Practical Continuously Non-Malleable Randomness Encoders in the Random Oracle Model

Antonio Faonio

EURECOM, France,
antonio.faonio@eurecom.fr

Abstract. A randomness encoder is a generalization of encoding schemes with an efficient procedure for encoding *uniformly random strings*. In this paper we continue the study of randomness encoders that additionally have the property of being continuous non-malleable. The beautiful notion of non-malleability for encoding schemes, introduced by Dziembowski, Pietrzak and Wichs (ICS'10), states that tampering with the codeword can either keep the encoded message identical or produce an uncorrelated message. Continuous non-malleability extends the security notion to a setting where the adversary can tamper the codeword polynomially many times and where we assume a self-destruction mechanism in place in case of decoding errors. Our contributions are: (1) two practical constructions of continuous non-malleable randomness encoders in the random oracle model, and (2) a new compiler from continuous non-malleable randomness encoders to continuous non-malleable codes, and (3) a study of lower bounds for continuous non-malleability in the random oracle model.

1 Introduction

The notion of non-malleable codes, Dziembowski, Pietrzak and Wichs [22], (NMC) has emerged at the intersection between cryptography and information theory. Non-malleable codes allow one to encode messages in such a way that, after malicious tampering, the modified codeword decodes to either the original message, or an unrelated one. Non-malleable codes find applications to cryptography, for example, for protecting arbitrary cryptographic primitives against related-key attacks [22] and commitments (Agrawal *et al.* [5]). Limitations on the nature of the tampering functions must be imposed, as otherwise NMCs are impossible to achieve [22]. One of the most studied settings for which NMCs are achievable is the split-state model [3,10,20], see also [2,4,15,31,33,34]. In this model we assume that the codeword is divided into two pieces, and that the tampering functions can alter the two pieces independently.

Continuous non-malleability. In the definition of non-malleable codes, the property is guaranteed as long as a *single* tampering function is applied to a target codeword. In particular, no security is guaranteed if an adversary can tamper multiple times with the target codeword. While "one-time" non-malleability is already sufficient in some cases, it comes with some shortcomings, among which, for instance, the fact that in applications, after a decoding takes place, we always need to re-encode the message using fresh randomness; the latter might be problematic, as such a re-encoding procedure needs to take place in a tamper-proof environment. Motivated by these limitations,Faust *et al.* [26] introduced a natural extension of non-malleable codes where the adversary is allowed to tamper a target codeword by specifying polynomially-many tampering functions; As argued in [26], such *continuously* non-malleable codes allow to overcome several limitations of one-time non-malleable codes, and further led to new applications where continuous non-malleability is essential [11,13]. Continuous non-malleability requires a special "self-destruct" capability that instructs the decoding algorithm to always output the symbol $\perp$ (meaning "decoding error") after the first invalid codeword is decoded, otherwise generic attacks are possible [26,28]. Faust *et al.*[26] showed that CNMCs are impossible in the information-theoretic setting, while Ostrovsky *et al.* [34] showed that CNMCs can be constructed assuming that one-to-one one-way functions exist.

Randomness Encoders. A randomness encoder consists of an encoding procedure which can produce a codeword for a random message and the relative decoding algorithm. Kanukurthi, Obbattu and Sekar [29] introduced the concept of non-malleable randomness encoders (NMREs) as a relaxation of NMCs. As shown by [29], NMREs are already sufficient for many of the applications of NMCs. For example, in the typical application of NMCs to tamper-resilient cryptography, the encoded messages are randomly generated secret keys. Moreover, they gave a construction of a NMC from a NMRE and (one-time) authenticated secret key encryption.

Continuous Non-Malleable Randomness Encoder. Dachman-Soled and Kulkarni [14] considered the natural notion of continuous non-malleable randomness encoders (CNMREs). A CNMREs is a relaxation of the concept of continuous non-malleable codes to the realm of randomness encoders. In particular, they showed a construction of CNMRE in the common-reference string (CRS) model assuming only the existence of injective one-way functions and showed a compiler from CNMRE to CNMC for 1-bit messages.

1.1 Our Contributions

As our main contribution, we present two practical CNMREs in the random oracle model. Our randomness encoders can encode random messages of size λ bits, where λ is the security parameter. The size of the codewords in the first randomness encoder is approximately 12λ bits, and the decoding function computes two cryptographic hash evaluations and an inner product between two vectors in $\mathbb{Z}_p^4$ for a prime $p \geq 2^\lambda$. The second randomness encoder has shorter codewords (the size of the codewords is approximately 8λ bits), but it has a more expensive decoding function which consists of four cryptographic hash evaluations and an inner product between two vectors in $\mathbb{Z}_p^4$. Compared to the state-of-art for CNMC (the construction of Ostrovsky *et al.* [34]) both our randomness encoders are thousands of times more efficient. Comparing with state-of-art for practical NMC (the construction of Fehr,Karpman and Mennick [27]), our randomness encoders are comparably similarly efficient both in terms of sizes of the codewords and in terms of the computational complexities of the algorithms. (We give more details in the next section.)

As second contribution, we show how to construct CNMCs from CNMREs, thus extending the result of [29] to the continuous setting. We consider the compiler of Coretti, Faonio and Venturi [12], which compiles a CNMC whose ratio between the messages and the codewords is small (asymptotically zero) to a CNMC where the ratio is $\frac{1}{2}$. Although our compiler and their compiler are similar, their analysis does not apply directly to our setting (we elaborate further in the next section). It is well known that continuous non-malleability is impossible in the split-state model with information-theoretic security [26]. As third contribution, we extend the lower bounds of [26] to the case of continuous non-malleability in the random oracle model. We show that we can have information-theoretic security, as long as the number of random oracle queries made by the adversary is bounded.

1.2 Technical Overview

In the continuous non-malleability experiment, the adversary receives two messages μ_0, μ_1 and gets oracle access to a target codeword (c_0, c_1) for the message μ_b with the goal of guessing the bit b. The adversary can submit tampering functions (f_0, f_1) receiving back the value $\mathsf{Dec}(f_0(c_0), f_1(c_1))$. If the output of the decoding algorithm is $\perp$ then the adversary loses access to the tampering oracle. In the very same vein, in the continuous non-malleability experiment for randomness encoders, the adversary gets input two uniformly random keys κ_0 and κ_1 (one of which is sampled using the randomness encoder) and gets oracle access to a target codeword (c_0, c_1). We proceed in two steps to construct our CNMREs. In the first step we reduce continuous non-malleability to leakage resilience. In particular for this step, we first define the notion of noisy leakage-resilient randomness encoders (LRREs, for short), then we show an efficient compiler from LRREs to CNMREs. In the second step, we give constructions of leakage-resilient randomness encoders. In the security game for the notion of LRREs the adversary has access to a leakage oracle to the codeword. The adversary can submit queries of the form (g_0, g_1) receiving back the values $g_0(c_0), g_1(c_1)$. In our definition we consider the so-called noisy-leakage model [6] where the leakage is measured as the drop of min-entropy of the codeword.

The compiler from LRREs to CNMREs is very similar to the original construction of CNMC of [26], and its proof of security follows a proof technique similar to [12,24,25]. Our LRREs are inspired by the leakage-resilient storage of Davi, Dziembowski and Venturi [16] based on the inner-product extractor (see also Dziembowski and Faust [19]). In more detail, let $\Pi' = (\mathsf{REncode}', \mathsf{Dec}')$ be a LRRE and let RO be a random oracle, we construct a CNMRE $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ where the encoding function samples a codeword c'_0, c'_1 from $\mathsf{REncode}'$ and then outputs $(c'_0, h_1), (c'_1, h_0)$ where $h_\beta = \mathsf{RO}(\beta \| c'_\beta)$. The decoding function on input a codeword $(c'_0, h_1), (c'_1, h_0)$ first checks that the hash values match, namely that $h_\beta = \mathsf{RO}(\beta \| c'_\beta)$ for $\beta \in \{0, 1\}$, and if so it decodes the codeword using Dec'. The main idea behind the security of the scheme is that if an adversary can tamper, let say c'_0, in a non-trivial way obtaining a value $\tilde{c}'_0$ then it must already know the tampered value $\tilde{c}'_0$, as otherwise the adversary would not be able to compute correctly $\mathsf{RO}(0 \| \tilde{c}'_0)$. (Recall that the adversary can only tamper c'_0 and h_0 independently, as they are in two different shares.) The latter implies that the output of the tampering oracle is predictable, and therefore we can simulate it using a leakage function that does not decrease the (average conditional) min-entropy of the target codeword.

The second step is to construct practical LRREs. Our first leakage-resilient randomness encoder encodes a random message by sampling two random vectors from a field with large enough cardinality, the decoding function outputs the inner product between the two vectors. Previous works considered the encoding scheme where, on input a message μ, the two vectors were sampled conditioned on their inner product being equal to μ. The proofs of security in the previous works relied on (1) the fact that the inner product is a two-source extractor and then (2) a complexity leveraging argument to break the dependence between the two vectors. By downgrading to randomness encoders, our proof of security does not need the complexity leveraging argument. This simple observation allows a significant gain in the concrete parameters of the scheme. The second scheme exploits the power of the random oracle. In fact, instead of sampling two vectors, we could sample two seeds which fed to the random oracle would produce the required vectors. By setting the parameters properly, the second randomness encoder has more compact codewords then our first one.

Compiler from randomness-encoders to codes. Similar to [1,12,29], the idea for our compiler is to encode a random key for an authenticated secret key encryption scheme using a CNMRE and then to encrypt the message we want to encode, thus obtaining a ciphertext γ. As proposed by [12], we store the resulting ciphertext in both sides of the codeword and check for equality of the two copies of the chipertext when decoding. The proof of security of [12] relies on the leakage resilience of the inner CNMC. We show that leakage resilience is not necessary. In fact, any adversarially generated codeword $(\tilde{c}_0\|\tilde{\gamma}_0,\ \tilde{c}_1\|\tilde{\gamma}_1)$ (for the compiled code) that successfully decodes must have $\tilde{\gamma}_0 = \tilde{\gamma}_1$. Our novel idea is to use this correlated information to synchronize the tampering functions performed by the reduction and to extract the adversarially generated ciphertext. In more detail, when reducing to the continuous non-malleability of the CNMRE, we additionally sample two valid codewords for two distinct keys κ_0, κ_1. Upon a tampering query for the compiled code, suppose that the tampered codeword is $(\tilde{c}_0\|\tilde{\gamma},\ \tilde{c}_1\|\tilde{\gamma})$, we first extract, bit-by-bit, the ciphertext $\tilde{\gamma}$ by sending tampering queries that output either κ_0 or κ_1 according to the bits of $\tilde{\gamma}$, and then we send an extra query that allows to decode $\tilde{c}_0, \tilde{c}_1$, thus obtaining the secret key for the chipertext $\tilde{\gamma}$.

Lower bounds on continuous non-malleability in the ROM. Very roughly speaking, the proof of the impossibility result of [26] shows that any CNMC musts have (at least) two special codewords. The strategy is to hardwire such codewords in their adversary and to use them to extract, bit-by-bit, all the information about the target codeword. However, in the random oracle model, the codeword space is a random variable that depends on the random oracle. Thus an adversary cannot simply have hardwired these two specials codewords, but it needs first to compute them. In other words, the complexity of the generic attack of [26], in our framework, depends on the random-oracle-query complexity of finding such two special codewords. Additionally, we show that, even if these two special codewords do not exist, we can still break continuous non-malleability when the adversary can query the random oracle enough time and de-randomize the full codeword space.

Lastly, we give a lower bound specific to our CNMRE construction. The number of random oracle queries that an adversary needs in order to break the security of our construction depends both on the random-oracle query complexity of the inner leakage-resilient randomness encoder and on the classical birthday-paradox lower bound.

