Leakage Resilient ℓ-more Extractable Hash and Applications to Non-Malleable Cryptography

Aggelos Kiayias[1], Feng-Hao Liu[2], and Yiannis Tselekounis[3]

[1] University of Edinburgh & IOG, akiayias@inf.ed.ac.uk
[2] Florida Atlantic University, fenghao.liu@fau.edu
[3] Carnegie Mellon University, itseleko@cs.cmu.edu

Abstract. ℓ-more extractable hash functions were introduced by Kiayias et al. (CCS '16) as a strengthening of extractable hash functions by Goldwasser et al. (Eprint '11) and Bitansky et al. (ITCS '12, Eprint '14). In this work, we define and study an even stronger notion of leakage-resilient ℓ-more extractable hash functions, and instantiate the notion under the same assumptions used by Kiayias et al. and Bitansky et al. In addition, we prove that any hash function that can be modeled as a Random Oracle (RO) is leakage resilient ℓ-more extractable, while it is however, not extractable according to the definition by Goldwasser et al. and Bitansky et al., showing a separation of the notions.
We show that this tool has many interesting applications to non-malleable cryptography. Particularly, we can derive efficient, continuously non-malleable, leakage-resilient codes against split-state attackers (TCC '14), both in the CRS and the RO model. Additionally, we can obtain succinct non-interactive non-malleable commitments both in the CRS and the RO model, satisfying a stronger definition than the prior ones by Crescenzo et al. (STOC '98), and Pass and Rosen (STOC '05), in the sense that the simulator does not require access to the original message, while the attacker's auxiliary input is allowed to depend on it.

1 Introduction

The notion of extractable collision-resistant hash functions (ECRHs) was originally proposed by [8,9,34] as a tool for building efficient succinct non-interactive arguments of knowledge (SNARKs). Informally, a family of functions, $\mathcal{H}$, is extractable, if for a uniform $h \in \mathcal{H}$, sampling an element $v \in \mathsf{Image}(h)$ without actually evaluating the function on a pre-image s (i.e., $h(s) = v$) is infeasible. This concept is formalized in the following way: for any algorithm $\mathcal{A}$ that produces some $v \in \mathsf{Image}(h)$, there exists an extractor that, possibly depending on the code of $\mathcal{A}$, outputs s such that $h(s) = v$. Typically, such families are interesting only if they possess some sort of hardness property, like one-wayness, or otherwise the problem can be trivial. In [8] the authors propose two constructions for ECRH: the first one is based on a variant of the Knowledge of Exponent

assumption, called t-KEA, and the hardness of the discrete logarithm problem, while the latter uses a lattice based knowledge assumption, called Knowledge of Knapsack.

ℓ-more extractable hash function families. An important observation regarding the setting described above is that ECRHs provide no guarantee against attackers that receive access to precomputed hash values, prior to producing their own. However, there are applications of the primitive that can deviate from the above setting. For instance, the attacker might be given access to a number of valid hash values for which it does not know the pre-images, prior to delivering its own hash value. In this setting, creating a new valid hash value could be achieved by mauling the received ones without knowing the pre-image. To tackle this issue, in [38], a new notion was introduced called ℓ-more extractable hash functions. Briefly speaking, ℓ-more extractable hash function families capture the following idea: if an adversary is given access to $\ell \in \mathbb{N}$ precomputed hash values $v_1, \ldots, v_\ell$, and produces a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$. As it is proven in [38], this notion is not implied by the one by Bitansky et al. [9] and Goldwasser et al. [34], which considers adversaries that get no access to precomputed hash values prior to producing their own value. The separation between the notions is based on the hardness of the discrete logarithm problem. Moreover, by requiring the attacker not only to produce some $\tilde{v} \in \mathsf{Image}(h)$ but also to come up with a valid pre-image for $\tilde{v}$, ℓ-more extractable hash functions are feasible under the same assumptions used in [9,34]. This puts forth a weaker form of extractability (we refer to it as wECRH) in the sense that the extractor is allowed to fail in case a pre-image exists but is not somehow efficiently computable based on the view of the adversary. There is no contradiction in terms here: this extra requirement does not trivialize the notion of ℓ-more wECRH, since the extractor is required to depend only on the adversarial program that produces $\tilde{v}$ and is independent of the program that produces the valid pre-image for $\tilde{v}$. In [38], the authors show that the weaker form of extractability provided by ℓ-more wECRHs is sufficient for constructing very efficient computationally secure non-malleable codes [27] against split-state attackers [27, 43].

The natural question that rises from these previous works is whether ℓ-more wECRHs (and suitable extensions of them) can be used to improve the efficiency and/or the security of other primitives in the context of non-malleable cryptography. This is the main subject of the present paper, which focuses on *continuous malleable codes* (CNMC) and *non-malleable commitments* (NMCOM) *with respect to opening*.

1.1 Non-malleable codes

Non-malleable codes (NMC) were introduced by Dziembowski, Pietrzak and Wichs [27], as a relaxation of error correction and error detection codes, aiming to provide strong privacy, but not necessarily correctness. Informally, non-malleability guarantees that any modified codeword decodes either to the original message or to a completely unrelated one, with overwhelming probability. The

definition of non-malleability is simulation-based, stating that for any tampering function f, there exists a simulator that simulates the tampering effect by only accessing f, i.e., without making any assumptions on the distribution of the encoded message.

The main application of non-malleable codes that motivated the seminal work by Dziembowski et al. [27] is the protection of cryptographic implementations from *active physical attacks* against memory, known as *tampering attacks*. In this setting, the adversary modifies the memory of the cryptographic device, receives the output of the computation, and tries to extract sensitive information related to the private memory. Security against such types of attacks can be achieved by encoding the private memory of the device using non-malleable codes. Besides that, various applications of non-malleable codes have been proposed in subsequent works, such as CCA secure encryption schemes [15], non-malleable commitments [1], and others [14, 28, 30, 31].

The split-state model. Due to the impossibility of non-malleable codes for arbitrary functions classes [27], various models have been proposed and studied over the years, including the extensively studied *split-state model*. The split-state model is a generalization of the bit-wise independent tampering function class [27], and it was originally introduced in the context of non-malleable codes, by Dziembowski et al. [27] and Liu and Lysyanskaya [43], who considered the case of *two split-states*. Briefly speaking, in the split-state model with two states, the codeword is split into two parts, c_0, c_1, and the attacker is allowed to apply on it any function $f = (f_0, f_1)$, that results in a tampered codeword equal to $(f_0(c_0), f_1(c_1))$. The critical point here is that each f_i, for $i \in \{0, 1\}$, tampers with c_i, independently of the value c_{1-i}. This is a plausible model to assume, since there are many scenarios in which sensitive data may be split into two storage devices, that are physically separated, for security reasons.

Continuous non-malleable codes. The notion of *continuous non-malleable codes* (CNMC) was introduced by Faust et al. [30], and considers adversaries that tamper continuously with the same codeword,[4] as opposed to the original notion that considers one-time attackers. In this setting, the adversary chooses the tampering function that he will apply to the codeword, based on the output of the tampering experiment in previous rounds. As it has been pointed by [30], CNMCs constitute a natural extension of the original notion that broadens the applicability of the primitive.

1.2 Non-malleable commitments

The notion of non-malleable commitments was introduced in the seminal work of Dolev, Dwork and Naor [24], as a countermeasure against *man-in-the-middle* adversaries. In the man-in-the-middle setting, we consider two parties that wish

[4] This is the notion of *non-persistent tampering*, which is stronger than *persistent tampering*.

to execute a protocol in the presence of an adversary, that fully controls the communication channel between the parties. The adversary is allowed to modify, block, or introduce messages, and also schedule the order of delivery, while the parties might not be aware of the adversarial presence. Protocols that remain secure against man-in-the-middle adversaries are said to be non-malleable [24].

In [24], Dolev, Dwork and Naor, propose security definitions for the notions of non-malleable commitments and non-malleable zero-knowledge, and assuming the existence of one-way functions, they construct secure protocols that require $\log(k)$ rounds of interaction, where k denotes the security parameter. Informally, a commitment scheme is non-malleable, if any man-in-the-middle adversary that is given a commitment over a message v, is not able to create a valid commitment of a message $\tilde{v}$, which is related to v. This idea has been modeled in two ways, yielding two different notions of security. The first one is *non-malleability with respect to commitment* [24], in which the adversary succeeds in breaking security, if he manages to commit to a related value, even without being able to produce a valid decommitment. This notion is meaningful only for *statistically-binding* commitments. The second one, is *non-malleability with respect to opening* [20], according to which the adversary breaks security if he manages to both commit and decommit to a related value. This notion is meaningful, both for the case of *statistically-binding* and *statistically-hiding* commitment schemes.

1.3 Our contributions

In the present work we formalize and study the notion of *leakage-resilient, ℓ-more, (weakly) extractable hash function families*, according to which, the adversary, in addition to receiving access to $\ell \in \mathbb{N}$ precomputed hash values, it also receives *bounded leakage* over the randomness used to compute those values. Subsequently, we illustrate the usefulness of this primitive by constructing *efficient, computationally secure*, *leakage-resilient continuous non-malleable codes* against *split-state* attackers [30], and finally, we show that assuming ℓ-more wECRH we can construct *succinct, non-interactive non-malleable commitments with respect to opening*, achieving a stronger definition than the ones by [20, 47]. Our contributions are informally summarized below.

First, we revisit the ℓ-more wECRH construction of [38] and we prove security in the presence of leakage over the randomness that is used to compute the hash. In particular, we prove the following theorem (informally stated).

Theorem 1.1 (Informal)**.** DLOG *and* t-KEA *imply leakage-resilient* ℓ*-more* wECRH.

Similar to [38], the main building block of our construction is a non-malleable code for the class of affine functions. Our construction can tolerate λ bits of leakage, for $m \geq \lambda + k + \omega(\log k)$, where k is the security parameter, and m is the min-entropy of the codewords produced by the underlying NMC scheme. Intuitively, an important property of the ℓ-more wECRH of [38], is that the hash values are indistinguishable from uniform, and this is essential for proving

extractability. In this work, we prove that the uniformity property can be still preserved in the presence of leakage, i.e., when the attacker receives partial access over the randomness (and is in full knowledge of the pre-images). For more details we refer the reader to Section 3.1.

Next, we prove that any hash function that can be modeled as a random oracle, is an ℓ-more wECRH. As the adversary receives black-box access to the hash, leakage-resilience in this setting is straightforward. Concretely, we prove the following informal theorem.

Theorem 1.2 (Informal)**.** *Let h be a hash function. If h can be modeled as a random oracle, then it is a leakage-resilient, ℓ-more* wECRH*.*

We note that, even though any function h that can be modeled as a random oracle is a leakage-resilient ℓ-more wECRH, it cannot be extractable according to [8, 9], as the range of the function is dense, thus an attacker can just output a random element in the range of the hash, and then there is no knowledge to extract, as the extractor needs to invert a uniform valid hash value. Given the results of [38], in which the authors prove (cf. Lemma 3.5 in [38]) that the notion of [9] does not imply 1-more (weak) extractability, and also general 1-more extractability (not the weaker form), we receive a separation between the notion of [9] and ℓ-more wECRH.[5]

In our next result, we leverage the power of leakage-resilient ℓ-more wECRH in the continuous setting, and we construct efficient, continuously non-malleable, leakage-resilient codes against split-state attackers [30]. Our result is summarized in the following informal theorem.

Theorem 1.3 (Informal)**.** *Assuming leakage resilient,* 1*-more* wECRH*, there exists an explicit, leakage-resilient continuous non-malleable code, against split-state functions.*

By instantiating the above theorem with the leakage-resilient ℓ-more wECRH of the Informal Theorems 1.2,1.1, we receive efficient split-state CNMCs in the random oracle and the CRS, model, respectively. The first instantiation is secure against arbitrary *polynomial-time adversaries*, while the latter is secure against PPT adversaries that make a *constant number* of tampering queries. Our constructions are way more efficient than the current state-of-the-art in CNMC and we provide a concrete comparison in Section 1.4. We notice that the notion of continuous non-malleable codes (even for 2-time tampering) cannot be achieved generically from a one-time non-malleable code, thus our latter result still improves the state of the arts in efficiency.

Our CRS-based construction tolerates a constant number of tampering queries due to the fact that, in the current proof, the extractor for round i depends on the extractor for round $i-1$ and extraction is non-black box. Thus, assuming a constant number of rounds the final extractor avoids the super-polynomial blow-up. If the extractor's overhead is linear additively, then the construction can tolerate

[5] From Lemma 3.5 of [38], we have that general ℓ-more extractability (not the weaker form), is a notion strictly stronger than the one by [9].

any arbitrary polynomial number of tampering rounds. Furthermore, our CRS-based ℓ-more wECRH (as in [38]), guarantees extractability only if the hash is indistinguishable from uniform, which is a property that cannot be achieved if the adversary is given access to the CRS and split-state leakage over the both parts of the codeword. Therefore, for the wECRH-based construction we consider non-adaptive leakage, i.e., the adversary defines the leakage queries only at the very beginning of the experiment.

In our final result, we construct *succinct, non-interactive non-malleable commitments* [47], *with respect to opening*, from ℓ-more wECRHs. Our result is summarized in the following informal theorem.

Theorem 1.4 (Informal)**.** *Assuming ℓ-more* wECRH*, there exists an explicit succinct non-interactive, non-malleable commitment scheme with respect to opening.*

Our primitive achieves a stronger definition of non-malleability, that allows the adversary's auxiliary input to depend on the message (this is not allowed in [20]), and in contrast to [47], our simulator is weaker, in the sense that it does not need access to the original message in order to simulate the decommitment phase. To our knowledge this is the first construction that achieves this type of security. Our KEA-based instantiation produces commitments of size $2k$, while for the random oracle based construction the commitment size is k. In Section 1.4, we highlight the importance of our primitive, by providing various useful applications.

1.4 Technical overview

Leakage-resilient ℓ-more wECRH. We formulate the notion of leakage-resilient ℓ-more wECRH, that considers attackers which, in addition to receiving access to $\ell \in \mathbb{N}$ precomputed hash values, they also receive bounded leakage over the randomness used to compute them, and we provide two instantiations of the primitive.

Our first instantiation is based on [38], thus we first revisit their construction. Let $\mathbb{G}$ be a group of prime order p and let g be a generator of $\mathbb{G}$. The construction of [38] is a composition between the extractable hash of [9], and a non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ against affine functions. In particular, an element from the hash function family of [9] is described by the pair $(g^{\mathbf{r}}, g^{a\mathbf{r}})$, for uniformly random vector $\mathbf{r}$, and element $a \in \mathbb{Z}_p$, where $g^{\mathbf{r}}$ denotes the value $(g^{r_1}, \ldots, g^{r_t})$, and $\mathbf{r} = (r_1, \ldots, r_t)$, $t \in \mathbb{N}$. The ℓ-more wECRH of [38], on input message $\mathbf{s} = (s_1, \ldots, s_t)$, computes the hash of it, $h(\mathbf{s}; \tau)$, as a pair $v := \left(g^{\langle \mathbf{r}, \mathsf{Enc}(\mathbf{s};\tau)\rangle}, g^{a\langle \mathbf{r}, \mathsf{Enc}(\mathbf{s};\tau)\rangle}\right)$, where $\langle \mathbf{r}, \mathsf{Enc}(\mathbf{s};\tau)\rangle$ is the inner product of $\mathbf{r}$, $\mathsf{Enc}(\mathbf{s};\tau)$, and τ is the randomness used by the encoder. In this work we prove that, if $\mathsf{Enc}(\mathbf{s};\tau)$ has sufficient entropy even given bounded leakage over τ, then v is indistinguishable from uniform due to the universality property (cf. Definition A.5) of the inner product, and we manage to reduce ℓ-more extractability in the presence of leakage to ℓ-more extractability without leakage, using a series of hybrids.

Our second instantiation is in the *random oracle model*. In particular, we prove that any hash function is ℓ-more wECRH, if it can be modeled as a random oracle. Briefly, the main idea behind our result, is as follows. The attacker, denoted as a pair of PPT algorithms $(\mathcal{A}_v, \mathcal{A}_s)$, is required (i) to produce a new valid hash value $\tilde{v}$ (this value is produced by $\mathcal{A}_v$) and (ii) to produce a valid pre-image for $\tilde{v}$ (this value is produced by $\mathcal{A}_s$). The extractor, who is given access only to the queries made by $\mathcal{A}_v$, checks if there is any query (pre-image) that hashes to $\tilde{v}$, and if so, it correctly outputs that pre-image, otherwise it outputs $\perp$. In the latter case, the extractor fails only if $\mathcal{A}_s$ manages to output a valid pre-image for $\tilde{v}$, which happens with negligible probability, as for any $\tilde{s}$ output by $\mathcal{A}_s$, when querying the oracle with $\tilde{s}$, the probability of receiving $\tilde{v}$ as a reply, is negligible. As we stated in Section 1.3, our result yields a separation between the notion of extractability by [9] and ℓ-more wECRHs [38].

