Practical Non-Malleable Codes from ℓ-more Extractable Hash Functions

Aggelos Kiayias
University of Edinburgh
akiayias@inf.ed.ac.uk

Feng-Hao Liu
Florida Atlantic University
fenghao.liu@fau.edu

Yiannis Tselekounis*
University of Edinburgh
tselekounis@sians.org

ABSTRACT

In this work, we significantly improve the efficiency of non-malleable codes in the split state model, by constructing a code with codeword length (roughly) $|s|+9k$, where $|s|$ is the length of the message, and k is the security parameter. This is a substantial improvement over previous constructions, both asymptotically and concretely.

Our construction relies on a new primitive which we define and study, called ℓ-more extractable hash functions. This notion, which may be of independent interest, is strictly stronger than the previous notion of extractable hash by Goldwasser et al. (Eprint '11) and Bitansky et al. (ITCS '12, Eprint '14), yet we can instantiate it under *the same* assumption used for the previous extractable hash function (a variant of the Knowledge of Exponent Assumption).

Keywords

Non-malleable codes, hash functions, split-state model

1. INTRODUCTION

Non-malleable codes were introduced by Dziembowski et al. [29] as a relaxation of error correction and error detection codes. They provide security in the following sense: any modified codeword decodes to the original message or to a completely unrelated one, with overwhelming probability. Non-malleability is defined through a simulation-based definition, which informally states that, for any tampering function f, we require the existence of a simulator that simulates the tampering effect, by only inspecting f, i.e., without making any assumptions on the distribution of the encoded message.

Various applications of non-malleable codes have been proposed, such as CCA secure encryption schemes [21], non-malleable commitments [5], and most notably, their application against malicious modification attacks, also known as *tampering attacks*. Indeed, using non-malleable codes to secure implementations against tampering attacks was the motivation in the original work by Dziembowski et al. [29]. Due to their important application, constructing non-malleable codes has drawn a lot of attention, as we elaborate below.

The split-state model [29, 40]. Ideally, we would like to achieve non-malleability against arbitrary function classes, yet, this task is not achievable, as it is also not achievable in the case of error correction/detection codes. As discussed in [29], assuming a tampering function f that computes the decoding of the codeword $c = \mathsf{Enc}(s)$, where s is the private message, and computes $\tilde{c} = \mathsf{Enc}(s+1)$, we receive a tampered codeword, $\tilde{c}$, that decodes to a message, highly related to the original one. Therefore, no secure construction can exist against any function class that contains f, which concludes that, restricting the function class, is inherent.

Motivated by the above, various function classes have been studied, and in particular, the *split-state* function class has been identified and extensively studied in the literature. Briefly speaking, in the split-state model, private memory is split in two parts, L, R, and the attacker may apply any function $f = (f_1, f_2)$ that results in a tampered memory equal to $(f_1(L), f_2(R))$. This is a plausible model since in many cases sensitive data may be split in two storage devices that are physically separated. Note that the model can generalize to multiple split states, with the two-state variant being the hardest to achieve; we only consider the two state variant in this paper.

Broadly speaking, (explicit) constructions of non-malleable codes in the split-state model can be categorized into information -theoretic and computational.[1] In a recent breakthrough result [4], Aggarwal et al. provide the first polynomial-time, information-theoretic, non-malleable code for multi-bit messages, thus significantly improving over the work of [28], which only supports single-bit messages. The encoder produces codewords of length $O((|s|+k)^7)$, where $|s|$

*Work performed while at the National and Kapodistrian University of Athens.

Research supported by ERC project CODAMODA, # 259152 and H2020 Project Panoramix # 653497.

CCS'16 October 24-28, 2016, Vienna, Austria

ACM ISBN 978-1-4503-4139-4/16/10.

DOI: http://dx.doi.org/10.1145/2976749.2978352

[1] The work of [29] showed that in the random oracle model, there exist efficient non-malleable codes against split-state tampering functions. However, their approach uses a probabilistic argument thus providing only a proof of existence and not an explicit construction. Therefore, their random oracle result does not count as an explicit construction. Currently, there is no known explicit constructions in the random oracle model to our knowledge.

denotes the length of the encoded message, s, and k is the security parameter.[2] Later Aggarwal et al. [3] proposed another construction that achieves codeword length roughly $O(|s|)$ (for sufficiently large $|s|$).[3]

In the computational setting, Liu and Lysyanskaya [40] construct a non-malleable code using cryptographic tools such as leakage resilient public-key encryption [43], and robust non-interactive zero-knowledge (NIZK) proofs [26]. The rate of their construction is not given in the original paper and a textbook instantiation with public-key encryption combined with NIZKs, would not yield a rate 1 code; however, using state of the art tools, we can provide a better instantiation of [40], with codeword length $|s| + O(k^2)$, see Table 1.1. Recently, Aggarwal et al. [2] presented a compiler that transforms any low rate, non-malleable code, to a rate 1, computationally secure, non-malleable code. The underlying encoding must satisfy a notion, strictly stronger than non-malleability, called *augmented* non-malleability, which, as it is stated in [2], can be satisfied by the construction of [4]. Thus, by instantiating the compiler of [2] with the construction of [4], the codeword's length becomes $|s| + O(k^7)$.

Although the above constructions achieve "rate 1" asymptotically, i.e., the ratio of message to codeword length is 1, as the message length, $|s|$, goes to infinity, in practice, the induced overhead can still be too large, when considering short messages (e.g., a 160-bit cryptographic key), even without counting the potentially *large hidden constants* in the asymptotic notation. Thus, even though the problem of "optimal-rate" has been solved in theory, it is still unclear what the practical implications of those constructions are. Given the current state of the art, as discussed above, constructing codes with very small overhead, including the hidden constant, remains still one of the most important open questions in the area. Note, that the natural lower bound for code length is merely $|s| + k$, and none of the known, computational or information-theoretic, constructions, match it, even asymptotically.

1.1 Our Results

In this work, we tackle the challenge to construct truly efficient non-malleable codes in the split-state model. To achieve this goal, we introduce a new cryptographic primitive, called *ℓ-more extractable hash function family*, and then we construct an efficient code, using our new tool. Our approach is modular: first we propose and formalize ℓ-more extractable hash function families, and then we demonstrate their application to non-malleable codes.

Briefly speaking, ℓ-more extractable hash function families capture the idea that, if an adversary, given ℓ hash values $v_1, \ldots, v_\ell$, produces a new valid hash value $\tilde{v}$, then it must know a pre-image of $\tilde{v}$. This is a generalization of the notion of extractable hash functions by Bitansky et al. [10] and Goldwasser et al. [49], which corresponds to the $\ell = 0$ case (i.e., the adversary gets no access to valid hash values, prior to producing its own value), and is somewhat reminiscent of the strengthening of simulation-soundness in the context of zero-knowlege proofs [48]. Our generalization is strict: we prove the following (informally stated):

THEOREM 1.1 (INFORMAL). *Extractable hash $\not\Rightarrow$ 1-more extractable hash.*

The subtlety comes from the fact that the ℓ-more attacker might get an "unfair advantage" in producing a valid hash value, for which it does not possess a pre-image, because of the ℓ additional inputs; e.g., by modifying the v_i's in some suitable way. Indeed, we show that the extractable hash function family of Bitansky et al. [10] is easily malleable, and thus exploitable by "1-more" attackers. This demonstrates that our new notion of ℓ-more extractability is strictly stronger than the previous one.

Our next step is to achieve such a stronger notion. We show, somewhat surprisingly, that the stronger notion can be achieved under the same assumptions used by the construction of Bitansky et al. [10], i.e., a variant of the *Knowledge of Exponent Assumption* (KEA) and DLOG. Thus, we conclude that, even though ℓ-more extractability is strictly stronger, KEA and DLOG are still sufficient to achieve it.

THEOREM 1.2 (INFORMAL). DLOG *and (a variant of) KEA imply ℓ-more extractable hash.*

We remark that KEA is non-falsifiable (cf. [42]), and it is indeed a strong assumption. However, one can argue that non-falsifiability might be inherent for extractable hash functions, and thus ℓ-more extractability. We recall that Bitansky et al. [10] showed that, extractable hash function families imply succinct non-interactive argument of knowledge (SNARK), and Gentry and Wichs [35] showed that SNARK is unlikely to be constructed based on falsifiable assumptions. Thus, non-falsifiable assumptions are likely to be inherent for achieving (ℓ-more) extractability. We note that some variants of KEA were shown to contradict (public-coin) differing-inputs obfuscation and indistinguishability obfuscation [12, 13]; the variant we use is suitably defined to circumvent this contradiction.

Next, we construct non-malleable codes using ℓ-more extractable hash functions. The crux of our methodology is to adapt the "public-key-encrypt-and-prove" method of [40], using our new ℓ-more extractable hash, yielding effectively a "(one-time-symmetric-key-encrypt)-and-hash" approach for obtaining non-malleable codes. In particular, we prove the following (informally stated):

THEOREM 1.3 (INFORMAL). *ℓ-more extractable hash (with some additional properties) implies non-malleable codes in the split-state model.*

Our scheme produces codewords of length $|s| + 9 \cdot k + 2 \cdot \log^2(k)$. In Table 1.1 we compare our construction with the current state of the art on the split-state setting. Our scheme is truly efficient in terms of codeword length, and it is one order of magnitude better than the combination of [40] + [2] + [43] + [37], which is the most competitive scheme that can be constructed,[4] based on the current state of the

[2]The result of [4] can be further improved assuming specific conjectures.

[3] The hidden constants might be "astronomical" as they depend on results in additive combinatorics, as pointed out in the conclusion of their work [3].

[4]For the sake of this comparison, we instantiate [40] with the efficient zero-knowledge proofs of [37] and the leakage resilient public-key encryption of [43]; moreover we observe that the resulting code is compatible with the compiler of [2] (it satisfies "augmented non-malleability", a property defined in the latter paper) and thus we can make the resulting system rate 1. This provides codeword length $|s| + O(k^2)$, cited in Table 1.1.

art. We note that, existing constructions in the information-theoretic setting, such as [3,4], and the work built on top of them, e.g., [2], might require very large constants, inherited by the results in additive combinatorics (cf. conclusion of the work [3]).

1.2 Technical Overview

Concepts of extractability and ℓ-more extractability. 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, such that $h(s) = v$, is infeasible. This idea is formalized in the following way: for any algorithm $\mathcal{A}_v$, there exists an extractor $\mathcal{E}_{\mathcal{A}_v}$, such that, if $\mathcal{A}_v$ produces some $v \in \mathsf{Image}(h)$, $\mathcal{E}_{\mathcal{A}_v}$ outputs s, such that $h(s) = v$. Clearly, such families are interesting only if they posses some sort of hardness property, like one-wayness, otherwise the problem is trivial.