1.3 Related Work

In the Table 1 we compare our results with the most relevant related works. We compare with the work of Kiayias, Liu and Tselekounis [30] (resp. the work of Fehr, Karpman and Mennink [27]) which showed a *practical* construction of NMC in the CRS model (resp. plain model), the work of Dachman-Soled and Kulkarni [14] which showed a construction of CNMRE in the CRS model and a general compiler from CNMREs to 1-bit messages CNMCs, and the work of Ostrovsky, Persiano, Visconti and Venturi [34] which showed a construction of CNMC in the standard model. The result of [34] makes use of a statistically binding commitment scheme and of the leakage-resilient (one-time) non-malleable code of Aggarwal *et al.* [4]. While, we could implement very efficiently the former ingredient in the random oracle model (by hashing the message together with some randomness), the latter ingredient is the bottleneck of their construction. In fact, the codeword size needs to be at least $O(\lambda^7)$ to encode a message of size λ. Without diving into the details, if we don't consider the cryptographic hash computations, for both our and their scheme the computational complexity of the decoding function is at least super-linear in the size of the codeword, which implies that our schemes are asymptotically faster than [34] of at least 7 orders of magnitude. We could obtain such a speed up because our schemes rely on the random-oracle methodology, on the other hand, the scheme of Ostrovsky *et al.* is in the standard model. We stress that the goal of this paper is, indeed, to construct very efficient schemes which could be already used in practice. Dachman-Soled and Kulkarni [14] give a compiler from CNMRE to CNMC. The idea of their compiler is to sample from the encoding procedure of the CNMRE until we obtain a valid codeword of the message to be encoded. The scheme $\mathsf{DK19}_2$ in Table 1 is the result of applying their compiler to their scheme $\mathsf{DK19}_1$. The codeword size is

Scheme	Non-Malleability	Codeword Size	Type	Model	Assumption
[14] DK19$_1$	continuous	$\approx 14\lambda^3$	R	CRS	inj. OWF
Π_1^*	continuous	6λ	R	ROM	-
Π_2^*	continuous	4λ	R	ROM	-
[27] FKM18	one-time	$2\lambda + k$	E	-	LR-PRP, RK-PRP
[30] KLT16	one-time	$9\lambda + 2\log^2 \lambda + k$	E	CRS	Ext-HF
[34] OPVV18	continuous	$\Omega(\lambda^6 k)$	E	-	inj. OWF
[14] DK19$_2$	continuous	$\approx 14\lambda^2$	E	CRS	inj. OWF
Π_3^*	continuous	$8\lambda + k$	E	ROM	OWF

Table 1. Comparison with related work. In the table λ is the security parameter and k is the length of the message; R stands for randomness encoders and E for encoding schemes; inj. OWF stands for injective one-way functions, Ext-HF stands for extractable hash functions, LR-PRP (resp. RK-PRP) stands for leakage-resilient (resp. related-key secure) pseudorandom permutation. OWF stands for one-way functions.

approx. $14\lambda^2$ while the codeword size of DK19$_1$ is approx. $14\lambda^3$. The reason is that the compiler works only for 1-bit messages. This limitation is due to the complexity-leveraging argument needed to prove the security of their compiler and the computational security of their scheme DK19$_1$. We notice that a natural extension to multi-bit messages of their compiler would work when applied to our scheme Π_1^* because our scheme is secure against unbounded adversaries in the random oracle model. However, the size of the codeword would have a multiplicative blow up in the security parameter[1]. On the other hand, our compiler has only an additive security loss, thus the resulting scheme is more efficient. Additionally, in terms of assumptions, Π_3^* is computationally secure when k, the length of the message, is such that $k > \lambda$, however, it can be information-theoretically secure when $k \leq \lambda/2$.

Non-malleability *in the multi-tampering* [9,32] model is related to the notion of continuous non-malleability. In the former notion, the number of tampering queries is a priori bounded, however there is no need for self-destruct mechanisms.

2 Preliminaries

We denote with $\lambda \in \mathbb{N}$ the security parameter. A function $\epsilon : \mathbb{N} \to [0, 1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ.

[1] The proof of security would work through a complexity-leveraging argument.

We recall a standard Lemma from Dodis, Reyzin and Smith [18]:

Lemma 1. *Let A, B, C be random variables, then*

1. *For any $\delta > 0$, we have $\mathbb{H}_\infty(A|B = b)$ is at least $\widetilde{\mathbb{H}}_\infty(A|B) - \log(1/\delta)$ with probability at least $1 - \delta$ over the choice of b.*
2. *If B has at most 2^λ values then $\widetilde{\mathbb{H}}_\infty(A|B, C) \geq \widetilde{\mathbb{H}}_\infty(A|C) - \lambda$.*

2.1 Split-State Codes and Randomness-Encoders in the ROM

Definition 1 (Split-State Encoding Scheme in the ROM, [12]). *Let $k(\lambda) = k \in \mathbb{N}$ and $n(\lambda) = n \in \mathbb{N}$ be functions of the security parameter $\lambda \in \mathbb{N}$. A (k, n)-split-state-code is a tuple of algorithms $\Sigma = (\mathsf{Enc}^{\mathsf{RO}}, \mathsf{Dec}^{\mathsf{RO}})$ specified as follows: (1) The randomized algorithm $\mathsf{Enc}^{\mathsf{RO}}$ takes as input a value $s \in \{0,1\}^k$, and outputs a codeword $(c_0, c_1) \in \{0,1\}^{2n}$; (2) The deterministic decoding algorithm $\mathsf{Dec}^{\mathsf{RO}}$ takes as input a codeword $(c_0, c_1) \in \{0,1\}^{2n}$, and outputs a value $s \in \{0,1\}^k \cup \{\bot\}$ (where $\bot$ denotes an invalid codeword).*

We say that Σ satisfies correctness if for all values $s \in \{0,1\}^k$, $\mathbb{P}\left[\mathsf{Dec}^{\mathsf{RO}}(\mathsf{Enc}^{\mathsf{RO}}(s)) = s\right] = 1$.

We introduce the notion of split-state randomness-encoders in the ROM.

Definition 2 (Split-State Randomness Encoders in the ROM). *Let $n(\lambda) = n \in \mathbb{N}$ be functions of the security parameter $\lambda \in \mathbb{N}$. A n-split-state-randomness-encoder is a tuple of algorithms $\Pi = (\mathsf{REncode}^{\mathsf{RO}}, \mathsf{Dec}^{\mathsf{RO}})$ specified as follows: (1) The randomized algorithm $\mathsf{REncode}^{\mathsf{RO}}$ (with the only input the security parameter) outputs a value $\kappa \in \{0,1\}^\lambda$ and a codeword $(c_0, c_1) \in \{0,1\}^{2n}$; (2) The deterministic decoding algorithm $\mathsf{Dec}^{\mathsf{RO}}$ takes as input a codeword $(c_0, c_1) \in \{0,1\}^{2n}$, and outputs a value $\kappa \in \{0,1\}^\lambda \cup \{\bot\}$ (where $\bot$ denotes an invalid codeword). We say that Σ satisfies correctness if for all λ the following holds:*
$\mathbb{P}\left[\kappa = \kappa' \wedge (\kappa, c) \leftarrow\$ \mathsf{REncode}^{\mathsf{RO}}(1^\lambda) \wedge \kappa' \leftarrow \mathsf{Dec}^{\mathsf{RO}}(c)\right] = 1.$

The contributions of this paper focus on split-state encoding schemes and split-state randomness encoders. To avoid redundancy, we therefore omit the adjective "split-state" whenever it is clear from the context. Many of the algorithms described in this paper make use of a random oracle, we avoid to upper script them with the oracle RO whenever it is clear from the context.

2.2 Continuous non-malleability in the ROM

An encoding scheme is non-malleable [22,33] if no adversary tampering *independently* with the two parts of the target encoding (c_0, c_1) can generate a modified codeword that decodes to a related value. Continuous non-malleability [26] strengthens this guarantee by allowing to tamper continuously and adaptively with (c_0, c_1), until a decoding error occurs, after which the system "self-destructs" and stops answering tampering queries. The self-destruct, that in practice could be implemented via a write-once flag, is strictly necessary (see [28]) for achieving continuous non-malleability whenever no other refresh mechanisms are in place (see [23,24]). Because we consider schemes in the random oracle model, we enlarge the class of possible tampering functions considering functions that additionally can query the random oracle RO. Consider the following class of tampering functions parameterized by two values $n(\lambda), q(\lambda)$:

$$\mathcal{F}_{n,q} = \left\{ (f_0, f_1) | \forall b : \begin{array}{c} f_b : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)} \\ f_b \text{ makes at most } q(\lambda) \text{ RO queries} \end{array}, \lambda \in \mathbb{N} \right\}$$

Definition 3 (Continuously non-malleable codes and randomness encoders in the ROM). *Let* $k(\lambda), n(\lambda), q(\lambda), q_T(\lambda), q_{\mathsf{RO}}(\lambda) \in \mathbb{N}$ *and let* $\epsilon(\lambda) \in \mathbb{R}$.

Let $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ *be a* (k, n)*-encoding scheme. We say that* Σ *is* $(\epsilon, q, q_{\mathsf{RO}})$-continuously non-malleable *code,* $(\epsilon, q, q_{\mathsf{RO}})$*-CNMC for short, if for all messages* μ_0, μ_1*, for all* $q_T = \mathtt{poly}(\lambda)$*, and for all unbounded adversaries* A *making up to* q_T *tampering oracle queries from the class of tampering functions* $\mathcal{F}_{n,q}$ *and up to* q_{RO} *random oracle queries, we have:*

$$\mathbf{Adv}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda) := \left|\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda, \mu_0, \mu_1) = 1\right] - 1/2\right| \leq \epsilon(\lambda). \quad (1)$$

Let $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ *be a* n*-randomness encoder. We say that* Π *is* $(\epsilon, q, q_{\mathsf{RO}})$-continuously non-malleable *randomness encoder,* $(\epsilon, q, q_{\mathsf{RO}})$*-CNMRE for short, if for all* $q_T = \mathtt{poly}(\lambda)$*, and for all (possibly unbounded) adversaries* A *making up to* q_T *tampering oracle queries from the class of tampering functions* $\mathcal{F}_{n,q}$ *and up to* q_{RO} *random oracle queries, we have:*

$$\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{A}}(\lambda) := \left|\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmre}}_{\Sigma,\mathsf{A}}(\lambda) = 1\right] - 1/2\right| \leq \epsilon(\lambda). \quad (2)$$

The experiments $\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda)$ *and* $\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda, \mu_0, \mu_1)$ *are described in Fig. 1.*

Remark 1 (On the choice of q_T*).* We could prove security of our constructions even when $q_T = \Omega(2^\lambda)$. However, in the definitions above we

$\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda, \mu_0, \mu_1)$:

$b \leftarrow\!\!\$ \{0,1\}$; $(c_0, c_1) \leftarrow\!\!\$ \mathsf{Enc}(\mu_b)$
$\mathcal{M} := \{\mu_0, \mu_1\}$
$\mathtt{stop} \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\cdot,\cdot)}(1^\lambda, \mu_0, \mu_1)$
Return $b \stackrel{?}{=} b'$

$\mathbf{G}^{\mathsf{cnmre}}_{\Pi,\mathsf{A}}(\lambda)$:

$(\kappa, (c_0, c_1)) \leftarrow\!\!\$ \mathsf{REncode}(1^\lambda)$
$b \leftarrow\!\!\$ \{0,1\}$; $\kappa_b \leftarrow \kappa$, $\kappa_{1-b} \leftarrow\!\!\$ \{0,1\}^\lambda$
$\mathcal{M} := \{\kappa_0, \kappa_1\}$
$\mathtt{stop} \leftarrow 0$
$b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\cdot,\cdot)}(\kappa_0, \kappa_1)$
Return $b \stackrel{?}{=} b'$