Continuous non-malleable codes. Our CNMC scheme is inspired by [30], thus we first revisit their construction. To encode a message m, the encoder of [30], computes $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$ and outputs $((s_0, v_1, \pi_1, \pi_0), (s_1, v_0, \pi_0, \pi_1))$, where $\mathsf{LRS}_{\mathsf{enc}}$ is the encoder of a leakage-resilient storage (LRS) scheme [19, 26, 30], $v_i = h(s_i)$ and h is a member of a collision resistant hash function family, and π_i is a robust non-interactive zero knowledge proof, proving knowledge of the witness (pre-image) s_i of v_i, with label v_{1-i}.[6]

Our goal is to construct an LRS based solution that will enable the simulation of tampering queries through leakage against the LRS, however without requiring the costly computation and storage of four NIZK proofs; replacing the NIZK proofs with a hash-based primitive, is what we aim for. In what follows we highlight why the use of ℓ-more extractable hash is necessary, i.e., the extractable hash function family of [9] is inadequate for our needs.

Towards that direction, assume an encoding scheme that computes the leakage resilient storage of the message m, i.e., it computes $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$ and outputs $((s_0, h(s_1)), (s_1, h(s_0)))$, where $h \in \mathcal{H}$, and $\mathcal{H}$ is an extractable hash function family [9]. There are two issues with this construction. First of all, it does not satisfy the *uniqueness* property [30], which informally states that it should be computationally infeasible for the adversary to find two valid codewords (c_0, c_1) and (c_0, c_1'), with $c_1 \neq c_1'$.[7] Thus the two parts of the codeword should be correlated in a way that modification over only one part, will yield an invalid codeword, with overwhelming probability. Secondly, since $\mathcal{H}$ is extractable according to [9], then it can be malleable (as it is proven in [38]) and security cannot be proved as generic LRS schemes do not provide non-malleability, thus the attacker could create related codewords.

Our construction resolves the above challenges by combining LRS with leakage-resilient 1-more wECRH [38], as follows: assuming $\mathcal{H}$ is a leakage-resilient, 1-more wECRH and $\bar{\mathcal{H}}$ is a collision resistant hash function family, our encoder on input message m, computes $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$, and outputs $((\tau_0, s_0, v_1), (\tau_1, s_1, v_0))$,

[6] The labels are used to bind together the two parts of the memory.

[7] And symmetrically for codewords (c_0, c_1) and (c_0', c_1) for which $c_0 \neq c_0'$.

where $v_i = h(s_i||\bar{v}_{1-i}; \tau_i)$, and $\bar{v}_{1-i} = \bar{h}(\tau_{1-i}||s_{1-i})$. Here, the collision resistance property of $\mathcal{H}$ (also w.r.t. τ) ensures the uniqueness property of our scheme. Also, $\bar{\mathcal{H}}_k$ is essential for proving security of our construction, as it enables simulatability of the hash values v_0, v_1, w.r.t. $\mathcal{H}_k$, via split-state leakage queries to the LRS oracles. I.e., our reduction computes v_1 (resp. v_0) by first leaking $\bar{v}_0$ (resp. $\bar{v}_1$) from the left (resp. right) LRS oracle, and then leaking v_1 (resp. v_0) via a leakage query that depends on $\bar{v}_0$ (resp. $\bar{v}_1$), from the right (resp. left) LRS oracle. If v_1 would be a hash w.r.t. $\mathcal{H}_k$ over both s_0 and s_1, then v_1 would be impossible to simulate via split-state leakage against LRS. Finally, the 1-more extractability property of $\mathcal{H}$ guarantees that, even if the attacker is given access to a valid hash value v_i, it cannot produce a new valid hash value, unless it knows a valid pre-image of it. Proving security of our scheme from any 1-more wECRH posses several challenges, mainly due to the following reasons: (1) 1-more wECRH [38] is a one-time primitive, thus it is not straightforward how it could provide security in the continuous setting, and (2) the two parts of the codeword are correlated, thus using the same wECRH hash function twice, could compromise the extactability property of the primitive. In our proof, we take advantage of the split-state structure of the codeword, and we are able to overcome those issues, while having a more efficient (in terms of running time and leakage) reduction than [30] (see below).

Our construction improves the efficiency of [30], while avoiding the need of a trapdoor CRS (the work of [30] computes and stores four NIZK proofs, while we only require two hashes of size at most $2k$, where k is the security parameter). In addition, the simulator of [30] requires leakage proportional to $O(k \log(q) + \lambda)$, while we only require $O(k + \lambda)$, where k is the security parameter, q is the number of rounds that the attacker tampers with the codeword, and λ is the leakage requested by the tampering adversary. As a result, in our case the size of the code becomes independent of q, which is essential when aiming for efficiency in the continuous setting. Regarding the reduction tightness, our reduction has $1/q$ loss in advantage compared to [30], but [30]'s reduction requires more time. By following a proof strategy similar to [30] we can achieve tightness w.r.t. the advantage but we chose to go with the simpler proof strategy. For more details we refer the reader to Section 3.

Non-malleable commitments. Man-in-the-middle (MIM) attacks, that non-malleable commitments are aiming to prevent, are modeled using a two stage adversary, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) participates in the commitment (resp. opening) stage. In this setting, a sender, Sender, sends a commitment c over a message v, to $\mathcal{A}$, and $\mathcal{A}_1$ sends a commitment $\tilde{c}$ to a receiver, $\mathsf{Receiver}$. In the second stage, Sender sends the opening of c to $\mathcal{A}$ and $\mathcal{A}_2$ sends the opening of $\tilde{c}$ to $\mathsf{Receiver}$. The goal of the adversary is to decommit to a value $\tilde{v}$, which is related to v. In addition, the adversary could also possess *auxiliary information*, which is fixed before initiating the protocol execution. The crucial point here is that such information can potentially depend on v: consider, for instance, a setting in which $\mathcal{A}$ has participated in the past in a protocol that depends on, and leaks some information about, v.

The work of Crescenzo et al. [20] constructed non-malleable commitments, assuming that the adversarial auxiliary information, denoted here by z, is *independent* of the message, v, while in case of dependency, the proof works only if there exists an efficient way to sample a message distribution which is consistent with v, given z. However, this cannot apply for all settings, as for instance, z could be the encryption of v, revealing no information about it.

The work by Pass and Rosen [47], made an important step towards solving the above problem, by considering a slightly different definition of security, in which: (1) z is allowed to depend on v, and (2) during the decommitment phase the simulator requires access to the original message, v, in order to simulate a valid decommitment for $\tilde{v}$. Although, this conforms with the MIM execution described above, in which the adversary learns the original message v (recall that in stage two the sender decommits to $\mathcal{A}$), and makes the simulator stronger, it is very specific to the way $\mathcal{A}$ accesses v, excluding cases in which $\mathcal{A}$ could potentially receive partial, or even no-information, over v. Consider for instance the *commit-and-prove* setting [10], in which the sender first sends a commitment, c, over the message v, and then, instead of revealing v, it sends a zero-knowledge proof, proving relations over v. In this setting, the simulator looses access to v and any attempt to prove security of the commitment scheme according to the definition of [47], would depend to the way $\mathcal{A}$ accesses v, or any sort of encoding of it. In this work, we achieve a stronger definition of non-malleability, that allows the adversary's auxiliary input to depend on the message (this is not allowed in [20]), and in contrast to [47], our simulator does not require access to the original message in order to simulate the decommitment phase, covering all possible settings, from zero/partial, to even full access to v.

We prove that any ℓ-more wECRH that produces outputs indistinguishable from *uniform*, is a *non-malleable commitment with respect to opening*. To commit to a message s, one just samples a uniformly random string r and outputs $c = h(s; r)$, as the commitment.[8] Intuitively, if the hash function produces outputs that are indistinghuishable from uniform, then the commitment scheme achieves the hiding property. In addition, if it is collision resistant, then the scheme is also binding. Finally, if the hash function is a 1-more wECRH, then for any man-in-the-middle attacker that produces a commitment (hash value) $\tilde{c}$, given c, where $\tilde{c} \neq c$, there exists an extractor that extracts $(\hat{s}, \hat{\tau})$, such that $h(\hat{s}; \hat{\tau}) = \tilde{c}$. Since c reveals no information about s, the extracted value, $\hat{s}$, is unrelated to s.

1.5 Related work

In [45], Ostrovsky et al., construct continuous non-malleable codes in the plain model, assuming one-to-one one-way functions. Their scheme uses a one-time, unconditionally secure, non-malleable code for split-state adversaries and non-interactive commitments, and the proposed instantiation produces codewords of length of $O(|s|^7)$.[9] In addition, their construction satisfies a weaker notion of

[8] In the random-oracle model the source of randomness is the oracle.

[9] Our scheme produces codewords of length roughly $O(|s|^2)$.

non-malleability, in which the adversary is only allowed to see the output of the decoded message, while our construction satisfies a stronger notion, in which the adversary receives the tampered codeword, in case it is valid and different than the original one.

In [28], Faonio et al. construct continuous non-malleable codes in the CRS model against split-state adversaries. Their work aims at a stronger flavor of continuous security, in which (1) the messages chosen by the adversary can depend on the CRS[10] and (2) the adversary is allowed to interact with the codeword even after creating an invalid one, by refreshing the codeword in a split-state manner.[11] In order to achieve this stronger property, their construction requires a combination of primitives, such as continuous leakage-resilient, storage friendly, public-key encryption [22], non-interactive commitments and zero-knowledge proofs. Finally, their simulation strategy is efficient, requiring only $O(k + \lambda)$ bits of leakage.

Given the impossibility of split-state CNMC in the information-theoretic setting for two states [30], Aggarwal et al. [25] construct information-theoretic, continuously non-malleable, codes in the split-state model with 8 states. Also, the works [14, 15] construct unconditionally secure continuous non-malleable codes for the bit-wise tampering model.

In [18, 20, 21], the authors construct non-interactive NMCOM, by either assuming that the adversary's auxiliary input does not depend on the message, or that the process of sampling a message that is consistent with the adversarial auxiliary input, is efficient. The work of [47], allows the adversarial auxiliary input to depend on the message, however the simulator requires access to the original message in order to simulate a valid commitment. In [11–13, 35, 41, 42] the authors construct interactive (concurrent) non-malleable commitments using various assumptions, while in [46] Pass proves that non-interactive NMCOM *cannot be proved using a black-box reduction to standard assumptions.*

A related line of work in tamper resilience aims to protect circuit computation against tampering attacks on circuit wires [16, 17, 32, 37] or gates [40], [4, 5] aim at protecting circuits against hardware Trojans, while [7] relies on trusted hardware. In this setting, using non-malleable codes for protecting the circuit's private memory is an option, still in order to achieve security the encoding and decoding procedures should be protected against fault injection attacks using the techniques from [16, 17, 32, 37, 40]. The work of [39] is the first that constructs (one-time) NMCs for the class of partial functions that tamper with almost the entire codeword. Whether NMCs could be useful in secure messaging remains an interesting open question [2, 3].

[10] In the original notion of continuous NMC the messages are fixed before sampling the CRS.

[11] The standard notion of continuous security [30] considers adversaries that loose access to the codeword, after an invalid one is produced, and this is unavoidable, otherwise security is impossible to achieve [33] without any sort of refreshing.

2 Preliminaries

In this section we present the basic definitions and notation that will be used throughout the paper. More standard definitions are presented in Section A.

Notation. Let t, i, j, be non-negative integers. Then, $[t]$ is the set $\{1,\ldots,t\}$. For bit-strings x, y, $x||y$, is the concatenation of x, y, $|x|$ denotes the length of x, for $i \in [|x|]$, $x[i]$ is the i-th bit of x, and for $i \leq j$, $x[i:j] = x[i]||\ldots||x[j]$. For a distribution D over a set $\mathcal{X}$, $x \leftarrow D$, denotes sampling an element $x \in \mathcal{X}$, according to D, $x \leftarrow \mathcal{X}$ denotes sampling a uniform element x from $\mathcal{X}$, $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$. The statistical distance between two random variables X, Y, is denoted by $\Delta(X,Y)$, "$\approx$" and "$\approx_c$", denote statistical and computational indistinguishability, respectively, and $\mathsf{negl}(k)$ denotes an unspecified, negligible function, in k. For a random variable X, $H_\infty(X)$ and $\tilde{H}_\infty(X)$, denote the min-entropy, and average min-entropy, of X, respectively. For any element g and vector $\mathbf{r} = (r_1,\ldots,r_t)$, we define $g^{\mathbf{r}} := (g^{r_1},\ldots,g^{r_t})$.

For an algorithm $\mathcal{A}$, using $y \leftarrow A(x)$ we denote the execution of $\mathcal{A}$ on input x, receiving output y. In case $\mathcal{A}$ is randomized, y is a random variable and $\mathcal{A}(x;r)$ denotes the execution of $\mathcal{A}$ on input x with randomness r. An algorithm $\mathcal{A}$ is probabilistic polynomial-time (PPT) if $\mathcal{A}$ is randomized and for any $x, r \in \{0,1\}^*$, the computation of $\mathcal{A}(x;r)$ terminates in at most $\mathrm{poly}(|x|+|r|)$ steps. For any algorithm that outputs a vector of values we use "$\sim$" to denote coordinates that are not being assigned to any variable, i.e., the calling program simply drops the returned values.

Below, we define encoding schemes, based on the definitions of [27, 43].

Definition 2.1. (Encoding scheme in the Common Reference String (CRS) Model [43]) *A (κ,ν)-coding scheme in the CRS model, $\kappa,\nu \in \mathbb{N}$, is a triple of algorithms $(\mathsf{Init},\mathsf{Enc},\mathsf{Dec})$ such that: Init is a randomized algorithm which receives 1^k, where k denotes the security parameter, and produces a common reference string $\Sigma \in \{0,1\}^{\mathrm{poly}(k)}$, and $(\mathsf{Enc}(1^k,\Sigma,\cdot),\mathsf{Dec}(1^k,\Sigma,\cdot))$ is a (κ,ν)-coding scheme, $\kappa,\nu = \mathrm{poly}(k)$.*

For brevity, 1^k will be omitted from the inputs of Enc and Dec.

The definition of the split-state continuous tampering oracle due to [30] follows.

Definition 2.2 (The split-state tampering oracle $\mathcal{O}_{\mathsf{cnm}}$ [30])**.**
Let $(\mathsf{Init},\mathsf{Enc},\mathsf{Dec})$ be a split-state, (κ,ν)-encoding scheme, in the CRS *model. For any $(c_0,c_1) \in \{0,1\}^{\nu/2} \times \{0,1\}^{\nu/2}$, and any split-state function $f = (f_0,f_1)$, f_0, $f_1 : \{0,1\}^{\nu/2} \to \{0,1\}^{\nu/2}$, define the stateful oracle $\mathcal{O}_{\mathsf{cnm}}(\cdot,\cdot)$ with initial state $st := 0$, as follows,*

$\mathcal{O}_{\mathsf{cnm}}((c_0,c_1),(f_0,f_1))$:
If $st = 1$, *return* $\perp$
$(\tilde{c}_0,\tilde{c}_1) \leftarrow (f_0(c_0),f_1(c_1))$
If $(c_0,c_1) = (\tilde{c}_0,\tilde{c}_1)$ *return* same^*
If $\mathsf{Dec}(\Sigma,(\tilde{c}_0,\tilde{c}_1)) = \perp$, *return* $\perp$ *and set* $st \leftarrow 1$
Else return $(\tilde{c}_0,\tilde{c}_1)$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$.

The λ-bit leakage oracle, returning a total of at most λ bits.

Definition 2.3 (The λ-bit leakage oracle $\mathcal{O}^\lambda(\cdot,\cdot)$)**.** *A leakage oracle* $\mathcal{O}^\lambda(\cdot,\cdot)$*, is a stateful oracle, with initial state* $st := 0$*. For any* $\lambda \in \mathbb{N}$*, string* s*, and function* $g : \{0,1\}^{|s|} \to \{0,1\}^{\lambda'}$*, if* $\lambda' + st \leq \lambda$*,* $\mathcal{O}^\lambda(s,g)$ *outputs* $g(s)$*, and updates it state to* $st \leftarrow st + \lambda'$*, otherwise it outputs* $\perp$*.*

The definition of split-state, leakage-resilient storage, due to [19, 26, 30], follows.

