On input a ciphertext and the secret key of the receiver i, it outputs either a message or $\perp$ (representing a decryption failure).

As in the original scheme, an ACE without sanitizer has to satisfy:

Correctness: a honestly generated ciphertext can always be decrypted by the designated receivers.

No Read Rule: only the designated receiver can extract information about the plaintext from a ciphertext; senders anonymity is guaranteed under natural assumptions.

No *Secret* Write Rule: parties cannot communicate secretly if the policy does not allow it. If parties manage to communicate despite being forbidden by the policy, then anyone can read their communication.

When compared to the security definitions of ACE, only the No write Rule requires major changes, as it is the only property where the sanitizer plays a fundamental role. Correctness and the No Real Rule only need small adjustments.

Definition 3.1 (Correctness). *An ACE without sanitizer is correct if for all $m \in \mathcal{M}$, $i \in [n_S]$, $j \in [n_R]$ such that $\mathcal{P}(i,j) = 1$ it holds*

$$\mathsf{Pr}\left[\mathsf{Dec}(pp, dk_j, \mathsf{Enc}(pp, ek_i, m)) \neq m \;:\; \begin{array}{c}(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P}),\\ ek_i \leftarrow \mathsf{KGen}(pp, msk, i, \mathsf{sen}),\\ dk_j \leftarrow \mathsf{KGen}(pp, msk, j, \mathsf{rec})\end{array}\right] \leq \mathsf{negl}(\lambda)\ ,$$

where the probabilities are taken over the random coins of all the algorithms.

The NRR models the case in which a coalition of parties (both senders and receivers) tries to either break the confidentiality of a message (payload privacy) or to break the anonymity of target senders. We consider the most powerful adversary, that has even access to the target senders' encryption keys. This guarantees sender's anonymity (and payload privacy) even for senders whose encryption key was leaked.

Definition 3.2 (No-Read Rule). *Consider the following security experiment, where A is a stateful adversary and $b \in \{0,1\}$,*

Experiment $\mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})$	***Oracles***	
$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ $(m_0, m_1, i_0, i_1, st) \leftarrow \mathsf{A}^{\mathcal{O}_G(\cdot),\ \mathcal{O}_E(\cdot)}(pp)$ $c_b \leftarrow \mathsf{Enc}(pp, \mathcal{O}_G(i_b, \mathsf{sen}), m_b)$ $b' \leftarrow \mathsf{A}^{\mathcal{O}_G(\cdot),\ \mathcal{O}_E(\cdot)}(st, c_b)$ *Return* b'.	$\mathcal{O}_G(j,t)$: *If* $\exists\ k_j$ *s.t.* $(k_j, j, t) \in \mathcal{L}$, *return* k_j. *Else* $k_j \leftarrow \mathsf{KGen}(pp, msk, j, t)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(k_j, j, t)\}$ *Return* k_j.	$\mathcal{O}_E(j, m)$: $ek_j \leftarrow \mathcal{O}_G(j, \mathsf{sen})$ $c \leftarrow \mathsf{Enc}(pp, ek_j, m)$. *Return* c.

Given the following requirement,

Necessary Condition: $b = b'$, $|m_0| = |m_1|$, $i_0, i_1 \in [n_S]$,

we say that A wins the experiment if one of the following holds:

Payload Privacy (PP). *The Necessary Condition holds, and for all queries $q = (j, \mathsf{rec})$ to $\mathcal{O}_G$ it holds that:*

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0\ .$$

Sender Anonymity (SA). *The Necessary Condition holds, and for all queries $q = (j, \mathsf{rec})$ to $\mathcal{O}_G$ it holds that:*

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) \text{ and } m_0 = m_1\ .$$

An ACE without sanitizer satisfies the No-Read rule if for all PPT A, $b \xleftarrow{\$} \{0,1\}$

$$2 \cdot \left| \mathsf{Pr}\left[(\mathsf{PP} \ \vee\ \mathsf{SA}) \ :\ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)\ .$$

NRR vs. Indistinguishability. The NRR corresponds to the indistinguishability properties of the PKE, which in fact can be seen as special cases of the NRR: Payload Privacy when $i_0 = i_1$ guarantees IND-CPA security, while the Sender Anonymity case is analogous of key-indistinguishability [7].

The goal of the No Secret Write Rule is to prevent unauthorized communications. However, as the sanitization step is not present anymore, there is no countermeasure in place to prevent parties to try to establish subliminal channels [25]: parties might try to embed messages in the bits of a valid ciphertext using some shared randomness (for example, bits of their secret keys). As completely preventing subliminal channels without some kind of sanitization step is impossible (cf. Section 7.2), we settle for preventing *secure* exfiltration of information: if two parties manage to communicate despite this being against the policy, they can only succeed in establishing an insecure subliminal channel (i.e., they can only send unencrypted messages). This is useful in scenarios where leaking information by broadcasting it in the clear is not an option (e.g., if the information allows to identify the party that leaked it). Thus we need to assume that the corrupted sender and receiver *do not share randomness or private communication channels*. An obvious implication is that they cannot corrupt the same party and they should only communicate through the bulletin board. In fact, this imposes much bigger limitations to their corruption abilities:

– They cannot corrupt parties that have parts of the key in common (e.g., in constructions relying on symmetric key cryptography), as in this case the common bits can be used as shared randomness.
– They cannot corrupt parties whose keys can be recovered from each other (as it is the case for public key cryptography, where usually the public key can be recovered from the secret key).
– Neither of them can have both read and write access to the board, otherwise they would have an (insecure but) two-way communication channel that would then allow for key-exchange. This means that the corrupted sender can only corrupt other senders, and analogously for the receiver. Moreover, corrupted senders should not have access to an encryption oracle, while corrupted receivers do: the first requirement is due to the fact that a corrupt sender could trivially break the property by "replaying" encryptions under keys of (honest) senders who are allowed to communicate to the corrupted receiver, while the latter is due to the fact that we want to model that receivers have access to the entire bulletin board, which may contain encryptions of known messages under keys of known identities.

The definition says that if a corrupted receiver can recover a message, then knowing some decryption keys did not help in the process. This is modeled by imposing that a party B without access to keys can recover the message with a similar success probability[4]. Remark that there is no consistency check on the ciphertext $\bar{s}$ output by the corrupted sender A_1: $\bar{s}$ could even be the entire view of A_1. In Section 7.2 we show that adding ciphertext verifiability yields stronger limitation on the communication between unauthorized parties.

Definition 3.3 (No Secret Write (NSW) Rule). *Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be an adversary and consider the following game:*

Experiments		***Oracles***	
$\mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda,\mathcal{P})$	$\mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda,\mathcal{P})$		
$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ $(\bar{m}, \bar{s}) \leftarrow \mathsf{A}_1^{\mathcal{O}_G(\cdot,\mathsf{sen})}(pp)$ $m' \leftarrow \mathsf{A}_2^{\mathcal{O}_G(\cdot,\mathsf{rec}),\ \mathcal{O}_E(\cdot)}(pp, \bar{s})$ *Return* 1 *if* $\bar{m} = m'$, 0 *otherwise.*	$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ $(\bar{m}, \bar{s}) \leftarrow \mathsf{A}_1^{\mathcal{O}_G(\cdot,\mathsf{sen})}(pp)$ $m'' \leftarrow \mathsf{B}^{\mathcal{O}_E(\cdot)}(pp, \bar{s})$ *Return* 1 *if* $\bar{m} = m''$, 0 *otherwise.*	$\mathcal{O}_G(j,t)$: *If* $\exists\ k_j$ *s.t.* $(k_j, j, t) \in \mathcal{L}$, *return* k_j. *Else* $k_j \leftarrow \mathsf{KGen}(pp, msk, j, t)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(k_j, j, t)\}$ *Return* k_j.	$\mathcal{O}_E(j,m)$: $ek_j \leftarrow \mathcal{O}_G(j, \mathsf{sen})$ $c \leftarrow \mathsf{Enc}(pp, ek_j, m)$. *Return* c.

Let $\mathcal{Q}_1$ (resp., $\mathcal{Q}_2$) be the set of all queries $q = (i, \mathsf{sen})$ (resp., $q = (j, \mathsf{rec})$) that A_1 (resp.,A_2) issues to $\mathcal{O}_G$. The adversary wins the experiment if $m' = \bar{m}$ while the following holds:

No Communication Rule (NCR). $\forall\ (i, \mathsf{sen}) \in \mathcal{Q}_1,\ (j, \mathsf{rec}) \in \mathcal{Q}_2,\ \mathcal{P}(i, j) = 0$.

Given λ and a policy $\mathcal{P}$, an ACE without sanitizer satisfies the No Secret Write rule if for all PPT $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ there exists a PPT algorithm B and a negligible function negl such that

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda,\mathcal{P})\right] \geq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda,\mathcal{P}) \ \wedge\ \mathsf{NCR}\right] - \mathsf{negl}(\lambda)\ .$$

Proving NSWR security. The security definition essentially implies that the views of A_1 and A_2 are independent given the public parameters. In practice, proving the validity of this property means showing that B can run A_2 as a black-box by simulating its view without having access to $\mathcal{O}_G$ (essentially, B performs a man-in-the-middle attack on the adversary, only relaying the forged ciphertext). In fact, our constructions satisfy a stronger version of the property: in all proofs B simulates the oracles even *without exploiting its access to $\mathcal{O}_E$*.

Impossibility of ACEnoS *from Symmetric Key Primitives.* Instantiating a protocol only using symmetric key cryptographic primitives is often desirable, both because they are usually more efficient than public key cryptographic primitives, and because their post-quantum security is easier to assess. However, instantiating an ACE without sanitization solely from symmetric key primitives seems impossible. The intuition is that the symmetry of the keys does not limit their intended use: the same key can be used both for encryption and decryption independently of what it was intended for. In details, without additional assumptions there seem to be only three ways to distribute (symmetric) keys:

[4] Alternatively one could require A_2 (and consequently B) to distinguish whether a ciphertext contains a subliminal message at all. This case is clearly implied by ours.

– Receivers get just one key (different for each of them), and every sender is given the key of every receiver it is allowed to communicate with.
– Senders get just one key (different for each of them), and every receiver is given the key of all the senders it is allowed to receive message from.
– Senders have a different key for every receiver they are allowed to communicate with, and *same for the receivers* (i.e., there is a different key pair for each pair of parties allowed to communicate).

Obviously, the last option is not viable, as it does not preserve sender anonymity: receivers can extract the identity of the sender of a message by simply observing which key they use to decrypt the message. The same holds for the second. The first option does not guarantee payload privacy (NRR): as sender keys can be used to decrypt as well, the adversary can win the NRR experiment simply by decrypting the challenge ciphertext c_b twice, once with the key of sender i_0 and once with the key of sender i_1. A solution has to ensure that sender keys cannot be used to decrypt, and, vice versa, receiver keys cannot be used to encrypt.

Updatable Policies and Policy Confidentiality. In our model, we consider the simplest case of a public and static communication policy (i.e., all parties know the communication policy, and it is not possible to add a party, or to change the policy, after the key generation phase). However, adapting the model to include updatable policies or to impose some level of policy confidentiality seems straightforward. In the construction from predicate encryption one could actually update the policy by just re-issuing the decryption keys, while in the construction from PKE one would have to re-issue all the keys.

CCA Security. The security definitions only consider chosen-plaintext attacks, as the adversary is never given a decryption algorithm. The CCA version of our definitions can be derived from them along the lines of what Badertscher et al. [4] did for the security definitions of ACE with sanitizers. Our constructions can be adapted to satisfy these new definitions adding a signature on the ciphertext with independently chosen keys, and using a public key encryption scheme that is IND-CCA2 instead of IND-CPA secure.

Relay Attacks. One could imagine a scenario where A_1 and A_2 exploit a third party allowed to communicate with both to relay messages between them. As we do not allow parties to be senders and receivers at the same time, this is outside of our model. The reason behind ignoring relay attacks is that this vulnerability is inherent to the design of the communication policy, not to the scheme itself[5]. That being said, in a potential implementation of our constructions one should make sure that the policy is coherent with the model. In particular, care should be taken when dealing with parties that have to be both senders and receivers, which we call *relaying parties*. In cases when it is not possible to implement such parties so that the sender and receiver parts are well separated (meaning: access to the sender side does not imply access to the receiver, and vice versa), our model has to be used with care. One way to adapt it to such scenario could be to define a new policy $\mathcal{P}'$ from the original policy $\mathcal{P}$ by representing $\mathcal{P}$ as a directed graph, and define $\mathcal{P}'(i,j) = 1$ if there exists a path that connects S_i to R_j, independently of whether there is a relaying party in between them. This essentially assumes that any relaying party could be corrupted at any time. By analyzing the structure of $\mathcal{P}'$ one could then understand which kind of security guarantee our model would yield in such scenario.

Remark about NCR. It is important that the rule is enforced in the experiment as a winning condition and not at a oracle level. Otherwise A_1 and A_2 could agree on a strategy to check that they have access to the same oracle by querying the key generation oracle on two pairs (i, sen), (j, rec) that are not allowed to communicate by policy (assuming the policy is public). This would make it impossible to construct B from A_2 in a black-box way in the security proof, and it does not seem to be a requirement coming from use cases of the protocol.

[5] Such attack is not considered in normal ACE either, for the same reasons. Hence the remarks about implementation apply to the case of ACE constructions too.

4 Linear ACE without Sanitizer from PKE

The first construction is akin to the original linear ACE from standard assumptions by Damgård et al. [12]. In such scheme, senders are given the public keys of all the receivers they are allowed to communicate with, and "decoy/placeholder" public keys for the receivers they are not allowed to communicated with (to make sure that ciphertexts generated by different senders have the same length). The encryption algorithm then encrypts the message under all the keys. The i-th receiver just decrypts the i-th ciphertext using its secret key. Sender's anonymity requires that ciphertexts do not leak any information about the key used to generate them, i.e., key indistinguishability [7].

Let $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption scheme, that is IND-CPA secure[6] and IK-CPA (Theorem A.3). An ACE without sanitizer from PKE (denoted $\mathsf{ACE}^{\mathsf{pke}}$) can be instantiated as follow.

Communication Model: parties communicate through a bulletin board. Only senders are allowed to write on the board. Receivers can only read from it.

Message Space: $\mathcal{M} := \{0,1\}^{\ell}$.

Setup: $(pp, msk) \leftarrow \mathsf{Setup}(1^{\lambda}, \mathcal{P})$

It generates the message set, and the number of parties n, of senders n_S, and of receivers n_R; all are included in pp, along with the policy. The master secret key is a list of $2n_R$ (distinct) pairs of asymmetric keys, i.e.,

$$msk = \left\{ ((pk_j^0, sk_j^0), (pk_j^1, sk_j^1)) \ : \ \begin{array}{l} \text{For } i = 1,2 \\ (pk_j^i, sk_j^i) \leftarrow \mathsf{PKE.KeyGen}(1^{\lambda}) \end{array} \right\}_{j \in [n_R]}.$$

Key Generation: $k_i \leftarrow \mathsf{KGen}(pp, msk, i, t)$

On input (i, t), the algorithm parses $msk = \{((pk_j^0, sk_j^0), (pk_j^1, sk_j^1))\}_j$, and behaves as it follows.

- If $i \neq 0$ and $t = \mathsf{sen}$, it returns a vector $ek_i = (ek_i[j])_{j \in [n_R]}$ such that $ek_i[j] \leftarrow pk_j^{\mathcal{P}(i,j)}$.
- If $i \neq 0$ and $t = \mathsf{rec}$, it returns $dk_i = sk_i^1$.
- If $i = 0$, returns $ek_0 = dk_0 = pp$.

Encryption: $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$

Run $c_j \leftarrow \mathsf{PKE.Enc}(ek_i[j], m; \rho_j)$ for all $j \in [n_R]$ (ρ_j is a random string). Return $c = (c_j)_{j=1,\ldots,n}$.

Decryption: $m' \leftarrow \mathsf{Dec}(pp, dk_j, c)$

Let $c = (c_1, \ldots, c_{n_R})$. Return the output of $\mathsf{PKE.Dec}(dk_j, c_j)$ (which could be either a message m or $\perp$).

Efficiency, Storage Requirements, and Optimizations. Both the encryption and the decryption runtimes are linear in the number of receivers n_R (in the case of the decryption algorithm, this is an upperbound). The ciphertext length is $\mathrm{O}(n_R)$. Senders have to store n_R public keys, while receivers only have to store one secret key. The senders memory requirement can be optimized by getting rid of the decoy keys, and instead giving to the sender only the public keys of the receivers they are allowed to communicate with. The encryption would then generate a random ciphertext (e.g., an encryption of the message under a random public key) for each missing receiver key. Security follows as long as the probability of collision with a receiver key when selecting a random public key is negligible. Finally, remark that if we drop the Sender Anonymity requirement, we can construct a compact (polylog) ACE already from public key encryption. Indeed, in this case each sender can have just one encryption key, whose corresponding decryption key can be given to every receiver it is allowed to communicate with. Hence each sender sends only one ciphertext. The SA is lost because by seeing which key is used to decrypt, the receiver can infer the identity of the sender. It remains an open question whether PKE is the minimal assumption to have ACE without sanitization and SA.

[6] It is enough that the PKE is IND-CPA, as whenever the receiver has to distinguish between the encryption of 2 different messages, it is not allowed to get the decryption key (as it would be in the Payload privacy game). In the sender anonymity game, when the adversary can ask for decryption keys, the only requirement is that is should be impossible to identify a sender from the encryption key it uses, which is guaranteed by the key-indistinguishability property.

Policy Confidentiality. In its current formulation, the scheme already can be used to enforce policy confidentiality (with the minor modification of removing the policy from the public parameters of the scheme). This cannot be done when using the optimized version proposed in the previous paragraph.

Theorem 4.1. *The* $\mathsf{ACE}^{\mathsf{pke}}$ *scheme is correct, and satisfies the properties of No-Read and No Secret Write as described in Section 3 if the public key encryption scheme is IND-CPA secure and key-indistinguishable.*

Proof. The proof is as follows.

Correctness. Correctness directly follows from the correctness of the PKE scheme.

No Read Rule. The No-Read Rule relies on both the IK-CPA and IND-CPA properties of the PKE scheme. The proof has a similar structure as the proof of Theorem 3 in [12], i.e., relies on a sequence of indistinguishable hybrid games. In the odd hybrid games, a ciphertext in the challenge ciphertext is changed from an encryption of m_0 under ek_{i_0} to an encryption of m_0 under ek_{i_1}. In the even hybrid games, the encryption of m_0 under ek_{i_1} is substituted with an encryption of m_1 under ek_{i_1}.

Game 0. This is the No-Read game with $b = 0$.

Hybrid $2k$ **for** $0 \leq k \leq n_R$**.** Replace the encryption algorithm used to generate the challenge ciphertext with $\mathsf{Enc}_k^*(ek_{i_0}, ek_{i_1}, m_0, m_1)$ which first encrypts m_0 and m_1 under ek_{i_0} and ek_{i_1} respectively to get:

$$c^0 = (c_1^0, \ldots, c_{n_R}^0) \leftarrow \mathsf{Enc}(pp, ek_{i_0}, m_0)$$
$$c^1 = (c_1^1, \ldots, c_{n_R}^1) \leftarrow \mathsf{Enc}(pp, ek_{i_1}, m_1)$$

Next, the algorithm outputs the following challenge ciphertext:

$$\bar{c}^{2k} = (c_1^1, \ldots, c_k^1, c_{k+1}^0, \ldots, c_{n_R}^0) \ .$$

Everything else is as in Game $2k - 1$.

Hybrid $2k + 1$ **for** $0 \leq k \leq n_R - 1$**.** Replace the encryption algorithm used to generate the challenge ciphertext with $\mathsf{Enc}_k^*(ek_{i_0}, ek_{i_1}, m_0, m_1)$ which first encrypts m_0 under both ek_{i_0} and ek_{i_1}, and m_1 under $ek_{i,1}$ to get:

$$c^0 = (c_1^0, \ldots, c_{n_R}^0) \leftarrow \mathsf{Enc}(pp, ek_{i_0}, m_0)$$
$$c^H = (c_1^H, \ldots, c_{n_R}^H) \leftarrow \mathsf{Enc}(pp, ek_{i_1}, m_0)$$
$$c^1 = (c_1^1, \ldots, c_{n_R}^1) \leftarrow \mathsf{Enc}(pp, ek_{i_1}, m_1) \ .$$

Then the algorithm outputs the following challenge ciphertext:

$$\bar{c}^{2k+1} = (c_1^H, c_2^0, \ldots, c_{n_R}^0) \ \text{ if } k = 0$$
$$\bar{c}^{2k+1} = (c_1^1, \ldots, c_k^1, c_{k+1}^H, c_{k+2}^0, \ldots, c_{n_R}^0) \ \text{ if } k > 0 \ .$$

Everything else is as in Game $2k$.

Game 1. This is the No-Read game with $b = 1$.

Let $\mathcal{Q}_G$ be the set of all queries $q = (j, t)$ that the adversary issues to the oracle $\mathcal{O}_G$, and J be the subset of $\mathcal{Q}_G$ composed by all the queries $q = (j, \mathsf{rec})$.

Payload Privacy. This case requires that for all queries $q = (j, \mathsf{rec}) \in J$ it holds that:

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0 \ .$$

This implies that the adversary cannot get the secret keys corresponding to the public keys contained in ek_{i_0} and ek_{i_1}.

We now prove that the games and their hybrid versions are indistinguishable.

Game $0 \approx$ **Hybrid** 0**.** In Game 0, the challenge ciphertext is generated as $c_0 \leftarrow \mathsf{Enc}(pp, ek_{i_0}, m_0)$. In Hybrid 0 the challenge ciphertext $\bar{c}^0$ is set to be exactly $c^0 = (c_1^0, \ldots, c_n^0) \leftarrow \mathsf{Enc}(pp, ek_{i_0}, m_0)$. Hence the games are identical.

Hybrid $2k \approx$ **Hybrid** $2k+1$ $(0 \leq k \leq n_R - 1)$**.** The only difference between the games is that in Hybrid $2k$ the $(k+1)$-th component of the challenge ciphertext is

$$c_{k+1}^0 \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^1, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_0, k+1) = 1 \ ,$$
$$c_{k+1}^0 \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^0, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_0, k+1) = 0 \ ,$$

while in Hybrid $2k+1$ is

$$c_{k+1}^H \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^1, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k+1) = 1 \ ,$$
$$c_{k+1}^H \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^0, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k+1) = 0 \ .$$

We can have 4 cases:

1. $\mathcal{P}(i_0, k+1) = \mathcal{P}(i_1, k+1) = 0$,
2. $\mathcal{P}(i_0, k+1) = 0 \ \wedge \ \mathcal{P}(i_1, k+1) = 1$,
3. $\mathcal{P}(i_0, k+1) = 1 \ \wedge \ \mathcal{P}(i_1, k+1) = 0$,
4. $\mathcal{P}(i_0, k+1) = \mathcal{P}(i_1, k+1) = 1$.

Due to the Payload Privacy condition, the adversary is allowed to query the decryption key dk_{k+1} only in the first case. In that case, the two ciphertexts c_{k+1}^0 and c_{k+1}^H are generated the same way as encryptions of m_0 under pk_{k+1}^0, thus the games are exactly equal.