$\mathbf{G}^{\mathsf{lrre}}_{\Pi,\mathsf{A}}(\lambda)$:

$(c_0, c_1) \leftarrow\!\!\$ \mathsf{REncode}(1^\lambda)$
$\kappa_b \leftarrow \mathsf{Dec}(c_0, c_1)$, $\kappa_{1-b} \leftarrow\!\!\$ \{0,1\}^\lambda$
$b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{leak}}(\cdot,\cdot)}(\kappa_0, \kappa_1)$
Return $b \stackrel{?}{=} b'$

Oracle $\mathcal{O}_{\mathsf{tamp}}(f_0, f_1)$:

If $\mathtt{stop} = 1$
  Return $\perp$
Else
  $(\tilde{c_0}, \tilde{c_1}) = (f_0(c_0), f_1(c_1))$
  $\tilde{\mu} = \mathsf{Dec}(\tilde{c_0}, \tilde{c_1})$
  If $\tilde{\mu} \in \mathcal{M}$ Return $\diamond$
  If $\tilde{\mu} = \perp$ Return $\perp$ and $\mathtt{stop} \leftarrow 1$
  Else return $\tilde{\mu}$

Oracle $\mathcal{O}_{\mathsf{leak}}(g_0, g_1)$:

  Return $(g_0(c_0), g_1(c_1))$

Fig. 1: Experiment defining continuously non-malleable codes and randomness-encoders in the split-state model, and leakage-resilient randomness encoders. The tampering oracle $\mathcal{O}_{\mathsf{tamp}}$ is implicitly parameterized by the flag $\mathtt{stop}$, the codeword c_0, c_1 and the set $\mathcal{M}$. Similarly, the leakage oracle is implicitly parameterized by the codeword c_0, c_1. If Π (resp. Σ) is in the random oracle model, then all the procedures and functions (including the adversary A, the leakage functions g_0, g_1 and the tampering functions f_0, f_1) implicitly have oracle access to RO.

limit the number of tampering queries to be a polynomial in the security parameter. The reason is that for each tampering query there is an associated call to the decoding algorithm of the attacked device. We can assume that the attacked device runs in polynomial time.

2.3 Noisy-leakage resilient randomness encoders.

As in previous works [7,12,24,25], we use the notion of *admissibility* to define noisy-leakage resilience. We extend this notion to the ROM.

Definition 4 (Admissible adversaries for randomness encoders). *Let* $n(\lambda), \ell(\lambda), q_{\mathsf{RO}}(\lambda) \in \mathbb{N}$ *such that* $\ell(\lambda) \leq n(\lambda)$*, let* $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ *be a* n*-randomness-encoder. An adversary* A *is* (ℓ, q_{RO})*-admissible if:*

1. *it outputs a sequences of leakage queries (chosen adaptively)* $(g_0^{(i)}, g_1^{(i)})_{i\in[q]}$ *for* $q \in \mathbb{N}$*, such that for any* i *the functions* $(g_0^{(i)}, g_1^{(i)})$ *can make queries to the random oracle,*

2. *it outputs a sequences of random oracle queries (adaptively)* $(x_i)_{i\in[q_{\mathsf{RO}}]}$, *such that:*

$$\widetilde{\mathbb{H}}_\infty\left(c_\beta|c_{1-\beta}, g_\beta^{(1)}(c_\beta), \cdots, g_\beta^{(p)}(c_\beta), (x_i, \mathsf{RO}(x_i))_{i\in[q_{\mathsf{RO}}]}\right) \geq \widetilde{\mathbb{H}}_\infty(c_\beta|c_{1-\beta}, \mathsf{RO}) - \ell \tag{3}$$

where (c_0, c_1) *is the joint random variable corresponding to* $\mathsf{REncode}(1^\lambda)$ *and* RO *is a shortcut for* $(\mathsf{RO}(x))_{x\in\{0,1\}^{\mathtt{poly}(\lambda)}}$.

We define the notion of noisy-leakage resilient randomness encoders (LR-REs).

Definition 5 (noisy-leakage resilient randomness encoders). *Let* $n(\lambda), \ell(\lambda), q_{\mathsf{RO}}(\lambda) \in \mathbb{N}$ *and* $\epsilon(\lambda) \in \mathbb{R}$, *and let* $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ *be a* n*-randomness encoder. We say that* Π *is* $(\epsilon, \ell, q_{\mathsf{RO}})$-noisy-leakage resilient *randomness encoder in the ROM,* $(\epsilon, \ell, q_{\mathsf{RO}})$*-LRRE for short, if for all* (ℓ, q_{RO})*-admissible adversaries* A*, we have that:*

$$\mathbf{Adv}^{\mathsf{lrs}}_{\Pi,\mathsf{A}}(\lambda) := \left|\mathbb{P}\left[\mathbf{G}^{\mathsf{lrre}}_{\Sigma,\mathsf{A}}(\lambda) = 1\right] - 1/2\right| \leq \epsilon(\lambda), \tag{4}$$

where experiment $\mathbf{G}^{\mathsf{lrre}}_{\Sigma,\mathsf{A}}(\lambda)$ *is depicted in Fig. 1.*

3 Our Continuous Non-Malleable Randomness Encoder

Let $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ be a n'-randomness-encoder, and let RO be a random oracle. Consider the following construction of a n-randomness-encoder $\Pi^* = (\mathsf{REncode}^*, \mathsf{Dec}^*)$ where $n := n' + 2\lambda$:

$\mathsf{REncode}^*(1^\lambda)$**:** Sample $\kappa, (c_0, c_1) \leftarrow\!\!\$\ \mathsf{REncode}(1^\lambda)$, compute $h_\beta \leftarrow \mathsf{RO}(\beta\|c_\beta)$ and set the codeword $c^*_\beta := (c_\beta, h_{1-\beta})$ for $i \in \{0,1\}$. Return $\kappa, (c^*_0, c^*_1)$.

$\mathsf{Dec}^*(c^*_0, c^*_1)$**:** Execute the following steps:

1. For $\beta \in \{0,1\}$, parse c^*_β as $(c_\beta, h_{1-\beta})$;
2. **(Hash Values Check.)** If $h_0 \neq \mathsf{RO}(0\|c_0)$ or $h_1 \neq \mathsf{RO}(1\|c_1)$ output $\perp$;
3. Else output $\mathsf{Dec}(c_0, c_1)$.

We give the following definition to simplify the notation in the statement of the theorem.

Definition 6. *Let* Π *be a* n*-randomness encoder, we define with* $\alpha_\Pi(\lambda) := \min_\beta\{\mathbb{H}_\infty(c_\beta) - \widetilde{\mathbb{H}}_\infty(c_\beta|c_{1-\beta}, \mathsf{RO})\}$ *where* $c_0, c_1 \leftarrow\!\!\$\ \mathsf{REncode}(1^\lambda)$.

Theorem 1 (LRREs ⇒ CNMREs in ROM). *For any $q_T := q_T(\lambda), q := q(\lambda)$, and for any adversary A that does up to q_T tampering oracle queries from the class of tampering functions $\mathcal{F}_{n,q}$ and up to $q_{\mathsf{RO}} := q_{\mathsf{RO}}(\lambda)$ random oracle queries there exists a (ℓ, q_{RO})-admissible adversary B where $\ell = 2\log q_{\mathsf{RO}} + \log q_T$ such that:*

$$\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi^*,\mathsf{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{lrs}}_{\Pi,\mathsf{B}}(\lambda) + \frac{q_T}{2^{2\lambda}} + \frac{(q_{\mathsf{RO}}+q\cdot q_T)^2}{2^{2\lambda}} + \frac{(q_{\mathsf{RO}}+q\cdot q_T)\lambda q_T}{2^{\alpha_\Pi(\lambda)-1}}.$$

If Π is $(\mathtt{negl}(\lambda), O(\lambda), \mathtt{poly}(\lambda))$-LRRE then Π^* is $(\mathtt{negl}(\lambda), \mathtt{poly}(\lambda), \mathtt{poly}(\lambda))$-CNMRE.

Proof. We give a reduction to the noisy-leakage resilience of Π. Before describing the reduction we introduce a sub-routine.

Procedure $\textsc{Leak}(g_0, g_1)$

- Let $g_{\beta,i}$ be the restriction of the function g_β to the i-th bit.
- For $i \in [\lambda]$ send the leakage oracle query $(g_{0,i}, g_{1,i})$:
  - let $z_{0,i}, z_{1,i}$ be the output of the oracle,
  - if $z_{0,i} \neq z_{1,i}$ output $\perp$,
  - if $z_{0,i} = z_{1,i} = \diamond$ output $\diamond$.
- Output $z = z_{0,0}, \dots, z_{0,l}$.

We are now ready to describe an adversary for Π [2]

We will keep track of the random oracle queries made by the adversary and by the tampering functions. We denote with $\mathcal{Q}_{\mathsf{A}}, \mathcal{Q}_0, \mathcal{Q}_1$ the lexicographically ordered set of tuple $x, \mathsf{RO}(x)$, and with $\bar{\mathcal{Q}}_{\mathsf{A}}, \bar{\mathcal{Q}}_0, \bar{\mathcal{Q}}_1$ the lexicographically oredered set of oracle queries (i.e., the inputs to the RO without the outputs).

Adversary $\mathsf{B}(\kappa_0, \kappa_1)$

1. **Hash values** h_0, h_1**.** Sample $h_\beta \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ for $\beta \in \{0,1\}$.
2. Run the adversary A with input (κ_0, κ_1).
3. **Random oracle queries.** Whenever A sends a query x to the random oracle, forward the query to random oracle RO. Add the query $(x, \mathsf{RO}(x))$ in the set $\mathcal{Q}_{\mathsf{A}}$.
4. **Tampering oracle queries.** When the adversary A sends its j-th tampering query $(f_0^{(j)}, f_1^{(j)})$, if the flag $\mathtt{stop} = 1$ return $\perp$, else run the sub-ruotine $\textsc{Leak}(g_0^{(j)}, g_1^{(j)})$, where the leakage

[2] Notice that Π might be a randomness encoder in the standard model (i.e. no random oracle), whilst our reduction makes random oracle queries. In this case we could assume that RO is a lazy-sampled, locally-stored random function, therefore B would be a standard-model adversary for Π.

functions $g_\beta^{(j)}$ is described below:

Leakage function $g_\beta^{(j)}(c_\beta)$:

(a) Compile the set $\mathcal{Q}_\beta$ of random oracle query made by the previous tampering functions by running $f_\beta^{(j')}(c_\beta, h_{1-\beta})$ for any $j' < j$ and collecting the queries. Whenever one of the tampering functions calls RO on $(\beta \| c_\beta)$ answer with h_β.
(b) Compute $(\tilde{c}_\beta, \tilde{h}_{1-\beta}) \leftarrow f_\beta^{(j)}(c_\beta, h_{1-\beta})$ and forward all the RO queries made by $f_\beta^{(j)}$ to the RO (but whenever the tampering function calls RO on $\beta \| c_\beta$ answer with h_β, instead of querying the RO).
(c) If $(\tilde{c}_\beta, \tilde{h}_{1-\beta}) = (c_\beta, h_{1-\beta})$ then output $\diamond$.
(d) If there is a tuple $(1-\beta \| c^\star_{1-\beta}, \tilde{h}_{1-\beta}) \in \mathcal{Q}_\mathsf{A} \cup \mathcal{Q}_\beta$,
– if $\beta = 0$ then output $\mathsf{Dec}(\tilde{c}_0, c^\star_1)$,
– if $\beta = 1$ then output $\mathsf{Dec}(c^\star_0, \tilde{c}_1)$,
else output $\perp$.