Definition 2.4 (Leakage-resilient storage [30])**.** *Let* $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ *be a coding scheme. For any* PPT *algorithm* $\mathcal{A}$*, message* m*,* $\theta \in \{0,1\}$*, and* $k \in \mathbb{N}$ *we define*

$$\mathsf{Leak}^{\theta}_{\mathcal{A},m}(k) := \left\{ (s_0,s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(1^k, m); out \leftarrow \mathcal{A}^{\mathcal{O}^\lambda(s_0,\cdot),\mathcal{O}^\lambda(s_1,\cdot)}; \textit{Output: } (s_\theta, out) \right\}.$$

Then, $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ *is a* λ*-leakage-resilient storage (*λ-LRS*), if for any* PPT *algorithm* $\mathcal{A}$*, messages* m_0*,* $m_1 \in \{0,1\}^{\mathrm{poly}(k)}$*,* $\theta \in \{0,1\}$*, and all, sufficiently large* $k \in \mathbb{N}$*,* $\left\{\mathsf{Leak}^{\theta}_{\mathcal{A},m_0}(k)\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Leak}^{\theta}_{\mathcal{A},m_1}(k)\right\}_{k\in\mathbb{N}}$.

Here we follow the definition of [30], which is stronger than previous definitions in the sense that the attacker is allowed to see one of the two shares, after the completion of the leakage experiment. As the authors suggest in [30], the scheme of [26] satisfies this stronger notion.

Below we provide the definition of *continuously non-malleable, leakage-resilient codes* due to [30].

Definition 2.5 (Continuously non-malleable, leakage-resilient codes [30])**.** *Let* $\mathsf{ES} = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be a split-state encoding scheme in the* CRS *model, and let* $\lambda, q \in \mathbb{N}$*. Then,* ES *is a* q*-continuously* λ*-leakage resilient (* (q,λ)-CNMLR *) code, if for every, sufficiently large* $k \in \mathbb{N}$*, any pair of messages* m_0*,* $m_1 \in \{0,1\}^{\mathrm{poly}(k)}$*, and any* PPT *algorithm* $\mathcal{A}$*,* $\left\{\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)\right\}_{k\in\mathbb{N}} \approx_c \left\{\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)\right\}_{k\in\mathbb{N}}$*, where,*

$$\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m}(k) := \left\{ \begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k); (c_0,c_1) \leftarrow \mathsf{Enc}(\Sigma, m); \\ out \leftarrow \mathcal{A}^{\mathcal{O}^\lambda(c_0,\cdot),\mathcal{O}^\lambda(c_1,\cdot),\mathcal{O}_{\mathsf{cnm}}((c_0,c_1),\cdot)}(\Sigma); \boldsymbol{Output} : out \end{array} \right\}$$

and $\mathcal{A}$ *makes at most* q *tampering queries against* $\mathcal{O}_{\mathsf{cnm}}$*.*

Next, we define the Uniqueness property by [30].

Definition 2.6 (Uniqueness [30])**.** *Let* $\mathsf{ES} = (\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *be a split-state coding scheme in the* CRS *model. Then,* ES *satisfies the uniqueness property if for any* PPT *algorithm* $\mathcal{A}$ *and all, sufficiently large* $k \in \mathbb{N}$*, we have:*

$$\Pr\left[\begin{array}{c} \Sigma \leftarrow \mathsf{Init}(1^k); (c_0,c_1,c_1') \leftarrow \mathcal{A}(1^k,\Sigma) : \\ \mathsf{Dec}(\Sigma,(c_0,c_1)) \neq \perp \wedge \mathsf{Dec}(\Sigma,(c_0,c_1')) \neq \perp \wedge c_1 \neq c_1' \end{array} \right] \leq \mathsf{negl}(k),$$

and symmetrically for the case in which we fix the right part of the codeword.

3 Continuous NMC from ℓ-more wECRH

In the current section we construct *leakage-resilient continuous non-malleable codes* from any *leakage-resilient, ℓ-more weakly extractable, hash function family*, and then, in Section 3.1 we provide instantiations.

First we define the notion of leakage-resilient, ℓ-more wECRH. Our definition is along the lines of the one given in [38], however in this work we allow the adversary to receive leakage over the randomness that is used to compute the hash.

Definition 3.1 (ℓ-more weakly extractable, leakage-resilient hash function families)**.** *Let $\ell, \lambda \in \mathbb{N}$. An efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$, is ℓ-more weakly extractable against λ bits of leakage, if for any* PPT *algorithm $\mathcal{A}_v$ and any $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, there exist a* PPT *extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all* PPT *algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$, we have*

$$\Pr_{h\leftarrow\mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})=1\right] \leq \mathsf{negl}(k),$$

where,

$\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_{\mathcal{E}})$:
$\tau_i \leftarrow \{0,1\}^{\mathrm{poly}(k)}, v_i = h(s_i;\tau_i), i \in [\ell]$ *(* **hash computation** *)*
$\mathbf{t} = (\tau_1,\ldots,\tau_\ell), \mathbf{v} = (v_1,\ldots,v_\ell)$
$(\tilde{v}, st) \leftarrow \mathcal{A}_v^{\mathcal{O}^\lambda(\mathbf{t},\cdot)}(h,\mathbf{v},z_v)$ *(* **hash tampering** *)*
$(\hat{\tau},\hat{s}) \leftarrow \mathcal{E}_{\mathcal{A}_v}(h,\mathbf{v},z_{\mathcal{E}})$ *(* **pre-image extraction** *)*
$(\tilde{\tau},\tilde{s}) \leftarrow \mathcal{A}_s\,(h,\mathbf{t},\mathbf{s},st)$ *(* **pre-image tampering** *)*
If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$, *return 1*
otherwise, return 0

Following [38], the main steps in the above experiment are the following. First, randomness is sampled and then the hash computation is performed over $\ell \in \mathbb{N}$, pre-images. For deterministic hash function families, randomness sampling is omitted and the hash is computed only over the messages. The challenge for the attacker $\mathcal{A}_v$, is to produce a valid hash value $\tilde{v}$, given ℓ hash values, denoted as $\mathbf{v}$, and auxiliary information z_v. Then, the extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ is executed, given $\mathbf{v}$ and its own auxiliary input $z_{\mathcal{E}}$. The adversary $\mathcal{A}_s$ is required to produce a valid pre-image for $\tilde{v}$, while given all information generated during the execution. The output of the experiment is 1, if $\mathcal{A}_v$ produces a valid hash value $\tilde{v}$, $\mathcal{A}_s$ produces a valid pre-image for $\tilde{v}$, while the extractor fails.

Notice that, in the above definition, the adversary is given leakage access to the randomness, still the extractor does require access to it. Also, similarly to [38], s can be any message, even one that the attacker knows. When considering the random oracle model, extractability is with respect to a single function h (not a family), and furthermore, we assume that all entities receive black-box

access to it. In addition, when the extractor receives black-box access to the adversary, we denote it as $\mathcal{E}$, i.e., we don't parameterize $\mathcal{E}$ with $\mathcal{A}_v$.

In the above definition, we only define extractability, however our constructions require *both extractability and collision resistance*, where the latter should also hold w.r.t. the input randomness, τ. Our instantiations satisfy both. Above we only present extractability to ease presentation, i.e., we don't want to mix the two properties into one. We refer to the primitive that satisfies both properties, by leakage-resilient ℓ-more wECRH.

Our construction of non-malleable codes is inspired by [30], thus we first revisit their construction. To encode a message m, the encoder of [30], computes $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$ and outputs $((s_0, v_1, \pi_1, \pi_0), (s_1, v_0, \pi_0, \pi_1))$, where $\mathsf{LRS}_{\mathsf{enc}}$ is the encoder of a leakage-resilient storage (LRS) scheme (cf. Definition 2.4), $v_i = h(s_i)$ and h is a member of a collision resistant hash function family, and π_i is a robust non-interactive zero knowledge proof, proving knowledge of the witness (pre-image) s_i of v_i, with label v_{1-i}.[12]

Our construction, which is defined below, improves the efficiency of [30] by combining LRS with leakage-resilient 1-more wECRH.

Construction 3.2 (A continuous non-malleable code)**.** *Let $\mathcal{H}_k$, $\bar{\mathcal{H}}_k$, be hash function families and $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ be a leakage-resilient storage scheme. We define an encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ as follows:*

- $\mathsf{Init}(1^k)$*: Sample $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$, and set $\Sigma = (h, \bar{h})$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: Let m be the input to the encoder. The encoder samples $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(1^k, m)$, $\tau_0, \tau_1 \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, computes $\bar{v}_0 \leftarrow \bar{h}(\tau_0 || s_0)$, $\bar{v}_1 \leftarrow \bar{h}(\tau_1 || s_1)$, and outputs $((\tau_0, s_0, v_1), (\tau_1, s_1, v_0))$, where $v_0 \leftarrow h(s_0 || \bar{v}_1; \tau_0)$, $v_1 \leftarrow h(s_1 || \bar{v}_0; \tau_1)$.*
- $\mathsf{Dec}(\Sigma, \cdot)$*: On input $((\tau_0, s_0, v_1), (\tau_1, s_1, v_0))$, for $i \in \{0,1\}$, if $h\left(s_i || \bar{h}(\tau_{1-i} || s_{1-i}); \tau_i\right) = v_i$, output $\mathsf{LRS}_{\mathsf{dec}}(1^k, (s_0, s_1))$, otherwise, output $\perp$.*

In what follows we will assume that $\mathcal{H}$ is a 1-more wECRH and this is essential for proving security. Observe that, if $\mathcal{H}$ is 0-more extractable, then it can be malleable (as it is proven in [38]) and security cannot be proved as generic LRS schemes do not provide non-malleability, thus the attacker could create related codewords. The 1-more extractability property resolves this issue: even if the attacker is given access to a valid hash value v_i, it cannot produce a valid hash value unless it knows a valid pre-image. Finally, $\bar{\mathcal{H}}_k$ is essential for proving security of our construction, as it enables simulatability of the hash values v_0, v_1, w.r.t. $\mathcal{H}_k$, via split-state leakage queries to the LRS oracles (cf. Section 1.4).

In what follows, we briefly discuss the main ideas behind our proof, while highlighting the differences from [30]. The security of our scheme relies on three primitives, namely, on leakage-resilient 1-more wECRH, on the collision resistance property of $\bar{\mathcal{H}}$ and the security of LRS. Our adversary is denoted by $\mathcal{A}'$ and is depicted in Figure 1, while its main subroutine, TComp, is depicted in Figure 2. $\mathcal{A}'$ simulates the tampering experiment against the codeword, while

[12] The labels are used to bind together the two parts of the memory.

given split-state leakage over $(s_0, s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$, where m denotes the message. In step 1, $\mathcal{A}'$ samples the elements required for simulating the codewords inside the leakage oracles, i.e., it samples hash functions $h, \bar{h}$, and randomness τ_i, for h. Then, it makes a guess, j^*, on the index of the round in which the attacker produces an invalid codeword. Such a round is called a "self-destruct" round.[13] Then, in step 2, the adversary leaks the actual hashes over s_0, s_1, and in this way it simulates perfectly the codewords inside the leakage oracles without using a trapdoor CRS, i.e., for $i \in \{0,1\}$, $c_i = (\tau_i, s_i, v_{1-i})$ is perfectly simulated inside $\mathcal{O}^{\lambda}(s_i, \cdot)$. This approach is in contrast to [30], in which the authors use robust NIZKs, and simulate the codeword inside the oracles using simulated proofs, that require a trapdoor CRS.

In step 3, $\mathcal{A}'$ executes $\mathcal{A}$ inside the leakage oracles and verifies if j^* is a correct guess for the self-destruct round. As we prove, this holds, if for all rounds before j^* the executions inside the two oracles are identical, while they differ in round j^*. The challenge here, is to execute $\mathcal{A}$ inside the oracles, as each $\mathcal{O}^{\lambda}(s_i, \cdot)$ gives access to $c_i = (\tau_i, s_i, v_{1-i})$, but provides no information about s_{1-i} (recall that τ_{1-i}, v_i can be simulated), thus it is unclear how to provide the adversary with $\tilde{c}_{1-i}$. We discuss how to resolve this issue, by first considering a non-leakage tampering adversary, $\mathcal{A}$. The main idea behind step 3 of $\mathcal{A}'$, is as follows: $\mathcal{A}'$ executes $\mathcal{A}$ inside $\mathcal{O}^{\lambda}(s_i, \cdot)$, and for each tampering query (f_0, f_1) of $\mathcal{A}$, $\mathcal{A}'$ computes $\tilde{c}_i \leftarrow f_i(c_i)$, and uses the 1-more wECRH property of $\mathcal{H}$ to extract $\tilde{c}_{1-i}$. When considering adversaries that issue leakage queries, $\mathcal{A}'$ replies to those queries by executing repeatedly TComp (cf. Figure 2) against the two oracles (cf. step 3-(a) in Figure 1). In steps 3-(b),(c), $\mathcal{A}'$ verifies if j^* is a self-destruct round, by leaking the hashes of the replies sent to $\mathcal{A}$ inside the oracles. We note that, our strategy is similar to [44], but different than [30] in which the adversary executes binary search to compute the exact value of the index, requiring leakage proportional to $O(k \log(q) + \lambda)$, while we only require $O(k + \lambda)$. In Step 4, $\mathcal{A}'$ learns s_0 and simulates the tampered execution in the same way it does in Step 3. Finally, in contrast to [30], in which extractability is easily implied by the robust NIZKs, proving extractability in the continuous setting using ℓ-more wECRH, which is a one-time primitive, is non-trivial.

For starters, we prove that the above construction satisfies the uniqueness property (cf. Definition 2.6), which is required for achieving non-malleability in the continuous setting.

Lemma 3.3. *Assuming $\mathcal{H}_k$ is a collision resistant hash function family, the split-state code of Construction 3.2 satisfies the uniqueness property.*

Proof. Let $(h, \bar{h}) \in \mathcal{H}_k \times \bar{\mathcal{H}}_k$, and let $\mathcal{H}_k$ be a collision resistant hash function family. Towards contradiction, assume there exists a PPT attacker $\mathcal{A}$ that, given $(h, \bar{h})$, it produces two distinct, valid codewords, (c_0, c_1), (c_0, c_1'), with probability greater than $\epsilon = 1/\text{poly}(k)$, i.e., $\mathcal{A}$ produces $c_0 = (\tau_0, s_0, v_1)$, $c_1 = (\tau_1, s_1, v_0)$,

[13] We can always assume that such a round exists, since for any $\mathcal{A}$ that is not producing an invalid codeword, we can construct another adversary that does so and has the same advantage with $\mathcal{A}$, cf. [30].

Algorithm $\mathcal{A}'$

1. **(Setup)**: Sample $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$, $\hat{h} \leftarrow \hat{\mathcal{H}}_k$, τ_0, $\tau_1 \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $j^* \leftarrow [q]$.
2. **(Hash leaking)**:
   (a) For $i \in \{0,1\}$, define $\mathsf{L}_i(s_i) := \bar{h}(\tau_i || s_i)$ and issue the leakage query $(\mathsf{L}_0, \mathsf{L}_1)$ against $\mathcal{O}^{\lambda}(s_0, \cdot)$, $\mathcal{O}^{\lambda}(s_1, \cdot)$. Let $\bar{v}_0$, $\bar{v}_1$, be the corresponding leaked values.
   (b) For $i \in \{0,1\}$, define $\mathsf{L}'_i(s_i) := h(s_i || \bar{v}_{1-i}; \tau_i)$ and issue the leakage query $(\mathsf{L}'_0, \mathsf{L}'_1)$ against $\mathcal{O}^{\lambda}(s_0, \cdot)$, $\mathcal{O}^{\lambda}(s_1, \cdot)$. Let v_0, v_1, be the leaked values.
3. **(Verifying j^*)**: Let $\mathbf{lk}$ be a $q \times 2$ zero matrix and for $j \in [q]$, $i \in \{0,1\}$ define the leakage function $\mathsf{L}^j_i(s_i, \mathbf{lk})$ that computes $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, j)$ (cf. Figure 2) and outputs its first coordinate, i.e., $\mathbf{lk}$.
   (a) For $j = 1, \ldots, q$: (i) $\mathbf{lk}' \leftarrow \mathsf{L}^j_0(s_0, \mathbf{lk})$, $\mathbf{lk}'' \leftarrow \mathsf{L}^j_1(s_1, \mathbf{lk}')$, (ii) set $\mathbf{lk} \leftarrow \mathbf{lk}''$.
   (b) Let $\bar{\mathsf{L}}_0(\cdot)$ be the leakage function that on input s_0, it computes $(\sim, \mathbf{t}_0) \leftarrow \mathsf{TComp}_q(0, \tau_0, s_0, v_1, \mathbf{lk}, q)$, $lk \leftarrow \hat{h}(\mathbf{t}_0[1 : j^* - 1])$, $lk' \leftarrow \hat{h}(\mathbf{t}_0[j^*])$, and outputs (lk, lk'). Send $\bar{\mathsf{L}}_0(\cdot)$ to $\mathcal{O}^{\lambda}(s_0, \cdot)$.
   (c) Define $\bar{\mathsf{L}}_1(s_1)$ that,
      i. Computes $(\sim, \mathbf{t}_1) \leftarrow \mathsf{TComp}_q(1, \tau_1, s_1, v_0, \mathbf{lk}, q)$.
      ii. If $lk = \hat{h}(\mathbf{t}_1[1 : j^* - 1])$ and $lk' \neq \hat{h}(\mathbf{t}_1[j^*])$, output 1, otherwise output 0.