In case 2 and 3, the adversary does not have the decryption key dk_{k+1}. In these cases it is possible to construct a PPT algorithm B that wins the IK-CPA experiment running A as a black-box. We show the reduction for case 2; case 3 is analogous. B simulates the No Read game as follows:

- B sets the public key pk_{k+1}^1 of the $(k+1)$-th receiver and the $(k+1)$-th decoy key pk_{k+1}^0 to be the challenge keys pk_1 and pk_0 from the IK-CPA experiment, respectively.
- When it is time to generate the challenge ciphertext $\bar{c}$, B sends m_0 as challenge plaintext to the IK-CPA challenger, and sets $\bar{c}_{k+1}$ to be the challenge ciphertext from the IK-CPA game.
- Everything else is done honestly according to the game specifications in Theorem 3.2.

B outputs 0 if A returns $2k$, 1 otherwise. If A wins the No Read game, then B wins the IK-CPA experiment with the same probability. Remark that the adversary cannot ask dk_{k+1}, thus it is not an issue that B has no access to the secret keys corresponding to pk_0 and pk_1.

In case 4 both ek_{i_0} and ek_{i_1} contain the public key pk_{k+1}^1 of the $(k+1)$-th receiver, thus both $(k+1)$-th ciphertexts in $\bar{c}_{2k}$ and in $\bar{c}_{2k+1}$ are encryptions of m_0 under pk_{k+1} and the games are identical.

Hybrid $2(k-1)+1 \approx$ **Hybrid** $2k$ $(1 \leq k \leq n_R)$**.** The only difference between the games is that in Hybrid $2(k-1)+1$ the k-th component of the challenge ciphertext is

$$c_k^H \leftarrow \mathsf{PKE.Enc}(pk_k^1, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k) = 1 \ ,$$
$$c_k^H \leftarrow \mathsf{PKE.Enc}(pk_k^0, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k) = 0 \ ,$$

while in Hybrid $2k$ is

$$c_k^1 \leftarrow \mathsf{PKE.Enc}(pk_k^1, m_1; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k) = 1 \ ,$$
$$c_k^1 \leftarrow \mathsf{PKE.Enc}(pk_k^0, m_1; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k) = 0 \ ,$$

We can have 4 cases:

1. $\mathcal{P}(i_0, k) = \mathcal{P}(i_1, k) = 0$,
2. $\mathcal{P}(i_0, k) = 0 \ \wedge \ \mathcal{P}(i_1, k) = 1$,
3. $\mathcal{P}(i_0, k) = 1 \ \wedge \ \mathcal{P}(i_1, k) = 0$,
4. $\mathcal{P}(i_0, k) = \mathcal{P}(i_1, k) = 1$.

Due to the Payload Privacy condition, the adversary is allowed to query the decryption key dk_k only in the first case. However, in that case, the two ciphertexts c_k^H and c_k^1 are encryptions of m_0 and m_1 respectively under the key pk_k^0 which is independent from $dk_k = sk_k^1$. Therefore, distinguishing the games implies breaking the IND-CPA property of the PKE scheme. Indeed, assume that A could distinguish the two games. Then an adversary C for the IND-CPA experiment could win the experiment running A as a black-box and simulating the NRR hybrid game as follows. C simulates the hybrid game according to the specifications, setting pk_k^0 to be the public key pk given as challenge in the IND-CPA game. It uses m_0 and m_1 as challenge messages in the IND-CPA game, and sets the k-th ciphertext in $\bar{c}$ to be the challenge ciphertext from IND-CPA game. The algorithm C outputs 1 if the distinguisher returns $2k$, 0 otherwise. The simulated hybrid game is perfectly indistinguishable from the real one, and C wins with the same success probability as the distinguisher (and runs in polynomial-time).
Case 2 and 3 are similar to the previous. Now the adversary does not have the decryption key dk_k, and the challenge ciphertext is such that $c_k^H \leftarrow \mathsf{PKE.Enc}(pk_k^0, m_0; \rho_k)$ and $c_k^1 \leftarrow \mathsf{PKE.Enc}(pk_k^1, m_1; \rho_k)$ (this is for case 2, case 3 is analogous). We can prove indistinguishability by adding an additional game hop in between Hybrid $2(k-1)+1$ and Hybrid $2k$ where the ciphertext c_k^H is set to be $\mathsf{PKE.Enc}(pk_k^1, m_0; \rho_{k+1})$. Remark that the adversary cannot query the decryption key $dk_k = sk_k^1$ associated with pk_k^1. Thus, distinguishing this from Hybrid $2(k-1)+1$ games implies breaking the IND-CPA property of the PKE scheme, and distinguishing it from Hybrid $2k$ implies breaking IK-CPA.
In case 4 the adversary still does not have the decryption key dk_k, but this time the challenge ciphertext contains either $c_k^H \leftarrow \mathsf{PKE.Enc}(pk_k^1, m_0; \rho_k)$ or $c_k^1 \leftarrow \mathsf{PKE.Enc}(pk_k^1, m_1; \rho_k)$. Thus, it is possible to construct a PPT algorithm B that wins the IND-CPA experiment running A as a black-box. B simulates the No Read game as follows:

- B sets pk_k^1 to be the challenge encryption key from the IND-CPA game. As A is not allowed to query for dk_k, this is indistinguishable from the standard game.
- When it is time to generate the challenge ciphertext $\bar{c}$, B sends m_1 and m_0 as challenge messages to the IND-CPA challenger, and sets $\bar{c}_k$ to be the challenge ciphertext from the game.
- Everything else is done honestly according to the game specifications in Theorem 3.2.

B outputs 1 if A returns $2k$, 0 otherwise. If A wins the No Read game, then B wins the IND-CPA experiment with the same probability.

Hybrid $2n_R \approx$ Game 1. In Game 1, the challenge ciphertext is generated as $c_1 \leftarrow \mathsf{Enc}(pp, ek_{i_1}, m_1)$. In Hybrid $2n_R$ the challenge ciphertext $\bar{c}^{2n}$ is set to be exactly the output of $\mathsf{Enc}(pp, ek_{i_1}, m_1)$. Thus the games are identical.

Sender Anonymity. This case requires that for all queries $q = (j, \mathsf{rec}) \in J$ it holds that:

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) \text{ and } m_0 = m_1 \ .$$

The indistinguishability of the games can be proved as follows.

Game $0 \approx$ Hybrid 0. As in the Payload Privacy case these games are identical.

Hybrid $2k \approx$ Hybrid $2k+1$ $(0 \le k \le n_R - 1)$. As before, the difference between the two games is how the $(k+1)$-th component of the challenge ciphertext is generated: in Hybrid $2k$ is

$$c_{k+1}^0 \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^1, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_0, k+1) = 1 \ ,$$
$$c_{k+1}^0 \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^0, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_0, k+1) = 0 \ ,$$

while in Hybrid $2k+1$ is

$$c_{k+1}^H \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^1, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k+1) = 1 \ ,$$
$$c_{k+1}^H \leftarrow \mathsf{PKE.Enc}(pk_{k+1}^0, m_0; \rho_{k+1}) \text{ if } \mathcal{P}(i_1, k+1) = 0 \ .$$

Again, we can have 4 cases:

1. $\mathcal{P}(i_0, k+1) = \mathcal{P}(i_1, k+1) = 0$,

2. $\mathcal{P}(i_0, k+1) = 0 \ \wedge \ \mathcal{P}(i_1, k+1) = 1$,
3. $\mathcal{P}(i_0, k+1) = 1 \ \wedge \ \mathcal{P}(i_1, k+1) = 0$,
4. $\mathcal{P}(i_0, k+1) = \mathcal{P}(i_1, k+1) = 1$.

From the Sender Anonymity condition we have that A can ask for the decryption key dk_{k+1} only in cases 1 or 4. In case 1 the two ciphertexts are both encryptions of m_0 under pk^0_{k+1}, hence the two games are equal. In case 4, the two ciphertexts are encryptions of the same message m_0 under the same key pk^1_{k+1}, thus again they are the same game. In case 2 and 3, the adversary does not have the decryption key dk_{k+1}, hence a successful distinguisher can be used to break the IK-CPA property of the PKE scheme as in the Payload Privacy case.

Hybrid $2(k-1)+1 \approx$ **Hybrid** $2k$ **(**$1 \leq k \leq n_R$**).** As $m_0 = m_1$ by the SA condition, these games are in fact equal.

Hybrid $n \approx$ **Game** 1**.** This can be proved using the same argument as in the Payload Privacy case.

No Secret Write Rule. Given an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ that wins the game $\mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P})$ with probability ϵ_{A}, to prove that the scheme satisfies the NSW Rule we need to construct an algorithm B that wins game $\mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})$ with essentially the same probability (up to a negligible difference). Upon receiving $\bar{s}$ from A_1, the algorithm B runs A_2 internally on input $(pp, \bar{s})$ simulating the oracles as follows. First, it generates $2n_R$ pairs of PKE keys, and collects them in a fresh master secret key $\bar{msk} = \{((\bar{pk}^0_j, \bar{sk}^0_j), (\bar{pk}^1_j, \bar{sk}^1_j))\}_{j \in [n_R]}$. The oracles are then simulated using msk':

$\mathcal{O}_G$: on input (j, t) from A_2, B generates new keys according to the policy $\mathcal{P}$ using $\bar{msk}$. If $t = \mathsf{sen}$ B sends $ek_j = (\bar{pk}_i^{\mathcal{P}(j,i)})_{i \in [n_R]}$; if $t = \mathsf{rec}$, it sends $dk_j = \bar{sk}^1_j$. It stores the keys in a list $\mathcal{K}$.

$\mathcal{O}_E$: simulates the encryption oracle as specified in the security experiment using the appropriate key from $\mathcal{K}$.

Finally, B outputs the message that A_2 returns. By the definition of conditional probability, the success probability of B is

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P}) \ \wedge \ \mathsf{NCR}\right] \cdot \mathsf{Pr}(\mathbf{E}) \ ,$$

where we denote by $\mathbf{E}$ the event "A_2 does not distinguish the simulated oracles from real ones". The only way A_2 could distinguish, is if the answers of the simulated oracles were inconsistent with the challenge ciphertext $\bar{s}$. However, in the real game the encryption keys queried by A_1 are statistically independent of the decryption keys queried by A_2, and they do not share state, thus any information encoded in $\bar{s}$ is statistically independent of the keys A_2 queries. The only way A_2 could get information about the encryption keys owned by A_1 would be by querying the encryption oracle on (i, m) for an i that A_1 has corrupted (such identity could be hardcoded in A_2, so the attack can be performed even in absence of shared state). If A_2 can distinguish that the ciphertext is not generated using the same key that A_1 received, then A can be exploited to break the key indistinguishability property of the PKE. Let q_E be the number of queries by A_2 to the encryption oracle. One can prove this by a sequence of hybrid games:

Game 0**.** This is the No-Secret-Write experiment.

Hybrid k **for** $0 \leq k \leq 2n_R$**.** In all hybrid games the view of A_1 is generated according to the NSWR experiment, i.e., using the master secret key msk generated at the beginning of the experiment. However, when generating the view of A_2, the challenger in Hybrid k generates the j-th key pairs in $\bar{msk}$ as follows:

Case $j < \lfloor k/2 \rfloor$**:** it generates fresh PKE key pairs $(\bar{pk}^0_j, \bar{sk}^0_j, \bar{pk}^1_j, \bar{sk}^1_j)$.

Case $j = k$**:** it generates a fresh key pair $(\bar{pk}, \bar{sk})$, and sets the j-th PKE pairs to be $(\bar{pk}, \bar{sk}, pk^1_j, sk^1_j)$ if k is odd and $(\bar{pk}^0_j, \bar{sk}^0_j, \bar{pk}, \bar{sk})$ if k is even, where $(pk^0_j, sk^0_j, pk^1_j, sk^1_j)$ is the j-th key pair in msk and $(\bar{pk}^0_j, \bar{sk}^0_j,) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.

Case $j > \lfloor k/2 \rfloor$**:** it uses the same PKE key pairs $(pk^0_j, sk^0_j, pk^1_j, sk^1_j)$ as in msk.

Let E_k be the event that A distinguishes between Hybrid k and Hybrid $k-1$. Theorem 4.2 shows that $\mathsf{Pr}(\mathrm{E}_k) \leq \frac{2}{3} q_E \epsilon_{\mathsf{ik\text{-}cpa}}(\lambda)$.

Game 1. This is the No-Secret-Write experiment as simulated by B. By definition, Hybrid 0 is exactly equal to Game 0, and Hybrid $2n_R$ is the same as Game 1. Therefore:

$$\mathsf{Pr}(\mathbf{E}) \geq 1 - 3n_R q_E \epsilon_{\mathsf{ik\text{-}cpa}}(\lambda) \ ,$$

where $\epsilon_{\mathsf{ik\text{-}cpa}}(\lambda)$ is the probability of breaking the IK-CPA property of the PKE scheme and $q_E = poly(\lambda)$ as A is a polynomial-time algorithm.

□

Lemma 4.2. $\mathsf{Pr}(E_k) \leq 3\epsilon_{ik\text{-}cpa}(\lambda) q_E$ *for all* $k \in [2n_R]$.

Proof. We split the proof in 3 cases:

Case 1: for all queries (j, sen) by A_1 to $\mathcal{O}_G$, $\mathcal{P}(j,k) = 0$.
Case 2: there are $\bar{i}, \bar{j} \in [n_S]$ such that A_1 queried $(\bar{i}, \mathsf{sen})$ and $(\bar{j}, \mathsf{sen})$ to $\mathcal{O}_G$ and $\mathcal{P}(\bar{i}, k) = 1$, $\mathcal{P}(\bar{j}, k) = 0$.
Case 3: for all queries (j, sen) by A_1 to $\mathcal{O}_G$, $\mathcal{P}(j,k) = 1$.

If $k = 0$ this is exactly the NSW experiment. If $k = 2n_R$ this is the NSW experiment as simulated by B. Assume now $0 < k < 2n_R$.

Let us start from k odd. In Case 1 A_1 sees pk_k^0 but not pk_k^1, while A_2 can query dk_k and receives sk_k^1 both in Hybrid k and in Hybrid $k-1$. The only difference is that in Hybrid k $\mathcal{O}_E$ uses $p\bar{k}_k, pk_k^1$ instead of pk_k^0, pk_k^1 as in Hybrid $k-1$. If A can distinguish in this case, we construct a PPT algorithm C that can win the IK-CPA experiment running A as a subroutine. C receives pk_0 and pk_1 from the IK-CPA experiment and generates msk setting $(pk_k^0, sk_k^0) = (pk_0, \perp)$ and $(p\bar{k}, s\bar{k}) = (pk_1, \perp)$. The rest of the master secret keys msk and $m\bar{s}k$ are generated as specified by Hybrid k. Then it answers to $\mathcal{O}_G$ using msk for the queries by A_1 and $m\bar{s}k$ for the queries by A_2. To answer queries from A_2 to $\mathcal{O}_E$, C selects a random $q \xleftarrow{\$} [q_E]$ and behaves as follows:

- C answers to the first $q-1$ queries using pk_0, pk_k^1 as the k-th encryption keys.
- When A_2 sends the q-th query (i, m), C returns m to the IK-CPA experiment and receives a challenge ciphertext $\bar{c}$. Then it generates the encryption of m as follows:

$$\begin{aligned}
&c_j \leftarrow \mathsf{PKE.Enc}(pk_j^{\mathcal{P}(i,j)}, m) \quad \text{for } j = 1, \ldots, k-1 \\
&c_j \leftarrow \bar{c} \quad \text{for } j = k \text{ if } \mathcal{P}(i,k) = 0 \\
&c_j \leftarrow \mathsf{PKE.Enc}(pk_j^1, m) \quad \text{for } j = k \text{ if } \mathcal{P}(i,k) = 1 \\
&c_j \leftarrow \mathsf{PKE.Enc}(\bar{pk}_j^{\mathcal{P}(i,j)}, m) \quad \text{for } j = k+1, \ldots, n_R
\end{aligned}$$

- C answers to the remaining $q_E - q + 1$ queries using pk_1, pk_k^1 as the k-th encryption keys.

Thus it follows that for k odd

$$\begin{aligned}
\mathsf{Pr}(\mathrm{E}_k \mid \text{Case 1}) &\leq |\mathsf{Pr}(\mathsf{A} \text{ wins Hybrid } k-1 \mid \text{Case 1}) - \mathsf{Pr}(\mathsf{A} \text{ wins Hybrid } k \mid \text{Case 1})| \\
&\leq |\sum_{Q=1}^{q_E} \mathsf{Pr}(\mathsf{A} \text{ wins the game } \mid q = Q \ \wedge \ \bar{c} \leftarrow \mathsf{PKE.Enc}(pk_0, m)) + \\
&\quad - \mathsf{Pr}(\mathsf{A} \text{ wins the game } \mid q = Q \ \wedge \ \bar{c} \leftarrow \mathsf{PKE.Enc}(pk_1, m))| \\
&\leq q_E \epsilon_{\mathsf{ik\text{-}cpa}}(\lambda) \ .
\end{aligned}$$

In Case 2 A_1 sees both pk_k^0 and pk_k^1, while A_2 cannot query dk_k both in Hybrid k and in Hybrid $k-1$. The difference is that in Hybrid k $\mathcal{O}_E$ uses $p\bar{k}_k, pk_k^1$ instead of pk_k^0, pk_k^1 as in Hybrid $k-1$. The reduction shown for Case 1 can be replicated without changes in this case. In Case 3 A_1 sees pk_k^1 but not pk_k^0, while A_2 cannot query dk_k both in Hybrid k and in Hybrid $k-1$. Again the only difference is that in Hybrid k $\mathcal{O}_E$ uses $p\bar{k}_k, pk_k^1$ instead of pk_k^0, pk_k^1 as in Hybrid $k-1$. Thus in this case the view of A in Hybrid k is statistically indistinguishable from the view of A in Hybrid $k-1$.

Finally, assume that k is even. In Case 1 A_1 sees pk_k^0 but not pk_k^1, while A_2 can query dk_k and receives $\bar{sk}_k^1$ in Hybrid k and sk_k^1 in Hybrid $k-1$. The encryption oracle $\mathcal{O}_E$ uses $\bar{pk}_k{}^0, \bar{pk}$ in Hybrid k, and $\bar{pk}_k^0, pk_k^1$ in Hybrid $k-1$. As the adversary does not see pk_k^1, the view of A in Hybrid k is statistically indistinguishable by the view of A in Hybrid $k-1$. In Case 3 A_1 sees pk_k^1 but not pk_k^0, while A_2 cannot query dk_k both in Hybrid k and in Hybrid $k-1$. The difference is that in Hybrid k $\mathcal{O}_E$ uses $\bar{pk}_k{}^0, \bar{pk}$ instead of $\bar{pk}_k^0, pk_k^1$ as in Hybrid $k-1$. The previous reduction can be adapted to this case as follows. C receives pk_0 and pk_1 from the IK-CPA experiment and generates msk setting $(pk_k^1, sk_k^1) = (pk_0, \perp)$ and $(\bar{pk}, \bar{sk}) = (pk_1, \perp)$. The rest of the master secret keys msk and $\bar{msk}$ are generated as specified by Hybrid k. Then it answers to $\mathcal{O}_G$ using msk for the queries by A_1 and $\bar{msk}$ for the queries by A_2. To answer queries from A_2 to $\mathcal{O}_E$, C selects a random $q \xleftarrow{\$} [q_E]$ and behaves as follows:

- C answers to the first $q-1$ queries using pk_0, pk_k^1 as the k-th encryption keys.
- When A_2 sends the q-th query (i, m), C returns m to the IK-CPA experiment and receives a challenge ciphertext $\bar{c}$. Then it generates the encryption of m as follows:

$$
\begin{aligned}
c_j &\leftarrow \mathsf{PKE.Enc}(pk_j^{\mathcal{P}(i,j)}, m) \quad \text{for } j = 1, \ldots, k-1 \\
c_j &\leftarrow \mathsf{PKE.Enc}(\bar{pk}_j^0, m) \quad \text{for } j = k \text{ if } \mathcal{P}(i,k) = 0 \\
c_j &\leftarrow \quad \bar{c} \quad \text{for } j = k \text{ if } \mathcal{P}(i,k) = 1 \\
c_j &\leftarrow \mathsf{PKE.Enc}(\bar{pk}_j^{\mathcal{P}(i,j)}, m) \quad \text{for } j = k+1, \ldots, n_R
\end{aligned}
$$

- C answers to the remaining $q_E - q + 1$ queries using pk_1, pk_k^1 as the k-th encryption keys.

Analogously to the case of k odd, for k even it holds that

$$\mathsf{Pr}(\mathrm{E}_k \mid \text{Case 3}) \leq q_E \epsilon_{\mathsf{ik\text{-}cpa}}(\lambda) \ .$$

Finally, in Case 2 A_1 sees both pk_k^0 and pk_k^1, while A_2 cannot query dk_k both in Hybrid k and in Hybrid $k-1$. The difference is again that in Hybrid k $\mathcal{O}_E$ uses $\bar{pk}_k{}^0, \bar{pk}$ instead of $\bar{pk}_k^0, pk_k^1$ as in Hybrid $k-1$. The reduction shown for Case 1 can be replicated without changes in this case. Therefore, for all k it holds that

$$\mathsf{Pr}(\mathrm{E}_k) = \sum_{i=1}^{3} \mathsf{Pr}(\text{Case } i)\mathsf{Pr}(\mathrm{E}_k \mid \text{Case } i) \leq 3 q_E \epsilon_{\mathsf{ik\text{-}cpa}}(\lambda) \ .$$

□

5 Compact ACE from Hybrid Encryption

The previous construction has the problem that the length of ciphertexts depends linearly on $\ell \cdot n_R$. This can be improved using a hybrid encryption technique: combining $\mathsf{ACE}^{\mathsf{pke}}$ with a rate-1 symmetric key encryption (SKE) scheme yields a more compact ACE (denoted by $\mathsf{ACE}^{\mathsf{he}}$), which outputs ciphertexts whose size scales with $\ell + n_R$ instead. Interestingly, there is no known analogous hybrid encryption version of the original construction of [12].

The idea is to encrypt each message m with the SKE using a one-time secret key sk, which is then encrypted using $\mathsf{ACE}^{\mathsf{pke}}$. Thus a ciphertext has two components, the SKE encryption of the message and the $\mathsf{ACE}^{\mathsf{pke}}$ encryption of the one-time secret key. The NRR security follows easily combining the NRR of $\mathsf{ACE}^{\mathsf{pke}}$ with the $\mathsf{lor\text{-}cpa}$ security of the SKE: intuitively, the Payload Privacy of the $\mathsf{ACE}^{\mathsf{pke}}$ implies that no information about sk is leaked by the ciphertext, thus sk can be used as encryption key and the $\mathsf{lor\text{-}cpa}$ security of SKE is preserved. The intuition behind the NSWR is the following: One can see the ciphertext as an $\mathsf{ACE}^{\mathsf{pke}}$ ciphertext padded with some additional bits (the SKE ciphertext); this is possible because there is no restriction on the bit length of the challenge ciphertext $\bar{s}$ in the definition of NSWR. Thus extracting a

subliminal message from a ciphertext generated with the compact ACE cannot be easier than extracting it from a ciphertext generated with $\mathsf{ACE}^{\mathsf{pke}}$. Interestingly enough, a formal proof of this intuition requires some non-trivial steps, and relies on both the NSWR and the NRR-PP security of the $\mathsf{ACE}^{\mathsf{pke}}$. Finally, observe that the construction can be instantiated with any ACE without sanitizer that satisfies the definitions in Section 3, not just with $\mathsf{ACE}^{\mathsf{pke}}$.