In this work, we introduce the notion of *ℓ-more extractable hash function families*, for which the extractability property holds, even if $\mathcal{A}_v$ is given access to ℓ valid hash values. Even though ℓ-more extractability looks similar to extractability (0-more extractability in our definition), we provide a separation between those two primitives. Before explaining further details, we first recall the underlying assumption, t-KEA, and the construction of Bitansky et al. [10].

t-KEA and the extractable hash function family of [10]. Assuming a group $\mathbb{G}$, of prime order p, the Knowledge of Exponent Assumption (KEA), introduced by Damgård [25], states the following: any adversary that is given a generator, g, of $\mathbb{G}$, and a random group element g^a, produces the pair (g^s, g^{as}), only if it "knows" the exponent s. The assumption was later extended by [8, 38], by requiring that, given g^{r_1}, g^{ar_1}, g^{r_2}, g^{ar_2}, it is infeasible to produce $v = g^{r_1 s_1 + r_2 s_2}$ and v^a, without "knowing" s_1, s_2. This assumption, generalized for $t = \text{poly}(\log |\mathbb{G}|)$ pairs g^{r_i}, g^{ar_i}, is referred to as t-KEA by [10].

An element from the hash function family of [10] is described by the pair $(g^{\mathbf{r}}, g^{a\mathbf{r}})$, for uniformly random vector $\mathbf{r}$, and element a. Note that, $g^{\mathbf{r}}$ denotes the value $(g^{r_1}, \ldots, g^{r_t})$, where $\mathbf{r} = (r_1, \ldots, r_t)$. The hash of a message $\mathbf{s} = (s_1, \ldots, s_t)$, is the pair $(g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{a\langle \mathbf{r},\mathbf{s}\rangle})$, where $\langle \mathbf{r}, \mathbf{s}\rangle$ denotes the inner product of $\mathbf{r}$, $\mathbf{s}$. It is not hard to see that the hash value can be computed efficiently given the message and the description of the hash function, and assuming the t-KEA, the above hash function family is extractable, or in our terminology, 0-more extractable. As we argue in the next paragraph, this family is not 1-more extractable, and thus, extractability does not imply ℓ-more extractability.

1-more Extractable Collision Resistant Hash (ECRH). Suppose the adversary receives a hash value $v = h(\mathbf{s}) = (g^{\langle \mathbf{r},\mathbf{s}\rangle}, g^{a\langle \mathbf{r},\mathbf{s}\rangle})$, for some unknown message $\mathbf{s}$, and then computes $v' = v^x = (g^{\langle \mathbf{r},x\mathbf{s}\rangle}, g^{a\langle \mathbf{r},x\mathbf{s}\rangle})$, for some non-zero x, of its choice. Clearly, the new hash value v' equals $h(x\mathbf{s})$, and thus, it is valid. Then, assuming an extractor for the current family, under the "1-more" setting, we can retrieve the original message $\mathbf{s}$, by first extracting $x\mathbf{s}$ and then dividing it by x. This idea can be turned into a DLOG solver, and thus, assuming the hardness of DLOG with respect to $\mathbb{G}$, we show in Lemma 3.5, that the above construction is not 1-more extractable.

Next we present our strategy for constructing 1-more ECRH. Our main observation is that, even though the above hash function family is malleable, the modified hash value, v', has some structure: it is the hash value of the message yielded after applying an affine transformation on the original message, $\mathbf{s}$, (in the above case, the affine transformation was $x \cdot \mathbf{s}$). Interestingly, we show that under the t-KEA, applying an affine transformation is the only thing the adversary can do! In particular, we show that, if the adversary outputs a valid, new hash value, v', then there exists an extractor that extracts an affine transformation on the underlying message. So, in order to make the hash non-malleable (and then 1-more extractable), we first encode $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s})$ using a non-malleable code against affine functions, and then we compute $v = (g^{\langle \mathbf{r},\mathbf{c}\rangle}, g^{a\langle \mathbf{r},\mathbf{c}\rangle})$. This approach can be viewed as a computational analogue of a non-malleable reduction, as previously used by [4], and then formally presented by [3] (both are in the information-theoretic setting).

It turns out that, in order to apply the methodology described above, a slightly stronger flavor of non-malleability is required for the underlying code, which we formalize as *randomness simulatable non-malleable codes*. Below, we briefly discuss this notion and we give the main idea of the proposed scheme.

Randomness simulatable NM codes against affine tampering. This notion of non-malleability is stronger than the standard one, in the sense that, besides simulating the pre-image of the tampered codeword, $\tilde{s}$, the simulator, also produces the randomness of the encoder, $\tilde{s}_{\mathsf{r}}$, such that the encoding of $\tilde{s}$ with randomness $\tilde{s}_{\mathsf{r}}$, produces the tampered codeword. The main idea of our construction method is given in the next paragraph.

For any message s, the encoder secret shares s into (s_1, s_2), using a two-out-of-two, additive, secret sharing scheme, and outputs $\mathbf{c} = (s_1, s_2, s_1^2, s_2^2)$. Then, for any codeword $\mathbf{c} = (s_1, s_2, s_1', s_2')$, decoding proceeds as follows: if $s_i^2 = s_i'$, for $i \in \{1, 2\}$, the decoder outputs $s_1 + s_2$, otherwise, it outputs $\perp$. An affine tampering function, f, against the code is described by the pair $(\mathbf{b}, d)$, and the application of f on a codeword $\mathbf{c}$, yields the codeword $d \cdot \mathbf{c} + \mathbf{b}$. We prove security of the above code by considering the following cases (roughly). If $d = 0$, then the tampered codeword is completely overwritten by $\mathbf{b}$, and clearly, the output of the decoder depends only on $\mathbf{b}$. If $d \neq 0$, then, we argue that, either the attack leaves the codeword intact, i.e., $d = 1, \mathbf{b} = \mathbf{0}$, or the decoding of the tampered codeword is $\perp$, with overwhelming probability.

In Section 5.2, we formally define randomness simulatable, non-malleable codes, and prove security for the proposed scheme. It is worth to point out that the idea of constructing a NM-code for affine functions, as an intermediate step for providing split-state codes, was also followed by [4], still, our technique differs significantly, and their code does not directly satisfy our requirements. Moreover, in [22] the authors construct AMD codes, still their notions are slightly different and do fit in our framework.

NM codes against split-state tampering. Our construction of non-malleable codes is inspired by the one of Liu and Lysyanskaya [40], so we first recall their construction. To encode a message $\mathbf{s}$, their encoder outputs $(\mathsf{sk}, (\mathsf{pk},$

[5]The size of the CRS is $O(k)$, see [37]. The size of the CRS in our construction is roughly $32k$ bits, cf. Section 4. CRS size is independent of $|s|$.

Scheme	Codeword length	Model	Assumption
[4]	$O\left((\lvert s\rvert + k)^7 \log^7(\lvert s\rvert + k)\right)$	Information-theoretic	N/A
[3][3]	$O(\max\{\lvert s\rvert, k\})$	Information-theoretic	N/A
[4] + [2]	$\lvert s\rvert + O\left(k'\right)$	Computational	Authenticated Encryption (AE)
[40] + [2] + [43] + [37]	$\lvert s\rvert + O\left(k^2\right)$	Computational, CRS[5]	Leakage-Resilient PKE + robust NIZK
This work	$\lvert s\rvert + 9 \cdot k + 2 \cdot \log^2(k)$	Computational, CRS	1-time Leakage-Resilient AE + KEA

Table 1: Comparison of multi-bit NMC's in the split-state model. k is the security parameter. In the information-theoretic setting, typically security breaks with probability $\epsilon = 2^{-\Omega(k)}$; in the computational setting, we have $\epsilon = \mathsf{negl}(k)$, e.g., $\epsilon = k^{-\omega(1)}$ or $2^{-\Omega(k)}$, depending on how strong the underlying computational assumption is.

$\mathsf{E}_{\mathsf{sk}}(s), \pi)$), where E is the encryption algorithm of a leakage resilient, semantically secure, public-key encryption scheme $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$, sk, pk, denote the secret key and public key, respectively, and π is a non-interactive proof of knowledge (robust NIZK), that proves the existence of a valid secret key, decrypting the ciphertext to the message $\mathbf{s}$.

Our construction significantly improves the efficiency of [40] by refining their approach: (1) we replace the leakage resilient public key encryption with a one-time, symmetric-key, leakage resilient authenticated encryption; (2) we replace the (robust) NIZK proof with our 1-more-ECRH. Our encoder works as follows: to encode a message $\mathbf{s}$, the encoder outputs $\Big((r, \mathsf{sk}), (e = \mathsf{E}_{\mathsf{sk}}(\mathbf{s}), v = h(r, \mathsf{sk}))\Big)$, where E is the encryption algorithm of a symmetric, leakage resilient authenticated encryption scheme, sk is the secret key, h is a (randomized) 1-more ECRH.

Here the reader can easily observe that, using a function h that is extractable, or in our terminology, 0-more extractable, is not a good idea. Since generic authenticated encryption schemes guarantee security only if the secret key remains the same, it is possible to break security if one modifies sk as well. In fact, it is possible to construct an authenticated encryption such that it becomes insecure if the secret key is modified. Therefore, if the hash is malleable, then the tampering function may compute $(e' = \mathsf{E}_{\mathsf{sk}'}(\mathbf{s} + \mathbf{1}), v' = h(r, \mathsf{sk}'))$, where the sk' is a bad key that does not provide security. The tampered codeword clearly decodes to a related message, and thus cannot be non-malleable. Our 1-more extractability property resolves this issue: even if the attacker is given access to a valid hash value v, it cannot produce a valid hash value v', unless it knows a valid pre-image. Proving security for the above construction requires to handle multiple subtleties, and we refer the reader to Section 4 for further details.

Putting things together. We construct a one-time, symmetric, leakage resilient authenticated encryption scheme, that in order to sustain $2 \cdot k + \log^2 k$ bits of leakage, it requires key and ciphertext length $|s| + 5 \cdot k + 2 \cdot \log^2(k)$ (cf. Section 7). In addition, for our 1-more ECRH we have $|r| = |v| = 2k$ (see Constructions 5.5 and 5.2). Therefore, the total codeword length is $|s| + 9 \cdot k + 2 \cdot \log^2(k)$. The encoding and decoding procedures require 128 group operations (64 exponentiations plus 64 multiplications), independently of the message length, plus the cost of one-time authenticated encryption and decryption, respectively.

1.3 Related work

The first non-malleable code in the split-state model, for the information-theoretic setting was proposed by [28], yet their scheme can only encode single-bit messages. Subsequent constructions for multi-bit messages are discussed in the previous section. Non-malleable codes for other function classes have been extensively studied, e.g., bit-wise independent tampering [29], bounded-size function classes [32], the k-split setting [18], block-wise tampering [16, 20], and bounded depth and fan-in circuits [6]. The work of [3] develops beautiful connections among different function classes.