   $\bar{\mathsf{L}}_1(\cdot)$ is executed against $\mathcal{O}^{\lambda}(s_1, \cdot)$, and let d be the bit output by the leakage query.
   (d) Receive s_0 (the leakage queries have ended).
4. **(Simulating tampering and leakage queries)**: Set $c_0 := (\tau_0, s_0, v_1)$.
   Execute $\mathcal{A}$ and for $j = 1, \ldots, q$:
   Receive (g^j_0, g^j_1) from $\mathcal{A}$, send $\mathbf{lk}[j]$ to it, receive $f^j = (f^j_0, f^j_1)$ from $\mathcal{A}$ and:
   – if $j \geq j^*$ send $\perp$ to $\mathcal{A}$,
   – otherwise, compute $\tilde{c}_0 := (\tilde{\tau}_0, \tilde{s}_0, \tilde{v}_1) = f^j_0(\tau_0, s_0, v_1)$ and
      (a) If $\tilde{c}_0 = c_0$, send same^* to $\mathcal{A}$.
      (b) If $\tilde{c}_0 \neq c_0$:
         If $\tilde{v}_1 = v_1$, send $\perp$ to $\mathcal{A}$, otherwise, $(\hat{\tau}_1, \hat{s}_1, \hat{v}_0) \leftarrow \mathcal{E}_{0,j}(h, v_1, z_{\mathcal{E}_{0,j}})$.
         - If $\bar{h}(\tilde{\tau}_0 || \tilde{s}_0) = \hat{v}_0$ and $h(\hat{s}_1 || \hat{v}_0; \hat{\tau}_1) = \tilde{v}_1$, set $v_0 := h\left(\tilde{s}_0 || \bar{h}(\hat{\tau}_1 || \hat{s}_1); \tilde{\tau}_0\right)$, and $\tilde{c}_1 := (\hat{r}_1, \hat{s}_1, v_0)$. Send $(\tilde{c}_0, \tilde{c}_1)$ to $\mathcal{A}$.
         - Otherwise, send $\perp$ to $\mathcal{A}$.
5. **(Output)**: Let out be the output of $\mathcal{A}$ after the completion of the previous step. $\mathcal{A}'$ outputs (out, d).

Fig. 1. The algorithm $\mathcal{A}'$ playing in $\mathsf{Leak}^0_{\mathcal{A}', m_b}(k)$.

Algorithm TComp_q

Input: $i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, r$.
Set $c_i := (\tau_i, s_i, v_{1-i})$ and let $\mathbf{t}$ be the zero vector with q coordinates.

For $j = 1, \ldots, r$:

– Receive (g^j_0, g^j_1) from $\mathcal{A}$ and
   1. If $j < r$, send $\mathbf{lk}[j]$ to $\mathcal{A}$ and receive (f^j_0, f^j_1) from it.
   2. If $j = r$, set $\mathbf{lk}[j, i+1] = g^j_i(s_i)$ and **break**.
– Compute $\tilde{c}_i := (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}) = f^j_i(\tau_i, s_i, v_{1-i})$.
   1. If $\tilde{c}_i = c_i$, set $\mathbf{t}[j] = \mathsf{same}^*$ and send same^* to $\mathcal{A}$.
   2. If $\tilde{c}_i \neq c_i$, then if $\tilde{v}_{1-i} = v_{1-i}$, set $\mathbf{t}[j] = \perp_i$ and send $\perp$ to $\mathcal{A}$, otherwise, sample $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,j}(h, v_{1-i}, z_{\mathcal{E}_{i,j}})$.
      - If $\bar{h}(\tilde{\tau}_i || \tilde{s}_i) = \hat{v}_i$ and $h(\hat{s}_{1-i} || \hat{v}_i; \hat{\tau}_{1-i}) = \tilde{v}_{1-i}$, set $\mathbf{t}[j] = (\tilde{c}_0, \tilde{c}_1)$, where $\tilde{c}_i$ is defined above, $\tilde{c}_{1-i} = (\hat{r}_{1-i}, \hat{s}_{1-i}, v_i)$ and $v_i = h\left(\tilde{s}_i || \bar{h}(\hat{\tau}_{1-i} || \hat{s}_{1-i}); \tilde{\tau}_i\right)$. Send $\mathbf{t}[j]$ to $\mathcal{A}$.
      - Otherwise, set $\mathbf{t}[j] = \perp_i$ and send $\perp$ to $\mathcal{A}$.

Output $(\mathbf{lk}, \mathbf{t})$.

Fig. 2. The algorithm TComp executed by $\mathcal{A}'$.

$c'_1 = (\tau'_1, s'_1, v'_0)$, where $c_1 \neq c'_1$,[14] with probability ϵ. We construct an adversary $\mathcal{A}'$, that given $h \leftarrow \mathcal{H}_k$, it breaks the collision resistance property of $\mathcal{H}_k$ with non-negligible probability: $\mathcal{A}'$ samples $\bar{h} \leftarrow \bar{\mathcal{H}}_k$ and

$$(\tau_0, s_0, v_1), (\tau_1, s_1, v_0), (\tau'_1, s'_1, v'_0) \leftarrow \mathcal{A}(h, \bar{h}),$$

and outputs (τ_1, s_1), (τ'_1, s'_1). Let $\bar{v}_0 \leftarrow \bar{h}(\tau_0 || s_0)$. By the validity of the codewords and Construction 3.2, we have that $h(s_1 || \bar{v}_0; \tau_1) = v_1 = h(s'_1 || \bar{v}_0; \tau'_1)$. Conditioned on $c_1 \neq c'_1$ we have that $(\tau_1, s_1) \neq (\tau'_1, s'_1)$: if $(\tau_1, s_1) = (\tau'_1, s'_1)$, we also have that $v_0 = v'_0$ and the codewords are equal. Thus, conditioned on $c_1 \neq c'_1$ we have that $(\tau_1, s_1) \neq (\tau'_1, s'_1)$ and $\mathcal{A}'$ breaks the collision resistance property of $\mathcal{H}_k$ with non-negligible probability, ϵ (here we assume collision resistance of $\mathcal{H}_k$ also w.r.t. to τ, which is achieved by our instantiations). ■

Below we prove non-malleability of Construction 3.2 in the continuous setting with respect to any 1-more wECRH.

Theorem 3.4. *Let $k, \lambda, \lambda' \in \mathbb{N}$, and let b be polynomial in k. Assuming $\mathcal{H}_k$ is a leakage-resilient 1-more* wECRH *function family against λ bits of leakage, that outputs $b(k)$ bits, $\bar{\mathcal{H}}_k$, $\hat{\mathcal{H}}_k$, are collision resistant hash function families that output k bits, and $(\mathsf{LRS}_{\mathsf{enc}}, \mathsf{LRS}_{\mathsf{dec}})$ is a λ'-*LRS*, for $\lambda' \geq 2\lambda + 2b(k) + 4k + 1$. Then, the encoding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ of Construction 3.2 is a (q, λ)-*CNMLR *code (cf. Definition 2.5), where $q = \mathrm{poly}(k)$ if extraction w.r.t. to $\mathcal{H}_k$ requires linear time, otherwise q is constant.*

Proof. Towards contradiction, assume there exists a pair of messages m_0, m_1, PPT adversary $\mathcal{A}$ and PPT distinguisher D, such that for infinitely many $k \in \mathbb{N}$,

$$\left| \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)\right) = 1\right] \right| > \epsilon,$$

for $\epsilon = 1/\mathrm{poly}(k)$. Here, $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_i}(k)$ is the experiment of Definition 2.5 with respect to $\mathcal{A}$, m_i. We will use m_0, m_1, $\mathcal{A}$, D, to construct m'_0, m'_1, $\mathcal{A}'$, D', for which

$$\left| \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m'_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m'_1}(k)\right) = 1\right] \right| > \epsilon',$$

for $\epsilon' = 1/\mathrm{poly}(k)$ and infinitely many k, where $\mathsf{Leak}^0_{\mathcal{A}',m'_i}(k)$ is the experiment of Definition 2.4, with respect to $\mathcal{A}'$, m'_i. The idea is that $\mathcal{A}'$ will play in $\mathsf{Leak}^0_{\mathcal{A}',m'_b}(k)$, for $b \in \{0,1\}$, interacting with $\mathcal{O}^\lambda(s_0, \cdot)$, $\mathcal{O}^\lambda(s_1, \cdot)$, where (s_0, s_1) follows $\mathsf{LRS}_{\mathsf{enc}}(m_b)$, and it will simulate $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, through its access to the aforementioned oracles. $\mathcal{A}'$ executes $\mathcal{A}$ using always the same randomness. Assume $\mathcal{A}$ issues q leakage/tampering queries and that there is always a round in which self-destruct occurs, i.e., a round in which the output of the decoder in the real experiment will be $\perp$. This round is denoted by j_{d}. We define $m'_0 := m_0$, $m'_1 := m_1$ and $\mathcal{A}'$ is defined in Figure 1. The definition of the distinguisher D'

[14] The case where $c_0 \neq c'_0$ is symmetric.

against $\mathsf{Leak}^0_{\mathcal{A}',m_b}(k)$ follows.

Algorithm D': D' receives the output of $\mathcal{A}'$, (out, d), and if $d = 1$ it outputs $b' \leftarrow \mathsf{D}(out)$, otherwise it outputs $b' \leftarrow \{0,1\}$.

Claim 3.5. *For any message m and all sufficiently large k, $\mathcal{A}'$ simulates perfectly $\mathsf{Enc}(\Sigma, m)$ inside the leakage oracles by leaking $2k + 2b(k)$ bits during the execution of $\mathsf{Leak}^0_{\mathcal{A}',s_b}(k)$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$.*

Proof. By Construction 3.2 and the definition of $\mathcal{A}'$, it is not hard to see that c_0 (resp. c_1) is perfectly simulated inside the oracle $\mathcal{O}^\lambda(s_0, \cdot)$ (resp. $\mathcal{O}^\lambda(s_1, \cdot)$). $\mathcal{A}'$ initially samples $h \leftarrow \mathcal{H}_k$, $\bar{h} \leftarrow \bar{\mathcal{H}}_k$ (the CRS), and $\tau_0, \tau_1 \leftarrow \{0,1\}^{\mathrm{poly}(k)}$. Then, by querying the leakage oracles with $(\mathsf{L}_0, \mathsf{L}_1)$, where $\mathsf{L}_i(s_i) := \bar{h}(\tau_i || s_i)$, it receives $\bar{v}_0, \bar{v}_1$ and finally, by leaking $(\mathsf{L}'_0, \mathsf{L}'_1)$, where $\mathsf{L}'_i(s_i) := h(s_i || \bar{v}_{1-i}; \tau_i)$, it receives v_0, v_1. All the remaining leakage queries against $\mathcal{O}^\lambda(s_0, \cdot)$ (resp. $\mathcal{O}^\lambda(s_1, \cdot)$) depend on τ_0, v_1 (resp. τ_1, v_0), and the execution inside the oracles takes place over (c_0, c_1). ■

Claim 3.6. *Let $\mathbf{lk}_{\mathsf{Real}}$ and $\mathbf{t}_{\mathsf{Real}}$ be the vectors of the replies to the first $j^* - 1$ leakage and tampering, respectively, queries made by $\mathcal{A}$ in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. Then, conditioned on $j_{\mathsf{d}} = j^*$, for any message m and all sufficiently large k, $\mathbf{lk}[1 : j^* - 1] \approx_c \mathbf{lk}_{\mathsf{Real}}$, $\mathbf{t}_{\mathsf{Real}} \approx_c \mathbf{t}_i[1 : j^* - 1]$, for $i \in \{0,1\}$, over the randomness of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, $\mathsf{Leak}^0_{\mathcal{A}',m_b}(k)$.*

Proof. Using strong induction, we prove that conditioned on $j_{\mathsf{d}} = j^*$, $\mathbf{lk}[1 : j^* - 1] \approx_c \mathbf{lk}_{\mathsf{Real}}$, $\mathbf{t}_{\mathsf{Real}} \approx_c \mathbf{t}_i[1 : j^* - 1]$, for $i \in \{0,1\}$, assuming that $j_{\mathsf{d}} > 1$, otherwise the claim holds trivially.

Base, $j = 1$*:* By executing $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, 1)$, for $i \in \{0,1\}$ (Step 3-a), it is clear that $\mathcal{A}'$ computes the replies to (g^1_0, g^1_1) as in the real execution. Thus, $\mathbf{lk}[1] = \mathbf{lk}_{\mathsf{Real}}[1]$. Regarding, the replies to the tampering queries produced by the execution of $\mathsf{TComp}_q(i, \tau_i, s_i, v_{1-i}, \mathbf{lk}, 1)$ inside $\mathcal{O}^\lambda(s_i, \cdot)$, we consider the following cases.

- $\exists i : \tilde{c}_i = c_i$: Assume that for some $i \in \{0,1\}$, $\tilde{c}_i = c_i$. Then by assumption we have that j is not a round in which self-destruct occurs, and in order for the tampered codeword to be valid, it must be the case that $\tilde{c}_{1-i} = c_{1-i}$ with overwhelming probability, otherwise we can use (f^1_0, f^1_1) to break the uniqueness property of the encoding scheme (cf. Lemma 3.3), by simulating the first round of execution. Thus, for $i \in \{0,1\}$, $\mathbf{t}_i[j] = \mathsf{same}^*$, which matches the reply $\mathbf{t}_{\mathsf{Real}}[1]$ of the tampering oracle in the real execution, since by Claim 3.5, c_i is perfectly simulated inside $\mathcal{O}^\lambda(s_i, \cdot)$.
- $\forall i : \tilde{c}_i \neq c_i \wedge \exists j : \tilde{v}_j = v_j$: Let E_i be the event in which $\tilde{c}_i \neq c_i \wedge \tilde{v}_{1-i} = v_{1-i}$, and E the event of not having a round with self-destruct. We prove that $\Pr[E_i \wedge E] \leq \mathsf{negl}(k)$, for $i \in \{0,1\}$. Towards contradiction, assume that for some i in $\{0,1\}$, $\Pr[E_i \wedge E] > \epsilon'_i$, for $\epsilon'_i = 1/\mathrm{poly}(k)$. Then, we have a valid codeword for which $(\tau_i, s_i) \neq (\tilde{\tau}_i, \tilde{s}_i)$ and $\tilde{v}_{1-i} = h(\tilde{s}_{1-i} || \bar{h}(\tilde{\tau}_i || \tilde{s}_i); \tilde{\tau}_{1-i}) =$

$h(s_{1-i}||\bar{h}(\tau_i, s_i); \tau_{1-i}) = v_{1-i}$. If $\bar{h}(\tilde{\tau}_i||\tilde{s}_i) = \bar{h}(\tau_i, s_i)$, then we can use f^j to break the collision resistance property of $\bar{h}$, otherwise f^j can be used to break the collision resistance property of h, with non-negligible probability ϵ_i, by simulating $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. Thus, such an event never happens with non-negligible probability during the execution of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.