Let $(\mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a rate-1 symmetric encryption scheme that is lor-cpa secure, and let $\mathsf{ACEnoS} = (\mathsf{ACE.Setup}, \mathsf{ACE.KGen}, \mathsf{ACE.Enc}, \mathsf{ACE.Dec})$ be an ACE without sanitizer that is NRR and NSWR secure.

Communication Model: parties communicate through a bulletin board; senders and receivers have write-only and read-only access respectively.
Message Space: $\mathcal{M} := \{0,1\}^\ell$.
Setup: $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$
Return $(pp, msk) \leftarrow \mathsf{ACE.Setup}(1^\lambda, \mathcal{P})$.
Key Generation: $k_i \leftarrow \mathsf{KGen}(pp, msk, i, t)$
Return $k_i \leftarrow \mathsf{ACE.KGen}(pp, msk, i, t)$.
Encryption: $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$
Generate a one-time secret key $sk \leftarrow \mathsf{SE.KeyGen}(1^\lambda)$, and encrypt the message using it: $c_1 \leftarrow \mathsf{SE.Enc}_{sk}(m)$. Then encrypt the key using the ACEnoS: $c_2 \leftarrow \mathsf{ACE.Enc}(pp, ek_i, sk)$. Return $c = (c_1, c_2)$.
Decryption: $m' \leftarrow \mathsf{Dec}(pp, dk_j, c)$
Parse $c = (c_1, c_2)$. Decrypt the secret key $sk' \leftarrow \mathsf{ACE.Dec}(pp, dk_j, c_2)$. If $sk' = \perp$, return $\perp$. Else return $m' \leftarrow \mathsf{SE.Dec}_{sk'}(c_1)$.

Efficiency, Storage Requirements, and Optimizations. The length of the ciphertext is $\mathrm{O}(n_R + \ell)$ using a rate-1 SKE. Appendix C contains a construction from predicate encryption that outputs more compact ciphertexts (of length $\mathrm{O}(\log(n_S) + \lambda)$) when instantiated for policies such that $\min_{j \in [n_R]} S_j = \mathrm{O}(\log n_S)$ where S_j is the set of senders allowed to communicate with the receiver j.

Theorem 5.1. *The protocol previously described is correct, and satisfies the properties of No-Read and No Secret Write if the SKE is lor-cpa secure, and* ACEnoS *satisfies correctness, NRR and NSWR as described in Section 3.*

The security proof is akin to that of $\mathsf{ACE}^{\mathsf{pke}}$, and is deferred to Appendix B.

6 Game-Specific Obfuscation

We suggest a variant of obfuscation that is weaker that Virtual Blackbox (VBB) obfuscation and hence may be possible to implement in general. VBB obfuscation requires that the obfuscated program gives nothing to the receiver, other than oracle access to the original program, and it is well known that no obfuscator can be capable of VBB-obfuscating every program.

Here, we consider instead a security game G (formalized as an experiment in Fig. 1), in which a challenger C plays against an adversary A, using an obfuscator Obf. The game comes with a specification of a family of programs $\mathcal{F} := \{\mathsf{P}_{k,\mathsf{p}}\}_{k \in \{0,1\}^\lambda, \mathsf{p} \in \{0,1\}^m}$, parameterized by k and by a label p of some length m, so we have one member of the family for each pair (k, p). This is meant to cover a wide range of applications where obfuscated programs may be used: very often, an application bakes one or more cryptographic keys into the program, this is modelled by the parameter k. The label p is useful in a multiparty scenario, where parties may be given programs that depend on their identity, for instance.

The game proceeds in rounds, where in each round of the game, A can query C on various labels p to obtain obfuscated programs $\hat{\mathsf{P}}^{\mathsf{p}}_k = \mathsf{Obf}(\mathsf{P}^{\mathsf{p}}_k)$, as well as for other data (such as public parameters). At the end of each round, A returns some final output z_i, which is remembered between rounds. Optionally, the game may allow A to remember additional state information between rounds (not represented in Fig. 1). In the

Experiment $\mathsf{Exp}^{G^0}_{\mathsf{A},\mathsf{Obf}}(\lambda,\mathcal{F},q)$	**Experiment** $\mathsf{Exp}^{G^1}_{\mathsf{B},\mathsf{Obf}}(\lambda,\mathcal{F},q)$
$st \leftarrow (\lambda,\mathcal{F},q)$ $(pp,k) \leftarrow \mathsf{C}(0;st)$ For $i=1,\ldots,q$: $\quad z_i^{\mathsf{A}} \leftarrow \mathsf{A}^{\mathsf{C}(1,k,\cdot;st),\mathsf{C}(3,\cdot;st)}(pp)$ $b \leftarrow \mathsf{C}(4,z_1^{\mathsf{A}},\ldots,z_q^{\mathsf{A}};st)$ Return b.	$st \leftarrow (\lambda,\mathcal{F},q)$ $(pp,k) \leftarrow \mathsf{C}(0;st)$ For $i=1,\ldots,q$: $\quad z_i^{\mathsf{B}} \leftarrow \mathsf{B}^{\mathsf{C}(2,k,\cdot,\cdot;st),\mathsf{C}(3,\cdot;st)}(pp)$ $b \leftarrow \mathsf{C}(4,z_1^{\mathsf{B}},\ldots,z_q^{\mathsf{B}};st)$ Return b.

Fig. 1. Security experiment for Game-Specific Obfuscation.

end, C decides if A won the game. Our definition compares this to a similar experiment where, however, the adversary B only gets oracle access to the programs.

Importantly, C can decide not to answer a query, based on the label and its current state. This models conditions one would typically have in a game, such as: the game models a scheme with several parties participating, some of which are corrupt, and the adversary is not allowed to query a program that was given to an honest player. Since the same C plays against both A and B, they are under the same constraints, so B cannot "cheat" and make queries that A could not.

For simplicity, we let C choose a single parameter k initially. We can easily generalize to cases where programs using several different values of k are used.

Definition 6.1 (Game-Specific Obfuscation). *We say that* Obf *is a* game-specific secure obfuscator *(GSO) relative to G and $\mathcal{F}$ if for every PPT adversary* A*, there exists a PPT adversary* B *which plays G using only oracle access to each obfuscated program, and where* $|\mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{G^0}_{\mathsf{A},\mathsf{Obf}}(\lambda,\mathcal{F},q)] - \mathsf{Pr}[1 \leftarrow \mathsf{Exp}^{G^1}_{\mathsf{B},\mathsf{Obf}}(\lambda,\mathcal{F},q)]|$ *is negligible, where the challenger behaves as follows:*

Challenge Generation: *on input* $(0;(\lambda,\mathcal{F},q))$*, it returns* $k \in \{0,1\}^\lambda$ *and some general public parameters pp.*

Program Obfuscation: *on input* $(1,k,\mathsf{p};st)$*, it returns the obfuscation* $\hat{\mathsf{P}}^{\mathsf{p}}_k \leftarrow \mathsf{Obf}(\mathsf{P}^{\mathsf{p}}_k)$ *of the program, or* $\perp$*.*

Oracle Access to Programs: *on input* $(2,(k,\mathsf{p},m);st)$ *it returns the evaluation of the program* $\mathsf{P}^{\mathsf{p}}_k(m)$*, or* $\perp$*.*

Other Data: *on input* $(3,\cdot;st)$ *it can return additional data.*

Winning Condition Check: *on input* $(4,z_1,\ldots,z_q;st)$ *it returns* 1 *if the adversary won the game, 0 otherwise.*

Every mode of operation can update the state st of the challenger too, if required by the game.

Note that this definition, while implied by VBB, makes a much weaker demand than VBB: we assume that the obfuscation gives nothing more than oracle access, only as far as winning G is concerned, and the obfuscator only needs to obfuscate programs in $\mathcal{F}$. Indeed, the impossibility result for VBB does not apply to game-specific obfuscation in general, it just rules out its existence for a specific game and family of programs. The notion is somewhat incomparable to iO obfuscation: obfuscators secure in the iO sense are usually claimed to be able to obfuscate any program, and can potentially be applied in any security game, but on the other hand, iO only guarantees indistinguishability between programs with the same input/output behavior. Even when restricting to assume the existence of iO/GSO for specific programs (as it happens in constructions relying on iO), still iO and GSO target different aspects: GSO has no specific requirement on the family of programs, while iO needs them to compute the same function; on the other hand, iO still guarantees indistinguishability for every game, while GSO targets a specific one.

As usual, we also require the obfuscators to *preserve functionality* (the input-output behaviour of the obfuscated program is equivalent to the original program) and *polynomial slowdown* (the obfuscated program should can at most be polynomially slower/larger than the original one). The formal definitions of these properties are deferred to Appendix A.

7 ACE with Ciphertext Verifiability

In this section we explore whether it is possible to obtain more than just preventing parties from establishing secure subliminal channels. The intuition is that it should be possible to restrict corrupted parties in the bandwidth of their subliminal channels by adding some form of *ciphertext verifiability* to our model. Ciphertext verifiability allows any party with access to the bulletin board to verify that ciphertexts appended to the public board have been generated honestly and according to policy, *even if the party is not allowed to decrypt them by the policy*. We then show a scheme that allows to restrict the bandwidth of corrupted senders to logarithmic in the security parameter under a novel variant of obfuscation, namely the GSO introduced in the previous section. We find this a promising indication that public verification can help to restrict subliminal communication between corrupted parties. As a byproduct, we get a construction whose complexity only scales polylogarithmically with the number of parties.

7.1 Ciphertext Verifiability

Parties, policies and the communication model are the same as in Section 3. The difference is that an ACE with ciphertext verifiability (VACE) is composed by 5 algorithms ($\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Verify}, \mathsf{Dec}$). The verification algorithm Verify allows receivers to verify that ciphertexts published in the bulletin board are well-formed according to their decryption key:

Verification $b \in \{0,1\} \leftarrow \mathsf{Verify}(dk_j, c)$

On input a ciphertext c and a decryption key, the algorithm outputs 1 if $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$ for some (unknown) honestly generated sender's key ek_i and message $m \in \mathcal{M}$, and 0 otherwise.

Remark that the definition implies that verification can be done using dk_0, i.e., the decryption key of the receiver with identity $j = 0$ which by policy cannot receive messages[7]. Differently from ACEnoS, now dk_0 might not be equal to the public parameters (while ek_0 still is). Moreover, dk_0 is not part of them: it is given only to receivers, not to the senders. This follows quite naturally from the communication model: as senders have write-only access to the public board, they cannot see (thus verify) ciphertexts by other senders than themselves[8].

The introduction of such algorithm requires to modify the properties of security and correctness as well. This new construction of ACE should satisfy both correctness as defined in Section 3 and a completeness requirement (i.e., that honestly generated ciphertexts pass verification).

Definition 7.1 (Completeness). *A* VACE *scheme is complete if for all* $\lambda, m \in \mathcal{M}$, $i \in [n_S]$, $j \in [n_R]$ *it holds*

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Verify}(dk_j, \mathsf{Enc}(pp, ek_i, m)) \;:\; \begin{array}{c}(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P}),\\ ek_i \leftarrow \mathsf{KGen}(pp, msk, i, \mathsf{sen}),\\ dk_j \leftarrow \mathsf{KGen}(pp, msk, j, \mathsf{rec})\end{array}\right] = 1 \ ,$$

where the probabilities are taken over the random coins of all the algorithms.

To ensure that verification is meaningful, the outcome of verification should be consistent when done with different keys.

Definition 7.2 (Verification Consistency). *Given a policy* $\mathcal{P}$*, a* VACE *scheme verifies consistently if, for every PPT adversary* A *there exists a negligible function* negl *such that*

$$\mathsf{Pr}\left[b_0 \neq b_1 \;\middle|\; \begin{array}{l}(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})\\ (i_0, i_1, c) \leftarrow \mathsf{A}^{\mathcal{O}_G(\cdot,\cdot)}(pp)\\ \textit{For } k = 0, 1\\ \quad dk_{i_k} \leftarrow \mathsf{KGen}(pp, msk, i_k, \mathsf{rec})\\ \quad b_k \leftarrow \mathsf{Verify}(dk_{i_k}, c)\end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

[7] The inclusion of the identity 0 for senders and receivers with no rights is standard in normal access control encryption, cf. [12].

[8] In fact, it seems to be necessary for a more technical reason related to the NSWR (as the verification key could be seen as shared randomness between corrupted senders and receivers).

where the $\mathcal{O}_G$ returns ek_j on input (j, sen), and dk_j on input (j, rec).

The No Read Rule remains unchanged as such property is not concerned with enforcing the policy, but with the anonymity and privacy of the scheme. On the other hand, the winning condition of the No Secret Write Rule changes to impose that the challenge ciphertext successfully verifies w.r.t. some fixed receiver key. This, combined with consistency of verification (which we just defined) implies that a successful verification w.r.t. even just the public verification key dk_0 is enough to guarantee it w.r.t. all receiver keys (which could be impossible to check efficiently in the game if the number of receivers is superpolynomial).

The verification key dk_0 is only given to the corrupted receiver A_2 and to the public verifier B but not to the corrupted sender A_1, as the latter cannot read from the public board, but just write on it.

Definition 7.3 (No Secret Write Rule). *Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be an adversary and consider the following game:*

Experiments

$\mathsf{Exp}^{\mathsf{nusw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P})$	$\mathsf{Exp}^{\mathsf{nusw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})$
$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ $(\bar{m}, c) \leftarrow \mathsf{A}_1^{\mathcal{O}_G(\cdot,\mathsf{sen})}(pp)$ $m' \leftarrow \mathsf{A}_2^{\mathcal{O}_G(\cdot,\mathsf{rec}),\ \mathcal{O}_E(\cdot)}(pp, c)$ *Return* 1 *if* $\bar{m} = m' \ \wedge\ 1 \leftarrow \mathsf{Verify}(dk_0, c), dk_0 \leftarrow \mathcal{O}_G(0, \mathsf{rec})$, 0 *otherwise.*	$(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ $(\bar{m}, c) \leftarrow \mathsf{A}_1^{\mathcal{O}_G(\cdot,\mathsf{sen})}(pp, \bar{m})$ $m'' \leftarrow \mathsf{B}^{\mathcal{O}_E(\cdot),\mathcal{O}_G(0,\mathsf{rec})}(pp, c)$ *Return* 1 *if* $\bar{m} = m'' \ \wedge\ 1 \leftarrow \mathsf{Verify}(dk_0, c), dk_0 \leftarrow \mathcal{O}_G(0, \mathsf{rec})$, 0 *otherwise.*

Oracles

$\mathcal{O}_G(j, t)$: *If* $\exists\ k_j$ *s.t.* $(k_j, j, t) \in \mathcal{L}$, *return* k_j. *Else* $k_j \leftarrow \mathsf{KGen}(pp, msk, j, t)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(k_j, j, t)\}$ *Return* k_j.	$\mathcal{O}_E(j, m)$: $ek_j \leftarrow \mathcal{O}_G(j, \mathsf{sen})$ *Return* $c \leftarrow \mathsf{Enc}(pp, ek_j, m)$.

Let $\mathcal{Q}_1$ (resp., $\mathcal{Q}_2$) be the set of all queries $q = (i, \mathsf{sen})$ (resp., $q = (j, \mathsf{rec})$) that A_1 (resp.,A_2) issues to $\mathcal{O}_G$. The adversary wins the experiment if $m' = \bar{m}$ and the ciphertext verifies while the following holds:

No Communication Rule (NCR). $\forall\ (i, \mathsf{sen}) \in \mathcal{Q}_1,\ (j, \mathsf{rec}) \in \mathcal{Q}_2$, *it should hold that* $\mathcal{P}(i, j) = 0$.

Given λ and a policy $\mathcal{P}$, a ACE without sanitizer with verifiable ciphertexts satisfies the No Secret Write rule if for all PPT $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ there exists a PPT algorithm B and a negligible function negl such that

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right] \geq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P}) \ \wedge\ \mathsf{NCR}\right] - \mathsf{negl}(\lambda)\ .$$

Ciphertext Verifiability vs. Sanitization. It is fair to wonder whether adding public verifiability yields an ACE with sanitization. This is not the case because: (1) the sanitizer/verification key is public; (2) in the VACE case, behavior of sanitizer/verifier is checkable by other receivers; (3) the access structure to a public board usually requires an authentication layer: verification (and possible identification of dishonest senders) can be enforced in that layer.

7.2 VACE from Game Specific Obfuscation

Verifiability of a ciphertext means that any party can verify that the ciphertext satisfies some relation, i.e., that *has some structure*, which bounds the entropy of the ciphertext. While this is not enough to prevent subliminal channels completely (as this seems to require the injection of true randomness, e.g., cf. [12, 23]), in this section we show that this is enough to meaningfully restrict the bandwidth of corrupted senders.

We build a VACE following the IND-CCA PKE construction from iO by Sahai and Waters [24], with the following changes: (1) we impose that every ciphertext encrypts the identity of the sender in addition to the message, and (2) decryption is done by an obfuscated program that checks the policy too. As the original protocol outputs ciphertexts composed by two parts, the encryption of the message and a value used as authentication/integrity check, we easily get a VACE construction that is NRR secure assuming iO with a proof similar to [24]. However, proving NSWR from iO seems impossible, thus we rely on a GSO assumption on the obfuscator (further details in Section 7.3).

We now consider messages to be just one bit, i.e., $\mathcal{M} = \{0,1\}$, and assume that $n_S = \mathsf{poly}(\lambda)$ (as this is needed when using the puncturable PRF in the proof of the NRR rule).

Setup: $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$
Generate the keys for the PRFs: $K_k \xleftarrow{\$} \{0,1\}^\lambda$, $k = 1, 2$. The algorithm returns $pp = (\lambda, \mathcal{P}, \mathcal{M})$ and the master secret key $msk = (K_1, K_2)$.

Key Generation: $k_i \leftarrow \mathsf{KGen}(pp, msk, i, t)$
Generate the obfuscated circuits $\hat{\mathsf{P}}^s_i \leftarrow \mathsf{Obf}(\lambda, \mathsf{P}^s_i)$, $i \in [n_S] \setminus \{0\}$, and $\hat{\mathsf{P}}^r_i \leftarrow \mathsf{Obf}(\lambda, \mathsf{P}^r_i)$, $i \in [n_R]$, of the programs P^s_i and P^r_j in Fig. 2, padded so that they are as long as the programs in the reductions (cf. proof of Theorem 7.4 and Theorem 7.9).

$\mathsf{P}^s_i(m, s)$

$t \leftarrow \mathsf{PRG}(s)$
$cipher \leftarrow \mathsf{PRF}_1(K_1, t) \oplus (m, i)$
$sig \leftarrow \mathsf{PRF}_2(K_2, (t \mid\mid cipher))$
$c \leftarrow (t, cipher, sig))$
Return c.

$\mathsf{P}^r_j(c)$

Parse $c = (t, cipher, sig)$.
$b \leftarrow (sig == \mathsf{PRF}_2(K_2, (t \mid\mid cipher)))$
If $(b == 1)$
  $(m, i) \leftarrow cipher \oplus \mathsf{PRF}_1(K_1, t)$
  If $(\mathcal{P}(i, j) == 1)$
    Return $(1, m)$
  Else return $(1, \perp)$.
Else Return $(0, \perp)$.

Fig. 2.

- If $i \neq 0$ and $t = \mathsf{sen}$, return $ek_i = (\hat{\mathsf{P}}^s_i)$.
- If $i = 0$ and $t = \mathsf{sen}$, return $ek_0 = pp$.
- If $t = \mathsf{rec}$, return $dk_i = (\hat{\mathsf{P}}^r_i)$.

Encryption: $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$
Sample $s \xleftarrow{\$} \{0,1\}^\lambda$. Return $c \leftarrow \hat{\mathsf{P}}^s_i(m, s)$.

Decryption: $m' \leftarrow \mathsf{Dec}(pp, dk_j, c)$
Run $(b, m') \leftarrow \hat{\mathsf{P}}^r_j(c)$ and return m'.

Verification: $b \in \{0,1\} \leftarrow \mathsf{Verify}(dk_j, c)$
Run $(b, m') \leftarrow \hat{\mathsf{P}}^r_j(c)$ and return b.

For ease of exposition we split the security proof of the VACE in two theorems, as the NRR and NSWR require different assumptions on Obf. In particular Theorem 7.4 shows NRR security and only requires the standard notion of iO, whereas Theorem 7.9 uses the novel GSO assumption. Note that one could also have chosen to prove the NRR security of the VACE assuming GSO instead of iO, but we opted for using the minimal assumptions for each theorem.

Theorem 7.4. *The* VACE *previously defined satisfies correctness and completeness, and has consistent verification, assuming the correctness of its building blocks. In addition, if* Obf *is a iO and* PRF_1, PRF_2 *and* PRG *are two puncturable PRF and a PRG respectively, the previous* VACE *and is NRR secure.*

Proof. Correctness and completeness of the VACE follows from the correctness of the building blocks. Verification consistency follows easily from the fact that the first bit of the output of $\hat{\mathsf{P}}^r_j$ is independent of the

value of j, thus it is the same for all $j \in [n_R]$. The NRR relies on the pseudorandomness property of the functions used in the obfuscated program, and on iO. The the proof relies on a sequence of hybrid games (when necessary, changes are pointed out in red):

Hybrid 0: this is the NRR experiment with $b = 0$.

Hybrid 1: in this game everything is the same as in Game 0 but the challenge ciphertext, which is generated as follows:

$$
\begin{aligned}
&t^* \xleftarrow{\$} \{0,1\}^{2\lambda} \\
&cipher^* \leftarrow \mathsf{PRF}_1(K_1, t^*) \oplus (m_0, i_0) \\
&sig^* \leftarrow \mathsf{PRF}_2(K_2, (t^* \mid\mid cipher^*)) \\
&c^* \leftarrow (t^*, cipher^*, sig^*) \ .
\end{aligned}
$$

The probability of distinguishing Hybrid 1 from Game 0 is exactly the probability of distinguishing a PRG from a uniform function.

Hybrid 2: in this game the challenger samples t^* at the beginning of the experiment, and it generates the receivers keys obfuscating the program $\bar{\mathsf{P}}^r_j$ in Fig. 3. The output of $\bar{\mathsf{P}}^r_j$ is equal to the output of P^r_j on all

$\mathsf{P}^r_j(c)$

Parse $c = (t, cipher, sig)$.
$b \leftarrow (sig == \mathsf{PRF}_2(K_2, (t \mid\mid cipher)))$
If $(b == 1)$
  If $t = t^*$
    $y \leftarrow \mathsf{PRF}_1(K_1, t^*)$
    $(m, i) \leftarrow cipher \oplus y$
  Else $(m, i) \leftarrow cipher \oplus \mathsf{PRF}_1(K_1, t)$
  If $(\mathcal{P}(i, j) == 1)$
    Return $(1, m)$
  Else return $(1, \perp)$.
Else Return $(0, \perp)$.

Fig. 3.

inputs, thus distinguishing Hybrid 2 from Hybrid 1 amounts to break iO security of the obfuscator.

Hybrid 3: in this game the challenger samples t^* at the beginning of the experiment, samples the PRF keys K_1, PRF_2 and generates the punctured key $K_{1,\{t^*\}}$ and the value $y^* \leftarrow \mathsf{PRF}_1(K_1, t^*)$. Then it generates the senders and receivers keys as obfuscation of the programs in Fig. 4. As t^* is sampled uniformly at random, the probability that t^* is in the image of the PRG is $1/2^\lambda$. Therefore, with probability $1 - 1/2^\lambda$ $\bar{\mathsf{P}}^s_i$ will not call $\mathsf{PRF}_1(K_1, t^*)$, and for all (m, s) it holds that $\mathsf{P}^s_i(m, s) = \bar{\mathsf{P}}^s_i(m, s)$. The functionality of $\bar{\mathsf{P}}^r_j$ does not change between Hybrid 3 and Hybrid 2. Hence distinguishing Hybrid 3 from Hybrid 2 amounts to breaking iO security.