Other aspects of non-malleable codes have also been studied, such as rate-function class tradeoff, in the information-theoretic setting [19]. Other variants of non-malleable codes have been proposed, such as continuous non-malleable codes [30], augmented non-malleable codes [2], locally decodable/updatable non-malleable codes [17, 23, 31], which were used to secure the implementation of RAM computation. Leakage resilience was also considered as an additional feature, e.g., [17, 23, 40].

KEAs and previous work. In [25], Damgård introduces KEA to construct a CCA-secure encryption scheme. In [8, 38], the authors extend the assumption of [25], and construct three-round, zero-knowledge arguments. Abe and Fehr [1] construct the first perfect NIZK for NP with adaptive soundness, by extending the assumption of [8]. Prabhakaran and Xue [47] constructed statistically-hiding sets for trapdoor DDH groups [27], by introducing a new knowledge assumption. Gennaro et al. [34] proved that a modified version of the Okamoto-Tanaka key-agreement protocol [44] satisfies perfect forward secrecy against fully active attackers, by introducing a new knowledge assumption. In [9–11, 33, 36], the authors construct succinct, non-interactive, arguments of knowledge (SNARKs), and NIZKs, while in [41], Mie presents a private information retrieval (PIR), scheme. In [14,15,24], Canetti and Dakdouk provide an extensive study on extractable functions. In [45], Parno et al. show how to perform verifiable computation, efficiently.

In [12,13], the authors show that, assuming indistinguishability obfuscation [7], extractable one-way functions, and thus ECRHs, does not exist against adversaries receiving arbitrary, polynomial-size, auxiliary input, if the extractor is fixed before the attacker's auxiliary input. On the other hand, they show that, under standard assumptions, extractable one-way functions, may exist against adversaries with bounded auxiliary input.

In this work, and as it is suggested by [12], we consider individual auxiliary, i.e., we allow the auxiliary info of the extractor to depend on the attacker's auxiliary info, and therefore, we do not contradict the impossibility results of [12, 13].

2. PRELIMINARIES

In this section we present basic primitives and notation that we use in our constructions.

DEFINITION 2.1 (NOTATION). *$\mathbb{N}^+$, $\mathbb{R}^+$, denote the set of positive natural and real numbers, respectively. For $t \in \mathbb{N}^+$, $[t]$ is the set $\{1,\ldots,t\}$. For vectors $\mathbf{x},\mathbf{y}$, $\langle \mathbf{x},\mathbf{y}\rangle$ is the inner product of $\mathbf{x}$, $\mathbf{y}$, and $[\mathbf{x}]_i$ is the i-th coordinate of $\mathbf{x}$. For strings x, y, $x||y$, is the concatenation of x, y, and $|x|$ denotes the length of x. For a distribution D over a set $\mathcal{X}$, $x \leftarrow D$, denotes sampling an element $x \in \mathcal{X}$, according to D, $x \xleftarrow{\$} \mathcal{X}$, denotes sampling a uniform element x, from $\mathcal{X}$, and $U_{\mathcal{X}}$ denotes the uniform distribution over $\mathcal{X}$. The statistical distance between two random variables X, Y, with range $\mathcal{D}$, is denoted by $\Delta(X,Y)$, i.e., $\Delta(X,Y) = \frac{1}{2}\sum_{u\in\mathcal{D}} |\Pr[X=u] - \Pr[Y=u]|$. Moreover, "$\approx$" and "$\approx_c$", denote statistical and computational indistinguishability, respectively. A function $f : \mathbb{N} \to \mathbb{R}^+$ is negligible, if for every positive polynomial $\mathrm{poly}(\cdot)$, and all sufficiently large k, $f(k) \le 1/\mathrm{poly}(k)$, 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. Finally, for any element g and vector $\mathbf{r} = (r_1,\ldots,r_t)$, $g^{\mathbf{r}} = (g^{r_1},\ldots,g^{r_t})$.*

Below, we define coding schemes, based on the definitions of [29, 40].

DEFINITION 2.2. (CODING SCHEME IN THE COMMON REFERENCE STRING (CRS) MODEL *[40])* *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. In the full version of the paper we provide the standard definitions of coding schemes and non-malleability. Now we state the definition of strong non-malleability in the CRS model based on the definitions of [29, 40].

DEFINITION 2.3. (STRONG NON-MALLEABILITY IN THE CRS MODEL *[29,40])* *Let $(\mathsf{Init},\mathsf{Enc},\mathsf{Dec})$ be a (κ,ν)-coding scheme in the common reference string model, and $\mathcal{F}$ be a family of functions $f : \{0,1\}^\nu \to \{0,1\}^\nu$. For any CRS Σ, $f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment*

$$\mathsf{Tamper}_s^{\Sigma,f} \stackrel{\mathrm{def}}{=} \left\{ \begin{array}{l} c \leftarrow \mathsf{Enc}(\Sigma,s), \tilde{c} \leftarrow f^\Sigma(c), \tilde{s} = \mathsf{Dec}(\Sigma,\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{c} = c, \textit{ and } \tilde{s} \textit{ otherwise.} \end{array} \right\}$$

which is a random variable over the randomness of Enc and Dec. The coding scheme $(\mathsf{Init},\mathsf{Enc},\mathsf{Dec})$ is strongly non-malleable with respect to the function family $\mathcal{F}$, if for each $f \in \mathcal{F}$ and any s_0, $s_1 \in \{0,1\}^\kappa$,

$$\left\{\left(\Sigma, \mathsf{Tamper}_{s_0}^{\Sigma,f}\right)\right\}_{k\in\mathbb{N}} \approx \left\{\left(\Sigma, \mathsf{Tamper}_{s_1}^{\Sigma,f}\right)\right\}_{k\in\mathbb{N}},$$

where $\Sigma \leftarrow \mathsf{Init}(1^k)$, and "$\approx$" may refer to statistical, or computational, indistinguishability, with parameter k.

According to the standard definition of non-malleability, the decoding procedure is not randomized, however, as it is suggested by Ball et al. [6] Dec may be randomized.

Next we state the t-variant, due to [10], of the *Knowledge of Exponent assumption* (KEA), [8, 25, 38], with individual auxiliary inputs for adversary and extractor, which is known not to contradict the impossibility results of [12, 13].

ASSUMPTION 2.4 (t-KEA ASSUMPTION). *Let $t \in \mathbb{N}$. There exists a group generation algorithm $\mathcal{G}$, such that for any pair $(\mathbb{G},g)$ sampled according to $\mathcal{G}(1^k)$, where $\mathbb{G}$ is a group of prime order $p \in (2^{k-1},2^k)$, the following holds: for any* PPT *algorithm $\mathcal{A}$ with auxiliary input $\mathsf{aux}_\mathcal{A} \in \{0,1\}^{\mathrm{poly}(k)}$, there exist* PPT *extractor $\mathcal{E}_\mathcal{A}$ with auxiliary input $\mathsf{aux}_\mathcal{E} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all sufficiently large $k \in \mathbb{N}$,*

$$\Pr_{\substack{(\mathbb{G},g)\leftarrow\mathcal{G}(1^k) \\ (a,\mathbf{r})\xleftarrow{\$}\mathbb{Z}_p\times\mathbb{Z}_p^t}} \left[\begin{array}{l} (v,v') \leftarrow \mathcal{A}(g^{\mathbf{r}}, g^{a\mathbf{r}}, \mathsf{aux}_\mathcal{A}), v' = v^a : \\ \mathbf{x} \leftarrow \mathcal{E}_\mathcal{A}(g^{\mathbf{r}}, g^{a\mathbf{r}}, \mathsf{aux}_\mathcal{E}) \wedge g^{\langle \mathbf{r},\mathbf{x}\rangle} \neq v \end{array} \right] \le \mathsf{negl}(k).$$

Below, we define the class of affine functions.

DEFINITION 2.5 (THE FUNCTION FAMILY $\mathcal{F}_{\mathsf{aff}}$). *For any set $\mathcal{M}$ and any $t \in \mathbb{N}^+$, we define the following function class*

$$\mathcal{F}_{\mathsf{aff}} = \{f(\mathbf{s}) = d\cdot\mathbf{s} + \mathbf{b} \mid \mathbf{b},\mathbf{s} \in \mathcal{M}^t, d \in \mathcal{M}\}.$$

Next we recall the definition of extractable hash of [10]. The definition can be modified to have different auxiliary inputs for adversary and extractor as the t-KEA above.

DEFINITION 2.6 (EXTRACTABLE HASH [10]). *An efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ is extractable, if for any PPT algorithm $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}_\mathcal{A}^\mathcal{H}$, such that for all large $k \in \mathbb{N}$ and any auxiliary input $\mathsf{aux} \in \{0,1\}^{\mathrm{poly}(k)}$:*

$$\Pr_{h\leftarrow\mathcal{H}_k} \left[\begin{array}{l} y \leftarrow \mathcal{A}(h,\mathsf{aux}), \exists x : h(x) = y : \\ x' \leftarrow \mathcal{E}_\mathcal{A}^\mathcal{H}(h,\mathsf{aux}) \wedge h(x') \neq y \end{array} \right] \le \mathsf{negl}(k).$$

Below, we define the split-state functions class, $\mathcal{F}_{\mathsf{ss}}$, and the λ-bit leakage function class $\mathcal{L}_\lambda$. A definition for split-state leakage functions was considered in [40].

DEFINITION 2.7 (THE SPLIT-STATE FUNCTION FAMILY $\mathcal{F}_{\mathsf{ss}}$). *For any, even, $\nu \in \mathbb{N}$ and any efficiently computable function $f : \{0,1\}^\nu \to \{0,1\}^\nu$, $f \in \mathcal{F}_{\mathsf{ss}}$, if there exist efficiently computable functions $f_1 : \{0,1\}^\nu \to \{0,1\}^{\nu/2}$, $f_2 : \{0,1\}^\nu \to \{0,1\}^{\nu/2}$, such that for every x_1, $x_2 \in \{0,1\}^{\nu/2} \times \{0,1\}^{\nu/2}$, $f(x_1||x_2) = f_1(x_1)||\ f_2(x_2)$.*

DEFINITION 2.8 (THE λ-BIT LEAKAGE FUNCTION CLASS $\mathcal{L}_\lambda$). *For any $\lambda \in \mathbb{N}$, $\mathcal{L}_\lambda$ is the set of the efficiently computable functions that output λ bits, i.e., for any $g \in \mathcal{L}_\lambda$, $g : \{0,1\}^* \to \{0,1\}^\lambda$.*

Next, we state the definition of semantically secure authenticated encryption, against one time leakage.