Let $\tilde{\mu}$ be the output of the Leak procedure, if $\tilde{\mu} = \perp$ then set the flag `stop` $\leftarrow 1$. Return $\tilde{\mu}$.

5. Eventually the adversary returns a bit b'. Output b'.

Claim. The adversary B is $(\log \lambda + \log q_T + 1,\ q_{\mathsf{RO}})$-admissible.

Proof. The number of random oracle queries made by B is equal to the number of random oracle queries made by A. We can assume w.l.g. that the last tampering query of A results to self-destruct. Let j^* be the index of the tampering query where the procedure $\textsc{Leak}(g_\beta^{(j^*)}, g_\beta^{(j^*)})$ outputs $\perp$ for the first time. Let i^* be the index of the iteration where $\textsc{Leak}(g_\beta^{(j^*)}, g_\beta^{(j^*)})$ stops, we have $i^* < \lambda$ and let $\bar{\kappa}_{0,0}, \bar{\kappa}_{1,0}, \ldots, \bar{\kappa}_{0,i^*}, \bar{\kappa}_{1,i^*}$ be the i^* bits leaked by the sub-routine. The adversary B leaks the values $\kappa_\beta^{(1)} := g_\beta^{(1)}(c_\beta), \ldots, \kappa_\beta^{(j^*-1)} := g_\beta^{(j^*-1)}(c_\beta)$ and the values $\bar{\kappa}_{\beta,0}, \ldots, \bar{\kappa}_{\beta,i^*}$ to answer the tampering queries made by A. Let $(x_1, \ldots, x_{q_{\mathsf{RO}}})$ be the random oracle queries made by B. For $\beta \in \{0,1\}$:

$$
\begin{aligned}
&\widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \kappa_\beta^{(1)}, \dots, \kappa_\beta^{(j^*-1)}, \bar{\kappa}_{\beta,0}, \dots, \bar{\kappa}_{\beta,i^*},\ (x_i, \mathsf{RO}(x_i))_{i\in[q_{\mathsf{RO}}]}))\\
&\quad \geq \widetilde{\mathbb{H}}_\infty\Big(c_\beta \mid h_\beta, c_{1-\beta}, \kappa_\beta^{(1)}, \dots, \kappa_\beta^{(j^*-1)}, \bar{\kappa}_{\beta,0}, \dots, \bar{\kappa}_{\beta,i^*}, \mathsf{RO}\Big)\\
&\quad \geq \widetilde{\mathbb{H}}_\infty\Big(c_\beta \mid h_\beta, c_{1-\beta}, \kappa_{1-\beta}^{(1)}, \dots, \kappa_{1-\beta}^{(j^*-1)}, \bar{\kappa}_{1-\beta,0}, \bar{\kappa}_{1-\beta,i^*-1}, \bar{\kappa}_{\beta,i^*}, \mathsf{RO}\Big) \quad (5)\\
&\quad \geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \bar{\kappa}_{\beta,i^*}, i^*, j^*, \mathsf{RO}) \quad (6)\\
&\quad \geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \mathsf{RO}) - (\log\lambda + \log q_T + 1) \quad (7)\\
&\quad \geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid c_{1-\beta}, \mathsf{RO}) - (\log\lambda + \log q_T + 1) \quad (8)
\end{aligned}
$$

Where Eq. (9) follows because, by the check performed by LEAK, for any $j < j^*$ we have $\kappa_0^{(j)} = \kappa_1^{(j)}$ for any $i < i^*$ we have $\bar{\kappa}_{0,i} = \bar{\kappa}_{1,i}$, Eq. (10) follows because $\kappa_{1-\beta}^{(j)}, \bar{\kappa}_{1-\beta,i}$ are function of $c_{1-\beta}$, Eq. (11) follows by the chain rule for average conditional min-entropy and noticing that the random variable Z_β and the random variable j^* need a total of $\log q_T$ bits to be represented and Eq. (12) follows by independence of h_β.

We now analyze the advantage of B. First notice that by the claim above and taking an union bound over the elements in $\bar{\mathcal{Q}}_{\mathsf{A}} \cup \bar{\mathcal{Q}}_{1-\beta}$ we have that for $\beta \in \{0,1\}$:

$$\mathbb{P}\left[c_\beta \in \bar{\mathcal{Q}}_{\mathsf{A}} \cup \bar{\mathcal{Q}}_{1-\beta}\right] \leq (q_{\mathsf{RO}} + q \cdot q_T) 2^{-\alpha_\Pi(\lambda) + \log\lambda + \log q_T + 1}.$$

We condition on the event that $\forall \beta : c_\beta \notin \bar{\mathcal{Q}}_{\mathsf{A}} \cup \bar{\mathcal{Q}}_{1-\beta}$. Under this condition the distributions of $(h_\beta)_{\beta\in\{0,1\}}$ and $(\mathsf{RO}(c_\beta))_{\beta\in\{0,1\}}$, given the full view of the adversary, are exactly the same, because the adversary could query c_β to the random oracle only inside the tampering functions $(f_\beta^{(j)})_{j\in[q_T]}$, but in this case the reduction would answer with h_β.

We further condition on the event that no collisions are found in RO on an execution of B. Notice that the probability of finding a collision is upper bounded by $\frac{(q_{\mathsf{RO}} + q\cdot q_T)^2}{2^{2\lambda}}$.

The adversary B simulates almost perfectly the experiment to A. Indeed, if the adversary B returns a message $\tilde{\mu} \neq \perp$ to A at the j-th tampering query then, since we assumed that there aren't collisions in the RO, it musts be that $c_\beta^\star = \tilde{c}_\beta$, where the former is computed by the leakage function $g_\beta^{(j)}$ and the latter is computed by the leakage function $g_{1-\beta}^{(j)}$.

The only difference between the simulation of B and the real experiment is that, at step 4 it could happen that B returns $\perp$ but the tampering query in the real experiment would output a message different than $\perp$.

Let j be the index when this event happens for the first time. If B returns $\perp$ then either the procedure LEAK finds two mismatching outputs from the leakage oracle or $\exists \beta$ s.t. the leakage function $g_\beta^{(j)}$ outputs $\perp$.

The first case reduces to the event of finding a collision in the RO which we assumed that cannot happen, in fact, $\mathsf{Dec}(\tilde{c}_0, c_1^\star) \neq \mathsf{Dec}(c_0^\star, \tilde{c}_1)$ but $\mathsf{RO}(\beta \| \tilde{c}_\beta) = \mathsf{RO}(\beta \| c_\beta^\star)$ for $\beta \in \{0,1\}$. The second case instead is more interesting. In fact, $\mathcal{Q}_\mathsf{A} \cup \mathcal{Q}_\beta$ might not cover the full set of random oracle queries that the adversary A can do through the tampering queries, thus, in principle, it could happen that the reduction cannot find a tuple $(1-\beta \| c_{1-\beta}^\star, \tilde{h}_{1-\beta}) \in \mathcal{Q}_\mathsf{A} \cup \mathcal{Q}_\beta$ but, nevertheless, the adversary queried $\tilde{c}_{1-\beta} = c_{1-\beta}^\star$ to the random oracle in one of the tampering queries $f_{1-\beta}^{(j')}$ for $j' \leq j$, i.e., $(1-\beta \| \tilde{c}_{1-\beta}) \in \bar{\mathcal{Q}}_{1-\beta}$. We show that the adversary cannot guess, using the tampering query $f_\beta^{(j)}$, the valid value for $\tilde{h}_{1-\beta}$ that would make pass the consistency check of the decoding algorithm Dec^*. Recall that we condition on j being the first index where the bad event described before could happen. Thus we have that for all $j' < j$ the output of the leakage functions $g_0^{(j')}$ and the output of $g_1^{(j')}$ agree. Also, as just said above, we condition on $(1-\beta \| \tilde{c}_{1-\beta}) \in \bar{\mathcal{Q}}_{1-\beta} \wedge (1-\beta \| c_{1-\beta}^\star, \tilde{h}_{1-\beta})) \notin \mathcal{Q}_\beta \cup \mathcal{Q}_\mathsf{A}$, and we want to compute the probability that $\tilde{h}_{1-\beta} = \mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta})$.

We compute the average conditional min-entropy of $\mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta})$ given the full view of the j-th leakage function $g_\beta^{(j)}$:

$$\begin{aligned}\widetilde{\mathbb{H}}_\infty(\mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta}) | \mathcal{Q}_\mathsf{A}, \mathcal{Q}_\beta, (\tilde{\mu}^{(j')})_{j'<j}, c_\beta, h_{1-\beta}) = \\ \widetilde{\mathbb{H}}_\infty(\mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta}) | \mathcal{Q}_\mathsf{A}, \mathcal{Q}_\beta, c_\beta, h_{1-\beta}) = \\ \widetilde{\mathbb{H}}_\infty(\mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta}) | c_\beta, h_{1-\beta}) = 2\lambda.\end{aligned}$$

First we notice that the tuple $(\mathcal{Q}_\mathsf{A}, \mathcal{Q}_\beta, (\tilde{\mu}^{(j')})_{j'<j}, c_\beta, h_{1-\beta})$ is indeed the full view of the leakage function $g_\beta^{(j)}$, as all the randomness in the experiment comes from the random oracle queries, the challenge codeword and, possibly, the outputs of the leakage oracle. In the derivation above, the first equation holds because $(\tilde{\mu}^{(j')})_{j'<j}$ can be computed as deterministic function of $\mathcal{Q}_\mathsf{A}, \mathcal{Q}_\beta, c_\beta, h_{1-\beta}$, the second equation holds because we assumed that $(1-\beta \| c_{1-\beta}^\star, \tilde{h}_{1-\beta}) \notin \mathcal{Q}_\mathsf{A} \cup \mathcal{Q}_\beta$. This shows that the probability that $g_\beta^{(j)}$ computes $\tilde{h}_{1-\beta}$ at the j-th query equal to $\mathsf{RO}(1-\beta \| \tilde{c}_{1-\beta})$ is $2^{-2\lambda}$, even when $(1-\beta \| \tilde{c}_{1-\beta}) \in \bar{\mathcal{Q}}_{1-\beta}$. We can prove that the same holds when $(1-\beta \| \tilde{c}_{1-\beta}) \notin \mathcal{Q}_{1-\beta}$, in this case the value was never queried to the RO, thus the adversary can guess it with probability $2^{-2\lambda}$. Taking an union bound over all the tampering oracle queries made by A the

probability that B outputs $\perp$ but the real experiment would have not is bounded by $q_T 2^{-2\lambda}$.

Putting all together, we can conclude that the advantage of B is bigger or equal to

$$\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi^*,\mathsf{A}}(\lambda) - \frac{q_T}{2^{2\lambda}} + \frac{(q_{\mathsf{RO}}+q\cdot q_T)^2}{2^{2\lambda}}.$$

Claim. The adversary B is $(2\lambda + 2\log q_{\mathsf{RO}} + \log q_T,\ q_{\mathsf{RO}})$-admissible.