Hybrid 4: in this game everything is the same as in Hybrid 3 except for the value y^*. Indeed, the challenger samples $y^* \xleftarrow{\$} \{0,1\}^{2\lambda}$ alongside t^* at the beginning of the game, and it uses it in $\bar{\mathsf{P}}^r_j$ from Fig. 4. When it is time to generate the challenge ciphertext, it does the following:

$$
\begin{aligned}
&cipher^* \leftarrow y^* \oplus (m_0, i_0) \\
&sig^* \leftarrow \mathsf{PRF}_2(K_2, (t^* \mid\mid cipher^*)) \\
&c^* \leftarrow (t^*, cipher^*, sig^*) \ .
\end{aligned}
$$

Then distinguishing Hybrid 4 from Hybrid 3 is equivalent to breaking the security of the constrained PRF PRF_1.

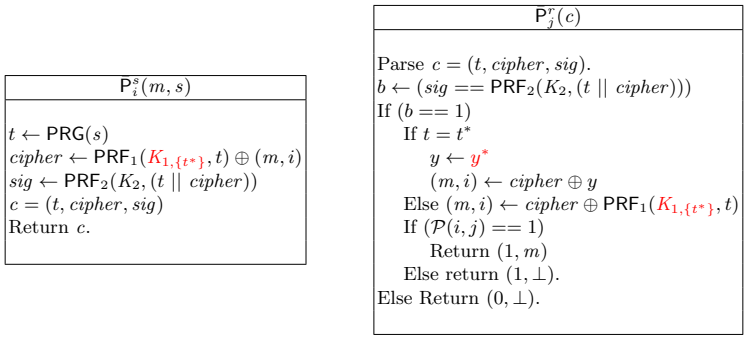

Fig. 4.

Hybrid 5: in this game everything is as in Hybrid 4 except for the challenge ciphertext, which is now generated as

$$\begin{aligned} cipher^* &\leftarrow y^* \oplus (m_1, i_0) \\ sig^* &\leftarrow \mathsf{PRF}_2(K_2, (t^* \mid\mid cipher^*)) \\ c^* &\leftarrow (t^*, cipher^*, sig^*) \ , \end{aligned}$$

where t^* and y^* are sampled at random at the beginning of the game as before. In the Sender Anonymity case, $m_0 = m_1$, thus Hybrid 5 is exactly equal to Hybrid 4. In the Payload Privacy case, the distribution of c_1^* is statistically close to uniform in both games, thus the only way the adversary can distinguish the games is by somehow decrypt $(t^*, cipher^*)$ using a decryption key. In Theorem 7.5 we show that this is essentially equivalent to guessing a random 2λ-long bit string.

Hybrid 6: in this game everything is as in Hybrid 5 except for the challenge ciphertext, which now encrypts identity i_1:

$$\begin{aligned} cipher^* &\leftarrow y^* \oplus (m_1, i_1) \\ sig^* &\leftarrow \mathsf{PRF}_2(K_2, (t^* \mid\mid cipher^*)) \\ c^* &\leftarrow (t^*, cipher^*, sig^*) \ , \end{aligned}$$

(t^* and y^* are sampled at random at the beginning of the game as before). In Theorem 7.6 we show that again this is essentially equivalent to guessing a random 2λ-long bit string. The proof is analogous to the proof of Theorem 7.5.

Hybrid 7: in this game everything is as in Hybrid 6 except for y^*, which is now generated as $\mathsf{PRF}_1(K_1, t^*)$. As the obfuscated programs are still generated using the punctured key $K_{1,\{t^*\}}$, distinguishing Hybrid 7 from Hybrid 6 is as hard as breaking the security of the PRF.

Hybrid 8: in this game everything is done as in Hybrid 7 except for the senders and receivers keys, which are now programmed to use the key K_1 instead of its punctured version. As the probability that t^* is in the image of PRG is $\frac{1}{2^\lambda}$, with probability $1 - \mathsf{negl}(\lambda)$ the functionality of the programs do not change, thus distinguishing Hybrid 7 from Hybrid 8 requires breaking iO.

Hybrid 9: in this game everything is as in Hybrid 8 but the receivers keys, which are generated as the obfuscation of the program in Fig. 2. As the input/output behavior of the program does not change, distinguishing Hybrid 9 from Hybrid 8 again requires to break iO.

Hybrid 10: in Hybrid 10 everything is generated as in the previous game except for the challenge ciphertext, which is now generated as:

$$
\begin{aligned}
&s \leftarrow \{0,1\}^\lambda \\
&t^* \leftarrow \mathsf{PRG}(s) \\
&cipher^* \leftarrow \mathsf{PRF}_1(K_1, t^*) \oplus (m_1, i_1) \\
&sig^* \leftarrow \mathsf{PRF}_2(K_2, (t^* \mid\mid cipher^*)) \\
&c^* \leftarrow (t^*, cipher^*, sig^*) \ .
\end{aligned}
$$

Distinguishing Hybrid 10 from Hybrid 9 would require to break the pseudorandomness of the PRG.
Game 1: this is the NRR game with $b = 1$. In fact, Game 1 is exactly identical to Hybrid 10.

□

In the following there are the two Lemmas required to complete the proof of Theorem 7.4.

Lemma 7.5. $\left|\mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H5}\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H4}\right]\right| \leq \frac{1+\mathsf{negl}_{iO}+\mathsf{negl}_{\mathsf{PRF}}}{2^{2\lambda}}$, *where* A^{Hi} *is the experiment in which the adversary is run in Hybrid* i*.*

Proof. The distribution of the challenge ciphertext c_1^* is statistically close to uniform in both games, thus the only way the adversary can distinguish the games is by somehow decrypt $(t^*, cipher^*)$ using a decryption key.
In the Sender Anonymity case it holds $m_0 = m_1$, thus the two hybrid games are identical.
In the Payload Privacy case, the only way the adversary can distinguish Hybrid 4 from Hybrid 5 is by somehow decrypting the ciphertext $(t^*, cipher^*)$. By definition of the PP case A can only query decryption keys of identities j such that $\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0$. To decrypt the ciphertext A has to change the identity it encrypts: Fist, A selects a sender identity i' such that there is a receiver's identity j such that $\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0 \wedge \mathcal{P}(i', j) = 1$. Then it modifies the challenge ciphertext setting the second component as $cipher' = cipher^* \oplus (0, i' \oplus i_0) = y^* \oplus (m_b, i')$. However, to have it decrypted, it needs to forge a valid tag sig' on $(t^*, cipher')$. We show that the success probability of A is essentially the probability of guessing a random 2λ-long bit string. The experiment is simulated as follows. Let B be the simulator. Before starting the interaction with A,

- B samples $t^* \xleftarrow{\$} \{0,1\}^\lambda$, $y^* \xleftarrow{\$} \{0,1\}^{2\lambda}$.
- It samples a random key K_2 for PRF_2, and punctures it on the set $S = \{(t^*, y^* \oplus (0, i))\}_{i \in [n_S]}$, obtaining $K_{2,S}$. Remark that $|S| = 2 * n_S$, which is polynomial in λ, as n_S is. This is why the proof does not require constrained PRFs [10], but just puncturable PRFs (cf. Definition 7 in [24]).
- It samples strings $\{sig'_{0,i}, sig'_{1,i}\}_{i \in [n_S]}$ at random from $\{0,1\}^{2\lambda}$, and sets them to be the signatures on the $(t^*, y^* \oplus (0, i))$, $(t^*, y^* \oplus (1, i))$ for the corresponding $i \in [n_S]$.

Then, B send the public parameters to A, and it simulates the oracles as follows:

$\mathcal{O}_G$**:** B generates the keys as obfuscations of the programs in Fig. 5.
$\mathcal{O}_E$**:** according to NRR experiment.
c^***:** B samples a random bit $b \xleftarrow{\$} \{0,1\}$, and sets the challenge ciphertext as

$$c^* \leftarrow (t^*, cipher^* = y^* \oplus (b, i_0), sig^* = sig_{b,i_0})) \ ,$$

where recall the messages are 1-bit long, thus we can use the convention $m_b = b$.

As t^* cannot be output by PRG but with probability $\frac{1}{2^\lambda}$, the functionalities of $\bar{\mathsf{P}}_i^s$ is exactly the same. Moreover, distinguishing the simulated tags from honestly generated ones amounts to break the security of the puncturable PRF PRF_2. Therefore,

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H5}\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H4}\right]\right| = \frac{1 + \mathsf{negl}_{iO} + \mathsf{negl}_{\mathsf{PRF}}}{2^{2\lambda}} \ .$$

□

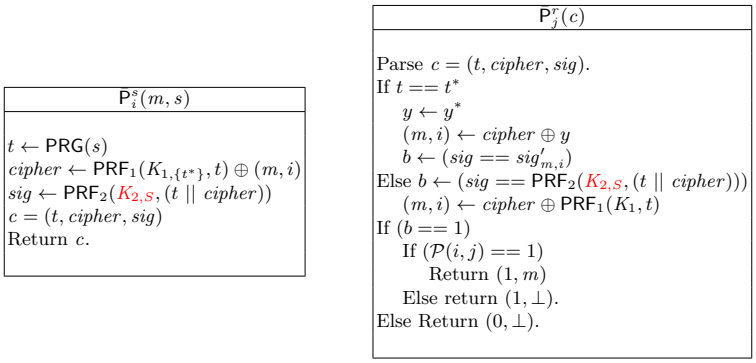

Fig. 5.

Lemma 7.6. $\left|\mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H6}\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H5}\right]\right| \leq \frac{1+\mathsf{negl}_{iO}+\mathsf{negl}_{\mathsf{PRF}}}{2^{2\lambda}}$, *where* A^{Hi} *is the experiment in which the adversary is run in Hybrid* i*.*

Proof. The distribution of the challenge ciphertext is the same in both games. Therefore, again the only way that A can distinguish is by observing discrepancies in the decryption of the string $(t^*, cipher^*)$. Now, for all receivers' keys queried by A to $\mathcal{O}_G$, it holds that:

Sender Anonymity: $\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j)$.
Payload Privacy: $\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0$.

Analogously as in the proof of Theorem 7.5, the only way A can decrypt the ciphertext is to change the identity it encrypts: Fist, A selects a sender identity i' such that there is a receiver's identity j such that $\mathcal{P}(i' \oplus i_0, j) \neq \mathcal{P}(i' \oplus i_1, j)$. Then it modifies the challenge ciphertext setting the second component as $cipher' = cipher^* \oplus (0, i') = y^* \oplus (m_1, i' \oplus i_b)$. To have such a ciphertext decrypted, A needs to forge a valid tag sig' on $(t^*, cipher')$. Again the experiment can be simulated so that distinguishing between the games is essentially equivalent to guessing a random string. The simulator B generated the simulation exactly as the simulator in the proof of Theorem 7.5, except for the challenge ciphertext, which is now generated as follows:

$$b \xleftarrow{\$} \{0,1\}$$
$$c^* \leftarrow (t^*, cipher^* = y^* \oplus (m_1, i_b), sig^* = sig'_{m_1,i_b})) \ .$$

As t^* cannot be output by PRG but with probability $\frac{1}{2^\lambda}$, the functionalities of $\bar{\mathsf{P}}_i^s$ is exactly the same, thus distinguishing the simulated keys from honestly generated ones is equivalent to breaking iO. Observe that if $b = 0$ (resp., $b = 1$) the challenge ciphertext (excluding the tag) is generated as in Hybrid 5 (resp., Hybrid 6); distinguishing the simulated hybrid games from the real ones requires breaking the security of the puncturable PRF. To understand how would A distinguish, assume $\mathcal{P}(i' \oplus i_0, j) = 1 \ \wedge \ \mathcal{P}(i' \oplus i_1, j) = 0$ (the opposite case is analogous). Clearly, if A guesses the tag sig' correctly in the case $b = 0$ (i.e., in Hybrid 5) it holds

$$cipher' = cipher^* \oplus (0, i') = y^* \oplus (m_1, i' \oplus i_0)$$

and $\mathsf{Obf}(\bar{\mathsf{P}}_j^r(t^*, cipher', sig'))$ returns $(1, m_1)$, while if $b = 1$ (i.e., in Hybrid 6) we have that

$$cipher' = cipher^* \oplus (0, i') = y^* \oplus (m_1, i' \oplus i_1)$$

and $\mathsf{Obf}(\bar{\mathsf{P}}_j^r(t^*, cipher', sig'))$ returns $(1, \perp)$. Therefore,

$$\left|\mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H6}\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{A}^{H5}\right]\right| = \frac{1 + \mathsf{negl}_{iO} + \mathsf{negl}_{\mathsf{PRF}}}{2^{2\lambda}} \ .$$

□

7.3 No Secret Write Rule of VACE

We argue that indistinguishability obfuscation does not seem enough to prove NSWR security for our VACE. A major hint in this direction is that proving that the NSWR holds seems to require to show that A_2 cannot distinguish the real experiment from an experiment where both the encryption oracle and the receiver keys are simulated using only the information available to B (i.e., the encryption oracle and the verification key). However, we do not see a way to simulate the decryption keys that preserves their I/O behavior without knowing the master secret keys. Such a consistency in the I/O behavior of the keys is needed because A_1 could still transmit information to A_2 related to the behavior of the senders' keys queried by A_1, e.g., the output on a particular input (s, m): as B does not know which keys have been queried by A_1, it cannot rely on the encryption oracle to answer these queries consistently. However, simulation can be done assuming Obf to be a secure GSO. In particular, the obfuscator is assumed to be GSO secure for the following family of programs and game.

Definition 7.7 ($\mathcal{F}$). *The family $\mathcal{F} = \{\mathsf{P}_{k,\mathsf{p}}\}_{k,\mathsf{p}}$ contains all the possible keys:*

- $k = msk = (K_1, K_2)$, *and*
- $\mathsf{p} = (j, t)$ *is the identity and type of party, and*
- $\mathsf{P}_{k,(j,t)} = \mathsf{P}^t_j$, $t \in \{s, r\}$ *as defined in Fig. 2.*

Definition 7.8 (G_{nsw}). *The game G_{nsw} runs for $q = 2$ rounds and is played by a challenger $\mathsf{C}_{\mathsf{nsw}}$ that behaves as follows:*

- $\mathsf{C}(0, \ldots; st)$ *returns* $(pp, k) = (pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$ *and stores them in st (alongside a round counter).*
- $\mathsf{C}(1, (k, (j, t)); st)$ *returns the output of* $\mathsf{KGen}(pp, msk, j, t)$ *and stores the query in a list q_i for $i = 1, \ldots, q$ in st.*
- $\mathsf{C}(2, (k, (j, t), m); st)$ *it returns the evaluation of the program* $\mathsf{P}^t_j(m)$.
- $\mathsf{C}(3, st)$ *returns $\perp$ during round 1, and $\bar{s}$ in round 2.*
- $\mathsf{C}(4, z_1, z_2; st)$ *parses $z_1 = (\bar{m}, \bar{s})$ and returns 1 if the following three conditions hold:*
  1. $z_2 = \bar{m}$
  2. $1 \leftarrow \mathsf{Verify}(dk_0, \bar{s})$
  3. q_1 *(resp., q_2) contains only queries for sender (resp., receiver) keys, and for every $(i, \mathsf{sen}) \in q_1$ and $(j, \mathsf{rec}) \in q_2$ it holds that $\mathcal{P}(i, j) = 0$.*

Note that we have chosen to only use the GSO assumption where it is necessary, namely the NSWR property. Therefore, since the NRR property is still proven using the iO assumption, the PRFs used in the construction are still puncturable even if this property is not explicitly used in the proof of the NSWR property.

Theorem 7.9. *Assuming* Obf *is a secure GSO for $\mathcal{F}$ and G_{nsw} as in Theorem 7.7 and Theorem 7.8, and given two puncturable PRF and a PRG, the previous* VACE *is NSWR secure. Moreover, assuming that only ciphertexts that pass the verification are posted, it only allows for subliminal channels of bandwidth at most $O(\log(\lambda))$.*

Proof. Proving the NSWR relies on the hypothesis that Obf is a secure GSO for $(\mathcal{F}, G_{\mathsf{nsw}})$. Indeed, the NSWR experiment in Theorem 7.3 is exactly equal to the game in Fig. 1 where $(\mathcal{F}, G_{\mathsf{nsw}})$ are as in Theorem 7.7 and Theorem 7.8, and the adversary A in the GSO experiment behaves like A_1 in the first round, and like A_2 in the second. This implies that the probability that $(\mathsf{A}_1, \mathsf{A}_2)$ win the NSWR experiment is the same as the probability that A wins the GSO experiment. In fact, from GSO security it follows that for any adversary A there exists a second adversary A' that has only oracle access to all the keys, but wins the game G_{nsw} (i.e., the NSWR experiment) with almost the same probability:

$$\begin{aligned}\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P}) \ \wedge \ \mathsf{NCR}\right] &= \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^0_{\mathsf{nsw}}}_{\mathsf{A},\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] \\ &\leq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^1_{\mathsf{nsw}}}_{\mathsf{A}',\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] + \epsilon_{GSO} \ . \end{aligned} \tag{1}$$

Let us now analyze the winning probability of A'. Denote by $(\mathsf{A}'_1, \mathsf{A}'_2)$ the execution of A' in the first and second round of G^1_{nsw} respectively. This adversary now does not receive the sender (or receiver) keys, but is only given oracle access to them. In fact, the oracle only evaluates the plaintext version of the keys, thus it is possible to substitute the PRF and PRG used in the keys with random functions, without significantly impacting the winning probability of A':

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^1_{\mathsf{nsw}}}_{\mathsf{A}',\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^2_{\mathsf{nsw}}}_{\mathsf{A}',\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] + \epsilon_\rho \ , \tag{2}$$

where ϵ_ρ is the probability of distinguishing the PRF and PRG from a random function, and the game G^2_{nsw} is a modification of G^1_{nsw} where $\mathsf{C}(2, \cdot)$ answers the queries executing the code of P^t_i where PRF_1, PRF_2 and PRG have been substituted by random functions.

At this point we can already observe that the bandwidth of the subliminal channel (for ciphertexts that pass the verification) has to be at most $\mathrm{O}(\log(\lambda))$. Indeed, in game G^2_{nsw} the components t and *cipher* of the ciphertext are uniformly random while the tag *sig* is deterministically generated from them, thus a corrupted sender is restricted to encode a subliminal message in a ciphertext only through rejection sampling: A'_1 can only try encrypting randomly chosen messages (the ciphertext does not reveal anything about the plaintext, thus it is fair to assume that the subliminal message and the plaintext are independently chosen) until the ciphertext finally encodes the subliminal message. If the sender runtime is restricted to be polynomial-time, this limits the amount of rejection sampling that it can do, restricting the amount of information encoded in the ciphertext (the subliminal message) to $\mathrm{O}(\log(\lambda))$. The GSO assumption allows to conclude the argument: as A_1 cannot do (much) better than A'_1, the adversary can send short subliminal messages in the real experiment too.

Finally, we conclude the proof of the NSWR by showing an algorithm B that can win the NSWR experiment running A'_2 internally, with almost the same probability as the adversary A. Recall that in game G^2_{nsw} the sender keys oracle (i.e., the simulated $\mathsf{C}(2, (\cdot, (\cdot, \mathsf{sen}), m); st)$, which can be called in both rounds) returns a uniformly random bit string. This in particular implies that the view of A'_1 (i.e., A' at round 1) and A'_2 are independent of the master secret key *msk* generated at the beginning of the game. Therefore, a simulator B can win the NSWR experiment running A' at round 2 internally by generating a fresh pp', msk' for the VACE and use them to simulate $\mathsf{C}(2, \cdot)$. As the public parameters of the scheme only contain λ, $\mathcal{M}$ and $\mathcal{P}$, there is no way for A'_2 to distinguish between the real and simulated experiment, and it holds that

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^2_{\mathsf{nsw}}}_{\mathsf{A}',\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{G^2_{\mathsf{nsw}}}_{(\mathsf{A}'_1,\mathsf{B}),\mathsf{Obf}}(\lambda, \mathcal{F}, 2)\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right] \ . \tag{3}$$

Combining Equations (1), (2), and (3) yields the claim. □

7.4 On the Need for Ciphertext Verifiability

Ciphertext verifiability is crucial for the previous argument to go through: if one cannot verify that the ciphertext has been generated by the obfuscated program, a corrupted sender could just set the ciphertext to be the (subliminal) message it wants to send. So long as the data structure of the ciphertext fits the specifications, the subliminal channel cannot be detected. The next lemma (which is a folklore result) shows that our result is optimal: stricter restrictions on the subliminal channel require sanitization.

Lemma 7.10. *Let λ be the security parameter. An encoding scheme $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ (either symmetric or asymmetric, deterministic or probabilistic) such that the domain of Enc_k has dimension at least $\mathsf{poly}(\lambda)$ for every k output by the key generation KG always allows for insecure subliminal channels with bandwidth $O(\log(\lambda))$ (in absence of a trusted sanitization step) assuming the adversary runs in polynomial-time and has* oracle access *to the encryption algorithm.*

Proof. Consider an encoding $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ that satisfies basic correctness (reportend in the following for completeness):

Correctness: $\forall \lambda \in \mathbb{N},\ \forall m \in \mathcal{M},\ \exists \epsilon = \mathsf{negl}(\lambda)\ :\ \mathsf{Pr}(m' \neq m \mid (k_e, k_d) \leftarrow \mathsf{KG}(1^\lambda),\ c \leftarrow \mathsf{Enc}(k_e, m),\ m' \leftarrow \mathsf{Dec}(k_d, c)) \leq \epsilon$.

Then a PPT adversary A_1 that has only *oracle access* to the encryption algorithm can transmit any subliminal message $\bar{m}$ such that $|\bar{m}| \leq \mathrm{O}(\log(\lambda))$ to a PPT receiver A_2 by sending a single valid ciphertext, even in the worst case scenario in which A_2 cannot decrypt the ciphertext, nor it shares state with A_1, and independently of the security guarantees of the encoding scheme.

The attack is very simple. Having oracle access to the encoding algorithm means that on input m, the oracle returns its encryption under a key fixed at the beginning of the game (and unknown to A_1). The adversary A_1 can query the encryption oracle to obtain $q = \mathsf{poly}(\lambda)$ *distinct* ciphertexts $\{c_i\}_{i=1,\ldots,q}$ (because it runs in polynomial-time, and the domain of the encryption algorithm is large enough for the ciphertexts to be distinct). As they are all distinct, there exists w.h.p. a ciphertext c_i whose (w.l.o.g.) first $|\bar{m}|$ bits are equal to $\bar{m}$. □

Thus, preventing insecure subliminal channels with bandwidth $\leq \mathsf{poly}(\log(\lambda))$ requires a somewhat trusted party (sanitizer) injecting honest randomness in the ciphertexts.