DEFINITION 2.9. (SEMANTICALLY SECURE AUTHENTICATED ENCRYPTION AGAINST ONE TIME LEAKAGE) *Let k be the security parameter, let $(\mathsf{KGen},\mathsf{E},\mathsf{D})$ be a symmetric encryption scheme and let $\mathcal{L}$ be a set of functions. Then, $(\mathsf{KGen},\mathsf{E},\mathsf{D})$ is authenticated, semantically secure against one-time leakage with respect to $\mathcal{L}$, if*

1. **(Correctness)***: For every message s, $\Pr[\mathsf{D}_{\mathsf{sk}}(\mathsf{E}_{\mathsf{sk}}(s)) = s] = 1$, where $\mathsf{sk} \leftarrow \mathsf{KGen}(1^k)$.*

2. **(Semantic security)**: *for any function $g \in \mathcal{L}$ and any two messages s_0, s_1, the following distributions are (either computationally or statistically) indistinguishable:*

$$\Big(\mathsf{E}_{\mathsf{sk}}(s_0), g(\mathsf{sk})\Big) \approx \Big(\mathsf{E}_{\mathsf{sk}}(s_1), g(\mathsf{sk})\Big),$$

where $\mathsf{sk} \leftarrow \mathsf{KGen}(1^k)$.

3. **(Unforgeability)**: *For every PPT algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the probability*

$$\Pr\left[e' \neq e \wedge \mathsf{D}_{\mathsf{sk}}(e') \neq \perp \;\middle|\; \begin{array}{c}\mathsf{sk} \leftarrow \mathsf{KGen}(1^k); (s, \mathsf{st}) \leftarrow \mathcal{A}_1(1^k); \\ e \leftarrow \mathsf{E}_{\mathsf{sk}}(s); e' \leftarrow \mathcal{A}_2(e, \mathsf{st})\end{array}\right]$$

is negligible in k.

Here, it should be noted that the leakage function is being defined by the attacker before receiving the challenge ciphertext, otherwise semantic security breaks.

3. ℓ-MORE EXTRACTABLE HASH FUNCTION FAMILIES

In this section we define the notion of *ℓ-more extractable hash function families*, and we provide a general discussion on the primitive.

DEFINITION 3.1. (ℓ-MORE EXTRACTABLE HASH FUNCTION FAMILIES) *For $\ell \in \mathbb{N}$, an efficiently samplable hash function ensemble $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$, is ℓ-more extractable, if for any PPT algorithm $\mathcal{A}_v$ and any $\mathsf{aux}_{\mathcal{A}_v} \in \{0,1\}^{\mathrm{poly}(k)}$, there exist a PPT extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and $\mathsf{aux}_{\mathcal{E}} \in \{0,1\}^{\mathrm{poly}(k)}$, such that for all PPT algorithms $\mathcal{A}_s$, any large $k \in \mathbb{N}$ and any vector of messages $\mathbf{s} = (s_1, \ldots, s_\ell)$,*

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

where,

$\mathsf{Exp}^{\mathbf{s}, h_z}_{\mathcal{A}_v, \mathcal{A}_s, \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, \mathsf{aux}_{\mathcal{A}_v}, \mathsf{aux}_{\mathcal{E}})$:
$\forall i \in [\ell], s_{\mathsf{r}_i} \leftarrow U_{\{0,1\}^{\mathrm{poly}(k)}}, v_i = h_z(s_{\mathsf{r}_i}, s_i)$ *(**hash computation**)*
$\mathbf{s}_{\mathsf{r}} = (s_{\mathsf{r}_1}, \ldots, s_{\mathsf{r}_\ell}), \mathbf{v} = (v_1, \ldots, v_\ell)$
$(\tilde{v}, \mathsf{st}) \leftarrow \mathcal{A}_v(h_z, \mathbf{v}, \mathsf{aux}_{\mathcal{A}_v})$ *(**hash tampering**)*
$(\hat{s}_{\mathsf{r}}, \hat{s}) \leftarrow \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}(h_z, \mathbf{v}, \mathsf{aux}_{\mathcal{E}})$ *(**pre-image extraction**)*
$(\tilde{s}_{\mathsf{r}}, \tilde{s}) \leftarrow \mathcal{A}_s(h_z, \mathbf{s}_{\mathsf{r}}, \mathbf{s}, \mathsf{st})$ *(**pre-image tampering**)*

If $h_z(\tilde{s}_{\mathsf{r}}, \tilde{s}) = \tilde{v} \wedge \forall i : \tilde{v} \neq v_i \wedge h_z(\hat{s}_{\mathsf{r}}, \hat{s}) \neq \tilde{v}$, return 1 otherwise, return 0

The main steps in the above experiment are the following. Initially, we sample randomness for the hash, and perform the hash computation over $\ell \in \mathbb{N}$, pre-images. For deterministic hash function families we just omit randomness sampling, and we compute the hash, only using the messages. The challenge for the attacker $\mathcal{A}_v$, is to produce a valid hash value $\tilde{v}$, given ℓ has values, denoted as $\mathbf{v}$, and auxiliary information $\mathsf{aux}_{\mathcal{A}_v}$. Then, the extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ is executed, given $\mathbf{v}$ and its own auxiliary input $\mathsf{aux}_{\mathcal{E}}$. Notice, that, we allow the auxiliary input of the extractor to depend on the attacker's auxiliary input.[6] Finally, the adversary $\mathcal{A}_s$ produces 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.

[6]For this reason our definition is not contradicting the impossibility results of [12, 13].

Leaving aside the fact that the above definition considers randomized function families, the major difference between the current definition and the one given by Bitansky et al. [9, 10] (Definition 2.6), is two-fold: first the "ℓ-more" generalization that allows the attacker to have access to ℓ valid hash values for which it does not know the pre-images, prior to delivering its own hash value. Second, the introduction of the algorithm $\mathcal{A}_s$, that takes the place of the existential quantifier that appears in the original definition. This is in fact a *weakening* of the original definition, in the sense that the extractor is allowed to fail in case a pre-image exists but is not efficiently computable based on the view of the adversary (this would not be allowed in the original definition).

Note, that, the existence of $\mathcal{A}_s$ does not trivialize the problem for the extractor since the extractor is challenged to produce a valid pre-image for $\tilde{v}$, given only the code of $\mathcal{A}_v$ and its own auxiliary input (and in particular it lacks access to the state of $\mathcal{A}_v$ and the program of $\mathcal{A}_s$).

It is easy to see that, constructing ℓ-more extractable hash function families that are non-compressing, can be achieved using existing tools, such as robust NIZKs [26]. Here we construct an ℓ-more extractable, collision resistant, hash (ECRH) function family, achieving length-efficiency comparable to that of a regular hash function.

In the following lemma we prove that, for any ℓ-more ECRH function family, the output of the extractor should match the output of $\mathcal{A}_s$, in case both of them output valid pre-images, otherwise we break collision resistance.

LEMMA 3.2. *Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ be a collision resistant, ℓ-more extractable, efficiently samplable, hash function ensemble. Then, for any $\mathcal{A}_v$, $\mathsf{aux}_{\mathcal{A}_v}$, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, $\mathsf{aux}_{\mathcal{E}}$, $\mathcal{A}_s$, $\mathbf{s} = (s_1, \ldots, s_\ell)$, ℓ, as they were defined in Definition 3.1, the probability*

$$\Pr_{h_z \leftarrow \mathcal{H}_k}\left[(\hat{s}_{\mathsf{r}}, \hat{s}) \neq (\tilde{s}_{\mathsf{r}}, \tilde{s}) \;\middle|\; \begin{array}{c}\mathsf{Exp}^{\mathbf{s}, h_z}_{\mathcal{A}_v, \mathcal{A}_s, \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(\ell, \mathsf{aux}_{\mathcal{A}_v}, \mathsf{aux}_{\mathcal{E}}) = 0 \\ h_z(\tilde{s}_{\mathsf{r}}, \tilde{s}) = \tilde{v}, \tilde{v} \neq v_i, i \in [\ell]\end{array}\right]$$

is negligible in k.

For the proof see the full version of the paper.

Next, we show a separation of 0-more extractability and general ℓ-more extractability as we discussed in the introduction. In particular, we prove that the 0-more extractable hash of [10] is not 1-more extractable. Before doing so, we first revisit their construction, which is based on the t-KEA assumption (Assumption 2.4).

CONSTRUCTION 3.3. (0-MORE EXTRACTABLE HASH FROM t-KEA *[10]*) *Let $\mathcal{G}$ be a group-generation algorithm. An instance of a $(kt, 2k)$-compressing, hash function family, $\mathcal{H}^* = (\mathsf{Gen}^*, h^*)$, with respect to $\mathcal{G}$, is defined as follows:*

1. $\mathsf{Gen}^*(1^k)$: *sample $(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k)$, $p \in (2^{k-1}, 2^k)$, $(a, \mathbf{r}) \xleftarrow{\$} \mathbb{Z}_p \times \mathbb{Z}_p^t$, where $p = |\mathbb{G}|$, and output $z = (\mathbb{G}, g^{\mathbf{r}}, g^{a\mathbf{r}})$.*
2. ***Hashing computation**: on input $\mathbf{s}$, compute $h^*_z(\mathbf{s}) = (g^{\langle \mathbf{r}, \mathbf{s}\rangle}, g^{\langle a\mathbf{r}, \mathbf{s}\rangle})$.*

In [10] the authors prove that Construction 3.3 is collision resistant.

LEMMA 3.4. (COLLISION RESISTANCE FOR CONSTRUCTION *3.3 [10]*) *Assuming the hardness of the discrete logarithm problem, with respect to a group $\mathbb{G}$, Construction 3.3 is collision resistant, with respect to $\mathbb{G}$.*

The above construction is also extractable with respect to Definition 2.6 and 0-more extractable, where both properties follow from the t-KEA assumption. In the following lemma we prove that Construction 3.3 is not 1-more extractable.

LEMMA 3.5. (CONSTRUCTION 3.3 IS NOT 1-MORE EXTRACTABLE) *Let $\mathcal{H}^*$ be the hash function family of Construction 3.3, with respect to a group generation algorithm $\mathcal{G}$. Then, assuming the difficulty of the discrete logarithm problem for $\mathcal{G}$, $\mathcal{H}^*$ is not 1-more extractable.*

In the introduction, we argued that the hash function family of Construction 3.3 is highly malleable, and thus not 1-more extractable, under the discrete logarithm assumption. We formalize the proof in the full version.

4. A NON-MALLEABLE CODE AGAINST SPLIT-STATE TAMPERING

In this section, we present our construction of non-malleable codes against split-state tampering functions. Our construction requires (i) a one-time, authenticated, symmetric-key encryption scheme that is also leakage resilient, and (ii) a 1-more ECRH.