Proof. The number of random oracle queries made by B is equal to the number of random oracle queries made by A. We can assume w.l.g. that the last tampering query of A results to self-destruct. Let j^* be the index of the tampering query where the procedure $\textsc{Leak}(g^{(j^*)}_\beta, g^{(j^*)}_\beta)$ outputs $\perp$ for the first time. Let i^* be the index of the iteration where $\textsc{Leak}(g^{(j^*)}_\beta, g^{(j^*)}_\beta)$ stops, we have $i^* < \lambda$ and let $\bar\kappa_{0,0}, \bar\kappa_{1,0}, \dots, \bar\kappa_{0,i^*}, \bar\kappa_{1,i^*}$ be the i^* bits leaked by the sub-routine. The adversary B leaks the values $\kappa^{(1)}_\beta := g^{(1)}_\beta(c_\beta), \dots, \kappa^{(j^*-1)}_\beta := g^{(j^*-1)}_\beta(c_\beta)$ and the values $\bar\kappa_{\beta,0}, \dots, \bar\kappa_{\beta,i^*}$ to answer the tampering queries made by A. Let $(x_1, \dots, x_{q_{\mathsf{RO}}})$ be the random oracle queries made by B. For $\beta \in \{0,1\}$:

$$\begin{aligned}
&\widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \kappa^{(1)}_\beta, \dots, \kappa^{(j^*-1)}_\beta, \bar\kappa_{\beta,0}, \dots, \bar\kappa_{\beta,i^*},\ (x_i, \mathsf{RO}(x_i))_{i\in[q_{\mathsf{RO}}]})) \\
&\quad\geq \widetilde{\mathbb{H}}_\infty\Big(c_\beta \mid h_\beta, c_{1-\beta}, \kappa^{(1)}_\beta, \dots, \kappa^{(j^*-1)}_\beta, \bar\kappa_{\beta,0}, \dots, \bar\kappa_{\beta,i^*}, \mathsf{RO}\Big) \\
&\quad\geq \widetilde{\mathbb{H}}_\infty\Big(c_\beta \mid h_\beta, c_{1-\beta}, \kappa^{(1)}_{1-\beta}, \dots, \kappa^{(j^*-1)}_{1-\beta}, \bar\kappa_{1-\beta,0}, \bar\kappa_{1-\beta,i^*-1}, \bar\kappa_{\beta,i^*}, \mathsf{RO}\Big) && (9)\\
&\quad\geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \bar\kappa_{\beta,i^*}, i^*, j^*, \mathsf{RO}) && (10)\\
&\quad\geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid h_\beta, c_{1-\beta}, \mathsf{RO}) - (\log\lambda + \log q_T + 1) && (11)\\
&\quad\geq \widetilde{\mathbb{H}}_\infty(c_\beta \mid c_{1-\beta}, \mathsf{RO}) - (\log\lambda + \log q_T + 1) && (12)
\end{aligned}$$

Where Eq. (9) follows because, by the check performed by $\textsc{Leak}$, for any $j < j^*$ we have $\kappa^{(j)}_0 = \kappa^{(j)}_1$ for any $i < i^*$ we have $\bar\kappa_{0,i} = \bar\kappa_{1,i}$, Eq. (10) follows because $\kappa^{(j)}_{1-\beta}, \bar\kappa_{1-\beta,i}$ are function of $c_{1-\beta}$, Eq. (11) follows by the chain rule for average conditional min-entropy and noticing that the random variable Z_β and the random variable j^* need a total of $\log q_T$ bits to be represented and Eq. (12) follows by independence of h_β.

□

Remark 2. Similarly to [29,30,34], we do not consider leakage resilience for our continuous non-malleable randomness encoder. Nevertheless, our reduction can easily handle leakage functions by hardcoding the hash values h_0, h_1 and forwarding the leakage queries to its own oracle. However, there is a catch: the leakage queries sent by the adversary cannot have access to the random oracle. In fact, an attacker could forward leakage functions that make random-oracle queries on behalf of the adversary. These *obfuscated* random oracle queries could not be seen by our reduction thus invalidating our observability-based argument.

The Theorem 1 gives an upper bound to the advantage of any adversary against the continuous non-malleability of Π^*. To give a full picture, in Sec. 6 (Corollary 1) we give a lower bound based on the random-oracle query complexity and randomness complexity of the underlying randomness encoder Π. Informally, the theorem states the existence of an adversary whose random-oracle query complexity is $\Omega(2^\lambda)$, tampering-oracle complexity is $O(n)$ and advantage is at least $(1/e)^8$.

4 Compiler from randomness encoders to code schemes

In this section we recall the compiler of Coretti, Faonio and Venturi [12]. The compiler makes use of an authenticated encryption scheme. Due to space constraints we defer its syntax and security definitions to Appendix A. A (k, m)-SKE scheme Ω encrypts k-bit messages and outputs ciphertexts of size m. We consider the standard security property of *authenticity* whose security game is denoted by $\mathbf{G}_\Omega^{\mathsf{auth}}$, and the standard security property of *indistinghuishability* which security game is denoted by $\mathbf{G}_\Omega^{\mathsf{ind}}$. Let $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ be a n-randomness-encoder, and $\Omega = (\mathsf{AEnc}, \mathsf{ADec})$ be a (k, m)-SKE scheme. Consider the following construction of a (k, n')-code $\Sigma' = (\mathsf{Enc}', \mathsf{Dec}')$, where $n' := m + n$.

$\mathsf{Enc}'(s)$**:** Upon input a value $s \in \{0,1\}^k$, compute $c_0, c_1 \leftarrow\$ \mathsf{REncode}(1^\lambda)$, let $\kappa \leftarrow \mathsf{Dec}(c_0, c_1)$, and compute $\gamma \leftarrow\$ \mathsf{AEnc}(\kappa, s)$; return c'_0, c'_1 where $c'_\beta = (c_\beta, \gamma)$ for $\beta \in \{0,1\}$.

$\mathsf{Dec}'(c'_0, c'_1)$**:** Parse $c'_\beta := (c_\beta, \gamma_\beta)$ for $\beta \in \{0,1\}$. If $\gamma_0 \neq \gamma_1$, return $\perp$ and self destruct; else let $\tilde{\kappa} = \mathsf{Dec}(c_0, c_1)$. If $\tilde{\kappa} = \perp$, return $\perp$ and self destruct; else return the same as $\mathsf{ADec}(\tilde{\kappa}, \gamma_0)$.

The difference between the compiler Σ described above and the compiler of [12] is that our compiler starts from a CNMRE, while their construction starts from a noisy-leakage-resilient CNMC. In particular, their proof strategy relies on the noisy-leakage resilience of the underlying CNMC while our proof strategy does not. A similar strategy to ours was recently used by Brian, Faonio and Venturi in the context of continuous non-malleable secret sharing schemes [8].

Theorem 2. *For any adversary* A *which makes at most* $q_T := q_T(\lambda)$ *tampering oracle queries there exist adversaries* B *which makes at most* $(m+1) \cdot q_T$ *tampering oracle queries, and adversaries* B' *and* B'' *such that*

$$\mathbf{Adv}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda) \leq 2\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{B}}(\lambda) + q_T \cdot \mathbf{Adv}^{\mathsf{auth}}_{\Omega,\mathsf{B}'}(\lambda) + \mathbf{Adv}^{\mathsf{ind}}_{\Omega,\mathsf{B}''}(\lambda).$$

5 Our Leakage-Resilient Randomness Encoders

We give two constructions Π_1 and Π_2 of LRREs, due to space constraints the proofs of security of Π_1 and Π_2 appear in the full version [**?**]. Notably the randomness encoder Π_2 has optimal leakage parameter, namely the leakage parameter is only λ bits smaller than the size of the codeword.

Let p be a prime such that $p \geq 2^\lambda$ and let $m \in \mathbb{N}$. Consider the following $(m \log p)$-randomness-encoder $\Pi_1 = (\mathsf{REncode}_1, \mathsf{Dec}_1)$:

$\mathsf{REncode}_1(1^\lambda)$**:** Sample column vectors $\boldsymbol{x}_0, \boldsymbol{x}_1 \leftarrow\$\ \mathbb{Z}_p^m$. Output c_0, c_1 where c_β is the binary representation of x_β

$\mathsf{Dec}_1(c_0, c_1)$**:** Parse c_β as a vector $\boldsymbol{x}_\beta \in \mathbb{Z}_p^m$. Return the binary representation of $\boldsymbol{x}_0^T \cdot \boldsymbol{x}_1 \in \mathbb{Z}_p$.

Theorem 3. *Let* $\ell \leq m \log p$*, for any* q*, the* Π_1 *scheme is* $(O(2^{-\lambda}), \ell, q)$*-noisy-leakage resilient for* $\ell \leq (m+1)\log p/2 - 2\lambda$*. In more detail, for any* (ℓ, q)*-admissible adversary* A*:* $\mathbf{Adv}^{\mathsf{lrs}}_{\Pi_1,\mathsf{A}}(\lambda) \leq 2^{-(m-1)\log p/2+\ell}$.

The theorem follows easily from the following two lemmas. The first lemma proves that the inner product over large field is an *average case* two-source extractor. the lemma is taken from Dodis et al. [17] (Lemma B.2 in full version). The second lemma was proved by Dziembowski and Pietrzak [21].

Lemma 2. *Let* $\mathbf{u}$ *be uniformly random over* $\mathbb{Z}_p$*, for any unbounded distinguisher* D*, any random variables* $\boldsymbol{x}_0, \boldsymbol{x}_1 \in \mathbb{Z}_p^m, \mathbf{Z} \in \{0,1\}^*$ *such that* $\boldsymbol{x}_0$ *and* $\boldsymbol{x}_1$ *are independent conditioned on* $\mathbf{Z}$*:*

$$\left|\mathbb{P}\left[\mathsf{D}(\boldsymbol{x}_0^T \cdot \boldsymbol{x}_1, \mathbf{Z}) = 1\right] - \mathbb{P}\left[\mathsf{D}(\mathbf{u}, \mathbf{Z}) = 1\right]\right| \leq 2^{((m+1)\log p - \ell_0 - \ell_1)/2}.$$

where $\ell_\beta = \widetilde{\mathbb{H}}_\infty(\boldsymbol{x}_\beta | \mathbf{Z})$.

The second lemma proves that independence is maintained even after split-state leakage.

Lemma 3. *Let $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ be two independent random variables. For any adversary $\mathbf{z} \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{leak}}((\boldsymbol{x}_0,\boldsymbol{x}_1),\cdot,\cdot)}$, the random variables $(\boldsymbol{x}_1|\mathbf{z})$ and $(\boldsymbol{x}_0|\mathbf{z})$ are independent.*

Proof (of Thm. 3). First notice that the randomness encoder Π_1 does not use the random oracle, so queries to the random oracle have no impact in our security analysis. Recall that in the game $\mathbf{G}^{\mathsf{lrre}}_{\Pi_1,\mathsf{A}}$ the adversary receives two keys κ_0 and κ_1, it has leakage oracle access to the codeword and it outputs a bit. Consider the hybrid experiment $\mathbf{H}$ that is identical to $\mathbf{G}^{\mathsf{lrre}}$ but where both the key κ_0 and κ_1 are chosen uniformly at random from $\{0,1\}^\lambda$. It is not hard to see that $\mathbb{P}[\mathbf{H}=1]=\frac{1}{2}$. We show that the distribution $\mathbf{G}^{\mathsf{lrre}}_{\Pi_1,\mathsf{A}}$ and the distribution $\mathbf{H}$ are statistically close.

Let $\mathbf{Z}$ be the state of A before it outputs its guess. W.l.o.g. the state contains all the leakage oracle queries and answers and all the randomness used by A. By Lemma 3 we have $(\boldsymbol{x}_0|\mathbf{Z})$ and $(\boldsymbol{x}_1|\mathbf{Z})$ are independent, moreover for $\beta \in \{0,1\} : \widetilde{\mathbb{H}}_\infty(\boldsymbol{x}_\beta|\mathbf{Z}) \geq m \log p - \ell$, thus by Lemma 2 the theorem follows.