Acknowledgment

Cecilia Boschini is supported by the Università della Svizzera Italiana under the SNSF project number 182452, and by the Postdoc.Mobility grant No. P500PT_203075. Claudio Orlandi is supported by: the Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

References

1. 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, Heidelberg, Dec. 2011.
2. S. Agrawal, M. Maitra, and S. Yamada. Attribute based encryption (and more) for nondeterministic finite automata from LWE. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2019, Part II*, LNCS, pages 765–797. Springer, Heidelberg, Aug. 2019.
3. J. Alwen, J. Katz, Y. Lindell, G. Persiano, a. shelat, and I. Visconti. Collusion-free multiparty computation in the mediated model. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 524–540. Springer, Heidelberg, Aug. 2009.
4. C. Badertscher, C. Matt, and U. Maurer. Strengthening access control encryption. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 502–532. Springer, Heidelberg, Dec. 2017.
5. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, Aug. 2017.
6. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001.
7. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, Dec. 2001.
8. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, Oct. 1997.
9. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, Feb. 2007.
10. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, Dec. 2013.
11. Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical hardness of learning with errors. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013.

12. I. Damgård, H. Haagh, and C. Orlandi. Access control encryption: Enforcing information flow with cryptography. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 547–576. Springer, Heidelberg, Oct. / Nov. 2016.
13. P. Datta, T. Okamoto, and K. Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In T. Peyrin and S. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 640–672. Springer, Heidelberg, Dec. 2018.
14. G. Fuchsbauer, R. Gay, L. Kowalczyk, and C. Orlandi. Access control encryption for equality, comparison, and more. In S. Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 88–118. Springer, Heidelberg, Mar. 2017.
15. O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
16. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, Aug. 2015.
17. 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, Heidelberg, Apr. 2008.
18. S. Kim and D. J. Wu. Access control encryption for general policies from standard assumptions. In T. Takagi and T. Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 471–501. Springer, Heidelberg, Dec. 2017.
19. V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2019, Part II*, LNCS, pages 671–700. Springer, Heidelberg, Aug. 2019.
20. L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.
21. Y. Lu, M. Ciampi, and V. Zikas. Collusion-preserving computation without a mediator. To appear at CSF 2022.
22. D. Micciancio. On the hardness of learning with errors with binary secrets. *Theory Comput.*, 14(1):1–17, 2018.
23. I. Mironov and N. Stephens-Davidowitz. Cryptographic reverse firewalls. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Heidelberg, Apr. 2015.
24. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
25. G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.
26. H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, Nov. 2017.

A Prerequisites

Let $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ be the sets of the natural, integer and real numbers.

Definition A.1. *A function* $\mathsf{negl} : \mathbb{N} \leftarrow [0,1]$ *is* negligible *if* $\mathsf{negl}(\lambda) < 1/\mathsf{poly}(\lambda)$ *for every univariate polynomial* $\mathsf{poly} \in \mathbb{R}[X]$ *and a large enough integer* $\lambda \in \mathbb{N}$.

A.1 Public Key Encryption Scheme

Let $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public key encryption scheme.

Definition A.2 (IND for PKE). *Consider the following game between a challenger* C *and a stateful adversary* A*,*

$\mathsf{Exp}^{\textit{ind-cpa}}_{\mathsf{A}}(\lambda)$
$(pk, sk) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$
$(m_0, m_1, st) \leftarrow \mathsf{A}(pk)$
$b \xleftarrow{\$} \{0,1\},\ \rho \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$
$cipher_b \leftarrow \mathsf{PKE.Enc}(pk, m_b, \rho)$
$b' \leftarrow \mathsf{A}(st, cipher_b)$
If $b' = b$ *return* 1*, else return* 0.

A PKE $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is IND-CPA secure if for all PPT* A *there exists a negligible quantity* $\epsilon_{\mathit{ind\text{-}cpa}} = \mathsf{negl}_{\mathit{ind\text{-}cpa}}(\lambda)$ *such that*

$$2 \cdot \left| \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathit{ind\text{-}cpa}}(\lambda)\right] - \frac{1}{2} \right| \leq \epsilon_{\mathit{ind\text{-}cpa}} \ .$$

Definition A.3 (IK-CPA, from [7]). *Consider the following game between a challenger* C *and an adversary* A*,*

$\mathsf{Exp}_{\mathsf{A},b}^{\mathit{ik\text{-}cpa}}(\lambda)$
$(pk_0, sk_0) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, $(pk_1, sk_1) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ $(m, st) \leftarrow \mathsf{A}(pk_0, pk_1)$ $b \xleftarrow{s} \{0,1\}$, $\rho \xleftarrow{s} \{0,1\}^{\ell(\lambda)}$ $cipher_b \leftarrow \mathsf{PKE.Enc}(pk_b, m, \rho)$ $b' \leftarrow \mathsf{A}(st, cipher_b)$ *If* $b' = b$ *return* 1*, else return* 0*.*

A PKE $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is IK-CPA secure if for all PPT* A *there exists a negligible quantity* $\epsilon_{\mathit{ik\text{-}cpa}} = \mathsf{negl}_{\mathit{ik\text{-}cpa}}(\lambda)$ *such that*

$$2 \cdot \left| \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^{\mathit{ik\text{-}cpa}}(\lambda)\right] - \frac{1}{2} \right| \leq \epsilon_{\mathit{ik\text{-}cpa}} \ .$$

A.2 Symmetric Encryption Scheme

Let $(\mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ be a symmetric encryption scheme (cf. [8] for a formal treatment). The *Left-or-Right* security of a symmetric encryption scheme can be defined as follows.

Definition A.4 (LOR-CPA). *Let* $(\mathsf{SE.KeyGen}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ *be a symmetric encryption scheme,* $b \in \{0,1\}$*,* λ *be the security parameter (i.e., the bit length of the key). Consider the following security experiment:*

$\mathsf{Exp}_{\mathsf{A},b}^{\mathit{lor\text{-}cpa}}(\lambda)$	$\mathcal{O}_{LR,b}(m_0, m_1)$:
$sk \leftarrow \mathsf{SE.KeyGen}(1^\lambda)$ $b' \leftarrow \mathsf{A}^{\mathcal{O}_{LR,b}(\cdot,\cdot)}(\lambda)$ *Return* b'	*If* $\lvert m_0 \rvert \neq \lvert m_1 \rvert$ *abort.* $c \leftarrow \mathsf{SE.Enc}_{sk}(m_b)$ *Return* c*.*

A symmetric encryption scheme satisfies Left-Or-Right Indistinguishability under Chosen-Plaintext Attacks (i.e., it is lor-cpa *secure) if for all PPT adversaries* A*,*

$$\left| \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A},1}^{\mathit{lor\text{-}cpa}}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A},0}^{\mathit{lor\text{-}cpa}}(\lambda)\right] \right| \leq \mathsf{negl}(\lambda) \ .$$

A.3 Obfuscation

Let C be a circuit. We denote by $|\mathsf{C}|$ the bit length of C.

Definition A.5 (Indistinguishability Obfuscator). *An indistinguishability obfuscator (iO) for circuits is a probabilistic algorithm* Obf *that takes as input a circuit* C *and a security parameter* λ *and outputs an obfuscated circuit* $\tilde{\mathsf{C}} \leftarrow \mathsf{Obf}(\lambda, \mathsf{C})$ *such that*

Functionality. *For every circuit* C*, and all valid inputs* x*, it holds that* $\hat{\mathsf{C}}(x) = \mathsf{C}(x)$*.*
Efficiency. $\mathsf{runtime}(\mathsf{Obf}) = \mathsf{poly}(|\mathsf{C}|, \lambda)$*.*
Polynomial Slowdown. $\mathsf{runtime}(\tilde{\mathsf{C}}) = \mathsf{poly}(\mathsf{runtime}(\mathsf{C}), \lambda)$*.*
Indistinguishability. *For any PPT* A *and security parameter* λ*, there exists a negligible quantity* $\mathsf{negl}_{iO} = \mathsf{negl}_{iO}(\lambda)$ *such that, for all two circuits* C_0 *and* C_1 *that compute the same function and are of the same size* k*, it holds that*

$$\left| \Pr\left[1 \leftarrow \mathsf{A}(\mathsf{Obf}(\mathsf{C}_0, \lambda))\right] - \Pr\left[1 \leftarrow \mathsf{A}(\mathsf{Obf}(\mathsf{C}_1, \lambda))\right] \right| \leq \mathsf{negl}_{iO} \ .$$

Experiment $\mathsf{Exp}^{\mathsf{su\text{-}cma}}_{\mathsf{A}}(\lambda)$
$m' \leftarrow \mathsf{A}_1(1^\lambda)$
$(otssk, otsvk) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$
$sig' \leftarrow \mathsf{OTS.Sign}(otssk, m')$
$(m^*, sig^*) \leftarrow \mathsf{A}_2(otsvk, m', sig')$
If $1 \leftarrow \mathsf{OTS.Ver}(otsvk, m^*, sig^*)$
and $m' \neq m^*$,
then return 1 else return 0.

Fig. 6. Selective unforgeability experiment for OTS.

A.4 Function Families

Definition A.6 (Pseudorandom Function ensemble (PRF), from [15]). *An (efficiently computable) PRF ensemble is a function ensemble* $\{f_K : \{0,1\}^{\kappa(|K|)} \to \{0,1\}^{\kappa(|K|)}\}_{K \in \{0,1\}^*}$ *where* $\kappa : \mathbb{N} \to \mathbb{N}$ *such that*

Efficient Evaluation: *there exists a polynomial-time algorithm* PRF *that on input* K *and* $x \in \{0,1\}^{\kappa(|K|)}$ *returns* $f_K(x)$.

Pseudorandomness: *The function ensemble* $\{\mathsf{PRF}_\lambda\}_{\lambda \in \mathbb{N}}$ *such that* PRF_λ *is uniformly distributed over the multi-set* $\{f_K\}_{K \in \{0,1\}^\lambda}$ *is pseudorandom*[9].

Definition A.7 (Pseudorandom Generator (PRG), from [15]). *A PRG is a deterministic polynomial-time algorithm* PRG *satisfying the following two conditions:*

Expansion: *There exists a function* $l : \mathbb{N} \to \mathbb{N}$ *such that* $l(n) > n$ *for all* $n \in \mathbb{N}$, *and* $|G(s)| = l(|s|)$. *The function* l *is called the* expansion factor *of* PRG.

Pseudorandomness: *The ensemble* $\{G(U_n)\}_{n \in \mathbb{N}}$ *is indistinguishable in polynomial-time from a truly uniformly random ensemble* $\{U_n\}_{n \in \mathbb{N}}$.

A.5 One-Time Signatures

A *One-Time Signature (OTS) scheme* for message set $\mathcal{M}$ is composed by a key generation algorithm $(otssk, otsvk) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$, a signing algorithm $sig \leftarrow \mathsf{OTS.Sign}(otssk, m)$ and a verification algorithm $b \leftarrow \mathsf{OTS.Ver}(otsvk, m, sig)$, $b \in \{0,1\}$.

Correctness requires that for all security parameters $\lambda \in \mathbb{N}$ it holds that:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{OTS.Ver}(otsvk, m, sig) \ : \ \begin{array}{c}(otssk, otsvk) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda), \\ sig \leftarrow \mathsf{OTS.Sign}(otssk, m)\end{array}\right] = 1 \ .$$

Definition A.8 (su-cma security of OTS). *A OTS scheme is said to be selectively unforgeable under chosen-message attacks (su-cma) if a PPT adversary* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *has negligible probability* $\epsilon_{\mathsf{su\text{-}cma}}$ *in winning the experiment* $\mathsf{Exp}^{\mathsf{su\text{-}cma}}_{\mathsf{A}}(\lambda)$ *in Fig. 6.*

A.6 One-Time Signatures from LWE

In this section we show how to construct a OTS from lattices whose security relies on the LWE problem. This scheme is used in Appendix C.1 to construct a secure predicate for the predicate encryption scheme by Katz et al. [17].

Definition A.9. *The* LWE distribution $A_{\mathbf{s},\chi}$ *outputs pairs* $(\mathbf{a}, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ *such that* $b = \mathbf{as} + e \ mod \ q$ *for a uniformly random* $\mathbf{a}$ *in* $\mathbb{Z}_q^n$ *and* $e \xleftarrow{\$} \chi$.

The (average-case) $\mathrm{LWE}_{k,\chi}$ decisional problem *(in the normal form) on* $\mathbb{Z}_q^n$ *with distribution* χ *and* k *samples is to distinguish whether* k *pairs* $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_k, b_k)$ *were sampled from* $A_{\mathbf{s},\chi}$ *for a random choice of* $\mathbf{s} \xleftarrow{\$} \chi^n$ *or from the uniform distribution over* $\mathbb{Z}_q^n$.

[9] Pseudorandom essentially means that no PPT algorithm can distinguish it from a source of uniformly random bits with non-negligible advantage.

The construction is analogous to the Lamport OTS [20], where the one-way function returns two instances of the $\mathrm{LWE}_{1,\chi}$ decisional problem.

Let the message space be a bit $m = \{0,1\}$. Let χ, B_χ be parameters of the LWE distribution.

$(otssk, otsvk) \leftarrow \mathsf{OTS.KeyGen}(1^\lambda)$**.** Generate two LWE samples: $\mathbf{s}_0, \mathbf{s}_1, e_0, e_1 \leftarrow \chi$, $\mathbf{a}_0, \mathbf{a}_1 \xleftarrow{\$} \mathbb{Z}_q^n$, $b_b \leftarrow \mathbf{a}_b \mathbf{s}_b + e_b \bmod q$. Set the keys as $otsvk = (\mathbf{a}_0, \mathbf{a}_1, b_0, b_1)$ and $otssk = (\mathbf{s}_0, \mathbf{s}_1, e_0, e_1)$.

$sig \leftarrow \mathsf{OTS.Sign}(otssk, m)$**.** On input a message $m \in \{0,1\}$, return $sig = (\mathbf{s}_m, e_m)$.

$b \leftarrow \mathsf{OTS.Ver}(otsvk, m, sig)$**,** $b \in \{0,1\}$**.** Parse $sig = (\mathbf{s}, e)$ Return 1 if $b_m = \mathbf{a}_m \mathbf{s} + e \bmod q$ and $\|e\|_\infty \leq B_\chi$, and 0 otherwise.

Theorem A.10. *The OTS signature from LWE is weakly selective unforgeable if $LWE_{1,\chi}$ is hard.*

Proof. Let A be an adversary that breaks the selective unforgeability of the OTS. We show that an algorithm B can solve LWE exploiting A as follows. On input the message, B sets the public key of the scheme to be $(\mathbf{a}_m, \mathbf{a}_{1-m}, b_m, b_{1-m})$ where $(\mathbf{a}_{1-m}, b_{1-m})$ are from the LWE distribution, and the rest is honestly generated. B returns the signature and the public key to A. For the adversary to output a successful forgery it has to output a $\mathbf{s}^*$ and e^* such that $b_{1-m} = \mathbf{a}_{1-m}\mathbf{s}^* + e^* \bmod q$ and $\|e\|_\infty, \|\mathbf{s}\|_\infty \leq B_\chi$, thus finding a solution to LWE. If A returns a forgery, B returns LWE, otherwise $uniform$. B breaks LWE with the same success probability of A (as the simulated experiment is indistinguishable from the real one). □

Longer messages. The signature can be extended to sign messages in $\mathcal{M} = \{0,1\}^\ell$ in different ways. The standard paradigm would be to sign each bit of the message separately then padding the signatures together. However, this blows up the length of the signature considerably. A better way to do it without expanding the length of the signature is to extend it to a full-fledged (**deterministic**) digital signature: For every message $m \in \mathcal{M}$, the key generation generates $(\mathbf{s}_m, e_m) \leftarrow \chi$ and one LWE sample $(\mathbf{a}_m, b_m)$. The signing algorithm just on m simply returns $(\mathbf{s}_m, e_m)$ on input m. Analogously to the LWE-based OTS, this signature is selectively unforgeable if $\mathrm{LWE}_{1,\chi}$ is hard.

Binary Secrets. The reader could wonder why we did not choose the LWE secret to be binary. The reason is that LWE with binary secrets in $\{0,1\}^n$ can be reduced to standard LWE with secrets in $\mathbb{Z}_q^k$ if $n = k \log q$, thus essentially the length of the secret would be exactly the same (cf. [11, 22]).

B Security Proof for $\mathsf{ACE}^{\mathsf{he}}$ (Theorem 5.1)

Proof. The proof is split in three parts: correctness, NRR security, and NSWR security.

Correctness. Directly follows from the correctness of the building blocks.

No Read Rule. Intuitively, the NRR follows from the NRR property of the ACEnoS and from the $\mathsf{lor\text{-}cpa}$ security of the SKE.

We treat the PP and SA case separately, to show that

$$
\begin{aligned}
\epsilon_{NRR} &= 2 \cdot \left| \Pr\left[(\mathsf{PP} \ \vee \ \mathsf{SA}) \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] - \frac{1}{2} \right| \\
&\leq 2 \cdot \left| \Pr\left[\mathsf{PP} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] + \Pr\left[\mathsf{SA} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] - \frac{1}{2} \right| \\
&\leq 2 \cdot \left| \Pr\left[\mathsf{PP} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] - \frac{1}{2} \right| + 2 \cdot \left| \Pr\left[\mathsf{SA} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda, \mathcal{P})\right] \right| \qquad (4) \\
&\leq \mathsf{negl}(\lambda) \ ,
\end{aligned}
$$

Payload Privacy. This case requires that for all queries $q = (j, \mathsf{rec}) \in J$ it holds that:

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) = 0 \ .$$

This implies that the adversary cannot get the secret keys corresponding to the public keys contained in ek_{i_0} and ek_{i_1}. The NRR security of the $\mathsf{ACE}^{\mathsf{he}}$ scheme in this case can be shown formally through a series of indistinguishable hybrid games, each of which consists of a slight modification of the original NRR security experiment (changes are highlighted in red).

Game 0: this game is the NRR-PP security experiment with $b = 0$.

Hybrid 0: in this hybrid game a simulator B runs A internally while simulating the security experiment. B does everything as specified in Theorem 3.2 except for the challenge ciphertext c^*, whose second component is now encrypted using the key of i_1 instead of i_0:

$$\begin{aligned} sk &\leftarrow \mathsf{SE.KeyGen}(1^\lambda) \\ c_1 &\leftarrow \mathsf{SE.Enc}_{sk}(m_0) \\ c_2 &\leftarrow \mathsf{ACE.Enc}(pp, ek_{i_1}, sk) \\ c^* &\leftarrow (c_1, c_2) \ . \end{aligned}$$

Hybrid 1: in this game B again acts as in Hybrid 0 except when generating the challenge ciphertext:

$$\begin{aligned} sk &\leftarrow \mathsf{SE.KeyGen}(1^\lambda) \\ c_1 &\leftarrow \mathsf{SE.Enc}_{sk}(m_0) \\ \rho &\xleftarrow{\$} \{0,1\}^{|sk|} \\ c_2 &\leftarrow \mathsf{ACE.Enc}(pp, ek_{i_1}, \rho) \\ c^* &\leftarrow (c_1, c_2) \ . \end{aligned}$$

Now B sets the second component of c^* to be the encryption of a random string instead of encrypting the key sk used to generate the first component of c^*; B still uses the key of i_1 to encrypt it, as in Hybrid 0.

Hybrid 2: in this game B acts exactly as in Hybrid 1, except when generating the challenge ciphertext, whose first component is now an encryption of m_1 instead of m_0:

$$\begin{aligned} sk &\leftarrow \mathsf{SE.KeyGen}(1^\lambda) \\ c_1 &\leftarrow \mathsf{SE.Enc}_{sk}(m_1) \\ \rho &\xleftarrow{\$} \{0,1\}^{|sk|} \\ c_2 &\leftarrow \mathsf{ACE.Enc}(pp, ek_{i_1}, \rho) \\ c^* &\leftarrow (c_1, c_2) \ . \end{aligned}$$

Hybrid 3: in this game B acts exactly as in Hybrid 2, except that now the second component of the challenge ciphertext is the encryption under the key of i_1 of the secret key sk used to encrypt m_1:

$$\begin{aligned} sk &\leftarrow \mathsf{SE.KeyGen}(1^\lambda) \\ c_1 &\leftarrow \mathsf{SE.Enc}_{sk}(m_1) \\ c_2 &\leftarrow \mathsf{ACE.Enc}(pp, ek_{i_1}, sk) \\ c &\leftarrow (c_1, c_2) \ . \end{aligned}$$

Game 1: this game is the NRRPP security experiment with $b = 1$.

We now proceed to prove that these games are in fact computationally indistinguishable; this then implies that (under computational assumptions) there is no adversary that can win the NRR-PP experiment. Let $\mathcal{Q}_G$ be the set of all queries $q = (j, t)$ that the adversary issues to the oracle $\mathcal{O}_G$, and J be the subset of $\mathcal{Q}_G$ composed by all the queries $q = (j, \mathsf{rec})$. Let $\mathbf{G}_i$ (resp. $\mathbf{H}_i$) be the probability that A returns 1 in Game (resp. Hybrid) i.

Game 0 ≈ Hybrid 0. Indistinguishability is guaranteed by the SA property of the underlying ACEnoS scheme used as building block. Indeed, if A can distinguish between these games, then B can exploit it to break the NRR-SA security of ACEnoS as follows.

public parameters pp**:** B sends the public parameters pp of the ACEnoS to A.

$\mathcal{O}_G(j,t)$**:** B queries (j,t) to the key generation oracle of the NRR-SA experiment of the ACEnoS, and returns its response to A.

$\mathcal{O}_E(j,m)$**:** B samples a secret key sk and uses it to generate the encryption c_1 of m, then it queries (j,sk) to the encryption oracle of the NRR-SA experiment of the ACEnoS to get c_2, and returns (c_1,c_2) to A.

c^***:** when A returns (m_0,m_1,i_0,i_1), B generates a secret key sk, and sends (sk,i_0,i_1) to the challenger of the NRR-SA experiment of the ACEnoS. Upon receiving c_b from the challenger, B sets the challenge ciphertext to be $c^* \leftarrow (\mathsf{SE.Enc}_{sk}(m_0), c_b)$.

The simulated oracles are identical to the real ones, so the view of A is the same as in the real game. Moreover, as B is simulating the NRR experiment for $\mathsf{ACE}^{\mathsf{he}}$ *in the* PP *case*, A can only query receivers keys to $\mathcal{O}_G$ for identities that cannot communicate with i_0 and i_1; thus the queries of B in the NRR-SA experiment of ACEnoS will satisfy the required restriction. Now, if $b=0$ (resp. $b=1$) in the NRR-SA experiment, then the game simulated by B is exactly Game 0 (resp. Hybrid 0). Thus the advantage of B in winning the security experiment is exactly equal to the success probability of A in distinguishing Game 0 from Hybrid 0:

$$|\mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{G}_0] - \mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{H}_0]| = \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} \ .$$

Hybrid 0 ≈ Hybrid 1. It is not hard to see that distinguishing these games is equal to break the NRR of the ACEnoS in the PP case where $i_0 = i_1$. Indeed, B can break it exploiting A by simulating the experiment as follows.

public parameters pp**:** B sends the public parameters pp of the ACEnoS to A.

$\mathcal{O}_G(j,t)$**:** B queries (j,t) to the key generation oracle of the NRR-PP experiment of the ACEnoS, and returns its response to A.

$\mathcal{O}_E(j,m)$**:** B samples a secret key sk and uses it to generate the encryption c_1 of m, then it queries (j,sk) to the encryption oracle of the NRR-PP experiment of the ACEnoS to get c_2, and returns (c_1,c_2) to A.

c^***:** when A returns (m_0,m_1,i_0,i_1), B generates a secret key sk and a random string ρ of equal length, and sends (sk,ρ,i_1,i_1) to the challenger of the NRR-PP experiment of the ACEnoS. Upon receiving c_b from the challenger, B sets the challenge ciphertext to be $c^* \leftarrow (\mathsf{SE.Enc}_{sk}(m_0), c_b)$.