CONSTRUCTION 4.1. *Let $\mathcal{H}_k = (\mathsf{Gen}, h)$ be a hash function family, and let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be a symmetric encryption scheme. We define a coding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, as follows:*

- $\mathsf{Init}(1^k)$*: sample $z \leftarrow \mathsf{Gen}(1^k)$ and set $\Sigma = z$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: let s be the input to the encoder. The encoder samples $\mathsf{sk} \leftarrow \mathsf{KGen}(1^k)$, $r \xleftarrow{\$} \{0,1\}^{\mathrm{poly}(k)}$, $e \leftarrow \mathsf{E}_{\mathsf{sk}}(s)$, and outputs $\Big(r, \mathsf{sk}, e, h_z(r, \mathsf{sk})\Big)$. In particular, the left part of the codeword is (r, sk), while the right part is $(e, h_z(r, \mathsf{sk}))$.*
- $\mathsf{Dec}(\Sigma, \cdot)$*: let (r, sk, e, v) be the input to* Dec*. If $h_z(r, \mathsf{sk}) = v$, the decoder outputs $\mathsf{D}_{\mathsf{sk}}(e)$, otherwise, it outputs $\perp$.*

Since the input message to h_z, sk, possesses adequate entropy, it is possible to omit r in the above construction, still for the sake of clarity we stick to the formulation provided in Definition 3.1 and we use independent randomness for hashing sk. In what follows we prove that Construction 4.1 is strongly non-malleable against $\mathcal{F}_{\mathsf{ss}}$ (Definition 2.7), assuming that for any $f = (f_1, f_2) \in \mathcal{F}_{\mathsf{ss}}$, f_1, f_2, affect (r, sk) and (e, v), independently, i.e., we assume the strings $r||\mathsf{sk}$, $e||v$, are of length $\nu/2$, where ν is the length the codeword.[7]

Intuition for the construction. Before formally analyzing the construction, we first discuss the ideas on why our construction is secure. Consider a split-state tampering function (f_1, f_2), where f_1 is applied to (r, sk), and f_2 is applied to (e, v). To prove non-malleability, roughly speaking, we need to simulate the tampering experiment without knowing the underlying message distribution. A first idea is to simulate the left side with (r', sk'), and the right side with $(e' = \mathsf{E}_{\mathsf{sk}'}(\mathbf{0}), v' = h(r', \mathsf{sk}'))$, where r', sk' is fresh randomness and key, respectively, hoping to infer the final outcome of the tampering experiment correctly due to the semantic security of the encryption.

There are several subtleties in doing so. First and foremost, the simulator needs to be able to produce the decoding of the codeword in case v' is modified by f_2. This is where 1-more extractability will be used to obtain a valid pre-image, $(\hat{r}, \hat{\mathsf{sk}})$. It might be very tempting to conclude the simulation by outputting the decrypted message $\mathsf{D}_{\hat{\mathsf{sk}}}(\hat{e})$ (where $\hat{e}$ is the modified codeword). However, this may not be consistent with the real-world experiment, as the values produced by the extractor $(\hat{r}, \hat{\mathsf{sk}})$ might not be consistent with the output of f_1. To check consistency, the simulator would want to check the output of f_1, yet such a simulation would be impossible to prove since it depends on sk, where the indistinguishability between e' and e does not hold in the presence of it. To go around this, we use a similar technique to Liu and Lysyanskaya [40], who observed that, the equality test between $f_1(\hat{r}, \hat{\mathsf{sk}})$ and $f_1(r, \mathsf{sk})$ can be performed via the leakage of a universal hash (cf. the full version) with $\log^2 k$ bits of output. Putting this to our setting, by requiring the encryption $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ to be a one-time semantically secure, symmetric-key authenticated encryption, that is secure under $2k + \log^2 k$ bits of leakage, is sufficient to facilitate the simulation. We also note that the case when v' is not modified by f_2 can be easily taken care of by the security of the authenticated encryption: as long as the key is not modified, any attempt to modify the ciphertext will result in an invalid ciphertext.

THEOREM 4.2. *Let k be the security parameter, $\mathcal{H}_k$ be a 1-more extractable hash function family that outputs $\beta(k)$ bits, $\beta(k) = \mathrm{poly}(k)$, and let $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ be an authenticated, semantically secure, symmetric encryption scheme, that is leakage resilient against $\mathcal{L}_\lambda$, $\lambda(k) = \omega(\log k) + \beta(k)$. Then, Construction 4.1 is strongly non-malleable against $\mathcal{F}_{\mathsf{ss}}$.*

PROOF. Following the definition of strong non-malleability (Definition 2.3), we need to prove that for any $f = (f_1, f_2) \in \mathcal{F}_{\mathsf{ss}}$ and any pair of messages s_0, s_1, $(\Sigma, \mathsf{Tamper}_{s_0}^{f,\Sigma}) \approx_c (\Sigma, \mathsf{Tamper}_{s_1}^{f,\Sigma})$, where $\Sigma \leftarrow \mathsf{Init}(1^k)$. We introduce a series of hybrids (see Figure 1), and the proof can be derived directly from the indistinguishability between adjacent hybrids. We first explain the hybrids and define the notation used in those experiments.

- Given a tampering function $f = (f_1, f_2)$ and message s, the first experiment, $\mathsf{Exp}_0^{f,\Sigma,s}$, is exactly the original tampering game, $\mathsf{Tamper}_s^{f,\Sigma}$, of Definition 2.3.
- In $\mathsf{Exp}_1^{f,\Sigma,s}$, we slightly modify the previous hybrid by checking whether the function f_2 has modified the hash value v. Intuitively, by the collision resistance property of the hash function family $\mathcal{H}_k$, if f_2 does not modify v, then the attack produces a valid codeword, $\tilde{c}$, only if the parts of $\tilde{c}$ that constitute the pre-image of $\tilde{v}$, are kept intact, i.e., $(r, \mathsf{sk}) = (\tilde{r}, \tilde{\mathsf{sk}})$, otherwise there is a collision. In addition, assuming $\mathsf{sk} = \tilde{\mathsf{sk}}$, we have that, if $\tilde{e} \neq e$, then the output of the decoder should be $\perp$, otherwise we break the authenticity under leakage (v is considered as leakage over sk) property of the

[7] This can always be achieved using padding.

encryption scheme. On the other hand, if $v \neq \tilde{v}$, the output of the current experiment is produced as in $\mathsf{Exp}_0^{f,\Sigma,s}$.

- In $\mathsf{Exp}_2^{f,\Sigma,s}$, we modify the previous experiment for the case in which v is modified: instead of using the real decoding procedure, we use the extractor of the hash function family, to extract a pre-image $(\hat{r}, \hat{\mathsf{sk}})$, for $\tilde{v}$, and then compute the output, $\tilde{s}$, with respect to that pre-image. However, we cannot output $\tilde{s}$ directly as we still need to check consistency with the output of f, i.e., we need to check whether $(\hat{r}, \hat{\mathsf{sk}})$ is equal to $(\tilde{r}, \tilde{\mathsf{sk}})$. The indistinguishability between the current hybrid and the previous one, follows by the 1-more extractability property of the hash function, which, informally, guarantees that if $\tilde{c}$ is a valid codeword, then $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ produces a valid pre-image for $\tilde{v}$, with overwhelming probability. If the extracted pre-image is consistent with the one output by f, the current hybrid outputs a non-bottom value, equal to the one output by the decoding procedure of $\mathsf{Exp}_1^{f,\Sigma,s}$. On the other hand, if $(\hat{r}, \hat{\mathsf{sk}}) \neq (\tilde{r}, \tilde{\mathsf{sk}})$, Lemma 3.2 guarantees that $(\tilde{r}, \tilde{\mathsf{sk}})$ is not a valid pre-image for $\tilde{v}$, with overwhelming probability, and the current experiment properly outputs $\perp$. Finally, it is straightforward to see, that if $\tilde{v}$ is invalid, both experiments output $\perp$.

  In order to define the extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, introduced in $\mathsf{Exp}_2^{f,\Sigma,s}$, we first need to define $\mathcal{A}_v$, $\mathsf{aux}_{\mathcal{A}_v}$, with respect to h_z, v, e, and $f = (f_1, f_2)$. Formally, we define the following:

  1. (**Define $\mathcal{A}_v$**): $\mathcal{A}_v(h_z, v, \mathsf{aux}_{\mathcal{A}_v}) := ([f_2(\mathsf{aux}_{\mathcal{A}_v}, v)]_2, \mathsf{st})$, where $\mathsf{st} = (f_2(\mathsf{aux}_{\mathcal{A}_v}, v), \mathsf{aux}_{\mathcal{A}_v}, v)$.
  2. (**Choose auxiliary info for $\mathcal{A}_v$**): set $\mathsf{aux}_{\mathcal{A}_v} = e$.
  3. (**Existence of the extractor, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, and auxiliary input,** $\mathsf{aux}_{\mathcal{E}}$): Given $\mathcal{A}_v$ and $\mathsf{aux}_{\mathcal{A}_v}$, by the 1-more extractability property of $\mathcal{H}_k$, there exists an extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, with hardwired auxiliary info, $\mathsf{aux}_{\mathcal{E}}$, that computes $(\hat{r}, \hat{\mathsf{sk}}) \leftarrow \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}(h_z, v)$. The extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ is used in $\mathsf{Exp}_2^{f,\Sigma,s}$ and all subsequent experiments (for brevity we denote it as $\mathcal{E}$).

  We remind, that, for any vector v, $[v]_i$, denotes the i-th coordinate of v.

- In $\mathsf{Exp}_3^{f,\Sigma,s}$, we modify the consistency check procedure, so that we access the right part of the codeword, only through leakage. Instead of checking consistency using directly the output of f_1, we do the check using a random hash function, $\bar{h}$, from a universal family (cf. full version), applied to the output of f_1, plus one more bit, that indicates whether f_1 has modified its input. Here, the hash v is computed through leakage over sk. The experiment differs from the previous one only when there is a collision against $\bar{h}$, which happens with negligible probability, as $\bar{h}$ is a universal hash function.