□

Let $(m-1)\log p/2 \geq n$ and let RO be a random oracle[3] with output $\mathbb{Z}_p^m$. Consider the following n-randomness-encoder $\Pi_2 = (\mathsf{REncode}_2, \mathsf{Dec}_2)$.

$\mathsf{REncode}(1^\lambda)$**:** Sample and output $c_0, c_1 \leftarrow\!\!\$\ \{0,1\}^n$.

$\mathsf{Dec}(c_0, c_1)$**:** Compute $\boldsymbol{x}_i \leftarrow \mathsf{RO}(c_i)$ for $i \in \{0,1\}$ and return the binary representation of $\boldsymbol{x}_0^T \cdot \boldsymbol{x}_1$.

Theorem 4. *Let $\ell + \lambda \leq n$, for any $q_{\mathsf{RO}}(\lambda) \in \mathbb{N}$, the encoding scheme Π_2 is $(O(2^{-\lambda} q_{\mathsf{RO}}), \ell, q_{\mathsf{RO}})$-noisy-leakage resilient. In more detail, for any (ℓ, q_{RO})-admissible adversary A: $\mathbf{Adv}^{\mathsf{lrs}}_{\Pi_2,\mathsf{A}}(\lambda) \leq 2^{\ell-n}(2q_{\mathsf{RO}}+2)$.*

The idea for the proof is that the adversary can either leak from c_i or directly from $\mathsf{RO}(c_i)$. The former kind of leakage cannot give any advantage to the adversary, since the adversary should be able to guess $n - \ell \geq \lambda$ bits to obtain any information about $\mathsf{RO}(c_i)$, the latter form of leakage is protected by the same argument of the leakage resilience of Π_1.

Proof. We reduce to the security of Π_1. Consider the hybrid experiment $\mathbf{H}_\mathsf{A}$ equivalent to the $\mathbf{G}^{\mathsf{lrre}}_{\Pi_2,\mathsf{A}}$ experiment but where the output of the

[3] It can be easily realized using a RO' with codomain $\{0,1\}^{2\lambda}$.

experiment is a random bit if A queries the random oracle either on c_0 or on c_1.

Claim. $\left|\mathbb{P}\left[\mathbf{G}^{\mathsf{lrre}}_{\Pi_2,\mathsf{A}}(\lambda)=1\right]-\mathbb{P}\left[\mathbf{H}_{\mathsf{A}}(\lambda)=1\right]\right| \leq 2q_{\mathsf{RO}}2^{\ell-n}$

Proof. The two experiments proceed exactly the same until the bad event that the adversary A queries the random oracle on c_0 or on c_1 happens. Let us call such event $\mathtt{Bad}$. Therefore we need only to bound the probability of $\mathtt{Bad}$. Let $\mathtt{Bad}_{i,b}$ the event that A queries c_b at its i-th query to RO.

$$\mathbb{P}\left[\mathtt{Bad}\right] \leq \sum_{i\in[q_{\mathsf{RO}}],b\in\{0,1\}} \mathbb{P}\left[\mathtt{Bad}_{i,b}\right] \leq 2q_{\mathsf{RO}} \max_{i,b} \mathbb{P}\left[\mathtt{Bad}_{i,b}\right]$$

Let $\mathbf{z}_i$ be the state of the adversary at the i-th query (which includes also all the random-oracle queries made), by the definition of average conditional min-entropy we have that $\mathbb{P}\left[\mathtt{Bad}_{i,b}\right] \leq 2^{-\widetilde{\mathbb{H}}_\infty(c_b|\mathbf{z}_i)}$. In fact, we can define a predictor that runs A with state set to $\mathbf{z}_i$ and outputs as its own guess the random oracle query made by A. By the (ℓ, q_{RO})-admissibility of A we have that $\widetilde{\mathbb{H}}_\infty(c_b|\mathbf{z}_i) \geq \widetilde{\mathbb{H}}_\infty(c_b|c_{1-b},\mathsf{RO}) - \ell = n-\ell$.

Claim. For any (ℓ, q_{RO})-admissible A there exists an $(\ell, 0)$-admissible A' such that: $|\mathbb{P}\left[\mathbf{H}_{\mathsf{A}}(\lambda)=1\right]-1|/2 = \mathbf{Adv}^{\mathsf{lrs}}_{\Pi_1,\mathsf{A}'}(\lambda)$.

Proof. Let A' be the adversary that simulates A and keeps the list $\mathcal{Q}$ of random oracle queries made by A. Additionally, A' samples a random function $\mathcal{H}$ with domain $\{0,1\}^n$ and co-domain $\mathbb{Z}_p^m$ and answers the random oracle query of A using $\mathcal{H}$. (Notice that A' does not need to be efficiently computable.) Also, A' samples two random values $c_0, c_1 \leftarrow\!\!\$\ \{0,1\}^n$. Whenever the adversary A sends a leakage oracle query (g_0, g_1) the adversary A' sends the leakage oracle query (g'_0, g'_1) where the leakage functions g'_β for $\beta\in\{0,1\}$ are defined below:

Leakage function $g'_\beta(x)$:

- Run $g_\beta(c_\beta)$;
- Upon random oracle query z from g_β if $z = c_\beta$ then return x else return $\mathcal{H}(z)$ to g_β.
- Output what g_β does.

Eventually A outputs its guess b, the adversary A' first checks that $c_\beta \notin \mathcal{Q}$ for $\beta \in \{0,1\}$, if so it returns a random bit else A' returns b.

Notice that A' simulates perfectly the hybrid experiment as long as neither c_0 nor c_1 are queried to the random oracle by A. In fact, if we condition on $\forall \beta : c_\beta \notin \mathcal{Q}$ then the adversary A' is running the hybrid experiment $\mathbf{H}$ with the random oracle $\mathcal{H}'(x)$ that answers $\mathcal{H}(x)$ if $x \notin \{c_0, c_1\}$, with $\boldsymbol{x}_0$ if $x = c_0$ and with $\boldsymbol{x}_1$ if $x = c_1$, where $(\boldsymbol{x}_0, \boldsymbol{x}_1)$ is the challenge codeword for A'. On the other hand, when this bad event happens the adversary A' will surely outputs a random bit, as the hybrid experiment does.

Putting the claims together, by the triangular inequality, and because of the relation $(m-1)\log p/2 + \lambda \geq n$ we have the statement of the theorem. $\square$

5.1 Instantiations

We present two instantiations for our continuous non-malleable randomness encoders. By joining together the results of Thm. 1 and Thm. 3 we obtain a $(m \log p + 2\lambda)$-randomness-encoders scheme Π_1^* with concrete security being:

$$\max_{\mathsf{A}} \mathbf{Adv}^{\mathsf{cnmre}}_{\Pi_1^*,\mathsf{A}}(\lambda) \leq \exp(-(m-1)\log p/2 + \log \lambda + \log q_T + 1) + \frac{q_T + (q_{\mathsf{RO}} + q \cdot q_T)^2}{2^{2\lambda}} + \frac{(q_{\mathsf{RO}} + q \cdot q_T)\lambda q_T}{2^{m \log p - 1}}.$$

For concreteness, suppose that an adversary can make $q_{\mathsf{RO}} + q \cdot q_T = 2^{40}$ random-oracle queries and $q_T = 2^{20}$ tampering-oracle queries, then to have ≈128-bits of security we need to set $(m-1)\log p \geq 312$ and $p \geq 2^{128}$, for example we can set we $m = 2$ and $p \geq 2^{312}$. Instantiating the random oracle using SHA256 then the codeword size would be approximately 2×880 bits. The time complexity of the decoding algorithm would be approximately the same as two SHA256 functions plus an inner-product between two vectors in $\mathbb{Z}_p^m$. Our second instantiation is derived by joining together the results of Thm. 1 and Thm. 4. We obtain a $n+\lambda$-randomness-encoders scheme Π_2^* with concrete security being:

$$\max_{\mathsf{A}} \mathbf{Adv}^{\mathsf{cnmre}}_{\Pi_2^*,\mathsf{A}}(\lambda) \leq \exp(-n + \log \lambda + \log q_T + \log q_{\mathsf{RO}} + 2) + \frac{q_T + (q_{\mathsf{RO}} + q \cdot q_T)^2}{2^{2\lambda}} + \frac{(q_{\mathsf{RO}} + q \cdot q_T)\lambda q_T}{2^{m \log p - 1}}.$$

Assuming the same setup of before, to get ≈128-bits of security we need to set $n \geq 128+69$. Using SHA256 the codeword size would be approximately 2×453 bits. The time complexity of the decoding algorithm would be the same of 8 SHA256 functions. In particular, the size of the codeword is in total only ≈ 7 times bigger than the size of the derived key.

6 Lower Bounds for CNMREs in the ROM

Definition 7. *Given a n-randomness encoder Π, an algorithm* A *is a* $(\epsilon, q_{\mathsf{RO}})$-finder *for Π if* $\mathsf{A}(1^\lambda)$ *makes at most* $q_{\mathsf{RO}}(\lambda)$ *random oracle queries and if:*

$$\mathbb{P}\left[\begin{matrix}\perp \neq \mathsf{Dec}(c_0, c_1) \neq \mathsf{Dec}(c_0, c'_1) \neq \perp \\ (c_0 = c'_0) \vee (c_1 = c'_1)\end{matrix} : (c_0, c_1, c'_0, c'_1) \leftarrow \mathsf{A}^{\mathsf{RO}}(1^\lambda)\right] \geq \epsilon(\lambda)$$

In the next theorem first we show that the existence of a finder is sufficient to break continuous non-malleability, then we show that even if there is no finder, we still can break continuous non-malleability given enough random oracle queries.

Theorem 5. *Let $n(\lambda) \in \mathbb{N}$ and let Π be a n-randomness encoder:*

1. *If there exists a $(\epsilon, q_{\mathsf{RO}})$-finder for Π then for any $\frac{\epsilon}{2} \geq \delta > 0$, Π is not a $(\frac{\epsilon}{2} - \delta, 0, q_{\mathsf{RO}})$-CNMRE. Namely, there exists an adversary A' making up to q_{RO} random oracle queries and $n+1$ tampering queries from $\mathcal{F}_{n,0}$, such that $\frac{\epsilon}{2} \leq \mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{A}'}(1^\lambda)$.*
2. *Suppose that $\mathsf{Enc}(1^\lambda)$ makes at most $q^{\mathsf{Enc}}_{\mathsf{RO}}(\lambda)$ random oracle queries and uses $r(\lambda)$ random bits. If for any $\epsilon, q'_{\mathsf{RO}}$, there does not exist a $(\epsilon, q'_{\mathsf{RO}})$-finder then for any $\delta > 0$ any q_{RO}, q such that $q_{\mathsf{RO}} + q \geq 2^r \cdot q^{\mathsf{Enc}}_{\mathsf{RO}}$ the scheme Π is not a $(1/2 - \delta, q, q_{\mathsf{RO}})$-CNMRE. Namely, there exists an adversary A' making up to q_{RO} random oracle queries and 1 tampering query from $\mathcal{F}_{n,q}$ such that $\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{A}'}(1^\lambda) = 1/2$.*

Proof (Sketch). For the first part of the theorem let the adversary A' first run the finder algorithm. For simplicity, let us assume that the finder outputs a tuple (c_0, c_1, c'_0, c'_1) where $c_0 = c'_0$. If the output of the finder is not valid then the adversary A' outputs a random bit. Else, for $i = 0, \ldots, n-1$ sends the tampering query $(f_0^{(i)}, f_1^{(i)})$ where $f_0^{(i)}$ returns c_0 and $f_1^{(i)}(c_1^*)$ returns either c_1 or to c'_1 depending on the on the i-th bit of c_1^*. After this process, the adversary can extract in full the value c_1^* of the target codeword, thus it can send a last tampering query that breaks non-malleability. It is clear that the adversary wins the game $\mathbf{G}^{\mathsf{cnmre}}$ with probability at least $\epsilon + (1-\epsilon)\frac{1}{2}$.