– $\underline{\forall i : \tilde{v}_i \neq v_i}$: In order to prove consistency between $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$, we need to define two extractors, $\mathcal{E}_{i,1}$, for $i \in \{0,1\}$, and the corresponding auxiliary inputs, and prove that the extracted values are consistent with $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. For this reason we relate (see below) the execution of TComp inside the oracles, with the ℓ-more experiment of Definition 3.1.

In the execution inside $\mathcal{O}^\lambda(s_i, \cdot)$, the adversary is given direct access to $c_i = (\tau_i, s_i, v_{1-i})$ and leakage access over c_{1-i}, and tampers with the hash v_{1-i} and some auxiliary values (τ_i, s_i). We relate this adversary to $\mathcal{A}_v$ of Definition 3.1 by defining a program $\mathcal{A}_{v,i}$ with auxiliary input $z_{v,i}$, as follows:[15]

1. **Program** $\mathcal{A}^{\mathcal{O}^\lambda(\tau_{1-i},\cdot)}_{v,i}(h, v_{1-i}, z_{v,i})$:
   – Sample $(g^1_0, g^1_1) \leftarrow \mathcal{A}(h)$ and parse $z_{v,i}$ as (τ_i, s_i, s_{1-i}).
   – Query $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ with $g_{s_{1-i}}(x) := \bar{h}(x||s_{1-i})$ and let $\bar{v}_{1-i}$ be the answer.
   – Set $v_i = h(s_i||\bar{v}_{1-i}; \tau_i)$.
   – Define $g_{v_i,s_{1-i}}(\tau_{1-i}) := g^1_{1-i}(\tau_{1-i}, s_{1-i}, v_i)$, send $g_{v_i,s_{1-i}}$ to $\mathcal{O}^\lambda(\tau_{1-i}, \cdot)$ and let w_{1-i} be the answer.
   – Compute $w_i \leftarrow g^1_i(\tau_i, s_i, v_{1-i})$, send (w_0, w_1) to $\mathcal{A}$ and receive (f^1_0, f^1_1).
   – **Output**: $([f^1_i(\tau_i, s_i, v_{1-i})]_3, st)$, where $st := (f^j_i(\tau_i, s_i, v_{1-i}), z_{v,i}, v_{1-i})$.
2. (**Auxiliary input for** $\mathcal{A}_{v,i}$): set $z_{v,i} = (\tau_i, s_i, s_{1-i})$.
3. (**Existence of the extractor,** $\mathcal{E}_{i,1}$**, and auxiliary input,** $z_{\mathcal{E}_{i,1}}$): Given $\mathcal{A}_{v,i}$ and $z_{v,i}$, by the 1-more extractability property of $\mathcal{H}_k$ under leakage, there exists an extractor $\mathcal{E}_{i,1}$ for $\mathcal{A}_{v,i}$, with auxiliary input, $z_{\mathcal{E}_{i,1}}$, that computes $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,1}(h, v_{1-i}, z_{\mathcal{E}_{i,1}})$.

Clearly, $\mathcal{A}_{v,i}$ is an admissible attacker against $\mathcal{H}_k$, that produces a tampered hash value $\tilde{v}_{1-i}$, as $\mathcal{A}$ does in the first round of the execution inside $\mathcal{O}^\lambda(s_i, \cdot)$. Now we relate the execution inside $\mathcal{O}^\lambda(s_{1-i}, \cdot)$ with a program $\mathcal{A}_{s,1-i}$ that outputs a tampered pre-image, and we define the message s of the experiment of Definition 3.1.

1. **Program** $\mathcal{A}_{s,1-i}(h, \tau, s, st)$:
   – Parse s as $s_{1-i}||\bar{v}_i$, and set $\tau_{1-i} := \tau$.
   – Compute $v_i := h(s_i||\bar{h}(\tau_{i-1}||s_{i-1}); \tau_i)$,[16] $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}, \tilde{v}_i) = f^1_{1-i}(\tau_{1-i}, s_{1-i}, v_i)$.
   – **Output**: $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i, \tilde{s}_i))$.
2. (**Define message** s): set $s := s_{1-i}||\bar{h}(\tau_i||s_i)$.

[15] $\mathcal{A}_{v,i}$ is implicitly parameterized by $\bar{h}$.
[16] $\mathcal{A}_s$ knows (τ_i, s_i), $(\tilde{\tau}_i, \tilde{s}_i)$, since they are stored in st.

By the 1-more extractability property of $\mathcal{H}_k$ under leakage, we have

$$\Pr_{h \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{s,h}_{\mathcal{A}_{v,i},\mathcal{A}_{s,1-i},\mathcal{E}_{i,1}}(1,\lambda,z_{v,i},z_{\mathcal{E}_{i,1}})=1\right] \leq \mathsf{negl}(k).$$

Let B be the event in which the extractor fails to produce a valid pre-image. Then, if B happens, and since we are not in a self-destruct round (an event denoted as E), we have that $\mathcal{A}_{v,i}$ produces a valid hash and $\mathcal{A}_{s,1-i}$, produces a valid pre-image, still the extractor fails, i.e., we have $\mathsf{Exp}^{s,h}_{\mathcal{A}_{v,i},\mathcal{A}_{s,1-i},\mathcal{E}_{i,1}}(1,\lambda,z_{v,i},z_{\mathcal{E}_{i,1}}) = 1$. Thus, by the above relation we receive $\Pr[B \wedge E] \leq \mathsf{negl}(k)$, and the extractor outputs a valid pre-image for $\tilde{v}_{1-i}$, i.e., $h(\hat{s}_{1-i}||\hat{v}_i;\hat{\tau}_{1-i}) = \tilde{v}_{1-i}$. Since the attacker creates a valid codeword, we also have $h(\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i);\tilde{\tau}_{1-i}) = \tilde{v}_{1-i}$. We prove that the extracted values are consistent with the ones produced by the attacker in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, i.e., we prove that $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i) = (\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$, with overwhelming probability.

Let B' be the event in which $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i) \neq (\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$. Then, assuming there exist m, $\mathcal{A}$, $\mathcal{A}'$, for which $\Pr[\neg B \wedge E \wedge B'] > \epsilon''$, for $\epsilon'' = 1/\mathrm{poly}(k)$, we build an attacker $\mathcal{A}''$ that breaks the collision resistance property of $\mathcal{H}_k$, with non-negligible probability: given m, $\mathcal{A}''$ simulates $\mathsf{Leak}^0_{\mathcal{A}',s}(k)$ while having full access to $(s_0,s_1) \leftarrow \mathsf{LRS}_{\mathsf{enc}}(m)$, and outputs $(\hat{\tau}_{1-i},\hat{s}_{1-i}||\hat{v}_i)$, $(\tilde{\tau}_{1-i},\tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i||\tilde{s}_i))$. We conclude that $\mathbf{t}_i[1] \approx_c \mathbf{t}_{\mathsf{Real}}[1]$. The case of $\mathbf{t}_{1-i}[1]$ is symmetric.

The proof for the inductive step is provided in Section A.2. ■

Claim 3.7. *Conditioned on $j_{\mathsf{d}} = j^*$, for any message m and all sufficiently large k, $\mathbf{t}_0[j_{\mathsf{d}}] \neq \mathbf{t}_1[j_{\mathsf{d}}]$, with overwhelming probability over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

Proof. By Claim 3.6, we have that the output of the decoder in $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, is simulated perfectly for the first $j_{\mathsf{d}}-1$ rounds, by the execution of TComp inside the leakage oracles. Thus, the tampering query made by the attacker in round j_{d} inside the oracles is consistent with the j_{d}-th tampering query made by the attacker in the execution of $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$. By assumption, j_{d} is self-destruct round, thus there exists $i \in \{0,1\}$, for which $h(\tilde{s}_i||\bar{h}(\tilde{\tau}_{1-i}||\tilde{s}_{1-i});\tilde{\tau}_i) \neq \tilde{v}_i$. We denote such an event by E_i and analyze the execution of TComp under the event $E_0 \vee E_1$. We consider the following cases.

- $\underline{\forall i\ \tilde{v}_i = v_i}$: Since $E_0 \vee E_1$ has occurred we know that for some $i \in \{0,1\}$, $\underline{(\tilde{\tau}_i,\tilde{s}_i) \neq (\tau_i,s_i)}$, and by the definition of TComp_q, $\mathbf{t}_i[j_{\mathsf{d}}] = \perp_i \neq \mathbf{t}_{1-i}[j_{\mathsf{d}}]$, independently of the value in $\mathbf{t}_{1-i}[j_{\mathsf{d}}]$.
- $\underline{\exists i : \tilde{v}_i \neq v_i \wedge \tilde{v}_{1-i} = v_{1-i}}$: If $(\tau_i,s_i) = (\tilde{\tau}_i,\tilde{s}_i)$, we have that $\mathbf{t}_i[j_{\mathsf{d}}] = \mathsf{same}^* \neq \underline{\mathbf{t}_{1-i}[j_{\mathsf{d}}]}$, since $\underline{\tilde{c}_{1-i} \neq c_{1-i}}$, else if $(\tau_i,s_i) \neq (\tilde{\tau}_i,\tilde{s}_i)$, we have that $\mathbf{t}_i[j_{\mathsf{d}}] = \perp_i \neq \mathbf{t}_{1-i}[j_{\mathsf{d}}]$, independently of the value in $\mathbf{t}_{1-i}[j_{\mathsf{d}}]$.
- $\underline{\forall i\ \tilde{v}_i \neq v_i}$: We prove the needed for the non-trivial case in which for all $\underline{i \in \{0,1\}}$, $\mathbf{t}_i[j_{\mathsf{d}}] \neq \perp_i$. Assuming the extractors executed inside the oracles output valid pre-images, we have $\mathbf{t}_0[j_{\mathsf{d}}] = ((\tilde{\tau}_0,\tilde{s}_0,\tilde{v}_1),(\hat{\tau}_1,\hat{s}_1,v_0))$, $\mathbf{t}_1[j_{\mathsf{d}}] = ((\hat{\tau}_0,\hat{s}_0,v_1),(\tilde{\tau}_1,\tilde{s}_1,\tilde{v}_0))$. Conditioned on E_i, we have $h(\tilde{s}_i||\bar{h}(\tilde{\tau}_{1-i},\tilde{s}_{1-i});\tilde{\tau}_i) \neq$

$\tilde{v}_i = h(\hat{s}_i||\bar{h}(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}); \hat{\tau}_i)$, which implies that $(\hat{\tau}_i, \hat{s}_i) \neq (\tilde{\tau}_i, \tilde{s}_i)$. Thus $\mathbf{t}_0[j_\mathsf{d}] \neq \mathbf{t}_1[j_\mathsf{d}]$, under $E_0 \vee E_1$. ■

Claim 3.8. *For any message m and all sufficiently large k, $j^* = j_\mathsf{d}$ iff $d = 1$, with overwhelming probability over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

Proof. Recall that d is the output bit of the leakage query defined in 1. By Claims 3.6, 3.7, we have that for all $j \in [j_\mathsf{d} - 1]$, $\mathbf{t}_0[j] = \mathbf{t}_1[j]$ and $\mathbf{t}_0[j_\mathsf{d}] \neq \mathbf{t}_1[j_\mathsf{d}]$, and clearly, assuming $j^* = j_\mathsf{d}$, we have that $\hat{h}(\mathbf{t}_0[1 : j^* - 1]) = \hat{h}(\mathbf{t}_1[1 : j^* - 1])$ and $\hat{h}(\mathbf{t}_0[j^*]) \neq \hat{h}(\mathbf{t}_1[j^*])$, with overwhelming probability, otherwise we can break the collision resistance property of $\hat{h}$ by simulating $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$. Thus $d = 1$. Symmetrically, assuming $d = 1$ we know that $\mathbf{t}_0[j^*] \neq \mathbf{t}_1[j^*]$, and using the collision resistance property of $\hat{h}$, with overwhelming probability, $\mathbf{t}_0[1 : j^* - 1] = \mathbf{t}_1[1 : j^* - 1]$. Thus, $d = 1$ iff $\mathcal{A}'$ makes a correct guess on j_d. ■

Claim 3.9. *Conditioned on $j^* = j_\mathsf{d}$, out $\approx_c \mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_b}(k)$, over the randomness of $\mathsf{Leak}^0_{\mathcal{A}',m}(k)$.*

The main arguments that prove the current claim, have already been proved in Claim 3.6, as the execution in of $\mathcal{A}'$ in Step 4, is similar to the TComp.

Claim 3.10. *For any pair of messages m_0, m_1,* PPT *adversary $\mathcal{A}$, all sufficiently large k, and all* PPT *distinguishers D, assuming that*

$$\left|\Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)\right) = 1\right]\right| > \epsilon,$$

for $\epsilon = 1/\mathrm{poly}(k)$, we have

$$\left|\Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_0}(k)\right) = 1\right] - \Pr\left[\mathsf{D}'\left(\mathsf{Leak}^0_{\mathcal{A}',m_1}(k)\right) = 1\right]\right| > \epsilon/q - \mathsf{negl}(k),$$

where D', $\mathcal{A}'$, have already been defined above with respect to D, $\mathcal{A}$, respectively.

The proof of the above claim can found in Appendix A.4.

Total leakage. Step 2 of Figure 1 leaks four hash values requiring $2k + 2b(k)$ bits of leakage. Step 3-a simulates the leakage queries of the adversary against the encoding scheme, i.e., λ bits for each part of the codeword, thus upper bounded by 2λ. Step 3-b leaks two hash values, requiring $2k$ bits of leakage, plus one bit. Therefore, the total amount of leakage is upper bounded by $2\lambda + 4k + 2b(k)$.

The above conclude the proof of the theorem. ■

3.1 Instantiating ℓ-more wECRH

In what follows, we prove that any hash function is an ℓ-more wECRH (cf. Definition 3.1) when it can be modeled as a random oracle. Since in the random oracle model, the randomness comes from the oracle, we do not hash the message using independent randomness τ. Leakage resilience is straightforward in this setting as the adversary has no access to the randomness.

Theorem 3.11. *Let $k, \ell \in \mathbb{N}$ and let h be a function, $h : \{0,1\}^* \to \{0,1\}^k$. Assuming h can be modeled as a random oracle, then h is an leakage resilient ℓ-more* wECRH *(cf. Definition 3.1).*

The intuition behind the proof of the above the theorem has been provided in the Introduction while the full proof is presented in Section A.3. By plugging the ℓ-more wECRH of Theorem 3.11 to Theorem 3.4 we obtain a leakage-resilient continuous NMC. In particular, we receive the following corollary.

Corollary 3.12. *Assuming $\mathcal{H}_k$ can be modeled as a random oracle and $\bar{\mathcal{H}}_k$ is collision resistant, then construction 3.2 is a q-*CNMLR *code against λ bits of leakage, for any $q = \text{poly}(k)$, assuming $\lambda' \geq 2\lambda + 6k + 1$, where $k, \lambda' \in \mathbb{N}$ are as in Theorem 3.4.*

The proof of Theorem 3.4 is with respect to any ℓ-more wECRH. When considering the random oracle model, and similarly to [29], the random oracle can be simulated inside the leakage oracles using a pseudo-random function.

As a second instantiation, in what follows, we prove that the ℓ-more wECRH of [38] satisfies a weaker form of leakage-resilience that is non-adaptive with respect to the CRS. This notion if formally defined below.

Definition 3.13 (ℓ-more weakly extractable, leakage-resilient hash function families)**.** *Let $\ell, \lambda \in \mathbb{N}$. An efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$, is ℓ-more weakly extractable against λ bits of leakage, if for any $g : \{0,1\}^* \to \{0,1\}^\lambda$, any* PPT *algorithm $\mathcal{A}_v$ and any $z_v \in \{0,1\}^{\text{poly}(k)}$, there exist a* PPT *extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $z_\mathcal{E} \in \{0,1\}^{\text{poly}(k)}$, such that for all* PPT *algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$,*

$$\Pr_{h\leftarrow\mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_\mathcal{E}) = 1\right] \leq \mathsf{negl}(k),$$

where,

$\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell,\lambda,z_v,z_\mathcal{E})$:
$\tau_i \leftarrow \{0,1\}^{\text{poly}(k)}, v_i = h(s_i;\tau_i), i \in [\ell]$ *(* **hash computation** *)*
$\mathbf{t} = (\tau_1,\ldots,\tau_\ell), \mathbf{v} = (v_1,\ldots,v_\ell)$
$(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, \mathbf{v}, g(\mathbf{t}), z_v)$ *(* **hash tampering** *)*
$(\hat{\tau}, \hat{s}) \leftarrow \mathcal{E}_{\mathcal{A}_v}(h, \mathbf{v}, z_\mathcal{E})$ *(* **pre-image extraction** *)*
$(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}_s\,(h, \mathbf{t}, \mathbf{s}, st)$ *(* **pre-image tampering** *)*

If $h(\tilde{s};\tilde{\tau}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s};\hat{\tau}) \neq \tilde{v}$, return 1
otherwise, return 0

Before presenting our result, we first revisit the ℓ-more wECRH construction of [38].