  Below, we formalize the above procedure: let $\bar{h} \leftarrow \bar{\mathcal{H}}_{\lambda-1}$ be a random hash function from a universal hash function family, that outputs $\lambda - 1$ bits. We define the function $g_{\bar{h},h_z}(\cdot)$ as follows:

$$g_{\bar{h},h_z}(x,y) = \begin{cases} (0, \bar{h}(f_1(x,y)), h_z(x,y)), & \text{if } f_1(x,y) = (x,y), \\ (1, \bar{h}(f_1(x,y)), h_z(x,y)), & \text{if } f_1(x,y) \neq (x,y). \end{cases}$$

  We view $g_{\bar{h},h_z}$ as a leakage function that outputs $\lambda = \omega(\log k) + \beta(k)$ bits in total. The experiment will then use the leaked value to check consistency, instead of using the whole string output by f_1. Concretely, we introduce the random variable b, which depends on the output of the leakage function, and we modify $\mathsf{Exp}_2^{f,\Sigma,s}$, so that the condition "If $(b = 1)$", introduced in $\mathsf{Exp}_3^{f,\Sigma,s}$, is exactly the same as the condition "If $(r, \mathsf{sk}, e) = (\tilde{r}, \tilde{\mathsf{sk}}, \tilde{e})$", of experiment $\mathsf{Exp}_2^{f,\Sigma,s}$. This modification does not induce any statistical difference. In the next modification, we check equality between $(\hat{r}, \hat{\mathsf{sk}})$, $(\tilde{r}, \tilde{\mathsf{sk}})$, by checking if $\bar{h}(\hat{r}, \hat{\mathsf{sk}}) = \bar{h}(f_1(r, \mathsf{sk}))$. Clearly, this part induces a statistical difference only if there is a collision against $\bar{h}$, which happens with negligible probability, since $\bar{h}$ is a universal hash function, chosen by the current experiment, independently.

- Finally, we are going to show that $\mathsf{Exp}_3^{f,\Sigma,s}$ is indistinguishable from $\mathsf{Exp}_3^{f,\Sigma,\vec{0}}$, for any message s, where $\vec{0}$ denotes the zero-message. This follows by the semantic security of the leakage resilient encryption scheme (Definition 2.9).

A concrete presentation of the hybrids, is given in Figure 1.

The indistinguishability between the hybrids in proved in the full version of the paper. ■

Length of the CRS. The length of the CRS in our construction is roughly $32k$ bits: we need to hash a $6k$-bit (roughly) key of an authenticated encryption scheme and then encrypt the message using that key; this would require the parameters for the 16-KEA to be on the CRS, resulting in the $32k$ bits length.

5. CONSTRUCTING 1-MORE ECRH

In this section, we present our construction of 1-more extractable hash function families. Our construction is in two steps: (1) we first present a construction assuming a coding scheme that satisfies randomness simulatable non-malleability (RSS-NM), against affine tampering functions, and (2) we show how to construct such a code. Finally, we present Corollary 5.7 to summarize our overall construction, by putting all things together in a single statement. As we have already discussed on the introduction, the idea of constructing a NM-code for affine functions, as an intermediate step for providing split-state codes, was also followed by [4], still, our technique differs significantly, and their code does not directly satisfy our requirements. Moreover, in [22] the authors construct AMD codes, still their notions are slightly different and do fit in our framework.

5.1 1-more extractable hash functions from RSS-NM codes against affine functions

In this section we construct a collision resistant, 1-more extractable hash function family. Before doing so, we present the notion of *"randomness simulatable, strongly non-malleable codes"*(RSS-NMC). This notion is stronger than strong non-malleability in the sense that besides simulating the pre-image, $\tilde{s}$, of the tampered codeword, the simulator also needs to produce the randomness of the encoder, $\tilde{s}_{\mathsf{r}}$, such that the encoding of $\tilde{s}$ with randomness $\tilde{s}_{\mathsf{r}}$, produces the tampered codeword. To ease the presentation of RSS-NMC, we modify the syntax of non-malleable codes, so that the Dec algorithm returns, not only the decoded message $\tilde{s}$, but also the

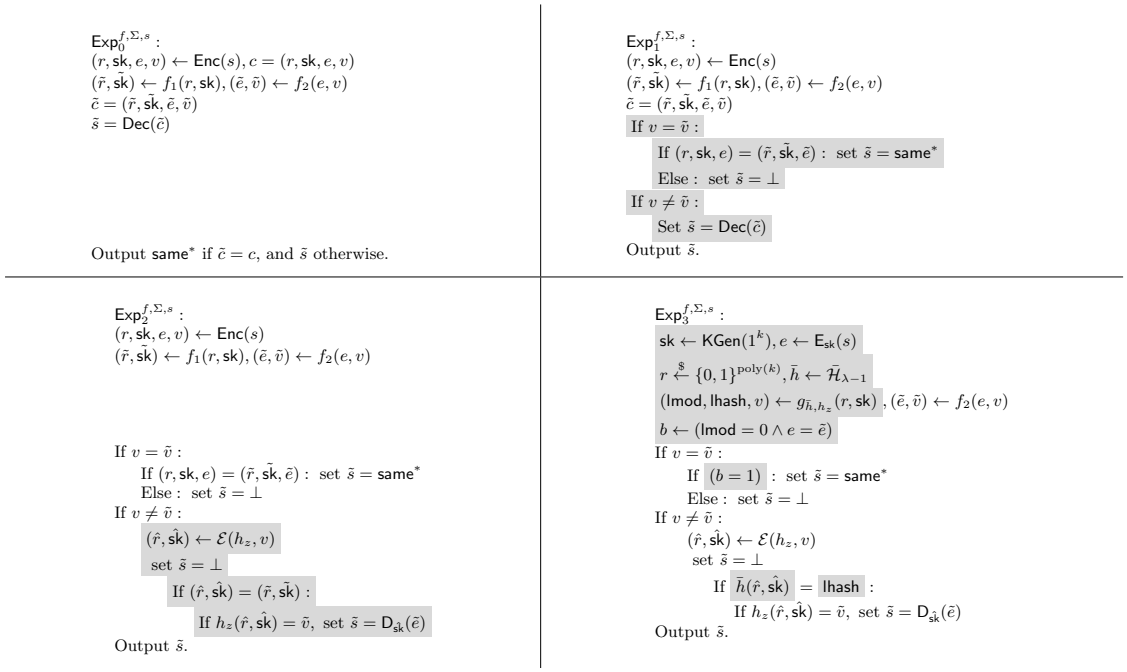

Figure 1: Hybrid experiments for the proof of Theorem 4.2. Their programs are based on $(\mathsf{Enc}, \mathsf{Dec})$, the encoding scheme, $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$ the encryption scheme, and $\mathcal{E}$, the extractor that is specified in the proof. The gray part signifies the portion of the code that differs from the previous experiment.

randomness string $\tilde{s}_\mathsf{r}$ for the encoder Enc. This is the string that in the tampering experiment the simulator should be able to match.[8]

Definition 5.1. (randomness simulatable, strongly non-malleable code) *Let $(\mathsf{Enc}, \mathsf{Dec})$ be a (κ, ν)-coding scheme and $\mathcal{F}$ be a family of functions $f : \{0,1\}^\nu \to \{0,1\}^\nu$. For every $f \in \mathcal{F}$ and $s \in \{0,1\}^\kappa$, define the tampering experiment*

$$\mathsf{Tamper}_s^f \stackrel{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(s), \tilde{c} \leftarrow f(c), (\tilde{s}_\mathsf{r}, \tilde{s}) = \mathsf{Dec}(\tilde{c}) \\ \textit{Output } \mathsf{same}^* \textit{ if } \tilde{c} = c, \textit{ and } (\tilde{s}_\mathsf{r}, \tilde{s}) \textit{ otherwise.} \end{array} \right\}$$

*which is a random variable over the randomness of Enc and Dec. A coding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is randomness simulatable, strongly non-malleable (*RSS-NM*), with respect to the function family $\mathcal{F}$, if for every $f \in \mathcal{F}$ and any s_0, $s_1 \in \{0,1\}^\kappa$, we have:*

$$\left\{\mathsf{Tamper}_{s_0}^f\right\}_{k\in\mathbb{N}} \approx \left\{\mathsf{Tamper}_{s_1}^f\right\}_{k\in\mathbb{N}}$$

where "$\approx$" may refer to statistical, or computational, indistinguishability. For coding schemes in the common reference string model, the definition is analogous.

Next we present our construction:

Construction 5.2 (1-more extractable hash). *Let $\mathcal{G}$ be a group-generation algorithm and let $(\mathsf{Enc}, \mathsf{Dec})$ be a (kt, kt')-coding scheme, $t, t' = O(\mathrm{poly}(k))$. An instance of a $(kt, 2k)$-compressing hash function family $\mathcal{H} = (\mathsf{Gen}, h)$ is defined as follows:*

1. $\mathsf{Gen}(1^k)$*: sample $(\mathbb{G}, g, p) \leftarrow \mathcal{G}(1^k)$, $(a, \mathbf{r}) \xleftarrow{\$} \mathbb{Z}_p \times \mathbb{Z}_p^{t'}$, where $p = |\mathbb{G}|$, and output $z = (\mathbb{G}, g^\mathbf{r}, g^{a\mathbf{r}})$.*
2. ***Hashing computation**: on input $\mathbf{s} = (s_1, \ldots, s_t)$, sample $\mathbf{s}_\mathsf{r} \xleftarrow{\$} U_{\{0,1\}^{\mathrm{poly}(k)}}$, compute $h_z(\mathbf{s}_\mathsf{r}, \mathbf{s}) = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{\langle a\mathbf{r}, \mathbf{c} \rangle})$, where $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s}_\mathsf{r}, \mathbf{s})$.*

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

In the following we prove that Construction 5.2, which is a composition of a coding scheme $(\mathsf{Enc}, \mathsf{Dec})$, with construction 3.3 (the 0-more extractable hash function by Bitansky et al. [10]), is collision resistant, 1-more extractable, and uniform under leakage, assuming that $(\mathsf{Enc}, \mathsf{Dec})$, satisfies certain properties. Then, in Section 5.2, we instantiate $(\mathsf{Enc}, \mathsf{Dec})$ with the desired properties. Below, we prove that Construction 5.2 is collision resistant.

Lemma 5.3. *Let $\mathcal{G}$ be any group generation algorithm. Then, assuming the hardness of the discrete logarithm problem on $\mathcal{G}$, and the underlying encoding algorithm is injective, Construction 5.2 is collision resistant with respect to $\mathcal{G}$.*

Proof. The work [10] proves that the hash function family of Construction 3.3, i.e., $\mathcal{H}^*$, is collision resistant, assuming the difficulty of the discrete logarithm problem. We note that Construction 5.2 is a composition of $\mathsf{Enc}(\cdot)$ and $\mathcal{H}^*$. Following a simple fact that *any injective function composed with a collision resistant hash function still results in a collision resistant hash function* (composition in any order), we can conclude that the hash function family of Construction 5.2 is collision resistant, under the same assumption. ■

In the following theorem, we prove that, under certain assumptions, Construction 5.2, is 1-more extractable.

[8] It is possible to define RSS-NMC without modifying the operation of Dec at the expense of slightly complicating the definition of non-malleability. Due to the fact that our RSS-NMC construction conforms to the modified syntax, we opt for the simpler alternative.