Analogously as before, the view of A is exactly the same as when given access to the real oracles, and the queries of A to $\mathcal{O}_G$ do not require B to break the PP condition when querying the key generation oracle in the NRR-PP experiment of the ACEnoS. Now, if $b=0$ (resp. $b=1$) in the NRR-PP experiment, then the game simulated by B is exactly Hybrid 0 (resp. Hybrid 1). Thus the advantage of B in winning the security experiment is exactly equal to the success probability of A in distinguishing Hybrid 0 from Hybrid 1:

$$|\mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{H}_0] - \mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{H}_1]| = \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{PP}} \ .$$

Hybrid 1 ≈ Hybrid 2. Distinguishing these games requires to break the lor-cpa security pf the SKE. Indeed, if A can distinguish with non-negligible advantage, B can win the lor-cpa experiment exploiting A as follows. B generates the public parameters pp and the master secret key msk of the ACEnoS honestly, and uses them to simulate the oracles according to the NRR experiment specifications. When it is time to generate the challenge ciphertext c^*, B sends the messages (m_0,m_1) sent by A as challenge messages in the lor-cpa game, thus receiving a ciphertext c_b. Finally, B generates the random string ρ, and sends $c^* \leftarrow (c_b, \mathsf{ACE.Enc}(pp, ek_{i_1}, \rho))$ to A. If $b=0$ (resp. $b=1$) in the lor-cpa experiment, then the game simulated by B is exactly equal to Hybrid 0 (resp. Hybrid 1). Hence,

$$|\mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{H}_1] - \mathsf{Pr}[\mathsf{PP} \ : \ \mathbf{H}_2]| = \epsilon_{\mathsf{lor\text{-}cpa}} \ .$$

Hybrid 2 ≈ Hybrid 3. These games are again indistinguishable because of the NRR-SA security of ACEnoS:

$$|\mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_2\right] - \mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_3\right]| = \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} \ .$$

The proof follows the line of the proof that Hybrid 0 and Hybrid 1 are indistinguishable, thus we omit it.

Hybrid 3 = Game 1. It is easy to see that the two games are exactly equal.

Thus it holds that

$$\begin{aligned}
&2\cdot\left|\mathsf{Pr}\left[\mathsf{PP} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda,\mathcal{P})\right] - \frac{1}{2}\right| = \\
&= 2\cdot\left|\frac{1}{2}\cdot\mathsf{Pr}\left[\mathsf{PP} \ : \ 1 \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},1}(\lambda,\mathcal{P})\right] + \frac{1}{2}\cdot\mathsf{Pr}\left[\mathsf{PP} \ : \ 0 \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},0}(\lambda,\mathcal{P})\right] - \frac{1}{2}\right| \\
&= \left|\mathsf{Pr}\left[\mathsf{PP} \ : \ 1 \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},1}(\lambda,\mathcal{P})\right] + (1 - \mathsf{Pr}\left[\mathsf{PP} \ : \ 1 \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},0}(\lambda,\mathcal{P})\right]) - 1\right| \\
&= |\mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{G}_1\right] - \mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{G}_0\right]| \\
&\leq |\mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{G}_1\right] - \mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_3\right]| + \sum_{i=0}^{2} |\mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_{i+1}\right] - \mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_i\right]| \\
&\quad + |\mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{H}_0\right] - \mathsf{Pr}\left[\mathsf{PP} \ : \ \mathbf{G}_0\right]| \\
&= 2\cdot\epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} + \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{PP}} + \epsilon_{\mathsf{lor\text{-}cpa}} \ . \qquad (5)
\end{aligned}$$

Sender Anonymity. This case requires that for all queries $q = (j, \mathsf{rec}) \in J$ it holds that:

$$\mathcal{P}(i_0, j) = \mathcal{P}(i_1, j) \text{ and } m_0 = m_1 \ .$$

Proving NRR security of the $\mathsf{ACE}^{\mathsf{he}}$ scheme in this case is quite easy. Indeed, observe that in the SA case the challenge ciphertext encrypts the same message both when $b = 0$ and when $b = 1$, as $m_0 = m_1$. Thus, the difference in the challenge ciphertext c is only the encryption key used to generate the second component c_2: when $b = 0$ c_2 is the encryption of the secret key sk under the key ek_{i_0}, while when $b = 1$ c_2 is the encryption of the secret key sk under the key ek_{i_1}. Thus, winning the NRR-SA experiment for the $\mathsf{ACE}^{\mathsf{he}}$ scheme is exactly equal to breaking the NRR-SA security of the ACEnoS:

$$\mathsf{Pr}\left[\mathsf{SA} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda,\mathcal{P})\right] = \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} \ . \qquad (6)$$

Therefore, combining Equation (4), (5), and (6) yields that the advantage of A in breaking the NRR security of the $\mathsf{ACE}^{\mathsf{he}}$ scheme is negligible:

$$\begin{aligned}
\epsilon_{NRR} &\leq 2\cdot\left|\mathsf{Pr}\left[\mathsf{PP} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda,\mathcal{P})\right] - \frac{1}{2}\right| + 2\cdot\left|\mathsf{Pr}\left[\mathsf{SA} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr}}_{\mathsf{A},b}(\lambda,\mathcal{P})\right]\right| \\
&\leq 4\cdot\epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} + \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{PP}} + \epsilon_{\mathsf{lor\text{-}cpa}} \ .
\end{aligned}$$

No Secret Write Rule. The NSWR security of the $\mathsf{ACE}^{\mathsf{he}}$ relies on the NSWR security and on the NRR-PP security of ACEnoS. To see this, observe that in the NSWR definition (cf. Theorem 3.3) the challenge ciphertext $\bar{s}$ *can be of any length.* This is crucial to show that the NSWR experiment for the ACEnoS is essentially equivalent to the NSWR experiment for the $\mathsf{ACE}^{\mathsf{he}}$. More in details, the proof is constructive, and has two steps: First, we prove that any adversary $(\mathsf{A}^{\mathsf{he}}_1, \mathsf{A}^{\mathsf{he}}_2)$ that wins the NSWR experiment for the $\mathsf{ACE}^{\mathsf{he}}$ with some probability immediately yields a pair $(\mathsf{A}_1, \mathsf{A}_2)$ that wins the NSWR experiment for the ACEnoS with the same probability. The key fact here is that the algorithms A_1 and $\mathsf{A}^{\mathsf{he}}_1$ are equal. This allows to show in the second step that a simulator B such that $(\mathsf{A}_1, \mathsf{B})$ win the experiment for ACEnoS with almost the same probability as $(\mathsf{A}_1, \mathsf{A}_2)$ can be used to construct a simulator B^{he} such that $(\mathsf{A}^{\mathsf{he}}_1, \mathsf{B}^{\mathsf{he}})$ wins the NSWR experiment for $\mathsf{ACE}^{\mathsf{he}}$ with almost the same probability as $(\mathsf{A}^{\mathsf{he}}_1, \mathsf{A}^{\mathsf{he}}_2)$. Thus, assuming that NSWR holds for

ACEnoS (i.e., for every pair of successful $(\mathsf{A}_1, \mathsf{A}_2)$ such a B exists), then the NSWR holds also for $\mathsf{ACE}^{\mathsf{he}}$ (i.e., a simulator B^{he} exists for every $(\mathsf{A}_1^{\mathsf{he}}, \mathsf{A}_2^{\mathsf{he}})$).

A successful adversary $\mathsf{A}^{\mathsf{he}} = (\mathsf{A}_1^{\mathsf{he}}, \mathsf{A}_2^{\mathsf{he}})$ in the NSWR experiment of the $\mathsf{ACE}^{\mathsf{he}}$ can be converted into a successful adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ in the NSWR experiment of the ACEnoS as follows. The corrupted sender algorithm A_1 runs $\mathsf{A}_1^{\mathsf{he}}$ internally, using its access to a key generation oracle to answer the queries of $\mathsf{A}_1^{\mathsf{he}}$. As the keys are identical in both the $\mathsf{ACE}^{\mathsf{he}}$ and the ACEnoS, the view of $\mathsf{A}_1^{\mathsf{he}}$ is exactly the same as in the real experiment. Then it outputs the same pair $(\bar{m}, \bar{s})$ output by $\mathsf{A}_1^{\mathsf{he}}$. As there is no limit on the length of the ciphertext containing the subliminal message, this pair is accepted as a valid challenge ciphertext in both NSWR experiments. The corrupted receiver A_2 simply gives $\bar{s}$ as input to $\mathsf{A}_2^{\mathsf{he}}$, and uses its access to the encryption and key generation oracle to answer queries from $\mathsf{A}_2^{\mathsf{he}}$:

Receiver Key: whenever $\mathsf{A}_2^{\mathsf{he}}$ queries a receiver key, A_2 returns the answer of the ACEnoS key generation oracle.

Encryption: when $\mathsf{A}_2^{\mathsf{he}}$ queries for the encryption of a message m with the key of the i-th sender, A_2 first generates a random key sk for the SKE, then it queries its encryption oracle on (i, sk) to get the ciphertext c_2, and finally it sets c_1 to be the SKE encryption of m under sk.

Finally, A_2 outputs the message m' returned by $\mathsf{A}_2^{\mathsf{he}}$. As A perfectly simulates the experiment, and does exactly the same queries as A^{he} to the key generation oracle, it wins the experiment without breaking the NCR with the same probability as A^{he}:

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACEnoS}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P}) \ \wedge \ \mathsf{NCR}\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{A}_2^{\mathsf{he}})}(\lambda, \mathcal{P}) \ \wedge \ \mathsf{NCR}\right] . \tag{7}$$

To conclude the proof, we need to show that an algorithm B that extracts the subliminal message in the NSWR experiment for ACEnoS without having access to receivers keys with probability

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACEnoS}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right] \geq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACEnoS}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P})\right] - \mathsf{negl}(\lambda) , \tag{8}$$

can be used to construct an algorithm B^{he} that extracts $\bar{m}$ from $\bar{s}$ in the NSWR of the $\mathsf{ACE}^{\mathsf{he}}$ scheme with probability

$$\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{B}^{\mathsf{he}})}(\lambda, \mathcal{P})\right] \geq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{A}_2^{\mathsf{he}})}(\lambda, \mathcal{P})\right] - \mathsf{negl}(\lambda) .$$

On input $\bar{s}$, B^{he} simulates the view of B as follows:

Input: B receives as input the public parameters pp of the $\mathsf{ACE}^{\mathsf{he}}$ (which are the same as the public parameters of the ACEnoS scheme by construction) and the challenge ciphertext $\bar{s}$.

Encryption Oracle: when B queries (j, m) to the encryption oracle of the ACEnoS, B^{he} queries (j, m) to the encryption oracle of the $\mathsf{ACE}^{\mathsf{he}}$ to get (c_1, c_2) and returns c_2 to B. Remark that the encryption oracle of the $\mathsf{ACE}^{\mathsf{he}}$ experiment returns an ACEnoS encryption of a random string (the secret key used to encrypt the message), not of the queried message. We show later in the proof that distinguishing the real from the simulated oracle amounts to breaking the NRR-PP security of the ACEnoS.

Finally, B^{he} returns the message m' returned by B. Clearly B^{he} wins the experiment with the same probability of B, assuming that the simulated experiment is indistinguishable from the real one. Denote by $\mathbf{E}$ the case in which B distinguishes the real from the simulated experiment. The winning probability of B^{he} becomes

$$\begin{aligned} \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{B}^{\mathsf{he}})}(\lambda, \mathcal{P})\right] &= \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACEnoS}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right] - \mathsf{Pr}\left[\mathbf{E}\right] \\ &\geq \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{A}_2^{\mathsf{he}})}(\lambda, \mathcal{P})\right] - \mathsf{Pr}\left[\mathbf{E}\right] - \mathsf{negl}(\lambda) , \end{aligned} \tag{9}$$

where the first equality holds because the algorithm A_1 is equal to $\mathsf{A}_1^{\mathsf{he}}$, and the second follows from Equation (7) and (8). Now, assume that B can distinguish between the simulated and real experiment with non

negligible probability ϵ_{B}. "Distinguishing" here means that the behavior of B is different in the real and simulated experiment, i.e., that $\epsilon_{\mathsf{B}} := \mathsf{Pr}\left[\mathbf{E}\right] = \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACEnoS}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda,\mathcal{P})\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw-ACE^{he}}}_{(\mathsf{A}_1^{\mathsf{he}},\mathsf{B}^{\mathsf{he}})}(\lambda,\mathcal{P})\right]$ (trivially implied by Equation (9)). Then one could break the NRR-PP security of ACEnoS exploiting $(\mathsf{A}_1^{\mathsf{he}},\mathsf{B})$ (remark that B is defined w.r.t. a corrupted sender algorithm A_1, which is exactly equal to $\mathsf{A}_1^{\mathsf{he}}$). We show this through a sequence of hybrid games. Let q be the number of the queries asked by B.

Game 0 = Hybrid 0: this is the real NSWR experiment for $\mathsf{ACE}^{\mathsf{he}}$.

Hybrid k, $k \in \{1,\ldots,q-1\}$: in this game everything is as in Game 0 except for the encryption oracle. Indeed, upon receiving (j,m) from B, the oracle behaves as follows.

- **query $i \leq k$:** generate the ciphertext as in Game 0.
- **query $i > k$:** generate the ciphertext as in Game 1, i.e., as an encryption of a random key $sk \leftarrow \mathsf{SE.KeyGen}(1^\lambda)$.

Game 1 = Hybrid q: in this game the view of B is simulated by B^{he}.

Let $\mathbf{H}_k$ be the event "B returns $m' = \bar{m}$ in Hybrid k". By construction,

$$\left|\mathsf{Pr}\left[\mathbf{G}_1\right] - \mathsf{Pr}\left[\mathbf{G}_0\right]\right| = \epsilon_{\mathsf{B}} \ .$$

Therefore, by the hybrid argument there exists $\bar{k} \in \{0,\ldots,q-1\}$ such that

$$\left|\mathsf{Pr}\left[\mathbf{H}_{\bar{k}+1}\right] - \mathsf{Pr}\left[\mathbf{H}_{\bar{k}}\right]\right| \geq \frac{\epsilon_{\mathsf{B}}}{q+1} - \mathsf{negl}(\lambda) \ .$$

Given $\bar{k}$, we show now how to construct an adversary D for the NRR-PP experiment of ACEnoS. D simulates the view of $(\mathsf{A}_1^{\mathsf{he}},\mathsf{B})$ as follows

Public parameters pp: the public parameters of the $\mathsf{ACE}^{\mathsf{he}}$ are set to be the pp of the ACEnoS, and given as input to $\mathsf{A}_1^{\mathsf{he}}$ (and, later on, to B).

$\mathcal{O}_G(j,t)$: when $\mathsf{A}_1^{\mathsf{he}}$ queries (j,sen) to the key generation oracle, the simulated oracle responds by querying the key generation oracle of the NRR-PP experiment of the ACEnoS. As the senders keys are the same for both the $\mathsf{ACE}^{\mathsf{he}}$ and ACEnoS, $\mathsf{A}_1^{\mathsf{he}}$ cannot distinguish between the real and simulated oracle.

$\mathcal{O}_E(j,m)$: when B queries (j,m), the oracle behaves as follows.

- **query $i < \bar{k}$:** generate the ciphertext by querying the key generation oracle in the NRR-PP game on (j,sen) to get the j-th sender key, and then return $c \leftarrow \mathsf{ACE.Enc}(pp, ek_j, m)$.
- **query $i > \bar{k}$:** generate the ciphertext by querying the encryption oracle in the NRR-PP game on (j,m).
- **query $i = \bar{k}$:** on input (j,m), sample a random key $sk \leftarrow \mathsf{SE.KeyGen}(1^\lambda)$, send (m,sk,j,j) as challenge to the NRR-PP game, and return the challenge ciphertext c to B.

Essentially, the behavior at the query k depends on the bit b in the NRR-PP experiment: if $b = 0$, the game is exactly Hybrid $\bar{k}$, otherwise it is Hybrid $\bar{k}+1$.

D returns 1 if $(\mathsf{A}_1,\mathsf{B})$ win the NSWR experiment for the ACEnoS, 0 otherwise. Remark that the simulation requires to query only sender keys to the key generation oracle of the NRR-PP experiment, thus its behavior always satisfies the PP condition. The advantage of D in the NRR-PP experiment is

$$\begin{aligned}\mathsf{Pr}\left[\mathsf{PP} \ : \ b' \leftarrow \mathsf{Exp}^{\mathsf{nr-ACEnoS}}_{\mathsf{D},b}(\lambda,\mathcal{P})\right] &= \epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{PP}} \\ &= \left|\mathsf{Pr}\left[\mathbf{H}_{\bar{k}+1}\right] - \mathsf{Pr}\left[\mathbf{H}_{\bar{k}+1}\right]\right| \geq \frac{\epsilon_{\mathsf{B}}}{q+1} - \mathsf{negl}(\lambda) \ ,\end{aligned}$$

which yields $\epsilon_{\mathsf{B}} \leq (q+1)\cdot\epsilon^{\mathsf{ACEnoS}}_{NRR-\mathsf{SA}} + \mathsf{negl}(\lambda)$, that concludes our proof.

□

C Linear ACEnoS from Predicate Encryption and Deterministic Signatures

In this section we present a construction (including a possible instantiation, cf. Appendix C.1) of ACEnoS that returns ciphertexts that are $O(n + \ell)$ and with complexity polylogarithmic in the number of parties. In fact, for particular policies this construction is more efficient than the one presented in Section 5: in Theorem C.6 we show that for policies such that $n_R \gg \min_{j\in[n_R]} |\{i \in [n_S] \ : \ \mathcal{P}(i, j) = 1\}| = O(\log n_S)$ the length of the ciphertext is $O(\log(n_S) + \ell)$, which is optimal.

The approach relies on Predicate Encryption (cf. [17]) and deterministic signatures, and it is akin to the ACE with sanitization for generic policies by Kim and Wu [18]. In fact, our construction can be extended to solve one of the problems left open in [18] (cf. Theorem C.9). In this section we first introduce two security properties (secure predicates and key-indistinguishability) of predicate encryption that are needed for our constructions, then show the construction itself. Finally, Appendix C.3 contains a possible instantiation of the scheme, and in particular a PE scheme that satisfies our new definition of key indistinguishability.

C.1 Key-Indistinguishable Predicate Encryption

PE is a special case of Attribute-based Encryption (ABE) where the attribute linked to the ciphertext is not leaked in the decryption procedure. It was formally introduced in [17], even if it was implicit already in [9].

Definition C.1 (Predicate Encryption). *A Predicate Encryption scheme (PE) for the class of predicates $\mathcal{F}$ over the set of attributes Σ and for message space $\mathcal{M}$ consists of four PPT algorithms* (PE.Setup, PE.KeyGen, PE.Enc, PE.Dec) *such that:*

$(PE.par, PE.msk) \leftarrow \mathsf{PE.Setup}(1^\lambda)$**:** *on input the security parameter 1^λ, outputs the public parameters $PE.par$ and the master secret key $PE.msk$.*

$PE.sk \leftarrow \mathsf{PE.KeyGen}(PE.msk, f)$**:** *on input the master secret key msk and a predicate $f \in \mathcal{F}$, outputs a decryption key $PE.sk$.*

$PE.c \leftarrow \mathsf{PE.Enc}(PE.par, m, \alpha)$**:** *on input the public parameters of the scheme $PE.par$, a message $m \in \mathcal{M}$ and an attribute $\alpha \in \Sigma$, outputs a ciphertext $PE.c$.*

$m \leftarrow \mathsf{PE.Dec}(PE.c, PE.sk)$**:** *on input a ciphertext $PE.c$ and a decryption key $PE.sk$, outputs a message $m \in \mathcal{M}$ or a special symbol $\perp$.*

Definition C.2 (Correctness). *A PE scheme* (PE.Setup, PE.KeyGen, PE.Enc, PE.Dec) *for $(\mathcal{M}, \mathcal{F}, \Sigma)$ is correct for all λ, $(PE.par, PE.msk) \leftarrow \mathsf{PE.Setup}(1^\lambda)$, $f \in \mathcal{F}$, $PE.sk \leftarrow \mathsf{PE.KeyGen}(PE.msk, f)$, $m \in \mathcal{M}$, and $\alpha \in \Sigma$:*

- *If $f(\alpha) = 1$ then $m \leftarrow \mathsf{PE.Dec}(\mathsf{PE.Enc}(PE.par, m, \alpha), PE.sk)$.*
- *If $f(\alpha) = 0$ then $\perp \leftarrow \mathsf{PE.Dec}(\mathsf{PE.Enc}(PE.par, m, \alpha), PE.sk)$.*

with all but negligible probability.

Security of PE has three different characteristics: selective/adaptive, IND-CPA/IND-CCA1/IND-CCA2, one-sided/attribute-hiding. We include the definition for the combination of them that we need in this work.

Definition C.3 (selective, IND-CPA AH for PE, original def from [17]). *Consider the following game between a challenger* C *and a stateful adversary* A*,*

$\mathsf{Exp}^{s-ind-ah}_{\mathsf{A}}(\lambda, \Sigma, \mathcal{F})$	***Oracle Definition***
$(\alpha_0, \alpha_1) \leftarrow \mathsf{A}(1^\lambda, \Sigma, \mathcal{F})$ *If* $(\alpha_0, \alpha_1) \notin \Sigma$ *abort.* $(PE.par, PE.msk) \leftarrow \mathsf{PE.Setup}(1^\lambda)$ $(m_0, m_1, st) \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{PE.KeyGen}, PE.msk}}(PE.par)$ $b \xleftarrow{\$} \{0, 1\}$, $cipher_b \leftarrow \mathsf{PE.Enc}(PE.par, m_b, \alpha_b)$ $b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{PE.KeyGen}, PE.msk}}(st, cipher_b)$ *If* $(\exists\ f_i \in \mathcal{Q} \ : \ f_i(\alpha_0) = f_i(\alpha_1) = 1) \ \wedge \ (m_0 \neq m_1)$ *return* 0. *ElseIf* $b' = b$ *Return* 1, *Else return* 0.	$\mathcal{O}_{\mathsf{PE.KeyGen}, PE.msk}(f_i)$ *If* $(f_i \in \mathcal{F}) \ \wedge \ (f_i(\alpha_0) = f_i(\alpha_1))$, $PE.sk \leftarrow \mathsf{PE.KeyGen}(PE.msk, f_i)$ *Store* $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{f_i\}$. *Else* $PE.sk \leftarrow \perp$. *Return* $PE.sk$.

A PE $(\mathsf{PE.Setup}, \mathsf{PE.KeyGen}, \mathsf{PE.Enc}, \mathsf{PE.Dec})$ *is (selective, IND-CPA) attribute hiding if for all PPT* A*, predicate sets* $\mathcal{F}$*, and attribute sets* Σ *there exists a negligible quantity* $\epsilon_{s-ind\text{-}ah} = \mathsf{negl}_{s-ind\text{-}ah}(\lambda)$ *such that*

$$2 \cdot \left| \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{A}}^{s-ind\text{-}ah}(\lambda)\right] - \frac{1}{2} \right| \leq \epsilon_{s-ind\text{-}ah} \ .$$

It is easy to see that this security definition is extremely similar to the No Read Rule: the winning condition is essentially NRR-Sender Anonymity (imagine $f_i = \mathcal{P}(\cdot, i)$), while the oracle condition enforces NRR-Payload Privacy.