For the second part of the theorem, for simplicity we consider first the case where $q = 0$. Since no finder exists, then for any c_0 there exists unique c_1 such that $(c_0, c_1) \in \{\mathsf{Enc}(1^\lambda; \rho) : \rho \in \{0,1\}^r\}$. Thus the the adversary

A' can compile a bijection L_0 such that $L_0(c_0) = c_1$, also let L_1 be the inverse of L_0. To compile such bijections the adversary A' needs at most $2^r \cdot q_{\mathsf{RO}}^{\mathsf{Enc}}$ random oracle queries. Then given such bijections, the adversary sends the tampering queries f_0, f_1 where $f_\beta(c_\beta)$ computes $c_{1-\beta} \leftarrow L_\beta(c_\beta)$, decodes $\kappa \leftarrow \mathsf{Dec}(c_0, c_1)$, and if $\kappa = \kappa_0$ sets the codeword to $\perp$, else leaves the codeword untouched. It is clear that the adversary wins the game $\mathbf{G}^{\mathsf{cnmre}}$ with probability 1.

For the case $q > 0$, we can consider an adversary that computes first partially the bijection L_0 using the budget of random-oracle queries q_{RO} and then finishes to computes the bijection L_0 using the budget of random-oracle queries that the tampering function can use.

Corollary 1. *Let $\Pi = (\mathsf{REncode}, \mathsf{Dec})$ be a randomness-encoder where $\mathsf{REncode}(1^\lambda)$ makes up to $q_{\mathsf{RO}}^{\mathsf{REncode}}(\lambda)$ queries to the random oracle and uses at most $r(\lambda)$ bits of randomness, and Dec makes up to $q_{\mathsf{RO}}^{\mathsf{Dec}}(\lambda)$ queries to the random oracle. Also, let Π be $(\epsilon, \ell, 2^r(q_{\mathsf{RO}}^{\mathsf{REncode}} + q_{\mathsf{RO}}^{\mathsf{Dec}}))$-noisy leakage resilient in the ROM, for any ϵ, ℓ where $\ell \geq 2$.*

Consider Π^* be our CNMRE from Sec. 3 instantiated with the randomness encoder Π. There exists an adversary A that makes up to $2^r(q_{\mathsf{RO}}^{\mathsf{REncode}} + q_{\mathsf{RO}}^{\mathsf{Dec}}) + 2^\lambda$ random oracle queries and up to $n+1$ tampering oracle queries from $\mathcal{F}_{n+2\lambda,0}$ such that:

$$(1/e)^{((\frac{1}{2}-\epsilon)2^\lambda-1)^2/2^{2\lambda-1}} \leq \mathbf{Adv}^{\mathsf{cnmre}}_{\Pi^*,\mathsf{A}}(\lambda).$$

In particular, when $\epsilon(\lambda) \in \mathtt{negl}(\lambda)$ then $(1/e)^8 \leq \mathbf{Adv}^{\mathsf{cnmre}}_{\Pi^*,\mathsf{A}}(\lambda)$.

Proof. We describe an $((1/e)^{((\frac{1}{2}-\epsilon)2^\lambda-1)^2/2^{2\lambda-1}}, 2^r(q_{\mathsf{RO}}^{\mathsf{REncode}} + q_{\mathsf{RO}}^{\mathsf{Dec}}) + 2^\lambda)$-finder for Π^*.

Finder $\mathsf{F}^{\mathsf{RO}}(1^\lambda)$:

1. Compute for any c_0 the set $\mathcal{E}(c_0) = \{c_1 | \exists \rho : (c_0, c_1) = \Pi.\mathsf{Enc}^{\mathsf{RO}}(1^\lambda; \rho)\}$;
2. Compute for any c_0 the set $\mathcal{M}(c_0) = \{\Pi.\mathsf{Dec}(c_0, c_1) | c_1 \in \mathcal{E}(c_0)\}$;
3. Find c_0^* such that $|\mathcal{M}(c_0^*)| = \max_{c_0} |\mathcal{M}(c_0)|$;
4. Find $c_1^{(1)}, c_1^{(2)} \in \mathcal{E}(c_0^*)$ such that $\mathsf{RO}(c_1^{(1)}) = \mathsf{RO}(c_2^{(2)})$ and $\Pi.\mathsf{Dec}(c_0^*, c_1^{(1)}) \neq \Pi.\mathsf{Dec}(c_0^*, c_1^{(2)})$.
5. If such tuple does not exist output $\perp$,
   else output $(\bar{c}_0, \bar{c}_1, \bar{c}'_0, \bar{c}'_1)$ such that:

$$\bar{c}_0 = \bar{c}'_0 := (c_0^*, \mathsf{RO}(c_1^{(1)})) \quad \bar{c}_1 := (c_1^{(1)}, \mathsf{RO}(c_0^*)) \quad \bar{c}'_1 := (c_1^{(2)}, \mathsf{RO}(c_0^*)).$$

We analyzes the probability that the finder outputs a valid triplet.

Claim. For any $\epsilon(\lambda) \in \mathbb{R}, \ell(\lambda) \in \mathbb{N}$, if Π is a $(\epsilon, \ell, 2^r(q_{\mathsf{RO}}^{\mathsf{REncode}} + q_{\mathsf{RO}}^{\mathsf{Dec}}))$-noisy-leakage resilient randomness encoder then $|\mathcal{M}(c_0^*)| \geq (\frac{1}{2} - \epsilon)2^\lambda$.

Proof (of the Claim). Suppose that for any c_0 we have $|\mathcal{M}(c_0)| < (\frac{1}{2} - \epsilon)2^\lambda$. Consider the following attacker against noisy-leakage resilience.

Adversary $\mathsf{B}(\kappa_0, \kappa_1)$:
1. Send the leakage function that on input c_0 outputs:
   - 1 if $\kappa_1 \in \mathcal{M}(c_0)$ but $\kappa_0 \notin \mathcal{M}(c_0)$,
   - 0 if $\kappa_0 \in \mathcal{M}(c_0)$ but $\kappa_1 \notin \mathcal{M}(c_0)$,
   - $\perp$ if $\kappa_0 \in \mathcal{M}(c_0)$ and $\kappa_1 \in \mathcal{M}(c_0)$.

   let b' be the output of the leakage function;
2. If $b' = \perp$ output a random bit, else output b'.

Let b be the challenge bit, the probability of $\kappa_{1-b} \in \mathcal{M}(c_0)$ is strictly smaller than $(\frac{1}{2} - \epsilon)2^\lambda/2^\lambda$. Notice that $\kappa_b \in \mathcal{M}(c_0)$, thus the adversary B successfully guesses the challenge bit whenever the output of the second leakage function is not $\perp$. We can conclude that the advantage of B is strictly greater then ϵ.

By the claim above there exists at least $(\frac{1}{2} - \epsilon)2^\lambda$ different values c_1 that decodes correctly with c_0^* and whose decoded messages are pairwise different. Thus applying the birthday-paradox bound the probability that the finder successfully outputs a valid tuple is at least $(1/e)^{((\frac{1}{2}-\epsilon)2^\lambda-1)^2/2^{2\lambda-1}}$. Finally, notice that the number of random oracle queries made by F are:

- $2^r(q_{\mathsf{RO}}^{\mathsf{Enc}} + q_{\mathsf{RO}}^{\mathsf{Dec}})$ to compute the sets $\mathcal{E}(c_0)$ for any c_0;
- At most 2^λ to compute the step 4.

By applying Theorem 5 point 1 we have the statement of the theorem.

References

1. Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *TCC 2016-A, Part II*, 2016.
2. Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *ACM STOC*, pages 459–468, 2015.
3. Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *ACM STOC*, pages 774–783, 2014.
4. Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *TCC 2015, Part I*, 2015.
5. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *CRYPTO*, pages 538–557, 2015.

6. Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT 2010*, 2010.
7. Gianluca Brian, Antonio Faonio, and Daniele Venturi. Continuously non-malleable secret sharing for general access structures. In *TCC 2019, Part II*, 2019.
8. Gianluca Brian, Antonio Faonio, and Daniele Venturi. Continuously non-malleable secret sharing: Joint tampering, plain model and capacity. Cryptology ePrint Archive, Report 2021/1128, 2021. https://ia.cr/2021/1128.
9. Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *48th ACM STOC*, 2016.
10. Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *TCC*, pages 440–464, 2014.
11. Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In *TCC*, pages 306–335, 2016.
12. Sandro Coretti, Antonio Faonio, and Daniele Venturi. Rate-optimizing compilers for continuously non-malleable codes. In *ACNS 19*, 2019.
13. Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In *TCC*, 2015.
14. Dana Dachman-Soled and Mukul Kulkarni. Upper and lower bounds for continuous non-malleable codes. In *PKC 2019, Part I*, 2019.
15. Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In *TCC*, pages 427–450, 2015.
16. Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN 10*, 2010.
17. Yevgeniy Dodis, Allison B. Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *52nd FOCS*, 2011.
18. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*.
19. Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In *ASIACRYPT 2011*, 2011.
20. Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO*, pages 239–257, 2013.
21. Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *48th FOCS*, 2007.
22. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS 2010*, 2010.
23. Antonio Faonio and Jesper Buus Nielsen. Non-malleable codes with split-state refresh. In *PKC 2017, Part I*, 2017.
24. Antonio Faonio, Jesper Buus Nielsen, Mark Simkin, and Daniele Venturi. Continuously non-malleable codes with split-state refresh. In *ACNS 18*, 2018.
25. Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In *CRYPTO 2019, Part II*, 2019.
26. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC 2014*, 2014.
27. Serge Fehr, Pierre Karpman, and Bart Mennink. Short non-malleable codes from related-key secure block ciphers. *IACR Trans. Symm. Cryptol.*, (1), 2018.
28. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

29. Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In *EUROCRYPT 2018*.
30. Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In *ACM CCS 2016*, 2016.
31. Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *ACM STOC*, pages 1144–1156, 2017.
32. Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *49th ACM STOC*, 2017.
33. Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.
34. Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In *CRYPTO 2018, Part III*, 2018.

$\underline{\mathbf{G}^{\mathsf{ind}}_{\Omega,\mathsf{A}}(\lambda)}$:	$\underline{\mathbf{G}^{\mathsf{auth}}_{\Omega,\mathsf{A}}(\lambda)}$:
$b \leftarrow\!\!\$\ \{0,1\};\ \kappa \leftarrow\!\!\$\ \{0,1\}^\lambda;$	$\kappa \leftarrow\!\!\$\ \{0,1\}^\lambda$
$(\mu_0, \mu_1, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0(1^\lambda);$	$(\mu, \alpha) \leftarrow\!\!\$\ \mathsf{A}_0(1^\lambda)$
$\gamma \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa, \mu_b);$	$\gamma \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa, \mu)$
$b' \leftarrow \mathsf{A}_1(\gamma, \alpha);$	$\gamma' \leftarrow\!\!\$\ \mathsf{A}_1(\gamma, \alpha)$
Return $b = b'$.	Return $\gamma' \neq \gamma \wedge \mathsf{ADec}(\kappa, \gamma') \neq \bot$.

Fig. 2: Experiments defining security of SKE.

A Preliminaries

We denote with $\lambda \in \mathbb{N}$ the security parameter. A function $\epsilon : \mathbb{N} \to [0,1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ.