THEOREM 5.4. *Let $t(k), t'(k) = O(\text{poly}(k))$, $(\mathsf{Enc}, \mathsf{Dec})$ be any* RSS*-non-malleable, (kt, kt')-coding scheme, against $\mathcal{F}_{\mathsf{aff}}$, let $\mathcal{H}$ be the hash function family of Construction 5.2 with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, and assume that for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$. Then, assuming t'-*KEA *and the hardness of* DLog*, $\mathcal{H}$ is* 1*-more extractable, with respect to $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$.*

PROOF. For $k \in \mathbb{N}$, let $(\mathsf{Enc}, \mathsf{Dec})$ be an RSS-NM, (kt, kt')-coding scheme, against $\mathcal{F}_{\mathsf{aff}}$, $t(k), t'(k) = O(\text{poly}(k))$, and let $\mathcal{H}$ be the $(kt, 2k)$-compressing, collision-resistant, hash function family of Construction 5.2. Following Definition 3.1, we need to prove that for any PPT algorithm $\mathcal{A}_v$ with auxiliary input $\mathsf{aux}_{\mathcal{A}_v}$, there exist extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$ and auxiliary input $\mathsf{aux}_{\mathcal{E}}$, such that for any PPT algorithm $\mathcal{A}_s$, any large k and every message $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{Z}_p^t$,

$$\Pr_{h_z \leftarrow \mathcal{H}_k}\left[\mathsf{Exp}^{\mathbf{s}, h_z}_{\mathcal{A}_v, \mathcal{A}_s, \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, \mathsf{aux}_{\mathcal{A}_v}, \mathsf{aux}_{\mathcal{E}}) = 1\right] \leq \mathsf{negl}(k). \quad (1)$$

Clearly, if $\mathcal{A}_v$ fails to produce a new valid hash, or, if $\mathcal{A}_s$ fails to produce a valid pre-image for the new hash, the experiment simply outputs 0, and there is no challenge for the extractor. Therefore, the interesting case is when $\mathcal{A}_v$ produces a valid hash value, say $\tilde{v}$, while having access to an element in the range of the hash, say v, and $\mathcal{A}_s$ produces a valid pre-image for $\tilde{v}$, while having access to $\mathbf{s}$, v, $\tilde{v}$, and any other state information produced by $\mathcal{A}_v$. Hence, for the rest of the proof we assume $\tilde{v} \neq v$, and $(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}})$ is a valid pre-image for $\tilde{v}$, i.e., $h_z(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) = \tilde{v}$.

Given any $\mathcal{A}_v$ with auxiliary input $\mathsf{aux}_{\mathcal{A}_v}$, the idea behind the definition of the extractor, $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$, and its auxiliary input, $\mathsf{aux}_{\mathcal{E}}$, goes as follows:

- First we define an adversary against the hash function family $\mathcal{H}^*$, of Construction 3.3: $\bar{\mathcal{A}}_v(h^*_{z'}, \mathsf{aux}_{\mathcal{A}_v}) := \mathcal{A}_v(h_z, v, \mathsf{aux}_{\mathcal{A}_v})$, where $\bar{\mathcal{A}}_v$ first interprets the description of the hash function $h^*_{z'}$, as (h_z, v), i.e., as a description of a hash function in $\mathcal{H}$ and a hash value v, and then executes $\mathcal{A}_v(h_z, v, \mathsf{aux}_{\mathcal{A}_v})$. The function $h^*_{z'}$ will be stated concretely below.
- Since $\mathcal{H}^*$ is a 0-more extractable hash function family, and assuming $h^*_{z'}$ is indistinguishable from an element in $\mathcal{H}^*$, there exists an extractor $\bar{\mathcal{E}}^{\mathcal{H}^*}_{\bar{\mathcal{A}}_v}$ with its auxiliary input $\mathsf{aux}_{\bar{\mathcal{E}}}$, that extracts a valid pre-image for $\tilde{v}$, with respect to $h^*_{z'}$. We define the auxiliary input $\mathsf{aux}_{\mathcal{E}} := \mathsf{aux}_{\bar{\mathcal{E}}}$.

The extractor is defined below.

The extractor $\mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}$:

Input: $(z = (g^{\mathbf{r}}, g^{a\mathbf{r}}), v = (g^{r'}, g^{ar'}), \mathsf{aux}_{\mathcal{E}})$.

1. Set $z' = (g^{\mathbf{r}}, g^{r'}, g^{a\mathbf{r}}, g^{ar'})$. Here, we interpret z' as a description of hash function $h^*_{z'} \in \mathcal{H}^*$, for vector messages with $t' + 1$ coordinates.
2. Sample $(b_1, \ldots, b_{t'}, d) \leftarrow \bar{\mathcal{E}}^{\mathcal{H}^*}_{\bar{\mathcal{A}}_v}(h^*_{z'}, \mathsf{aux}_{\mathcal{E}})$ and set $f = (b_1, \ldots, b_{t'}, d) = (\mathbf{b}, d) \in \mathbb{Z}_p^{t'+1}$.
3. Interpret f as an affine function that on input $(x_1, \ldots x_{t'})$ outputs $(dx_1 + b_1, dx_2 + b_2, \ldots, dx_{t'} + b_{t'})$, and then sample $(\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}}) \leftarrow D^{\mathsf{aff}}_f$, where D^{aff}_f is the simulator of the underlying RSS-NM code, $(\mathsf{Enc}, \mathsf{Dec})$, parameterized by the affine function f.
4. **Output**: $(\hat{\mathbf{s}}_{\mathsf{r}}, \hat{\mathbf{s}})$.

The extractor is defined with respect to any input v, still by the definition of the ℓ-more experiment, v is always a valid hash value, i.e., $v = h_z(\mathbf{s}) = (g^{\langle \mathbf{r}, \mathbf{c} \rangle}, g^{a\langle \mathbf{r}, \mathbf{c} \rangle})$, where $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{s}_{\mathsf{r}}, \mathbf{s})$, for some message $\mathbf{s}$. Then, for any $\mathcal{A}_s$, and message $\mathbf{s}$, we are going to analyze the execution of $\mathsf{Exp}^{\mathbf{s}, h_z}_{\mathcal{A}_v, \mathcal{A}_s, \mathcal{E}^{\mathcal{H}}_{\mathcal{A}_v}}(1, \mathsf{aux}_{\mathcal{A}_v}, \mathsf{aux}_{\mathcal{E}})$. We first prove that with overwhelming probability, the following events happen:

- E_1: $h^*_{z'}(b_1, \ldots, b_{t'}, b_0) = \tilde{v}$. Recall that $\tilde{v}$ is the output of $\mathcal{A}_v$ on input (h_z, v).
- E_2: $\mathsf{Enc}(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}}) = f(\mathbf{c})$. Recall that $(\tilde{\mathbf{s}}_{\mathsf{r}}, \tilde{\mathbf{s}})$ is the output of $\mathcal{A}_s$.

We formalize those ideas in a series of claims, where their concrete proofs are given in the full version of the paper. ■

5.2 Constructing RSS-NM codes

In this section, we construct RSS-NM codes as required by the previous section.

CONSTRUCTION 5.5 (THE CODE FOR $\mathcal{F}_{\mathsf{aff}}$). *For any $k \in \mathbb{N}$, $t = O(\text{poly}(k))$, we define a $(kt, (2t+4)k)$-coding scheme $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ in the CRS model,*[9] *as follows:*

- $\mathsf{Init}(1^k)$*: sample a k-bit prime $p \in (2^{k-1}, 2^k)$ and set $\Sigma = p$.*
- $\mathsf{Enc}(\Sigma, \cdot)$*: let $\mathbf{s} = (s_1, \ldots, s_t) \in \mathbb{F}_p^t$ be the input to* Enc*. Sample two random field elements $v, r \xleftarrow{\$} \mathbb{F}_p$, and then output $\mathbf{c} = \left(v, v^2, r, r^2, u_1, u_1^2, \ldots, u_t, u_t^2\right) \in \mathbb{F}_p^{2t+4}$, where $u_i = s_i - r$ for $i \in [t]$.*
- $\mathsf{Dec}(\Sigma, \cdot)$*: on input $\mathbf{c} = (v, \bar{v}, r, \bar{r}, u_1, \bar{u}_1, \ldots, u_t, \bar{u}_t)$, the decoder checks whether $\bar{v} = v^2$, $\bar{r} = r^2$, and $\bar{u}_i = u_i^2$ for all $i \in [t]$. If so, then it outputs $(v, r, u_1 + r, u_2 + r, \ldots, u_t + r)$, otherwise, outputs $\perp$.*

All operations are performed modulo p. We also consider the deterministic version of Enc *by allowing the randomness to be given on the input. In that case we have $\mathbf{c} = \mathsf{Enc}(\Sigma, \mathbf{s}_{\mathsf{r}}, \mathbf{s})$, where $\mathbf{s}_{\mathsf{r}} = (v, r)$.*

Notice, that, the randomness employed by the above construction is $2k$, independently of the message length.

THEOREM 5.6. *The code of Construction 5.5 is randomness simulatable, strongly non-malleable (Definition 5.1), with respect to $\mathcal{F}_{\mathsf{aff}}$. In addition, for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$.*

The proof intuition is presented in the introduction and the concrete proof is given in the full version of this paper.

5.3 Our resulting instantiation

By plugging Construction 5.5, as the underlying coding scheme to Construction 5.2, we receive the corollary:

COROLLARY 5.7. *Under the* DLog *assumption and t-KEA, there exists a* 1*-more extractable, collision resistant, hash function family $\mathcal{H}_k$.*

PROOF. Let $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ be the $(kt, (2t+4)k)$, RSS-NM code of Construction 5.5. Then we construct $\mathcal{H}_k$ by plugging in $(\mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$, as the underlying coding scheme to the hash function family of Construction 5.2. Clearly,

[9] Note that the CRS is not essential for this encoding, but for simplicity we describe the code in this model.

by Lemma 5.3, $\mathcal{H}_k$ is collision resistant as the underlying encoding algorithm is injective. By Theorem 5.6, the underlying coding scheme is RSS-non-malleable against $\mathcal{F}_{\mathsf{aff}}$, and moreover, for any message $\mathbf{s}$, $H_\infty(\mathsf{Enc}(\mathbf{s})) \geq k + \omega(\log k)$. Thus, by Theorem 5.4, $\mathcal{H}_k$ is 1-more extractable. This concludes the proof of this corollary. ■

6. CONSTRUCTING ℓ-MORE EXTRACTABLE HASH

In the "ℓ-more" setting, the attacker is given $v_1, \ldots, v_\ell$, hash values, and produces a new hash value $\tilde{v}$. Having the techniques from the "1-more" setting, one can easily argue the attack against $\tilde{v}$ (in the ℓ-more setting), can be reduced to an affine attack against the codewords $c_1, \ldots, c_\ell$, that are related to $v_1, \ldots, v_\ell$, respectively. In order to construct ℓ-more ECRH, for $\ell > 1$, we generalize the notion of RSS-NM codes, for multiple codewords. The generalization is a straightforward extension of Definition 5.1, where the tampering function receives ℓ codewords and the simulator needs to recover the message and randomness in case the output of the tampering function is not among the given codewords. The formal is definition is given in the full version.