Construction 3.14 (ℓ-more weakly extractable hash [38])**.** *Let $k \in \mathbb{N}$, $\mathcal{G}$ be a group-generation algorithm and let $(\mathsf{Enc}, \mathsf{Dec})$ be a (kt, kt')-coding scheme, $t, t' = O(\text{poly}(k))$. An instance of a $(kt, 2k)$-compressing hash function family $\mathcal{H} = (\mathcal{G}, h)$ is defined as follows:*

1. $\mathsf{Gen}(1^k)$*: sample* $(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k)$, $(a, \mathbf{r}) \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p^{t'}$, *where* $p = |\mathbb{G}|$, *and output* $h = (\mathbb{G}, g^{\mathbf{r}}, g^{a\mathbf{r}})$.[17]
2. ***Hashing****: on input* s, *sample* $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, $\mathbf{c} \leftarrow \mathsf{Enc}(s;\tau)$ *and output* $h(s;\tau) = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{\langle a\mathbf{r}, \mathbf{c} \rangle})$.

For coding schemes $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *in the CRS model,* $\mathcal{G}(1^k)$ *outputs* (h, Σ), *where* $\Sigma \leftarrow \mathsf{Init}(1^k)$.

In [38] the authors prove that if $(\mathsf{Enc}, \mathsf{Dec})$ is a non-malleable code against affine functions, and $\mathsf{Enc}(s)$ has sufficient entropy, then construction 3.14, is an ℓ-more wECRH under the t-KEA assumption and DLOG. In the following theorem, we reduce the ℓ-more extractability in the presence of leakage to standard, i.e., ℓ-more extractability without leakage, under the same assumptions.

Theorem 3.15. *For* $k \in \mathbb{N}$, *let* $\mathcal{H}$ *be the* ℓ*-more* wECRH *family of Construction 3.14 instantiated with an* NMC, $(\mathsf{Enc}, \mathsf{Dec})$, *such that for any message* s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$. *Then,* $\mathcal{H}$ *is an* ℓ*-more* wECRH *against* λ *bits of non-adaptive leakage (cf. Definition 3.13).*

Proof. We prove the needed for the 1-more case (the ℓ-more case is identical) using a series of hybrid experiments that we describe below and they are depicted in Figure 3.

- For any $g : \{0,1\}^* \to \{0,1\}^\lambda$, PPT $\mathcal{A}_v$ with auxiliary input z_v, any $\mathcal{A}_s$, any message s and $h \in \mathcal{H}$, Exp_0 is the ℓ-more experiment under leakage, $\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}_{\mathcal{A}_v}}(\ell, \lambda, z_v, z_{\mathcal{E}})$, of Definition 3.13. In order to fully define the experiment we need to define $\mathcal{E}$, $z_{\mathcal{E}}$: for any g, $\mathcal{A}_v$, z_v, we define the non-leakage attacker $\mathcal{A}'_v$ such that $\mathcal{A}'_v(h, v, z_v)$ samples independent randomness $\tau' \leftarrow \{0,1\}^{\mathrm{poly}(k)}$ and executes $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau'), z_v)$. By the 1-more extractability of $\mathcal{H}$ (without leakage) there exists extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}'_v}$ with hard-coded auxiliary information $z_{\mathcal{E}}$, for $\mathcal{A}'_v$, and we define $\mathcal{E} := \mathcal{E}^{\mathcal{H}}_{\mathcal{A}'_v}$. We prove that for any $\mathcal{A}_v$, z_v, $\mathcal{A}_s$ and any message s, $\Pr[\mathsf{Exp}_0 = 1] \leq \mathsf{negl}(k)$, otherwise we break 1-more extractability, without leakage, of $\mathcal{H}$, with non-negligible probability.
- In Exp_1 we modify Exp_0 in two ways. First the leakage is computed over the independent randomness τ', instead of τ, which is used to compute v. As we prove, if the adversary is not leaking more than λ bits in total, v is statistically close to a uniform element in the range of the hash even if the attacker receives leakage over τ, and this modification does not induce any statistical difference between the two experiments. Thus, $\mathcal{A}_v$ cannot distinguish between the two experiments. However, $\mathcal{A}_s$ might do so since for some leakage query g we might have $g(\tau) \neq g(\tau')$. Hence, for any $\mathcal{A}_s$, we define an $\mathcal{A}'_s$, that given τ, s, computes the output of $\mathcal{A}_s$ exactly as it happens in Exp_0, i.e., $\mathcal{A}'_s(h, \tau, s, st)$ computes (i) $v \leftarrow h(s;\tau)$, (ii) $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau), z_v)$, (iii) outputs $(\tilde{\tau}, \tilde{s}) \leftarrow \mathcal{A}_s(h, \tau, s, st)$, and clearly, the output of $\mathcal{A}_s$ in Exp_0, matches the output of $\mathcal{A}'_s$ in Exp_1.

[17] For a vector $\mathbf{r} = (r_1, \ldots, r_t)$, $g^{\mathbf{r}} := (g^{r_1}, \ldots, g^{r_t})$.

– In Exp_2, for any $\mathcal{A}_v$, we substitute $\mathcal{A}_v$ with $\mathcal{A}'_v$ such that $\mathcal{A}'_v(h, v, z_v)$ samples $\tau' \leftarrow \{0,1\}^{\mathrm{poly}(k)}$ and outputs $(\tilde{v}, st) \leftarrow \mathcal{A}_v(h, v, g(\tau'), z_v)$. Exp_2 is the original ℓ-more experiment (without leakage) and it is not hard to see that $\mathsf{Exp}_1 \approx \mathsf{Exp}_2$.

In the following claims we prove statistical indistinguishability between the hybrids. The statistical distance between Exp_0 and Exp_1 is bounded by the distance of the input/output variables to $\mathcal{A}_v$, $\mathcal{A}_s$, $\mathcal{A}'_s$ and $\mathcal{E}$.

Claim 3.16. *For any any leakage function g, any s, $\mathcal{A}_v$, $\mathcal{A}_s$, z_v, $(h(s;\tau), g(\tau)) \approx (h(s;\tau), g(\tau'))$, over the randomness of Exp_0, Exp_1.*

Proof. By assumption we have that for any s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$. Moreover, each leakage query g can leak at most λ bits of τ (as all queries cannot leak more than λ bits, in total). Since the randomness of the encoder is independent of z_v, h, we have that for $Z = (z_v, h, g(h))$, $\mathbf{H}_\infty(\mathsf{Enc}(s) \mid Z) \geq k+\omega(\log k)$. Thus, $\tilde{\mathbf{H}}_\infty(\mathsf{Enc}(s) \mid Z) \geq k+\omega(\log k)$. By the Left-Over Hash Lemma (Lemma A.3) and the universality of the inner product function (Lemma A.5), the distribution $\langle \mathbf{r}, \mathsf{Enc}(s;\tau)\rangle$ is statistically close to uniform over $\mathbb{Z}_p$ and we have

$$(h(s;\tau), g(\tau)) = \left(\left(g^{\langle \mathbf{r}, \mathsf{Enc}(s;\tau)\rangle}, g^{a\langle \mathbf{r}, \mathsf{Enc}(s;\tau)\rangle}\right), g(\tau)\right) \approx ((g^r, g^{ar}), g(\tau)),$$

for uniform r, τ. Since τ, τ', r, are uniform and independent we have $((g^r, g^{ar}), g(\tau))$ $\approx ((g^r, g^{ar}), g(\tau'))$, and thus $(h(s;\tau), g(\tau)) \approx (h(s;\tau), g(\tau'))$. This concludes the proof of the claim and the input and output distributions for $\mathcal{A}_v$ and $\mathcal{E}$ in both experiments are the same. ■

By the above claim and the fact that the input and output distributions of $\mathcal{A}_s$ and $\mathcal{A}'_s$ are the same (by the definition of $\mathcal{A}'_s$), we have that $\mathsf{Exp}_0 \approx \mathsf{Exp}_1$.

Finally, in Exp_2 $\mathcal{A}'_v$ is just sampling τ' internally and then executes $\mathcal{A}_v$. Again the output distributions of $\mathcal{A}_v$ and $\mathcal{A}'_v$ are the same, thus $\mathsf{Exp}_1 \approx \mathsf{Exp}_2$.

From the above we have that $\mathsf{Exp}_0 \approx \mathsf{Exp}_2$ and $|\Pr[\mathsf{Exp}_0 = 1] - \Pr[\mathsf{Exp}_2 = 1]| \leq \mathsf{negl}(k)$. Thus, assuming $\Pr[\mathsf{Exp}_0 = 1] > \epsilon$, for $\epsilon = 1/\mathrm{poly}(k)$, we receive that $\Pr[\mathsf{Exp}_2 = 1] > \epsilon' = \epsilon - \mathsf{negl}(k)$, and 1-more extractability of $\mathcal{H}$ breaks with non-negligible probability (recall that Exp_2 is the 1-more experiment without leakage). Thus, $\Pr[\mathsf{Exp}_0 = 1] \leq \mathsf{negl}(k)$, and 1-more extractability under leakage for $\mathcal{H}$ follows. The collision resistance property of the construction follows by [38] (cf. Lemma 5.3). ■

By plugging the non-malleable encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ against affine functions (cf. Construction 5.8 of [38]) with $|p| = k+\lambda$, as the underlying encoding scheme to Construction 3.14, we have that for any message s, $\mathbf{H}_\infty(\mathsf{Enc}(s)) \geq \lambda + k + \omega(\log k)$, and Construction 3.14 is an ℓ-more wECRH against λ bits of leakage, which by Theorem 3.4 yields a continuous NMC against split-state adversaries for a constant number of rounds and tolerating λ bits of non-adaptive leakage. In particular, we receive the following corollary.

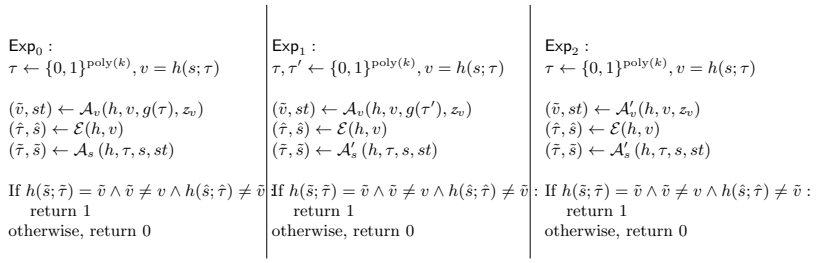

Fig. 3. The hybrid experiments for the proof of Theorem 3.15.

Corollary 3.17. *For $k, t, \lambda' \in \mathbb{N}$ as in Theorem 3.4, assuming* DLOG*, t-*KEA*, and collision resistant hash function families, Construction 3.2 is a q-*CNMC *against λ bits of non-adaptive leakage, for $\lambda' \geq 2\lambda + 8k + 1$ and any constant round* PPT *adversary.*

4 Non-malleable commitments

In the present section, we construct *succinct,*[18] *non-interactive non-malleable commitments with respect to opening*, from ℓ-more wECRHs. Our result is summarized in the following informal theorem.

Theorem 4.1 (Informal)**.** *Assuming ℓ-more* wECRH*, there exists an explicit succinct non-interactive, non-malleable commitment scheme with respect to opening.*

Our primitive achieves a stronger definition of non-malleability, that allows the adversary's auxiliary input to depend on the message (this is not allowed in [20]), and in contrast to [47], our simulator is weaker, in the sense that it does not need access to the original message in order to simulate the decommitment phase. Our KEA based instantiation produces commitments of size $2k$, while for the random oracle based construction the commitment size is k.

We start by presenting the notion of non-interactive commitments in the CRS model.

Definition 4.2 (Non-interactive commitment in the CRS model)**.**
Let $(\mathsf{Init}, \mathsf{Commit}, \mathsf{Open})$ *be a commitment scheme and let* $\Sigma \leftarrow \mathsf{Init}(1^k)$*. Then, it satisfies the following properties:*

- **(Computational binding)***: It is computationally infeasible to find $s \neq s'$, and τ, τ', such that* $\mathsf{Commit}(\Sigma, s; \tau) = \mathsf{Commit}(\Sigma, s'; \tau')$.
- **(Statistical hiding)***: For any two messages s, s',* $\mathsf{Commit}(\Sigma, s) \approx \mathsf{Commit}(\Sigma, s')$.

[18] The length of the commitment is independent of the message length.

For brevity, the CRS is omitted when calling Commit and Open.

We define the notion of non-malleable, non-interactive commitments, in the standalone setting, following the definition of Pass and Rosen [47], with some simplifications for the non-interactive settings. First, present the *man-in-the-middle execution* with respect to the *real game* and the *ideal experiment*, as follows.

Man-in-the-middle experiment (real game). Here we consider a two-stage man-in-the-middle adversary, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ participates in the commitment stage and $\mathcal{A}_2$ participates in the opening stage. More specifically, given a binary relation $\mathcal{R}(\cdot,\cdot)$, a man-in-the-middle adversary, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, a sender, Sender, a receiver, $\mathsf{Receiver}$, a message s, and auxiliary input z, we define the real experiment $\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z)$ as follows: (1) Sender sends a commitment $c \leftarrow \mathsf{Commit}(s;\tau)$ to $\mathcal{A}$; (2) $\mathcal{A}_1(c, z)$ sends a commitment $\tilde{c}$ to $\mathsf{Receiver}$; (3) Sender sends the opening of c to $\mathcal{A}$; (4) $\mathcal{A}_2(z, s, \tau)$ sends the opening of $\tilde{c}$ to $\mathsf{Receiver}$. The experiment outputs 1 if and only if $\mathcal{A}$ produces a valid $\tilde{s}$ for the commitment $\tilde{c}$, and $\mathcal{R}(s, \tilde{s}) = 1$. The message s is chosen prior to the experiment and $\mathcal{A}$ is allowed to receive auxiliary input, z, that might depend on s.

The Ideal experiment. Given a binary relation $\mathcal{R}(\cdot,\cdot)$, an ideal adversary, $\mathcal{S}$, a sender, Sender and receiver $\mathsf{Receiver}$, message s, and auxiliary input z, the Ideal experiment, $\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)$, is defined as follows: $\mathcal{S}$ only interacts with $\mathsf{Receiver}$ by (1) sending a commitment to $\tilde{c}$ it and (2) sending the corresponding decommitment. The experiment outputs 1 if and only if $\mathcal{S}$ produces a valid decommitment, $\tilde{s}$, for $\tilde{c}$ and $\mathcal{R}(s, \tilde{s}) = 1$. The message s is chosen prior to the experiment and $\mathcal{S}$ receives the auxiliary input z, as $\mathcal{A}$ does in the real experiment.

Having defined the real and ideal executions, we define the notion of non-malleable (non-interactive) commitments.

Definition 4.3 (Non-malleable non-interactive commitment)**.** *A non-interactive commitment scheme is said to be non-malleable (with respect to opening) if for every* PPT *man-in-the-middle adversary $\mathcal{A}$, there exists a* PPT *adversary $\mathcal{S}$ and a negligible function $\mathsf{negl}(\cdot)$, such that for every non-reflexive polynomial-time computable relation $\mathcal{R} \subseteq \{0,1\}^n \times \{0,1\}^n$, every $s \in \{0,1\}^n$ and every $z \in \{0,1\}^*$, we have that*

$$\Pr\left[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1\right] < \Pr\left[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)\right] + \mathsf{negl}(n).$$

It should be noted that, the definition presented above is stronger from the ones presented in [20, 47], since (1) we allow the attacker's auxiliary input to depend on the message s, and (2), our simulator does not need the original message in order to simulate the decommitment phase.

In the CRS model, both the real and ideal experiments will generate the CRS by running Init and all parties will receive access to it.[19]

[19] There is a weaker model called trapdoor CRS, in which the simulator $\mathcal{S}$ generates an indistinguishable CRS with a trapdoor. The construction that is proposed in this work uses the honestly generated CRS, i.e., it does not require trapdoor information.

Below we define our construction.