We recall a standard Lemma from Dodis, Reyzin and Smith [18]:

Lemma 4. *Let A, B, C be random variables, then*

1. *For any $\delta > 0$, we have $\mathbb{H}_\infty(A|B = b)$ is at least $\widetilde{\mathbb{H}}_\infty(A|B) - \log(1/\delta)$ with probability at least $1 - \delta$ over the choice of b.*
2. *If B has at most 2^λ values then $\widetilde{\mathbb{H}}_\infty(A|B, C) \geq \widetilde{\mathbb{H}}_\infty(A|C) - \lambda$.*

A.1 Authenticated Encryption

A secret-key encryption (SKE) scheme is a tuple of algorithms $\Omega := (\mathsf{AEnc}, \mathsf{ADec})$ specified as follows: (1) The randomized algorithm AEnc takes as input a key $\kappa \in \{0,1\}^\lambda$, a message $\mu \in \{0,1\}^k$, and outputs a ciphertext $\gamma \in \{0,1\}^m$; (2) The deterministic algorithm ADec takes as input a key $\kappa \in \{0,1\}^\lambda$, a ciphertext $\gamma \in \{0,1\}^m$, and outputs a value $\mu \in \{0,1\}^k \cup \{\bot\}$ (where $\bot$ denotes an invalid ciphertext). The values $k(\lambda), m(\lambda)$ are all polynomials in the security parameter $\lambda \in \mathbb{N}$, and sometimes we call Ω an (k, m)-SKE scheme.

We say that Ω meets correctness if for all $\kappa \in \{0,1\}^\lambda$, all messages $\mu \in \{0,1\}^k$, we have that $\mathbb{P}\left[\mathsf{ADec}(\kappa, \mathsf{AEnc}(\kappa, \mu)) = \mu\right] = 1$ (the probability is taken over the randomness of AEnc).

Definition 8 (Security of SKE). *Let $\Omega = (\mathsf{KGen}, \mathsf{AEnc}, \mathsf{ADec})$ be a SKE scheme. We say that Ω is (ϵ, δ)-secure if the following holds for the games defined in Fig. 2. For all (possibly unbounded) adversaries A the following advantages $\mathbf{Adv}^{\mathsf{auth}}_{\Omega,\mathsf{A}}(\lambda) := \mathbb{P}\left[\mathbf{G}^{\mathsf{auth}}_{\Omega,\mathsf{A}}(\lambda) = 1\right]$ and $\mathbf{Adv}^{\mathsf{ind}}_{\Omega,\mathsf{A}}(\lambda) := \left|\mathbb{P}\left[\mathbf{G}^{\mathsf{ind}}_{\Omega,\mathsf{A}}(\lambda) = 1\right] - \frac{1}{2}\right|$ are negligible.*

Note that since both authenticity and indistinguishable encryption are one-time properties, information-theoretic constructions with such properties exist.

B Compiler to encoding schemes

Theorem 2. *For any adversary* A *which makes at most* $q_T := q_T(\lambda)$ *tampering oracle queries there exist adversaries* B *which makes at most* $(m+1) \cdot q_T$ *tampering oracle queries, and adversaries* B' *and* B'' *such that*

$$\mathbf{Adv}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(\lambda) \leq 2\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{B}}(\lambda) + q_T \cdot \mathbf{Adv}^{\mathsf{auth}}_{\Omega,\mathsf{B}'}(\lambda) + \mathbf{Adv}^{\mathsf{ind}}_{\Omega,\mathsf{B}''}(\lambda).$$

Proof. The proof proceeds by a sequence of hybrid experiments. We assume, w.l.o.g., that $\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(1^\lambda, \mu_0, \mu_1) = 1\right] \geq \frac{1}{2}$. We underline in gray the differences between consecutive hybrids. Fix μ_0, μ_1 and let $\mathbf{H}_1(1^\lambda)$ be the same as $\mathbf{G}^{\mathsf{cnmc}}_{\Pi,\mathsf{A}}(1^\lambda, \mu_0, \mu_1)$ but where:

- The target codeword is generated by computing $c_0, c_1 \leftarrow\!\!\$\ \mathsf{REncode}(1^\lambda)$, setting $\kappa_1 \leftarrow \mathsf{Dec}(c_0, c_1)$, sampling $\kappa_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and computing the chipertext $\gamma \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa_0, s)$;
- At each tampering query, let $f = (f_0, f_1)$ be the tampering function sent by the adversary and compute, let $(\tilde{c}_\beta, \tilde{\gamma}_\beta) = f_\beta(c'_\beta)$ for $\beta \in \{0,1\}$. The decoding algorithm checks if $\mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) = \kappa_1$, if this is the case (and if $\tilde{\gamma}_0 = \tilde{\gamma}_1$) then it decrypts the tampered ciphertext $\tilde{\gamma}_0$ using the key κ_0.

Consider the following reduction:

Adversary $\mathsf{B}(1^\lambda, \kappa_0, \kappa_1)$:

1. Sample a random bit b^* and set $\gamma \leftarrow \mathsf{AEnc}(\kappa_0, \mu_{b^*})$. Compute an auxiliary codewords $c_0^{(\alpha)}, c_1^{(\alpha)} \leftarrow \mathsf{REncode}(1^\lambda)$ such that $\mathsf{Dec}(c_0^{(0)}, c_1^{(1)}) = \mathsf{Dec}(c_0^{(1)}, c_1^{(0)}) = \bot$ and let $\kappa^{(\alpha)} \leftarrow \mathsf{Dec}(c_0^{(\alpha)}, c_1^{(\alpha)})$ for $\alpha \in \{0,1\}$. Start the adversary $\mathsf{A}(1^\lambda, \mu_0, \mu_1)$, set the flag $\mathtt{stop} \leftarrow 0$.
2. **Random oracle queries.** Whenever A sends a query x to the random oracle, forward the query to random oracle RO.
3. **Tampering oracle queries.** When the adversary A sends its j-th tampering query $(f_0^{(j)}, f_1^{(j)})$, if the flag $\mathtt{stop} = 1$ return $\bot$ else consider the following tampering function:

   Tampering function $f_\beta^{j,i}(c_\beta)$:

(a) Compute $(\tilde{c}_\beta, \tilde{\gamma}_\beta) \leftarrow f_\beta^{(j)}(c_\beta, \gamma)$;

(b) Let α be the i-th bit of $\tilde{\gamma}_\beta$, return $c_\beta^{(\alpha)}$.

For $i \in [m]$ send the tampering query $(f_0^{j,i}, f_1^{j,i})$ and:

- If the tampering oracle outputs $\kappa^{(\alpha)}$ then set $\alpha_i \leftarrow \alpha$;
- Else the tampering oracle outputs $\perp$ then return $\perp$ to A.

Let $\tilde{\gamma} = (\alpha_0, \ldots, \alpha_{m-1})$. Consider the tampering function $f'_\beta(c_\beta)$ that computes $(\tilde{c}_\beta, \tilde{\gamma}) \leftarrow f_\beta^{(j)}(c_\beta, \gamma)$ and outputs $\tilde{c}_\beta$.
Send the tampering query (f'_0, f'_1) and obtain $\tilde{\kappa}$.

- If $\tilde{\kappa} = \perp$ return $\perp$ to A and set the flag `stop` $\leftarrow 1$,
- if $\tilde{\kappa} = \diamond$ then set $\tilde{\kappa} \leftarrow \kappa_0$,
- compute $\tilde{\mu} \leftarrow \mathsf{ADec}(\tilde{\kappa}, \tilde{\gamma})$ if $\tilde{\mu} = \perp$ then set the flag `stop` $\leftarrow$ 1 and return $\perp$ to A else return $\tilde{\mu}$.

4. Eventually the adversary return a bit b', return $b^* = b'$.

Claim. $\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(1^\lambda, \mu_0, \mu_1) = 1\right] \leq 2\mathbf{Adv}^{\mathsf{cnmre}}_{\Pi,\mathsf{B}}(1^\lambda) + \mathbb{P}\left[\mathbf{H}_1(1^\lambda) = 1\right]$.

Proof (of the Claim). It is sufficient to prove that $\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmre}}_{\Pi,\mathsf{B}}(1^\lambda, \mu_0, \mu_1) = 1 | b = 0\right] = \mathbb{P}\left[\mathbf{G}^{\mathsf{cnmc}}_{\Sigma,\mathsf{A}}(1^\lambda) = 1\right]$, $\mathbb{P}\left[\mathbf{G}^{\mathsf{cnmre}}_{\Pi,\mathsf{B}}(1^\lambda, \mu_0, \mu_1) = 0 | b = 1\right] = \mathbb{P}\left[\mathbf{H}_1(1^\lambda) = 1\right]$. First we notice that at step 1 of B, the reduction samples two auxiliary codewords $c^{(0)}$ and $c^{(1)}$ such that $\mathsf{Dec}(c_0^{(0)}, c_1^{(1)}) = \mathsf{Dec}(c_0^{(1)}, c_1^{(0)}) = \perp$. If such constraint does not hold then we can break the non-malleability of Σ by applying Thm. 5. Also notice that, assuming that the constraint does not hold, then the advantage of the reduction to non-malleability of Σ is tighter than the advantage of B. Thus we assume that the condition holds. Independently of the challenge bit b, the adversary B simulates perfectly the tampering oracle queries. In fact, if the j-th tampering oracle query of A, on input the tampering function $(f_0^{(j)}, f_1^{(j)})$, outputs $\perp$ then either (1) the ciphertexts $\tilde{\gamma}_0$ and $\tilde{\gamma}_1$ are different or (2) $\mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) = \perp$ or (3) $\mathsf{ADec}(\tilde{\kappa}, \tilde{\gamma}_0) = \perp$. If the event (1) happens, let i be the first index where the ciphertexts $\tilde{\gamma}_0$ and $\tilde{\gamma}_1$ differ, the tampering query $(f_0^{j,i}, f_1^{j,i})$ returns either $c_0^{(0)}, c_1^{(1)}$ or $c_0^{(1)}, c_1^{(0)}$ which makes B to return $\perp$ to A. In the event (2) happens, then B would receive $\perp$ from the tampering oracle query with input (f'_0, f'_1). If the event (3) happens, then by definition of B, it would return $\perp$ to A.

On the other hand, if the j-th tampering oracle query of A outputs a message $\tilde{\mu} \neq \perp$ then, by inspection of B, it is easy to show that B would return the same message $\tilde{\mu}$ to A.

Finally notice that if $b = 0$ then the target codeword for Π encodes κ_0, thus the codeword $(c_0, \gamma), (c_1, \gamma)$ is distributed exactly as the output of Enc'. Else, if $b = 1$ the target codeword for Π encodes κ_1 so the codeword is distributed exactly as generated by $\mathbf{H}_1$.

Let $\mathbf{H}_2$ be the same as $\mathbf{H}_1$ but where for any tampering query (f_0, f_1) let $(\tilde{c}_0, \tilde{\gamma}), (\tilde{c}_1, \tilde{\gamma}')$ be the tampered codeword as computed by the tampering oracle, if $\mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) = \kappa_1$ and $\tilde{\gamma} \neq \gamma$ then the hybrid $\mathbf{H}_2$ directly returns $\perp$ and self destructs else, if $\mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) = \kappa_1$ and $\tilde{\gamma} = \gamma$, it directly returns $\diamond$. It is easy to show that, by reduction to the *authenticity* of Ω, the advantages of $\mathbf{H}_2$ and $\mathbf{H}_3$ are negligibly close.

Let $\mathbf{H}_3$ be the same as $\mathbf{H}_2$ but where the ciphertext in the target codeword is computed as $\gamma \leftarrow \mathsf{AEnc}(\kappa_0, 0^k)$. We can show that, by reduction to the *indistinguishability* of Ω, the advantages of $\mathbf{H}_2$ and $\mathbf{H}_3$ are negligibly close. Finally, it is easy to show that the advantage in $\mathbf{H}_3$ is $\frac{1}{2}$. □