Clearly, for $\ell = 1$, the notion of multi-codeword RSS-NMC matches Definition 5.1. In order to construct, ℓ-more ECRH, for $\ell > 1$, we need an RSS-NM code, for the following function class.

DEFINITION 6.1 (THE FUNCTION CLASS $\bar{\mathcal{F}}^{\ell}_{\mathsf{aff}}$). *We define the following function class*

$$\bar{\mathcal{F}}^{\ell}_{\mathsf{aff}} = \{f(x_1, \ldots, x_\ell) = f_1(x_1) + \ldots + f_\ell(x_\ell) \mid f_i \in \mathcal{F}_{\mathsf{aff}}\}.$$

We present the following lemma.

LEMMA 6.2. *The code of Construction 5.5,* $(\mathsf{Enc}, \mathsf{Dec})$*, is a multi-codeword* RSS-NM *code against* $\bar{\mathcal{F}}^{\ell}_{\mathsf{aff}}$*, for* $\ell > 1$.

PROOF. A proof sketch is given in the full version. ■

In the "ℓ-more" setting the attacker receives $v_i = (g^{\langle \mathbf{r}, \mathbf{c}_i \rangle}, g^{a\langle \mathbf{r}, \mathbf{c}_i \rangle})$, $i \in [\ell]$, and constructs a valid hash $\tilde{v}$. The proof of Theorem 5.4 easily extends to the "ℓ-more" setting by proving that $\tilde{v} = g^{\langle \mathbf{r}, \sum_{i=1}^{\ell} f_i(\mathbf{c}_i) \rangle}, g^{a\langle \mathbf{r}, \sum_{i=1}^{\ell} f_i(\mathbf{c}_i) \rangle})$, where $(f_1, \ldots, f_\ell) \in \bar{\mathcal{F}}^{\ell}_{\mathsf{aff}}$, and we achieve extractability using the simulator of the underlying, RSS-NMC for multiple codewords. Thus, we are able to show the following theorem.

THEOREM 6.3. *Under the* DLOG *assumption and t-KEA, Construction 5.2, instantiated with the Construction 5.5, is an ℓ-more extractable hash function family.*

The proof is essentially the same as that of Theorem 5.4, as we discussed above.

7. INSTANTIATING AUTHENTICATED ENCRYPTION

In the following we instantiate one-time leakage-resilient, authenticated, semantically secure symmetric encryption (Definition 2.9), against λ bits of leakage. The idea is to combine a leakage-resilient pseudorandom generator [46] with a message authentication code that outputs k bits.

CONSTRUCTION 7.1 (AUTHENTICATED ENCRYPTION). *Let* PRG *be a pseudo-random generator, i.e.,* $\mathsf{PRG} : \{0,1\}^{2\lambda} \to \{0,1\}^{|s|+k}$*, and let* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a message authentication code that outputs tags of length k (cf. [39]). We define a symmetric encryption scheme* $(\mathsf{KGen}, \mathsf{E}, \mathsf{D})$*, as follows:*

- $\mathsf{KGen}(1^k)$*: sample* $\mathsf{sk} \xleftarrow{\$} \{0,1\}^{2\lambda}$.
- $\mathsf{E}_{\mathsf{sk}}(\cdot)$*: On input message s, compute* $(r_0, r_1) = \mathsf{PRG}(\mathsf{sk})$*, where* $|r_0| = |s|$ *and* $|r_1| = k$*,* $e = r_0 + s$*,* $t = \mathsf{Mac}_{r_1}(e)$*, and outputs* (e, t).
- $\mathsf{D}_{\mathsf{sk}}(\cdot)$*: On input* (e, t)*, compute* $(r_0, r_1) = \mathsf{PRG}(\mathsf{sk})$*, and if* $\mathsf{Vrfy}_{r_1}(e, t) = 1$*, output* $s = r_0 - e$*, otherwise output* $\perp$.

The PRG of [46] considers $|\mathsf{sk}| = 2\lambda/\alpha$, and sustains $\alpha\lambda$ bits of leakage (cf. [50]), where $\alpha \in [0, 1]$ depends on how strong the underlying assumption is. In the above construction we use the strongest assumption, i.e., $\alpha = 1$, which yields $|\mathsf{sk}| = 2\lambda$. The ciphertext length is $|s|+k$, and by setting $\lambda = 2k + \log^2 k$, which is adequate for our needs, we receive $|\mathsf{sk}| + |e| + |t| = 5k + 2\log^2 k + |s|$. In the full version we analyze the above construction and we provide instantiations that use weaker assumptions. By plugging the above instantiation to our split-state non-malleable code, the total codeword length is $|s| + 9 \cdot k + 2 \cdot \log^2(k)$, since the hash and the randomness for computing it, are of size $2k$, each.

8. REFERENCES

[1] M. Abe and S. Fehr. Perfect nizk with adaptive soundness. In *TCC*, pages 118–136, 2007.

[2] D. Aggarwal, S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran. Optimal computational split-state non-malleable codes. Cryptology ePrint Archive, Report 2015/1063, 2015.

[3] D. Aggarwal, Y. Dodis, T. Kazana, and M. Obremski. Non-malleable reductions and applications. In *STOC*, pages 459–468, 2015.

[4] D. Aggarwal, Y. Dodis, and S. Lovett. Non-malleable codes from additive combinatorics. In *STOC*, pages 774–783, 2014.

[5] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran. *CRYPTO*, chapter Explicit Non-malleable Codes Against Bit-Wise Tampering and Permutations, pages 538–557. 2015.

[6] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin. *Advances in Cryptology – EUROCRYPT 2016*, chapter Non-malleable Codes for Bounded Depth, Bounded Fan-In Circuits. 2016.

[7] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18. 2001.

[8] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, pages 273–289. 2004.

[9] 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.

[10] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[11] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

[12] N. Bitansky, R. Canetti, O. Paneth, and A. Rosen. On the existence of extractable one-way functions. In *STOC*, pages 505–514, 2014.

[13] E. Boyle and R. Pass. Limits of extractability assumptions with distributional auxiliary input. In *ASIACRYPT*, pages 236–261. 2015.

[14] R. Canetti and R. Dakdouk. Extractable perfectly one-way functions. In *Automata, Languages and Programming*, pages 449–460. 2008.

[15] R. Canetti and R. Dakdouk. Towards a theory of extractable functions. In *TCC*, pages 595–613. 2009.

[16] N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, and J. Upadhyay. Block-wise non-malleable codes. *IACR Cryptology ePrint Archive*, page 129, 2015.

[17] N. Chandran, B. Kanukurthi, and S. Raghuraman. *Information-Theoretic Local Non-malleable Codes and Their Applications*, pages 367–392. TCC 2016-A. 2016.

[18] E. Chattopadhyay and D. Zuckerman. Non-malleable codes against constant split-state tampering. In *FOCS*, pages 306–315, 2014.

[19] M. Cheraghchi and V. Guruswami. Capacity of non-malleable codes. In *ITCS*, pages 155–168, 2014.

[20] S. G. Choi, A. Kiayias, and T. Malkin. Bitr: Built-in tamper resilience, 2011.

[21] S. Coretti, U. Maurer, B. Tackmann, and D. Venturi. *TCC*, chapter From Single-Bit to Multi-bit Public-Key Encryption via Non-malleable Codes, pages 532–560. 2015.

[22] R. Cramer, Y. Dodis, S. Fehr, C. Padró, and D. Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT*, pages 471–488. 2008.

[23] D. Dachman-Soled, F.-H. Liu, E. Shi, and H.-S. Zhou. *TCC 2015*, chapter Locally Decodable and Updatable Non-malleable Codes and Their Applications, pages 427–450. 2015.

[24] R. R. Dakdouk. Theory and application of extractable functions, 2009.

[25] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456. 1992.

[26] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598. 2001.

[27] A. Dent and S. Galbraith. Hidden pairings and trapdoor ddh groups. In *Algorithmic Number Theory*, pages 436–451. 2006.

[28] S. Dziembowski, T. Kazana, and M. Obremski. Non-malleable codes from two-source extractors. In *CRYPTO*, pages 239–257. 2013.

[29] S. Dziembowski, K. Pietrzak, and D. Wichs. D.: Non-malleable codes. In *ICS*, 2010.

[30] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. *TCC 2014*, chapter Continuous Non-malleable Codes, pages 465–488. 2014.

[31] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. *PKC 2015*, chapter A Tamper and Leakage Resilient von Neumann Architecture, pages 579–603. 2015.

[32] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs. *EUROCRYPT*, chapter Efficient Non-malleable Codes and Key-Derivation for Poly-size Tampering Circuits, pages 111–128. 2014.

[33] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. *EUROCRYPT '13*, chapter Quadratic Span Programs and Succinct NIZKs without PCPs, pages 626–645. 2013.

[34] R. Gennaro, H. Krawczyk, and T. Rabin. Okamoto-tanaka revisited: Fully authenticated diffie-hellman with minimal overhead. In *ACNS*, pages 309–328. 2010.

[35] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. Cryptology ePrint Archive, Report 2010/610, 2010.

[36] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340. 2010.

[37] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

[38] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In *CRYPTO '98*, pages 408–423. 1998.

[39] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. 2007.

[40] F.-H. Liu and A. Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532. 2012.

[41] T. Mie. Polylogarithmic two-round argument systems, 2008.

[42] M. Naor. *CRYPTO '03*, chapter On Cryptographic Assumptions and Challenges, pages 96–109. 2003.

[43] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. *SIAM Journal on Computing*, pages 772–814, 2012.

[44] E. Okamoto and K. Tanaka. Key distribution system based on identification information. *Selected Areas in Communications, IEEE Journal on*, pages 481–485, 1989.

[45] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *Security and Privacy*, pages 238–252, 2013.

[46] K. Pietrzak. *Advances in Cryptology - EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, chapter A Leakage-Resilient Mode of Operation. 2009.

[47] M. Prabhakaran and R. Xue. Statistically hiding sets. In *CT-RSA*, pages 100–116. 2009.

[48] A. Sahai. Simulation-sound non-interactive zero knowledge. Technical report, IBM RESEARCH REPORT RZ 3076, 2001.

[49] A. R. Shafi Goldwasser, Huijia Lin. Delegation of computation without rejection problem from designated verifier cs-proofs. Cryptology ePrint Archive, Report 2011/456, 2011.

[50] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. chapter Leakage Resilient Cryptography in Practice, pages 99–134. 2010.