Key Indistinguishability with Secure Predicate. We define key-indistinguishability property for PE schemes as a combination of key indistinguisability in the PKE case [7] and left-or-right indistinguishability for the SKE case [8]. Essentially, we want that, given two public parameters sets $PE.par_0$, $PE.par_1$, an adversary is not able to distinguish whether some decryption keys are generated from $PE.msk_0$ or $PE.msk_1$. As it is stated, the property clearly never holds: an adversary A can just generate a predicate f along with an attribute α such that $f(\alpha) = 1$, query the key generation oracle for a decryption key $PE.sk$ for f, generate two ciphertexts $PE.c_b \leftarrow \mathsf{PE.Enc}(PE.par_b, m, \alpha)$, $b \in \{0,1\}$ for a random message m, and return the bit b such that $PE.c_b$ can be correctly decrypted using $PE.sk$. To ensure that the definition is not trivially broken, we then need to require that generating such a pair of attribute and predicate is hard.

Definition C.4 (Secure Predicate Family). *A* secure predicate family *is a family of predicates* $\mathcal{F}$ *such that:*

Generation. *There exists a PPT algorithm* $(\mathcal{F}, \Sigma) \leftarrow \mathsf{PredGen}(1^\lambda)$ *that generates the family of predicates* $\mathcal{F}$*, and a collection* $\Sigma = \{\Sigma_f\}_{f \in \mathcal{F}}$ *of sets* Σ_f *of all the attributes satisfying each predicate.*

Security. *For all PPT algorithm* A *there exists a negligible quantity* $\epsilon_s = \mathsf{negl}(\lambda)$ *such that*

$$\Pr_{(\mathcal{F},\Sigma) \leftarrow \mathsf{PredGen}(1^\lambda)} (f(\alpha) = 1 \ : \ (\alpha, f) \leftarrow \mathsf{A}(\mathcal{F})) \leq \epsilon_s \ .$$

We now define key indistinguishability with respect to secure predicates. Essentially we require that, given two public parameters sets $PE.par_0$, $PE.par_1$ and a secure predicate family $\mathcal{F}$, an adversary should not be able to distinguish whether some decryption keys *and ciphertexts*[10] are generated from $(PE.par_0, PE.msk_0)$ or $(PE.par_1, PE.msk_1)$.

Definition C.5 (IK-CPA). *Let* $\mathcal{F}$ *be a secure predicate family. Consider the following game between a challenger* C *and an adversary* A*,*

$\mathsf{Exp}_{\mathsf{A},b}^{ik\text{-}cpa}(\lambda)$
$(\mathcal{F}, \Sigma) \leftarrow \mathsf{PredGen}(1^\lambda)$ $(PE.par_c, PE.msk_c) \leftarrow \mathsf{PE.Setup}(1^\lambda)$ *for* $c \in \{0,1\}$ $b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{PE.KeyGen},b},\ \mathcal{O}_{\mathsf{PE.Enc},b},\ \mathcal{O}_\Sigma}(PE.par_0, PE.par_1, \mathcal{F})$ *Return* b'*.*

Key Generation Oracle	***Encryption Oracle***	***Attribute Oracle***
$\mathcal{O}_{\mathsf{PE.KeyGen},b}(f)$*:* *If* $(f \notin \mathcal{F}) \ \vee \ (f \in \mathcal{Q}_\Sigma)$*, return* $\perp$*.* *Else If* $(\exists \ PE.sk \ : \ (f, PE.sk) \in \mathcal{Q})$ *return* $PE.sk$*.* *Else return* $PE.sk \leftarrow \mathsf{PE.KeyGen}(PE.msk_b, f)$ *Store* $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(f, PE.sk)\}$*.*	$\mathcal{O}_{\mathsf{PE.Enc},b}(m, f)$*:* *If* $(f \notin \mathcal{F})$*, return* $\perp$*.* *Else* $\alpha \xleftarrow{s} \Sigma_f$*,* *return cipher* $\leftarrow \mathsf{PE.Enc}(PE.par_b, m, \alpha)$*.*	$\mathcal{O}_\Sigma(f)$*:* *If* $(f \notin \mathcal{F}) \ \vee \ (\exists \ PE.sk \ : \ (f, PE.sk) \in \mathcal{Q})$*, return* $\perp$*.* *Else* $\alpha \xleftarrow{s} \Sigma_f$ *store* $\mathcal{Q}_\Sigma \leftarrow \mathcal{Q}_\Sigma \cup \{f\}$*.* *return* α*.*

[10] As we are interested in PE schemes that are attribute-hiding, we allow the adversary to see ciphertexts generated using random attributes (chosen by the encryption oracle) satisfying a predicate of its choice. The definition of key indistinguishability for PE that are partially or not hiding can be obtained from Theorem C.5 by removing the encryption oracle.

A PE $(\mathsf{PE.Setup}, \mathsf{PE.KeyGen}, \mathsf{PE.Enc}, \mathsf{PE.Dec})$ *satisfies the key-indistinguishability property if for all PPT* A *and attribute sets* Σ *there exists a negligible quantity* $\epsilon_{\textit{ik-cpa}} = \mathsf{negl}(\lambda)$ *such that*

$$\left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\textit{ik-cpa}}_{\mathsf{A},1}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\textit{ik-cpa}}_{\mathsf{A},0}(\lambda)\right]\right| \leq \epsilon_{\textit{ik-cpa}} \ .$$

Remark that the definition is quite weak, as the adversary is even allowed to query valid attributes w.r.t. some of the predicates. However, this requires to make sure that A does not get not the decryption keys corresponding to such predicates. Indeed, assume A could get both the decryption key $PE.sk_f$ corresponding to some predicate f and a valid attribute α w.r.t. f. Then it could trivially win the experiment by choosing a message m, generating $PE.c_0 \leftarrow \mathsf{PE.Enc}(PE.par_0, m, \alpha)$ and $PE.c_1 \leftarrow \mathsf{PE.Enc}(PE.par_1, m, \alpha)$, and checking which of the two ciphertexts can be decrypted using $PE.sk_f$.

C.2 Polylog ACEnoS from Predicate Encryption with Secure Predicates

Predicate encryption (PE) decrypts a ciphertext only if it is associated to a (possibly secret) attribute that verifies a certain predicate f. Intuitively, this is how a policy can be enforced: a ciphertext is decrypted only if the sender and the receiver are allowed to communicate[11]. This requires to add an *authentication layer* to verify the identities of the parties involved. During key generation each sender receives a signed identity along with the PE encryption key, and each receiver a decryption key connected to its identity. Such keys are generated so that decryption outputs a plaintext only if it is associated with a valid sender identity allowed to communicate with the receiver. The *attribute-hiding* property of the PE (i.e., ciphertexts do not leak information about the attribute they are linked to, cf. Appendix C.1) allows to keep the sender's signed identity hidden from the receiver, thus ensures the NRR. Essentially, encryption enforces secrecy, while authentication guarantees that the party is allowed to send messages to the decrypting party.

Thus, given the following building blocks,

- a OTS scheme $(\mathsf{OTS.KeyGen}, \mathsf{OTS.Sign}, \mathsf{OTS.Ver})$ that is selectively unforgeable and outputs signatures in a set $\mathcal{S}$.
- a predicate encryption scheme $(\mathsf{PE.Setup}, \mathsf{PE.KeyGen}, \mathsf{PE.Enc}, \mathsf{PE.Dec})$ that is attribute-hiding selective IND-CPA secure and IK-CPA for:
  Message Space: $\mathcal{M} := \{0,1\}^\ell$
  Attribute Space: $\Sigma := \{(i, sig_i)\}_{i\in[n_S]}$
  Predicate Space: $\mathcal{F} := \{f_i^S\}_{i\in[n_S]} \cup \{f_i^R\}_{i\in[n_R]}$, where $f_i^x : [n_S] \times \mathcal{S} \to 0,1$ for $x \in \{S, R\}$ is defined as:

$$f_j^S(i, sig) = ((i == j) \ \wedge \ (\mathsf{OTS.Ver}_{otsvk_i}(sig, i))) \tag{10}$$

$$f_j^R(i, sig) = (\mathcal{P}(i, j) \ \wedge \ (\mathsf{OTS.Ver}_{otsvk_i}(sig, i))) \tag{11}$$

  where $\mathcal{S}$ is the set of possible signatures output by the OTS, $\{otsvk_i\}_{i\in[n_S]}$ are verification keys of the OTS. The f_j^S's are not used in the construction, but they are needed when reducing breaking the NSWR to breaking the IK-CPA property of the PE.

we construct an ACE without sanitizer as follows.

Communication Model: same as before. There is a public bulletin board; only senders can write on it, receivers have read-only access.
Message Space: $\mathcal{M} := \{0,1\}^\ell$.

[11] The intuition is similar to the intuition behind compact ACE from iO in [12]. In fact, it is possible to also obtain a compact ACE *without* sanitizer from iO through a similar construction (cf. Section 7.2: removing verifiability from the construction presented there only influences the bound on the insecure subliminal channel bandwidth). The construction in this section is obtained substituting the iO and NIZK proof with Predicate Encryption (PE) (with an approach similar to [18]). Hence we can fit all the checks that were previously done by the obfuscated program in the f. By choosing a PE that is attribute-hiding, we do not need the NIZK proof.

Setup: $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{P})$

Generate the parameters and master secret key of the PE $(PE.par, PE.msk) \leftarrow \mathsf{PE.Setup}(1^\lambda)$, and the OTS key pairs for message space $[n_S]$: $(otssk_i, otsvk_i) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda, [n_S])$, $i \in [n_S]$. Let $otssk := \{otssk_i\}_{i\in[n_S]}$ and $otsvk := \{otsvk_i\}_{i\in[n_S]}$. The algorithm returns the public parameters $pp = (\lambda, \mathcal{M})$ and the master secret key $msk = (\mathcal{P}, PE.par, PE.msk, \mathcal{F}, \Sigma, otssk, otsvk)$.

Key Generation: $k_i \leftarrow \mathsf{KGen}(pp, msk, i, t)$

- If $i \neq 0$ and $t = \mathsf{sen}$, generate a signature on the identity $sig_i \leftarrow \mathsf{OTS.Sign}_{otssk_i}(i)$. Return $ek_i = (PE.par, i, sig_i)$.
- If $i \neq 0$ and $t = \mathsf{rec}$, return $dk_i = PE.sk_i \leftarrow \mathsf{PE.KeyGen}(PE.msk, f_i^R)$.
- If $i = 0$, return $ek_0 = dk_0 = pp$.

Encryption: $c \leftarrow \mathsf{Enc}(pp, ek_i, m)$

Encrypt the message using the PE key: $PE.c \leftarrow \mathsf{PE.Enc}(PE.par, m, (i, sig_i))$ (where the identity i and its signature compose the attribute). Return $c = PE.c$.

Decryption: $m' \leftarrow \mathsf{Dec}(pp, dk_j, c)$

Output $m' \leftarrow \mathsf{PE.Dec}(PE.c, dk_j)$.

Remark C.6. This construction returns ciphertexts that are $\mathrm{O}(n+\ell)$ bits long instead of $\mathrm{O}(n\ell)$ as in Section 4. In fact, our instantiation (cf. Appendix C.3) is linear in $\min_{j\in[n_R]} S_j + \ell$, where S_j is the set of senders allowed to communicate with the receiver j. Thus, for policies such that $\min_{j\in[n_R]} S_j = \mathrm{O}(\log n_S)$ this is already optimal. The scheme presented in Section 7.2 is better, as the ciphertext length is linear in $\log(n)$. Remark that this estimate is related to the current state of the art in PE: better, more efficient schemes might be available following a PE with better check. Finally, the runtime of the construction is polylog in the number of parties (as it generates only one ciphertext, which contains the identity of the sender as an attribute of the message).

Remarks about the Authentication Layer. We decided to use (one-time) signatures instead of MACs because the verification of the signature is included in the predicate. If we were to use a MAC, the predicate would depend on the secret key, and we would have to worry about leakage during decryption or in the decryption key. Using a signature we do not have to worry about the secrecy of the predicate in the PE, as the predicate is not generated from values that have to remain hidden to the receiver – besides $PE.par$ (whose leakage would break the secrecy of the encryption anyway). Moreover, using a *different* key pair for each sender allows to keep the keys of parties that are not allowed to communicate independent. Finally, remark that it is enough for the signature to be selectively weakly unforgeable, as in an impersonation attack the A does not even see a signature w.r.t. the verification key of the target sender i, but at most it sees vk_i.

Theorem C.7. *The protocol previously described is correct, and satisfies the properties of No-Read and No Undetectable Secret Write as described in Section 3 and Section 7.1 assuming that the signature is selectively unforgeable and the predicate encryption is attribute-hiding selective IND-CPA secure and key indistinguishable.*

Proof (sketch).

Correctness. From correctness of building blocks.

NRR. This follows from the attribute hiding property of the PE. Indeed, assume A is a PPT adversary that wins the NRR experiment with advantage ϵ_{A}. We construct a PPT algorithm B that wins the $\mathsf{ind\text{-}ah}$ experiment with advantage $\frac{\epsilon_{\mathsf{A}}}{2^{2n_S}}$, where the loss in the number of senders is due to the fact that the PE is only selectively secure; in case the PE is adaptively secure, B wins with the same advantage as A. B simulates the NRR experiment as follows:

- B generates the signature key pairs $(otssk_i, otsvk_i) \leftarrow \{\mathsf{OTS.KeyGen}(1^\lambda, [n_S])$ for all $i \in [n_S]$, guesses the identities (i'_0, i'_1), signs them, and sends $((i'_0, sig_{i_0}), (i'_1, sig_{i_1}))$ to the $\mathsf{ind\text{-}ah}$ challenger. Then it sends the public parameters pp (i.e., $\mathcal{M}$ and λ) to A.

– Upon receiving the public key $PE.par$ of the PE, B simulates the oracles as follows:
  $\mathcal{O}_G$**:** on input (i, sen) returns $(PE.par, i, \mathsf{OTS.Sign}_{otssk_i}(i))$. On input (j, rec) queries the key generation oracle of the ind-ah experiment on f_j^R and returns the oracle answer to A.
  $\mathcal{O}_E$**:** on input (i, m) generates the encryption key $ek_i = (PE.par, i, sig_i) \leftarrow \mathcal{O}_G(j, \mathsf{sen})$ and returns $PE.c \leftarrow \mathsf{PE.Enc}(PE.par, m, (i, sig_i))$ to A.
  When needed, B uses the previously generated sig_{i_0}, sig_{i_1}.
– When A returns (m_0, m_1, i_0, i_1), B aborts if $(i_0, i_1) \neq (i'_0, i'_1)$. Else it sends (m_0, m_1) to the PE challenger, and returns the challenge ciphertext $PE.c_b$ to A.
– B answers the oracle queries round as previously defined.
– Upon receiving the bit guess b from A, B sends it as bit guess in the ind-ah experiment.

To compute the advantage of B we need to analyze three points:

– whether the output of A satisfies the winning condition of the ind-ah experiment: to win the ind-ah experiment the queries and challenge messages/identities chosen by A should be such that: $f_j^R(i_0) = f_j^R(i_1)$ for all (j, rec) queried to the key generation oracle and, if $f_j^R(i_0) = f_j^R(i_1) = 1$, then $m_0 = m_1$. These are exactly the condition of PP and SA in the NRR experiment: thus the interaction of B with the ind-ah challenger does not end in an abort.
– whether A can distinguish the simulated from the original NRR experiment: the experiment simulated by B is exactly equal to the NRR experiment, thus A cannot distinguish.
– whether B aborts: this happens only if B wrongly guessed the challenge identities, thus with probability $\frac{1}{2^{2n_S}}$.

Hence the advantage of B is $\epsilon_{\mathsf{A}} 2^{-2n_S}$.

NSWR. Assume that there is a successful adversary in the NSWR experiment. Then the algorithm B could win the game with essentially the same probability as A, by running A_2 as a subroutine, simulating the oracles as follows: B generates the signature key pairs $(otssk_i, otsvk_i) \leftarrow \{\mathsf{OTS.KeyGen}(1^\lambda, [n_S])$ for all $i \in [n_S]$, and a pair of keys for the predicate encryption $(PE.par, PE.msk) \leftarrow \mathsf{PE.Setup}(1^\lambda, ([n_S], \mathcal{S}))$ and,

$\mathcal{O}_G$**:** on input (j, rec) from A_2, it generates keys according to the policy $\mathcal{P}$ using $(\{otsvk_i\}_{i \in [n_S] \,:\, \mathcal{P}(i,j)=1}$, $PE.par, PE.msk)$. The keys are stored in a list $\mathcal{K}$.
$\mathcal{O}_E$**:** on input (i, m), B executes the honest encryption oracle as specified in the security experiment generating the encryption key of the sender i using $(otssk_i, PE.par)$. It stores the keys (with corresponding identity and signature) in the list $\mathcal{K}$.

B outputs whatever A_2 returns, thus wins with essentially the same probability. To estimate the loss in the success probability, we need to analyze the probability that A_2 can distinguish between the real and simulated oracles. We do this through a series of games.

Game 0**:** this is the NSWR experiment.
Game 1**:** same as Game 0, but now the key generation oracle is split into two independent oracles, $\mathcal{O}_{K,1}$ accessed by A_1 and $\mathcal{O}_{K,2}$ accessed by A_2, that do not share state. In particular, this means that the verification keys used by $\mathcal{O}_{K,2}$ and the signatures on the senders identities used by $\mathcal{O}_E$ are not the same used by $\mathcal{O}_{K,1}$.
Game 2**:** same as Game 1, but now $\mathcal{O}_{K,2}$ generates the keys using a freshly generated pair $(PE.par', PE.msk')$. This is the NSWR experiment in which the oracles accessed by A_2 are in fact simulated by B.

Let $\mathbf{G}_i$ be the probability that A_2 returns a message $m' = \bar{m}$ in Game i. We show that:

$$\begin{aligned}
&\left|\mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{A}_2)}(\lambda, \mathcal{P}) \ \wedge \ \mathsf{NCR}\right] - \mathsf{Pr}\left[1 \leftarrow \mathsf{Exp}^{\mathsf{nsw}}_{(\mathsf{A}_1,\mathsf{B})}(\lambda, \mathcal{P})\right]\right| \\
&= |\mathsf{Pr}(\mathbf{G}_0) - \mathsf{Pr}(\mathbf{G}_1) + \mathsf{Pr}(\mathbf{G}_1) - \mathsf{Pr}(\mathbf{G}_2)| \\
&\leq \left| q_2^E \epsilon_{\mathsf{ind\text{-}ah}} + \frac{\epsilon_{\mathsf{ik\text{-}cpa}}}{1 - n_S \cdot \epsilon_{\mathsf{su\text{-}cma}}} \right| .
\end{aligned}$$

Indeed it holds that

Game $0 \approx$ **Game** 1. The decryption keys queried by A_2 do not make use of the $\{otsvk_i\}_i$ generated to answer queries from A_1, as the two adversaries are not allowed to corrupt parties that can communicate. Thus, the fact that $\mathcal{O}_{K,2}$ does not know the verification keys generated by $\mathcal{O}_{K,1}$ is undetectable by A_2. The only way that A could distinguish is if it could tell that the signatures used by $\mathcal{O}_{K,1}$ are different from the signatures used by $\mathcal{O}_E$. This can only happen in case A_2 queries $\mathcal{O}_E$ on a (j, m) where j was queried by A_1 to $\mathcal{O}_{K,1}$. Then there exists an algorithm C that can break the ind-ah property of the PE scheme as follows. C sets the challenge messages in the ind-ah experiment to be $m_0 \leftarrow (j, sig, m)$ and $m_1 \leftarrow (j, sig', m)$ (where sig is the signature given to A_1 as part of the j-th sender's key). Then it sets the answer to the query (j, m) as the challenge ciphertext of the ind-ah experiment. If A_2 returns $m' = \bar{m}$ C returns 1, otherwise 0. The success probability of C is $\frac{1}{q_2^E}(\mathsf{Pr}(\mathbf{G}_1) - \mathsf{Pr}(\mathbf{G}_0)) \leq \epsilon_{s-\mathsf{ind\text{-}ah}}$, where q_2^E is the number of queries by A_2 to $\mathcal{O}_E$.

Game $1 \approx$ **Game** 2. Distinguishing these games is essentially the same as winning the ik-cpa experiment for the PE scheme. An algorithm C running A as a black-box can win the ik-cpa experiment with probability: $\epsilon_{\mathsf{ik\text{-}cpa}} \geq \mathsf{Pr}(\mathbf{G}_2) - \mathsf{Pr}(\mathbf{G}_1)$. Indeed, upon receiving $\mathcal{F}$, $PE.par_0$, and $PE.par_1$ from the ik-cpa experiment, C simulates the adversary's inputs and the oracles as follows:

Input to A_1**:** λ and $\mathcal{M}$ from the PE scheme.

$\mathcal{O}_{K,1}$**:** on input i, if there is an entry $(i, sig_i) \in \mathcal{Q}_K^1$, returns $(PE.par_0, i, sig_i)$ to A_1. Otherwise, C queries the attribute oracle $\mathcal{O}_\Sigma$ on (f_i^s) to get a signature sig_i on i. C stores (i, sig_i) in $\mathcal{Q}_K^1$, and sends $(PE.par_0, i, sig_i)$ to A_1 (C aborts if $i \notin [n_S]$).

Input to A_2**:** upon receiving $(\bar{m}, \bar{s})$ from A_1, C stores $\bar{m}$ and gives $\bar{s}$ in input to A_2 (along with λ and $\mathcal{M}$).

$\mathcal{O}_{K,2}$**:** on input j, queries the key generation oracle of the ik-cpa experiment on f_j^R and returns the oracle answer to A_2 (aborts if $j \notin [n_R]$). As A_2 is not allowed to communicate with A_1, no attribute queried by C to $\mathcal{O}_\Sigma$ is valid w.r.t. any of the f_j^R queried to the key generation oracle of the ik-cpa experiment. Thus C can always return a key to A_2.

$\mathcal{O}_E$**:** on input (j, m), queries the encryption oracle of the ik-cpa experiment on (m, f_j^S), and returns the oracle answer to A_2 (aborts if $j \notin [n_S]$).

Upon receiving m' from A_2, C returns 1 if $m' = \bar{m}$, and 0 otherwise. Thus the winning probability of C is $(1 - \epsilon_s)(\mathsf{Pr}(\mathbf{G}_2) - \mathsf{Pr}(\mathbf{G}_1)) \leq \epsilon_{\mathsf{ik\text{-}cpa}}$, where ϵ_s is the probability of breaking the security of the predicate family $\mathcal{F}$. In Theorem C.8 we show that the family $\mathcal{F}$ defined in Equation (10) is in fact a secure predicate family, and that $\epsilon_s \leq \epsilon_{\mathsf{su\text{-}cma}} \cdot n_S$. Thus, $\mathsf{Pr}(\mathbf{G}_2) - \mathsf{Pr}(\mathbf{G}_1) \leq \frac{\epsilon_{\mathsf{ik\text{-}cpa}}}{1 - n_S \cdot \epsilon_{\mathsf{su\text{-}cma}}}$.

Lemma C.8. *Given a OTS that is weekly selectively unforgeable, the predicate family* $\mathcal{F} := \{f_i^S\}_{i \in [n_S]} \cup \{f_i^R\}_{i \in [n_R]}$ *with* n_S *subexponential in* λ*, defined in Equation* (10) *is a secure predicate family (cf. Theorem C.4).*

Proof (Sketch.). The generation algorithm for $(\mathcal{P}, n_S, n_R)$ simply generates the signature key pair and a signature on each $i \in [n_S]$, then generates the f_i^x's in $\mathcal{F}$ according to Equation (10).