Construction 4.4 (Non-malleable non-interactive commitment)**.** *Let $\mathcal{H}_k$ be a hash function family. We define a non-interactive commitment scheme* $(\mathsf{Init}, \mathsf{Commit}, \mathsf{Open})$*, as follows:*

- $\mathsf{Init}(1^k)$*: Sample $h \leftarrow \mathcal{H}_k$ and set $\Sigma := h$.*
- $\mathsf{Commit}(\Sigma, \cdot)$*: on input string $s \in \{0,1\}^{\mathrm{poly}(k)}$, the algorithm selects random string $\tau \in \{0,1\}^{\mathrm{poly}(k)}$ and outputs $h(s;\tau)$.*
- $\mathsf{Open}(\Sigma, \cdot)$*: on input a commitment c, the algorithm outputs s, τ. The receiver accepts if $h(s;\tau) = c$.*

In the following statement we formalize the properties that $\mathcal{H}_k$ should meet, in order for the above scheme to be non-malleable.

Theorem 4.5. *Let $\mathcal{H}_k$ collision resistant hash function family, such that for $h \leftarrow \mathcal{H}_k$ and any message s, $h(s)$ is statistically close to uniform. Then the commitment scheme of Construction 4.4 is statistically hiding and computationally binding. Furthermore, if the hash function family $\mathcal{H}_k$, is a* 1*-more* wECRH*, then the commitment scheme is non-malleable with respect to opening (cf. Definition 4.3).*

Proof. The first part of the proof, i.e., proving the statistical hiding property of the scheme, is straightforward, as by assumption, the distribution of $h(s;\tau)$ is statistically close to uniform. In addition, since the hash function family is collision resistant, no PPT adversary can find two distinct valid openings for the same commitment, i.e., computing efficiently $(s,\tau) \neq (s',\tau')$, for which $h(s;\tau) = h(s';\tau')$, happens only with negligible probability, assuming the collision resistance property of $\mathcal{H}_k$. The binding property of the scheme follows.

Next we are going to prove non-malleability. Given any man-in-the-middle adversary $\mathcal{A}$, we define an ideal adversary $\mathcal{S}$ as follows: $\mathcal{S}$, (1) samples $\tau \leftarrow \{0,1\}^{\mathrm{poly}(k)}$ and sends $c := h(0;\tau)$ to $\mathcal{A}$, (2) then executes $\tilde{c} \leftarrow \mathcal{A}_1(h, c, z)$ and forwards $\tilde{c}$ to the external receiver $\mathsf{Receiver}$, and (3) for the opening, if $\tilde{c} = c$, then $\mathcal{S}$ just sends $(0,\tau)$. Otherwise, $\mathcal{S}$ runs the extractor $\mathcal{E}$ (defined below) to extract $(\tilde{s},\tilde{\tau})$ and forwards the extracted value to $\mathsf{Receiver}$.

Below, we relate the above execution, with the execution of the ℓ-more ECRH experiment, of Definition 3.1, without considering any sort of leakage.
We first define $\mathcal{A}_v$:

> $\mathcal{A}_v$ on input the description of a hash function h, a hash value $c = h(s;\tau)$, and the auxiliary input z, outputs $\tilde{c} \leftarrow \mathcal{A}_1(h, c, z)$. By the properties of the 1-more wECRH (cf. Definition 3.13), there exist auxiliary input z' and ex tractor $\mathcal{E}$ that on input z' and c outputs $(\hat{s},\hat{\tau})$.

We will prove that for any $\mathcal{A}$, $\Pr[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1] < \Pr[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)] + \mathsf{negl}(k)$, using a hybrid argument presented below.

H_1: this hybrid is the same as the Real execution, for the first two steps. In the third step, the sender Sender does not provide an opening for the commitment. Instead, if the man-in-the middle adversary $\mathcal{A}$ forwards the commitment sent by Sender, i.e., $\tilde{c} = c$ the experiment just outputs $\mathcal{R}(s, s)$. Otherwise, the experiment runs the extractor $\mathcal{E}$ (defined above) to extract a value $(\hat{s}, \hat{\tau})$, and sends $(\hat{s}, \hat{\tau})$ to $\mathsf{Receiver}$. The experiment finally outputs $\mathcal{R}(s, \hat{s})$.

H_2: this hybrid is the same as H_1 except that Sender commits to zero in the first step.

Claim 4.6. *Assuming that $\mathcal{H}_k$ is a 1-more* wECRH*, then for any non-reflexible polynomially computable relation $\mathcal{R}$ and any s, z we have*

$$\Pr[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1] < \Pr[H_1 = 1] + \mathsf{negl}(k).$$

The proof of the above claim can be found in Appendix A.5.

Claim 4.7. *Assuming the hiding property of the commitment scheme, we have* $|\Pr[H_1 = 1] - \Pr[H_2 = 1]| < \mathsf{negl}(k)$.

Proof. We observe that both H_1 and H_2 do not depend on the opening of the commitment. Therefore, if one can distinguish H_1 from H_2, then it can distinguish between $\mathsf{Commit}(0)$ from $\mathsf{Commit}(s)$. ■

It is clear that $\Pr[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z) = 1] = \Pr[H_2 = 1]$, as $\mathcal{S}$ simulates perfectly the experiment H_2. By the above claims, we receive

$$\begin{aligned}\Pr\left[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1\right] &< \Pr[H_1 = 1] + \mathsf{negl}(k) \leq \Pr[H_2 = 1] + \mathsf{negl}(k) \\ &= \Pr\left[\mathsf{Ideal}_{\mathcal{S}}(\mathcal{R}, s, z)\right] + \mathsf{negl}(k).\end{aligned}$$

This concludes the proof of the theorem. ■

Instantiations. Construction 4.4 can be instantiated using the 1-more (ℓ-more) wECRH of Construction 3.14, since it produces hashes that are indistinguishable from uniform (see Claim 3.16 in the proof of Theorem 3.15). It can also be instantiated in the random oracle model, as the uniformity property of the random oracle based ℓ-more wECRH (cf. Theorem 3.11) is straightforward.

References

1. S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 538–557. Springer, Heidelberg, Aug. 2015.

2. J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis. Security analysis and improvements for the ietf mls standard for group messaging. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 248–277, Cham, 2020. Springer International Publishing.
3. J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis. Modular design of secure group messaging protocols and the security of mls. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1463–1483, New York, NY, USA, 2021. Association for Computing Machinery.
4. G. Ateniese, A. Kiayias, B. Magri, Y. Tselekounis, and D. Venturi. Secure outsourcing of circuit manufacturing. Cryptology ePrint Archive, Paper 2016/527, 2016. `https://eprint.iacr.org/2016/527`.
5. G. Ateniese, A. Kiayias, B. Magri, Y. Tselekounis, and D. Venturi. Secure outsourcing of cryptographic circuits manufacturing. In J. Baek, W. Susilo, and J. Kim, editors, *Provable Security*, pages 75–93, Cham, 2018. Springer International Publishing.
6. B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover hash lemma, revisited. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 1–20. Springer, Heidelberg, Aug. 2011.
7. P. Bhatotia, M. Kohlweiss, L. Martinico, and Y. Tselekounis. Steel: Composable hardware-based stateful and randomised functional encryption. In J. A. Garay, editor, *Public-Key Cryptography – PKC 2021*, pages 709–736, Cham, 2021. Springer International Publishing.
8. N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580, 2014. `http://eprint.iacr.org/2014/580`.
9. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In S. Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, Jan. 2012.
10. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.
11. Z. Cao, I. Visconti, and Z. Zhang. Constant-round concurrent non-malleable statistically binding commitments and decommitments. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 193–208. Springer, Heidelberg, May 2010.
12. M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 270–299. Springer, Heidelberg, Aug. 2016.
13. M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Four-round concurrent non-malleable commitments from one-way functions. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 127–157. Springer, Heidelberg, Aug. 2017.
14. S. Coretti, Y. Dodis, B. Tackmann, and D. Venturi. Non-malleable encryption: Simpler, shorter, stronger. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 306–335. Springer, Heidelberg, Jan. 2016.
15. S. Coretti, U. Maurer, B. Tackmann, and D. Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 532–560. Springer, Heidelberg, Mar. 2015.

16. D. Dachman-Soled and Y. T. Kalai. Securing circuits against constant-rate tampering. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 533–551, 2012.
17. D. Dachman-Soled and Y. T. Kalai. Securing circuits and protocols against 1/poly(k) tampering rate. In Y. Lindell, editor, *Theory of Cryptography: 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*. 2014.
18. I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *35th ACM STOC*, pages 426–437. ACM Press, June 2003.
19. F. Davì, S. Dziembowski, and D. Venturi. Leakage-resilient storage. In J. A. Garay and R. D. Prisco, editors, *SCN 10*, volume 6280 of *LNCS*, pages 121–137. Springer, Heidelberg, Sept. 2010.
20. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *30th ACM STOC*, pages 141–150. ACM Press, May 1998.
21. G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 40–59. Springer, Heidelberg, May 2001.
22. Y. Dodis, A. B. Lewko, B. Waters, and D. Wichs. Storing secrets on continually leaky devices. In R. Ostrovsky, editor, *52nd FOCS*, pages 688–697. IEEE Computer Society Press, Oct. 2011.
23. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004.
24. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
25. N. Döttling, J. B. Nielsen, and M. Obremski. Information theoretic continuously non-malleable codes in the constant split-state model. Cryptology ePrint Archive, Report 2017/357, 2017. `http://eprint.iacr.org/2017/357`.
26. S. Dziembowski and S. Faust. Leakage-resilient cryptography from the inner-product extractor. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 702–721. Springer, Heidelberg, Dec. 2011.
27. S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. In A. C.-C. Yao, editor, *ICS 2010*, pages 434–452. Tsinghua University Press, Jan. 2010.
28. A. Faonio, J. B. Nielsen, M. Simkin, and D. Venturi. Continuously non-malleable codes with split-state refresh. In B. Preneel and F. Vercauteren, editors, *Applied Cryptography and Network Security*, pages 121–139, Cham, 2018. Springer International Publishing.
29. S. Faust, K. Hostáková, P. Mukherjee, and D. Venturi. Non-malleable codes for space-bounded tampering. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 95–126. Springer, Heidelberg, Aug. 2017.
30. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. Continuous non-malleable codes. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 465–488. Springer, Heidelberg, Feb. 2014.
31. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. A tamper and leakage resilient von neumann architecture. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 579–603. Springer, Heidelberg, Mar. / Apr. 2015.
32. S. Faust, K. Pietrzak, and D. Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. In *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 391–402, 2011.

33. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 258–277. Springer, Heidelberg, Feb. 2004.
34. S. Goldwasser, H. Lin, and A. Rubinstein. Delegation of computation without rejection problem from designated verifier CS-Proofs. Cryptology ePrint Archive, Report 2011/456, 2011. `http://eprint.iacr.org/2011/456`.
35. V. Goyal, O. Pandey, and S. Richelson. Textbook non-malleable commitments. In D. Wichs and Y. Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016.
36. J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
37. Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. Private circuits ii: Keeping secrets in tamperable circuits. In *Advances in Cryptology - EUROCRYPT 2006, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings*. Springer Berlin Heidelberg, 2006.
38. A. Kiayias, F.-H. Liu, and Y. Tselekounis. Practical non-malleable codes from l-more extractable hash functions. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 1317–1328. ACM Press, Oct. 2016.
39. A. Kiayias, F.-H. Liu, and Y. Tselekounis. Non-malleable codes for partial functions with manipulation detection. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 577–607, Cham, 2018. Springer International Publishing.
40. A. Kiayias and Y. Tselekounis. Tamper resilient circuits: The adversary at the gates. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pages 161–180, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
41. H. Lin and R. Pass. Constant-round non-malleable commitments from any one-way function. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 705–714. ACM Press, June 2011.
42. H. Lin, R. Pass, and P. Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In *58th FOCS*, pages 576–587. IEEE Computer Society Press, 2017.
43. F.-H. Liu and A. Lysyanskaya. Tamper and leakage resilience in the split-state model. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532. Springer, Heidelberg, Aug. 2012.
44. A. S. Mortazavia, M. Salmasizadeh, and A. Daneshgar. FMNV continuous non-malleable encoding scheme is more efficient than believed. Cryptology ePrint Archive, Report 2016/604, 2016. `http://eprint.iacr.org/2016/604`.
45. R. Ostrovsky, G. Persiano, D. Venturi, and I. Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 608–639, Cham, 2018. Springer International Publishing.
46. R. Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In A. Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 334–354. Springer, Heidelberg, Mar. 2013.
47. R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 533–542. ACM Press, May 2005.

A Basic notions

A.1 Randomness extractors and universal hash function families

Using extractors we can extract randomness from sources that produce weakly-random values, assuming those values have sufficient min-entropy. Here, we follow the definition given by [23], that uses average conditional min-entropy $\tilde{H}_\infty(\cdot)$.

Definition A.1 (Randomness Extractor [23]). *A polynomially time computable function* $\mathsf{Ext} : \mathcal{M} \times \{0,1\}^n \to \{0,1\}^k$ *is an average case, strong,* (m, ϵ)*-extractor, if for all random variables* S*,* Z*, where* S *is a variable over* $\mathcal{M}$ *and* $\tilde{H}_\infty(S|Z) \geq m$*, it holds that*

$$\Delta(\mathsf{Ext}(S; R),\ U_k \mid (R, Z)) \leq \epsilon,$$

where R *denotes the random coins of* Ext*. The value* $L = m - k$ *is called the entropy loss of* Ext*, and* n *is the seed length of* Ext*.*

Universal hash functions are good randomness extractors, and they are defined as follows:

Definition A.2 (ρ-Universal Hashing). *A family* $\mathcal{H}$ *of deterministic functions* $h : \mathcal{M} \to \{0,1\}^k$ *is called a* ρ*-universal hash family, if for any* $s_1 \neq s_2 \in \mathcal{M}$*,* $\Pr_{h \leftarrow \mathcal{H}}[h(s_1) = h(s_2)] \leq \rho$*. If* $\rho = 1/2^k$*,* $\mathcal{H}$ *is called universal.*

Now we state the leftover-hash lemma [36], following the definition given in [6].

Lemma A.3 (Leftover-Hash Lemma [6, 36]). *Assume that the family* $\mathcal{H}$ *of functions* $h : \mathcal{M} \to \{0,1\}^k$ *is a* $\frac{1+\gamma}{2^k}$*-universal hash family. Then, the extractor* $\mathsf{Ext}(s; h) = h(s)$*, where* h *is sampled according to* $\mathcal{H}$*, is an average case, strong* (m, ϵ)*-extractor, where* $\epsilon = \frac{1}{2} \cdot \sqrt{\gamma + \frac{1}{2^L}}$ *and* $L = m - k$ *is the entropy loss.*

Below, we define the inner product hash function family and in Lemma A.5 we prove that it is universal.

Definition A.4 (The inner product hash function family). *Let* $\mathbb{F}_p$ *be a finite field of prime order* p*, where* p *is a* k*-bit integer. For any* $t \in \mathbb{N}$*, the inner-product function family* $\mathcal{H}_{\mathsf{ip}} = (\mathsf{Gen}, h)$*, for messages over* $\mathbb{F}_p^t$ *is defined as follows:*

- $\mathsf{Gen}(1^k)$*: sample* $(r_1, \ldots, r_t) \leftarrow \mathbb{F}_p^t$ *and set* $z = (r_1, \ldots, r_t)$*.*
- ***Hash computation****: on input message* $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{F}_p^t$*, compute* $h_z(\mathbf{s}) = \sum_{i=1}^t s_i \cdot r_i$*, where the summation refers to the addition operation, and* $\cdot$ *is the multiplication operation, over* $\mathbb{F}_p$*.*

Lemma A.5. *The function family* $\mathcal{H}_{\mathsf{ip}}$ *of Definition A.4 is universal.*

Proof. For any k in $\mathbb{N}$, let $\mathbb{F}_p$ be any field of order p, where p is a k-bit integer, and let $\mathbf{s} = (s_1, \ldots, s_t)$, $\bar{\mathbf{s}} = (\bar{s}_1, \ldots, \bar{s}_t)$ be two distinct messages, i.e., $\mathbf{s}$ and $\bar{\mathbf{s}}$ differ in at least one coordinate. Without loss of generality, we assume that $s_1 \neq \bar{s}_1$. Then,

$$\Pr_{h_z \leftarrow \mathcal{H}_{\mathsf{ip}}}[h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})] = \Pr\left[\sum_{i=1}^t r_i \cdot (s_i - \bar{s}_i) = 0\right] = \Pr\left[r_1 = \frac{-\sum_{i=2}^t r_i \cdot (s_i - \bar{s}_i)}{(s_1 - \bar{s}_1)^{-1}}\right]$$

Hence, for any choice of $r_2, \ldots, r_t$, there is a unique r_1 for which $h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})$. Since r_1 is random over $\mathbb{F}_p$, we have that $\Pr[h_z(\mathbf{s}) = h_z(\bar{\mathbf{s}})] \leq 1/p \leq 1/2^k$. ■

A.2 Inductive step for the proof of Claim 3.6

Inductive step:
Inductive hypothesis: *For* $i \in \{0,1\}$, $\mathbf{lk}[1:n] \approx_c \mathbf{lk}_{\mathsf{Real}}[1:n]$, $\mathbf{t}_i[1:n] \approx_c \mathbf{t}_{\mathsf{Real}}[1:n]$. *In particular, for* $j \in [n]$, $i \in \{0,1\}$, *there exist* $\mathcal{E}_{i,j}$ *with auxiliary inputs* $z_{\mathcal{E}_{i,j}}$, *that output valid pre-images in rounds* $1,\ldots,n$.

Proving that $\mathbf{lk}[n+1] \approx_c \mathbf{lk}_{\mathsf{Real}}[n+1]$ is straightforward, as the tampering and leakage queries are simulated correctly in the previous rounds: by leaking the first coordinate of $\mathsf{TComp}_q(i,\tau_i,s_i,v_{1-i},\mathbf{lk},n+1)$, for $i \in \{0,1\}$, it is clear that $\mathcal{A}'$ computes the replies to (g_0^{n+1}, g_1^{n+1}) as in the real execution, as previous tampering and leakage queries are simulated correctly. Thus, $\mathbf{lk}[n+1] \approx_c \mathbf{lk}_{\mathsf{Real}}[n+1]$. Regarding, the replies to the tampering queries the proof for the two cases, "$\exists i : \tilde{c}_i = c_i$" and "$\forall i : \tilde{c}_i \neq c_i \wedge \exists j : \tilde{v}_j = v_j$" are identical to the base case. For the last case "$\forall i : \tilde{v}_i \neq v_i$", the proof slightly changes, as the extractors $\mathcal{E}_{i,n+1}$ for the round $n+1$ depend on all previous extractors, and $\mathcal{A}$. As in the base case, we define $\mathcal{A}_{v,i}$, $z_{v,i}$ and $\mathcal{A}_{s,1-i}$.

1. **Program** $\mathcal{A}_{v,i}^{\mathcal{O}^\lambda(\tau_{1-i},\cdot)}(h, v_{1-i}, z_{v,i})$:
   Parse $z_{v,i}$ as (τ_i, s_i, s_{1-i}), query $\mathcal{O}^\lambda(\tau_{1-i},\cdot)$ with $g_{s_{1-i}}(x) := \bar{h}(x||s_{1-i})$ and let $\bar{v}_{1-i}$ be the answer. Compute $v_i \leftarrow h(s_i||\bar{v}_{1-i};\tau_i)$.
   For $j = 1,\ldots,n+1$:
   - Sample $(g_0^j, g_1^j) \leftarrow \mathcal{A}(h)$.
   - Define $g_{v_i,s_{1-i}}(\tau_{1-i}) := g_{1-i}^j(\tau_{1-i}, s_{1-i}, v_i)$, send $g_{v_i,s_{1-i}}$ to $\mathcal{O}^\lambda(\tau_{1-i},\cdot)$ and let w_{1-i} be the answer.
   - Compute $w_i \leftarrow g_i^j(\tau_i, s_i, v_{1-i})$.
   - Send (w_0, w_1) to $\mathcal{A}$ and receive (f_0^j, f_1^j).
   - Compute $(\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}) \leftarrow f_i^j(\tau_i, s_i, v_{1-i})$ and set $\tilde{c}_i := (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i})$.
   - If $j \leq n$:
     - Sample $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,j}(h, v_{1-i}, z_{\mathcal{E}_{i,j}})$.
     - Compute $v_i \leftarrow h\left(\tilde{s}_i||\bar{h}(\hat{\tau}_{1-i}||\hat{s}_{1-i});\tilde{\tau}_i\right)$ and set $\tilde{c}_{1-i} := (\hat{r}_{1-i}, \hat{s}_{1-i}, v_i)$.
     - Send $(\tilde{c}_0, \tilde{c}_1)$ to $\mathcal{A}$.

   Output: $(\tilde{v}_{1-i}, st)$, where $st = (\tilde{\tau}_i, \tilde{s}_i, \tilde{v}_{1-i}, \tau_i, s_i, v_{1-i})$.
2. (**Auxiliary input for** $\mathcal{A}_{v,i}$): set $z_{v,i} = (\tau_i, s_i, s_{1-i})$.
3. (**Existence of the extractor,** $\mathcal{E}_{i,n+1}$**, and auxiliary input,** $z_{\mathcal{E}_{i,n+1}}$): Given $\mathcal{A}_{v,i}$ and $z_{v,i}$, by the 1-more extractability property of $\mathcal{H}_k$ under leakage, there exists an extractor $\mathcal{E}_{i,n+1}$ for $\mathcal{A}_{v,i}$, with auxiliary input, $z_{\mathcal{E}_{i,n+1}}$, that computes $(\hat{\tau}_{1-i}, \hat{s}_{1-i}, \hat{v}_i) \leftarrow \mathcal{E}_{i,n+1}(h, v_{1-i}, z_{\mathcal{E}_{i,n+1}})$.

The definition of $\mathcal{A}_{s,1-i}$, s, follows.

1. **Program** $\mathcal{A}_{s,1-i}(h, \tau, s, st)$:
   - Parse s as $s_{1-i}||\bar{v}_i$, and set $\tau_{1-i} := \tau$.
   - Compute $v_i \leftarrow h(s_i||\bar{h}(\tau_{i-1}||s_{i-1});\tau_i)$.[20]
   - Compute $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}, \tilde{v}_i) \leftarrow f_{1-i}^{n+1}(\tau_{1-i}, s_{1-i}, v_i)$.
   - **Output**: $(\tilde{\tau}_{1-i}, \tilde{s}_{1-i}||\bar{h}(\tilde{\tau}_i, \tilde{s}_i))$.
2. (**Define message** s): set $s := s_{1-i}||\bar{h}(\tau_i||s_i)$.

By the 1-more extractability property of $\mathcal{H}_k$ under leakage, $\mathcal{E}_{i,n+1}$ outputs a valid pre-image with overwhelming probability and using the same arguments that we used for the base case, we prove that the extracted value is consistent with $\mathbf{t}_{\mathsf{Real}}[n+1]$. This concludes the proof of the claim and implies the correctness of the values computed by $\mathcal{A}'$ in step 3, $\mathbf{lk}$, $\mathbf{t}_0$, $\mathbf{t}_1$, up to round $j^* - 1$, conditioned on $j_{\mathsf{d}} = j^*$.

[20] $\mathcal{A}_s$ knows (τ_i, s_i), $(\tilde{\tau}_i, \tilde{s}_i)$, since they are stored in st.

A.3 Proof of Theorem 3.11

Proof. Let h be a random function that will be accessed by the extractor and the attacker in a black-box way. For any $\mathcal{A}_v$ with auxiliary input z_v we define $z_{\mathcal{E}} := z_v$ and $\mathcal{E}^{h(\cdot)}$ as follows:

$\mathcal{E}^{h(\cdot)}(\mathbf{v}, z_{\mathcal{E}})$:

1. (**Initialization**): Set $Q_v := \emptyset$.
2. (**Execute** $\mathcal{A}_v$): Execute $\mathcal{A}_v(\mathbf{v}, z_v)$ and for each oracle query q of $\mathcal{A}_v$, query the oracle with q, set $Q_v = Q_v \cup \{\mathsf{q}\}$ and send $h(\mathsf{q})$ to $\mathcal{A}_v$. At the end of the execution receive $\tilde{v}$ from $\mathcal{A}_v$.
3. (**Output**): If there exists $\mathsf{q} \in Q_v$ such that $h(\mathsf{q}) = \tilde{v}$, set $\hat{s} := \mathsf{q}$, otherwise, set $\hat{s} := \perp$. Output $\hat{s}$.

Clearly, the running time of $\mathcal{E}^{h(\cdot)}$ is linear in the running time of $\mathcal{A}_v$. According to Definition 3.1 (the leakage is omitted here), we need to prove that for any any PPT algorithm $\mathcal{A}_v$, any $z_v \in \{0,1\}^{\mathrm{poly}(k)}$, all PPT algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$,

$$\Pr_h\left[\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k),$$

where,

$$\begin{array}{l}
\mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) : \\
v_i = h(s_i), i \in [\ell] \\
\mathbf{v} = (v_1, \ldots, v_\ell) \\
(\tilde{v}, st) \leftarrow \mathcal{A}_v^{h(\cdot)}(\mathbf{v}, z_v) \\
\hat{s} \leftarrow \mathcal{E}^{h(\cdot)}(\mathbf{v}, z_{\mathcal{E}}) \\
\tilde{s} \leftarrow \mathcal{A}_s^{h(\cdot)}(\mathbf{s}, st) \\
\\
\text{If } h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s}) \neq \tilde{v}, \text{ return 1} \\
\text{otherwise, return 0}
\end{array}$$

Let $\mathcal{Q} = \{s_i \mid i \in [\ell]\}$. We define the following events,

$$B\text{: } \mathsf{Exp}^{\mathbf{s},h}_{\mathcal{A}_v,\mathcal{A}_s,\mathcal{E}}(\ell, z_v, z_{\mathcal{E}}) = 1,\ E\text{: } h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i.$$

Clearly, $\Pr[B \wedge \neg E] = 0$, thus we only need to bound $\Pr[B \wedge E]$.

$$\begin{aligned}
\Pr[B \wedge E] &= \Pr[h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h(\hat{s}) \neq \tilde{v}] \\
&= \Pr[h(\tilde{s}) = \tilde{v} \wedge \forall i : \tilde{s} \neq s_i \wedge \hat{s} = \perp] \\
&\leq \Pr[h(\tilde{s}) = \tilde{v} \wedge \tilde{s} \notin (\mathcal{Q} \cup Q_v)] \leq \frac{1}{2^k} = \mathsf{negl}(k),
\end{aligned}$$

where the last inequality follows from the fact that $\tilde{s}$ does not belong to the set of queries made to h, thus $h(\tilde{s})$ is completely random over $\{0,1\}^k$. This completes the proof of the theorem. ∎

A.4 Proof of Claim 3.10

Proof. Let $L_i := \mathsf{Leak}^0_{\mathcal{A}',m_i}(k)$ and $T_i := \mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_i}(k)$. For $i \in \{0,1\}$ we have,

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i)=1|j^*=j_{\mathsf{d}}\right] &\geq \Pr\left[\mathsf{D}'(L_i)=1|j^*=j_{\mathsf{d}}, d=1\right]\cdot\Pr[d=1|j^*=j_{\mathsf{d}}]\\ &\geq \Pr\left[\mathsf{D}(T_i)=1\right]-\mathsf{negl}(k).\end{aligned}$$

The last inequality follows from Claims 3.8, 3.9 and the definition of D'. Symmetrically,

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i)=1|j^*\neq j_{\mathsf{d}}\right] &\geq \Pr\left[\mathsf{D}'(L_i)=1|j^*\neq j_{\mathsf{d}}, d\neq 1\right]\cdot\Pr[d\neq 1|j^*\neq j_{\mathsf{d}}]\\ &\geq \frac{1}{2}-\mathsf{negl}(k).\ \text{(Claim 3.8, }\mathsf{D}')\end{aligned}$$

and from the above relations we receive

$$\begin{aligned}\Pr\left[\mathsf{D}'(L_i)=1\right] &= \Pr\left[\mathsf{D}'(L_i)=1|j^*=j_{\mathsf{d}}\right]\cdot\Pr[j^*=j_{\mathsf{d}}]\\ &\quad+\Pr\left[\mathsf{D}'(L_i)=1|j^*\neq j_{\mathsf{d}}\right]\cdot\Pr[j^*\neq j_{\mathsf{d}}]\\ &\geq \Pr\left[\mathsf{D}(T_i)=1\right]\cdot\frac{1}{q}+\frac{q-1}{2q}-\mathsf{negl}(k).\end{aligned}\tag{1}$$

Similarly, we obtain an upper bound on $\Pr\left[\mathsf{D}'(L_i)=1\right]$,

$$\Pr\left[\mathsf{D}'(L_i)=1\right]\leq \Pr\left[\mathsf{D}(T_i)=1\right]\cdot\frac{1}{q}+\frac{q-1}{2q}+\mathsf{negl}(k).\tag{2}$$

From 1, 2 we receive

$$\left|\Pr\left[\mathsf{D}'(L_i)=1\right]-\left(\frac{\Pr\left[\mathsf{D}(T_i)=1\right]}{q}+\frac{q-1}{2q}\right)\right|\leq\mathsf{negl}(k).\tag{3}$$

Let $\delta := |\Pr[\mathsf{D}'(L_0)=1]-\Pr[\mathsf{D}'(L_1)=1]|$. We compute,

$$\begin{aligned}\delta &= \left|\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}-\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}+\Pr[\mathsf{D}'(L_0)=1]-\Pr[\mathsf{D}'(L_1)=1]\right.\\ &\left.-\left(\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}+\frac{q-1}{2q}\right)+\left(\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}+\frac{q-1}{2q}\right)\right|\\ &\geq \epsilon/q-\left|\frac{\Pr\left[\mathsf{D}(T_0)=1\right]}{q}+\frac{q-1}{2q}-\Pr[\mathsf{D}'(L_0)=1]\right.\\ &+\left.\Pr[\mathsf{D}'(L_1)=1]-\frac{\Pr\left[\mathsf{D}(T_1)=1\right]}{q}-\frac{q-1}{2q}\right|\geq\epsilon/q-\mathsf{negl}(k).\end{aligned}\tag{4}$$

The above inequalities follow from 3, the triangle inequality, and the assumption that $\mathcal{A}$ distinguishes between $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_0}(k)$ and $\mathsf{Tamper}^{\mathsf{cnmlr}}_{\mathcal{A},m_1}(k)$, with non-negligible probability ϵ. Assuming such an attacker $\mathcal{A}$, we proved that D' distinguishes between $\mathsf{Leak}^0_{\mathcal{A}',m_0}(k)$ and $\mathsf{Leak}^0_{\mathcal{A}',m_1}(k)$, with non-negligible probability $\epsilon/q-\mathsf{negl}(k)$, and the proof is complete. ■

A.5 Proof of Claim 4.6

Proof. Let $\mathcal{A}_v$ and $\mathcal{E}$ be the adversary and extractor as defined above, and let $(\hat{s},\hat{\tau})$ be the extracted value in H_1. We further define $\mathcal{A}_s$ as follows: on input (h,τ,s,z), $\mathcal{A}_s$

outputs $(\tilde{s}, \tilde{\tau}) \leftarrow \mathcal{A}_2(h, \tau, s, z)$. We define the following events:
E_1: $h(\tilde{s}; \tilde{\tau}) = h(\hat{s}; \hat{\tau}) \ \wedge \ (\tilde{s}; \tilde{\tau}) \neq (\hat{s}; \hat{\tau})$, E_2: $h(\tilde{s}; \tilde{\tau}) = \tilde{c} \ \wedge \ \tilde{c} \neq c \ \wedge \ h(\hat{s}; \hat{\tau}) \neq c$.

Since $\mathcal{H}_k$ is collision resistant, $\Pr[E_1] = \mathsf{negl}(k)$. Otherwise, one can find a collision with non-negligible probability by simulating H_1 with $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $\mathcal{E}$. By the 1-more extractability property of $\mathcal{H}_k$, we have that $\Pr[E_2] = \mathsf{negl}(k)$.

Finally, we observe that Real outputs 1 and $H_1 = 0$ only when the extractor $\mathcal{E}$ fails to extract a valid decommitment, or it extracts a value such that $(\hat{s}, \hat{\tau}) \neq (\tilde{s}, \tilde{\tau})$. This is captured by the events E_1, E_2. Therefore, conditioned on $\neg(E_1 \vee E_2)$, $\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1$ implies that $H_1 = 1$. We can then conclude:

$$\Pr[\mathsf{Real}_{\mathcal{A}}(\mathcal{R}, s, z) = 1] < \Pr[H_1 = 1] + \Pr[\neg(E_1 \vee E_2)] = \Pr[H_1 = 1] + \mathsf{negl}(k).$$

This completes the proof of the claim. ■