Security follows from the unforgeability of the signature. Assume there exists a PPT algorithm A that on input $\mathcal{F}$ returns (i, sig_i) such that $f_i^x(i, sig_i) = 1$ with probability ϵ_s. Then an algorithm B can win the su-cma experiment in Fig. 6 as follows. B commits to a random message $\bar{i} \in [n_S]$ in the su-cma experiment, and gets back $ot\bar{s}vk$. Then it sets $otsvk_{\bar{i}} = ot\bar{s}vk$, generates the rest of the OTS keypairs, and constructs the predicates as specified in Equation (10). Finally, it runs $\mathsf{A}(\mathcal{F})$. There are two cases:

- If A returns $((i, sig), f_k^S)$, this means that $i = k$ and sig is a valid signature on i w.r.t. $otsvk_i$.
- If A returns $((i, sig), f_j^R)$, this means that $\mathcal{P}(i, j) = 1$, and sig is a valid signature on i w.r.t. $otsvk_i$.

In both cases, if $i = \bar{i}$ B can submit (i, sig) as a forgery and wins with probability $\epsilon_{\mathsf{B}} = \frac{\epsilon_s}{n_S}$. Thus, $\epsilon_{\mathsf{B}} = \epsilon_{\mathsf{su\text{-}cma}} \geq \frac{\epsilon_s}{n_S}$, and $\epsilon_s \leq \epsilon_{\mathsf{su\text{-}cma}} \cdot n_S$. □

Remark C.9. Our construction guarantees full anonymity: the identity of the sender is hidden even from the receiver of the message (NRR-SA case). This is different from the construction by Kim and Wu [18]: in fact, how to build an ACE scheme for general policies with such a strong anonymity guarantee was left as an open

problem. We indirectly solve such problem: our construction can be modified to get a fully anonymous ACE *with* sanitization by adding the sanitization layer from Kim and Wu, i.e. the functional encryption scheme. The main reason for it is that our scheme relies on an attribute-hiding IND-CPA PE, instead of a one-sided IND-CPA predicate encryption scheme.

C.3 Instantiations

Not all PE scheme are suitable to instantiate this ACE scheme. Beyond the obvious security requirements, there are some small design details that are pivotal in the security reduction, but might get overlooked when designing a PE. In particular, a suitable PE should generate the keys without having access to the predicates, which why we choose the construction by Katz et al. [17] over the construction by Baltico et al. [5][12]. All other existing PE cannot be used as they either only guarantee one-sided security (it does not guarantee that the attribute is hidden if it satisfies the predicate, cf. [1, 19]), or because they require part of the attribute to decrypt (partially hiding security, cf. [16, 26, 13]). In general, it seems that one cannot use schemes that have simulation based security definitions, as they do not allow the adversary to query keys after it has gotten the challenge ciphertext (so e.g. [2] does not work).

We now prove that scheme from [17] satisfies the IK-CPA property when instantiated with a family of secure predicates. Let us start by recalling the scheme in the following. What we present is a slight modification of the original protocol, where the generators of two of the three subgroups are part of the public parameters instead of part of the public key for the sake of simplicity.

Setup: Generate $N = pqr \in \mathbb{N}$, with p, q, r primes, the cyclic groups $\mathbb{G} = \mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$ and $\mathbb{G}_T$ of order N, and a pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Computes g_q, g_r, g_p generators of $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$ respectively, and random $\gamma \xleftarrow{\$} \mathbb{Z}_p$, $R_0, R_{1,i}, R_{2,i} \xleftarrow{\$} \mathbb{G}_r$, $h, h_{1,i}, h_{2,i} \xleftarrow{\$} \mathbb{G}_p$ for $i = 1, \ldots, n$. Let $(N, \mathbb{G}, \mathbb{G}_T, \hat{e}, g_r, g_p)$ be the public parameters, and let the public key $PE.par$ and the master secret key $PE.msk$ be:

$$\begin{aligned} PE.par &= (Q = g_q R_0,\ P = \hat{e}(g_p, h)^{\gamma},\ \{H_{1,i} = h_{1,i} \cdot R_{1,i},\ H_{2,i} = h_{2,i} \cdot R_{2,i}\}_{i=1,\ldots,n}) \\ PE.msk &= (p, q, r, g_q, h^{-\gamma},\ \{h_{1,i}, h_{2,i}\}_{i=1,\ldots,n})\ . \end{aligned}$$

Encryption: on input an attribute $\alpha = (\alpha_0, \ldots, \alpha_n) \in \mathbb{Z}_N^n$ and a message $m \in \mathbb{G}_T$, choose random $s, \alpha, \beta \xleftarrow{\$} \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \xleftarrow{\$} \mathbb{G}_r$ for $i = 1, \ldots, n$, and return the ciphertext

$$PE.c = \left(C = mP^s,\ C_0 = g_p^s,\ \left\{C_{1,i} = H_{1,i}^s Q^{\alpha\alpha_i} R_{3,i},\ C_{2,i} = H_{2,i}^s Q^{\beta\alpha_i} R_{4,i}\right\}_{i=1,\ldots,n}\right)\ .$$

Key Generation: on input $PE.msk$ and the predicate $f = (f_1, \ldots, f_n) \in \mathbb{Z}_N^n$, generate random $r_{1,i}, r_{2,i} \xleftarrow{\$} \mathbb{Z}_p$ for $i = 1, \ldots, n$, $R_5 \xleftarrow{\$} \mathbb{G}_r$, $\phi_1, \phi_2 \xleftarrow{\$} \mathbb{Z}_q$, $Q_6 \xleftarrow{\$} \mathbb{G}_q$. Return the decryption key

$$PE.sk_f = \left(K = R_5 Q_6 h^{-\gamma} \textstyle\prod_{i=1}^n h_{1,i}^{-r_{1,i}} h_{2,i}^{-r_{2,i}},\ \left\{K_{1,i} = g_p^{r_{1,i}} g_q^{\phi_1 f_i}, K_{2,i} = g_p^{r_{2,i}} g_q^{\phi_2 f_i}\right\}_{i=1,\ldots,n}\right)\ . \tag{12}$$

Decryption: on input $PE.c$ and $PE.sk_f$ as defined above, return

$$m' = C \cdot \hat{e}(C_0, K) \cdot \prod_{i=1}^{n} \hat{e}(C_{1,i} K_{1,i}) \cdot \hat{e}(C_{2,i} K_{2,i})$$

In the following theorem we show that given a family of secure predicates the PE is IK-CPA under the same assumptions that underlie its ind-ah security.

Theorem C.10. *The PE scheme from [17] is IK-CPA secure if the predicates family is a family of secure predicates and Assumption 1 and 2 from [17] hold.*

[12] Remark that one *can* adapt their construction to circumvent this limitation, but still would need to prove it satisfies key-indistinguishability as defined in this section.

Proof. We prove the statement with a series of game hops to transform the ik-cpa experiment with $b = 0$ into the ik-cpa experiment with $b = 1$. With each hybrid game, we will substitute one part of the ciphertexts or secret keys generated using elements of $PE.par_0$ with parts generated using the corresponding components of $PE.par_1$. The thesis essentially follows from the fact that the secret keys and ciphertexts computationally hide the master secret key. Let A be the adversary in the ik-cpa experiment, and let

$$PE.par_0 = \Big(Q^0 = g_q R_0^0,\ P^0 = \hat{e}(g_p, h^0)^{\gamma^0},\ \{H_{1,i}^0 = h_{1,i}^0 \cdot R_{1,i}^0,\ H_{2,i}^0 = h_{2,i}^0 \cdot R_{2,i}^0\}_{i=1,\dots,n}\Big), \tag{13}$$

$$PE.par_1 = \Big(Q^1 = g_q R_0^1,\ P^1 = \hat{e}(g_p, h^1)^{\gamma^1},\ \{H_{1,i}^1 = h_{1,i}^1 \cdot R_{1,i}^1,\ H_{2,i}^1 = h_{2,i}^1 \cdot R_{2,i}^1\}_{i=1,\dots,n}\Big)\ . \tag{14}$$

be the challenge keys given to A alongside the public parameters and the encryption and key generation oracles $\mathcal{O}_{\mathsf{PE.Enc},b}$, $\mathcal{O}_{\mathsf{PE.KeyGen},b}$.

Game 0: this is the ik-cpa experiment with $b = 0$.

Hybrid 0: in this hybrid game the oracle $\mathcal{O}_{\mathsf{PE.Enc},b}$ generates the ciphertexts using $Q^1 = g_q R_0^1$ instead of $Q^0 = g_q R_0^0$. Everything else is the same as in Game 0. This does not impact decryption, as the elements in $\mathbb{G}_r$ are used as masking elements and get canceled out when applying $\hat{e}$. Moreover, the distribution of the ciphertext is exactly the same, as its projection in $\mathbb{G}_r$ sends $C_{1,i}$ to $(R_{1,i}^0)^s (R_0^1)^{\alpha\alpha_i} R_{3,i} = g_r^{a_{1,i}^0 s + a_0^1 \alpha\alpha_i + a_{3,i}}$, where $a_{1,i}^0 = \log_{g_r} R_{1,i}^0$, $a_0^1 = \log_{g_r} R_0^1$ and $a_{3,i} = \log_{g_r} R_{3,i}$. Now, recall that sampling a random element R in $\mathbb{G}_r$ is equivalent to sampling a random exponent $a \in \mathbb{Z}_r$ and setting $R = g_r^a$, where g_r is a generator of $\mathbb{G}_r$. Therefore, the element $g_r^{a_{1,i}^0 s + a_0^b \alpha\alpha_i + a_{3,i}}$ for $b \in \{0, 1\}$ is in fact a random element of $\mathbb{G}_r$, as $a_{3,i} \xleftarrow{\$} \mathbb{Z}_r$, independently of whether the oracle uses R_0^0 or R_0^1 to generate it[13]. An analogous analysis can be done for $C_{2,i}$.
Therefore Hybrid 1 is statistically indistinguishable from Game 0.

Hybrid 1: in this hybrid game the oracle $\mathcal{O}_{\mathsf{PE.Enc},b}$ generates the ciphertexts using $R_{1,i}^1$ instead of $R_{1,i}^0$. Everything else is the same as in Hybrid 0. Analogously to the previous case we can argue that Hybrid 1 is statistically indistinguishable from Hybrid 0.

Hybrid 2: in this hybrid game the oracle $\mathcal{O}_{\mathsf{PE.Enc},b}$ generates the ciphertexts using $R_{1,i}^2$ instead of $R_{2,i}^0$. Everything else is the same as in Hybrid 1. Analogously to the previous case we can argue that Hybrid 2 is statistically indistinguishable from Hybrid 1.

Hybrid 3: in this hybrid game $\mathcal{O}_{\mathsf{PE.Enc},b}$ and $\mathcal{O}_{\mathsf{PE.KeyGen},b}$ are simulated as follows. The encryption oracle $\mathcal{O}_{\mathsf{PE.Enc},b}$ does everything as in Hybrid 2 except for the exponent s. Instead of $s \xleftarrow{\$} \mathbb{Z}_N$, the oracle samples $s_0, s_1 \xleftarrow{\$} \mathbb{Z}_N$ and sets $s \leftarrow s_0 + s_1 \cdot (\gamma_0^{-1} k_0^{-1} \gamma_1 k_1)$, where $k_b := \log_{g_p} h^i$ (the simulator knows already k_b because it generates the public keys itself, so it does not have to compute the logarithm). The key generation oracle $\mathcal{O}_{\mathsf{PE.KeyGen},b}$ does everything as in Hybrid 2 except for the exponent $r_{1,1}$. Instead of $r_{1,1} \xleftarrow{\$} \mathbb{Z}_p$, the oracle samples $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p$ and sets $r_{1,1} \leftarrow r_0 - (k_{1,1}^0)^{-1}(\gamma_0 k_0 - \gamma_1 k_1)$ where $k_{1,1}^0 = \log_{g_p} h_{1,1}^0$. Remark that $r_{1,1}$ is still in $\mathbb{Z}_p$, as all the logarithms are modulo g_p. A standard computation shows that decryption still works. The distributions of s and $r_{1,1}$ are the same in both Hybrid 3 and Hybrid 2, thus the games are statistically indistinguishable.

Hybrid 4: in this game the secret keys are generated using $h^{1-\gamma_1}$, and the ciphertexts output by $\mathcal{O}_{\mathsf{PE.Enc},b}$ are generated using P^1 and $s \xleftarrow{\$} \mathbb{Z}_N$. A standard computation shows that decryption still works. In Theorem C.11 we show that the two games are indistinguishable if the PE is ind-ah secure, thus under Assumption 1 and 2 from [17] (cf. Theorem C.1 in [17]).

Hybrid 5: in this hybrid game $\mathcal{O}_{\mathsf{PE.Enc},b}$ and $\mathcal{O}_{\mathsf{PE.KeyGen},b}$ are simulated as follows. The encryption oracle $\mathcal{O}_{\mathsf{PE.Enc},b}$ does everything as in Hybrid 4 except for the exponent s. Instead of $s \xleftarrow{\$} \mathbb{Z}_N$, the oracle samples $s_0, s_1 \xleftarrow{\$} \mathbb{Z}_N$ and sets $s \leftarrow s_0 + s_1 \cdot ({k_{1,1}^0}^{-1} k_{1,1}^1)$, where $k_{1,1}^b := \log_{g_p} h_{1,1}^b$. The key generation oracle $\mathcal{O}_{\mathsf{PE.KeyGen},b}$ does everything as in Hybrid 2 except for the exponent $r_{1,1}$. Instead of $r_{1,1} \xleftarrow{\$} \mathbb{Z}_p$, the oracle samples $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p$ and sets $r_{1,1} \leftarrow r_0 + r_1(k_{1,1}^0 + k_{1,1}^1)$. A standard computation shows that decryption still works, and that $(h_{1,1}^0)^s = (h_{1,1}^1)^{s_1 + s_0 \cdot ({k_{1,1}^1}^{-1} k_{1,1}^0)}$ and $(h_{1,1}^0)^{r_{1,1}} = (h_{1,1}^1)^{-(r_1 + r_0 {k_{1,1}^1}^{-1})}$. The distributions of s and $r_{1,1}$ are the same in both Hybrid 4 and Hybrid 5, thus the games are statistically indistinguishable.

[13] Another way of seeing this is by setting $a'_{3,i} = a_{3,i} + (a_0^0 - a_{0,1})\alpha\alpha_i$ and observing that $a'_{3,i} \bmod r$ is still a random element in $\mathbb{Z}_r$.

Hybrid 6: in this game both the secret keys and the ciphertexts are generated using $h^1_{1,1}$ and sampling fresh $r_{1,1} \xleftarrow{\$} \mathbb{Z}_p$, $s \xleftarrow{\$} \mathbb{Z}_N$. A standard computation shows that decryption still works. With an analogous proof to the case of Hybrid 3 and 4 one can show that the two games are indistinguishable if the PE is ind-ah secure.

For $i = 2, \ldots, n$

Hybrid 5+i: in each game the secret keys and the ciphertexts are generated using $h^1_{1,i}$. Indistinguishability follows from the proof of indistinguishability of Hybrid 4, 5, 6.

For $i = 1, \ldots, n$**:**

Hybrid (5+n)+i: in each game the secret keys and the ciphertexts are generated using $h^1_{2,i}$. The argument for indistinguishability is the same as before.

Game 1: this is the ik-cpa experiment with $b = 1$. In fact, this is exactly equal to Hybrid $5 + 2n$.

Lemma C.11. *If the PE is ind-ah secure, Hybrid 4 and Hybrid 3 are indistinguishable.*

Proof. First, let us consider a ciphertext in Hybrid 3. By standard algebraic computation we obtain that the components of $PE.c = (C, C_0, \{C_{1,i}, c_{2,i}\}_i)$ are such that

$$\begin{aligned}
C &= mP^{0^s} \underset{k_b := \log_{g_p} h^b,\ b=0,1}{=} m \cdot \hat{e}(g_p, g_p)^{\gamma_0 k_0 \cdot s} \underset{E \leftarrow \hat{e}(g_p, g_p)}{=} m \cdot E^{\gamma_0 k_0 \cdot s} \\
&= m \cdot E^{\gamma_0 k_0 \cdot (s_0 + s_1 \cdot (\gamma_0^{-1} k_0^{-1} \gamma_1 k_1))} \\
&= m \cdot E^{\gamma_0 k_0 \cdot s_0} \cdot E^{\gamma_0 k_0 \cdot s_1 \cdot \gamma_0^{-1} k_0^{-1} \gamma_1 k_1} \\
&= m \cdot P^{0^{s_0}} P^{1^{s_1}} = m \cdot P^{1^{s_1 + s_0 \cdot (\gamma_1^{-1} k_1^{-1} \gamma_0 k_0)}} \\
C_0 &= g_p^s \\
&\text{For all } i = 1, \ldots, n \\
C_{1,i} &= (h^0_{1,i} R^1_{1,i})^s Q^{1^{\alpha \alpha_i}} R_{3,i} \\
C_{2,i} &= (h^0_{2,i} R^1_{2,i})^s Q^{1^{\beta \alpha_i}} R_{4,i} \ .
\end{aligned}$$

The exponents $s = s_0 + s_1 \cdot (\gamma_0^{-1} k_0^{-1} \gamma_1 k_1$ and $s' := s_1 + s_0 \cdot (\gamma_1^{-1} k_1^{-1} \gamma_0 k_0)$ are uniformly distributed over $\mathbb{Z}_N$ as $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_N$. Therefore the distribution of the ciphertext is statistically indistinguishable from a ciphertext $PE.c = (m \cdot P^{1^{u'}},\ g_p^u,\ \{(h^0_{1,i} R^1_{1,i})^u Q^{1^{\alpha \alpha_i}} R_{3,i},\ (h^0_{2,i} R^1_{2,i})^u Q^{1^{\beta \alpha_i}} R_{4,i}\}_i)$ for $u, u' \xleftarrow{\$} \mathbb{Z}_N$.

Analogously, the secret key generated in Hybrid 3 is such that

$$\begin{aligned}
K &= R_5 Q_6 h^{0^{-\gamma_0}} \prod_{i=1}^{n} {h^0_{1,i}}^{-r_{1,i}} {h^0_{2,i}}^{-r_{2,i}} \\
K_{1,i} &= g_p^{r_{1,i}} g_q^{\phi_1 f_i} \\
K_{2,i} &= g_p^{r_{2,i}} g_q^{\phi_2 f_i} \ ,
\end{aligned}$$

where $r_0, r_{1,i}, r_{2,j} \xleftarrow{\$} \mathbb{Z}_p$ for $i \neq 1$ and for all j, and $r_{1,1} \leftarrow r_0 - (k^0_{1,1})^{-1}(\gamma_0 k_0 - \gamma_1 k_1)$. by a standard algebraic computation we get that:

$$\begin{aligned}
{h^0_{1,1}}^{-r_{1,1}} &= {h^0_{1,1}}^{-r_0 + (k^0_{1,1})^{-1}(\gamma_0 k_0 - \gamma_1 k_1)} \\
&= {h^0_{1,1}}^{-r_0} \cdot g_p^{\gamma_0 k_0 - \gamma_1 k_1} \\
&= {h^0_{1,1}}^{-r_0} \cdot h^{0^{\gamma_0}} \cdot h^{1^{-\gamma_1}} \ .
\end{aligned}$$

Therefore

$$K = R_5 Q_6 h^{1^{-\gamma_1}} \cdot {h^0_{1,1}}^{-r_0} {h^0_{2,1}}^{-r_{2,1}} \cdot \prod_{i=2}^{n} {h^0_{1,i}}^{-r_{1,i}} {h^0_{2,i}}^{-r_{2,i}} \ ,$$

and the distributions of the decryption keys in Hybrid 3 and Hybrid 4 are statistically indistinguishable. From the previous observations it follows that distinguishing Hybrid 3 from Hybrid 4 entails distinguishing whether $s' = s \in \mathbb{Z}_N$ given ciphertexts (for convenience we drop the indices $0, 1$ of the elements of $PE.par_0,\ PE.par_1$)

$$PE.c = \left(C = mP^{s'-s}P^s,\ C_0 = g_p^s,\ \left\{C_{1,i} = H_{1,i}^s Q^{\alpha\alpha_i} R_{3,i},\ C_{2,i} = H_{2,i}^s Q^{\beta\alpha_i} R_{4,i}\right\}_{i=1,\ldots,n}\right) . \tag{15}$$

A successful distinguisher could then break the indistinguishability property of the PE (with challenge messages $(m, m \cdot P^t)$ for some random $t \xleftarrow{\$} \mathbb{Z}_N$). □

C.4 Secure Predicate Family Instantiation

The PE scheme from [17] can be used to verify predicates that can be expressed as an inner product of the attribute vector α with a predicate vector f. This in particular means that they can verify that the attribute is a solution of a particular polynomial, thus obtaining conjunctions and disjunctions of inequalities too (cf. Sections 5.3 and 5.4 in [17]). We now show how to instantiate the predicate family in Eq. (10) as an inner product using the deterministic signature in Appendix A.6. Essentially, we check that a signature $(\mathbf{s}, e)$ satisfies the following logical expression:

$$\left(\bigvee_{i\in[B_\chi]} (|e| = i)\right) \wedge \left(\bigvee_{k\in S_j} (\mathbf{a}_k\mathbf{s} + e - b_k)\right) ,$$

where the first disjunction checks that the norm of the error is smaller than B_χ, and S_j is the set of all the senders allowed to communicate with the j-th receiver, $S_j := \{k \in [n_S] \ :\ \mathcal{P}(k, j) = 1\}$. To convert this expression into a polynomial we have to be careful, ad the cardinality of S_j (the set of senders allowed to communicate with receiver j) may vary depending on the j. This would imply that the total degree of the polynomial representing this expression might vary, thus the length of the attribute (that has to contain one value for all the monomials in such polynomial) cannot be fixed. As a workaround we add bogus signature verifications for all the senders not in S_j, by sampling random $\mathbf{a}_k^j,\ b_k^j$. As the predicate is generated by the setup, which is trusted, we can assume that these are generated such that no signature (s_i, e_i) is such that $\mathbf{a}_k^j\mathbf{s}_i + e_i = b_k^j \bmod q$.

Given random $r_i \in \mathbb{Z}_N$ for all $i \in [n_S]$, and $\mathbf{a}_h^j \xleftarrow{\$} \mathbb{Z}_q^n$, $b_h^j \xleftarrow{\$} \mathbb{Z}_q$ for all $h \in [n_S] \setminus S_j$, $j \in [n_R]$, the predicate family is $\mathcal{F} := \{f_i^S\}_{i\in[n]} \cup \{f_i^R\}_{i\in[n]}$ where

$$f_j^S(\mathbf{s}, e) = (\mathbf{a}_j\mathbf{s} + e - b_j) + r_j \sum_{i=1}^{B_\chi} (e - i) \bmod N$$

$$f_j^R(\mathbf{s}, e) = \prod_{k\in S_j} (\mathbf{a}_k\mathbf{s} + e - b_k) \prod_{h\in[n_S]\setminus S_j} (\mathbf{a}_h^j\mathbf{s} + e - b_h^j) + r_j \sum_{i=1}^{B_\chi} (e - i) \bmod N .$$

The total degree of this polynomial is $\max\{B_\chi, n_S\}$, thus the length of the vector representation of an attribute is

$$\max\left\{\binom{n+2}{n_S} + (B_\chi - n_S), \binom{n+2}{n_S}\right\} .$$