Continuously Non-Malleable Codes in the Split-State Model from Minimal Assumptions*

Rafail Ostrovsky[1], Giuseppe Persiano[2], Daniele Venturi[3], and Ivan Visconti[4]

[1] Computer Science Department, University of California Los Angeles, USA
[2] DISA-MIS, University of Salerno, Italy
[3] Computer Science Department, Sapienza University of Rome, Italy
[4] DIEM, University of Salerno, Italy

Abstract. At ICS 2010, Dziembowski, Pietrzak and Wichs introduced the notion of *non-malleable codes*, a weaker form of error-correcting codes guaranteeing that the decoding of a tampered codeword either corresponds to the original message or to an unrelated value. The last few years established non-malleable codes as one of the recently invented cryptographic primitives with the highest impact and potential, with very challenging open problems and applications.

In this work, we focus on so-called *continuously* non-malleable codes in the split-state model, as proposed by Faust *et al.* (TCC 2014), where a codeword is made of two shares and an adaptive adversary makes a polynomial number of attempts in order to tamper the target codeword, where each attempt is allowed to modify the two shares independently (yet arbitrarily). Achieving continuous non-malleability in the split-state model has been so far very hard. Indeed, the only known constructions require strong setup assumptions (i.e., the existence of a common reference string) and strong complexity-theoretic assumptions (i.e., the existence of non-interactive zero-knowledge proofs and collision-resistant hash functions).

As our main result, we construct a continuously non-malleable code in the split-state model without setup assumptions, requiring only one-to-one one-way functions (i.e., essentially optimal computational assumptions). Our result introduces several new ideas that make progress towards understanding continuous non-malleability, and shows interesting connections with protocol-design and proof-approach techniques used in other contexts (e.g., look-ahead simulation in zero-knowledge proofs, non-malleable commitments, and leakage resilience).

* Research supported in part by "GNCS - INdAM", FARB 300392FRB15VISCO, NSF grant 1619348, DARPA SafeWare subcontract to Galois Inc., DARPA SPAWAR contract N66001-15-1C-4065, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the authors and do not reflect position of the Department of Defense or the U.S. Government. Work partially done while the second and fourth authors were visiting UCLA. The final authenticated version is available online at https://doi.org/10.1007/978-3-319-96878-0_21

Keywords: continuously non-malleable codes, split-state model, minimal assumptions.

1 Introduction

Dziembowski, Pietrzak and Wichs introduced the notion of a non-malleable code (NMC) in [27]. Their new notion generated tremendous interest in recent years both for the challenging theoretical questions raised by such codes, and for their interesting applications in cryptography. An NMC is a key-less procedure that allows to encode a message m in such a way that, upon input the encoding of m, it is not possible (or it is hard in the computational case) to produce an encoding of a value related to m.[5]

Obviously an NMC requires some restrictions on the view of the adversary. Indeed, as the encoding/decoding are key-less procedures, an adversary could always decode a codeword, change the underlying message to a related value, and encode the result. For this reason, non-malleability is typically parameterized by the set of allowed modifications Φ that can be applied by the adversary to a target encoding, and previous work on NMCs focused on constructing non-malleable codes for restricted (yet meaningful) classes Φ.

The split-state model. One of the most natural and investigated models is to assume that a codeword c consists of two shares $c = (c_0, c_1)$, and that each tampering attempt $\phi = (\phi_0, \phi_1) \in \Phi$ is characterized by two arbitrary functions that can be applied to each share independently. Note that the two tampering functions cannot run the decoding procedure, because both shares are needed in order to decode a codeword, whereas each of the functions ϕ_0, ϕ_1 can access only one share. This setting is often called the *split-state model* and is the focus of this paper; we often use the terminology *split-state code* to denote a code in the split-state model. We refer the reader to §1.5 for an overview of known constructions of non-malleable codes for different classes Φ. Previous work showed how to construct split-state non-malleable codes, both for the information-theoretic setting [27,25,4,18,6,3,13,7,37] and the computational setting [38,30,22,2].

1.1 Continuous Non-Malleability

The original notion of NMCs provides a security guarantee only against adversaries that try to tamper the codeword once. The more general case of *continuous non-malleability* was introduced by Faust *et al.* [30], with the goal of guaranteeing non-malleability even after multiple (adaptively chosen) tampering attempts; that is, the adversary is allowed to choose the tampering functions to apply in the next round based on the answers obtained in the previous rounds. As pointed out also in [30], continuously non-malleable codes (CNMCs) are arguably the most natural generalization of standard NMCs, and allow to significantly strengthen their applications [31,20,19].

[5] In this paper, we will only focus on *efficient* NMCs where both the encoding and decoding procedures run in polynomial time.

Different flavors of non-malleability. The work of [27] considered a default and a strong flavor of non-malleability. In both cases, the adversary is allowed to see the decoding $\tilde{m}$ of the modified codeword $\tilde{c} = \phi(c)$. However, the default notion only guarantees non-malleability as long as the decoded message is different from the original message, i.e. it might be possible for the attacker to create an encoding $\tilde{c} \neq c$ such that $\tilde{c}$ still decodes to the original message m. In contrast, this is not allowed in the case of *strong* non-malleability which guarantees that whenever $\tilde{c} \neq c$ the decoded value $\tilde{m}$ will be unrelated to m. An even stronger flavor, known as *super* non-malleability [30,32,36], ensures that $\tilde{c}$ is independent of c whenever $\tilde{c} \neq c$ is a valid codeword. This is modeled by allowing the adversary to actually see $\tilde{c}$ (as long as $\tilde{c} \neq c$ and $\tilde{c}$ is valid).

Clearly, the above flavors of non-malleability can also be considered in the continuous setting. In this paper, we focus only on the "default flavor" of continuous non-malleability. This in contrast to previous work on continuously non-malleable codes (except [20,19,28]), which instead by default considered continuous super non-malleability. While the notion we consider is strictly weaker than continuous strong or super non-malleability, to the best of our knowledge, it is sufficient for all known applications of continuously non-malleable codes, in particular [30,31,20,19,28].

Depending on the tampering functions being applied always to the initial encoding c, or to the result of the previous tampering attempt, one can also have notions called *non-persistent* and *persistent* tampering. In this paper we focus on the setting of non-persistent tampering, which is the strongest[6] flavor of continuous non-malleability (and also the variant most useful for applications).

Self destruction. Unfortunately, even for very simple classes Φ, continuous non-malleability as hinted above is actually impossible to achieve. Indeed, a simple attack—proposed for the first time by Gennaro *et al.* [33] in the context of "Algorithmic Tamper-Proof Security"—allows to completely recover a target encoding by simply trying to guess each of its bits individually: the output of the decoding corresponding to each tampering attempt will yield either the original message or the special symbol $\perp$ (denoting an invalid codeword), thus revealing the entire codeword (and thus the underlying message) in a bit-wise fashion. Remarkably, such an attack can be performed by looking at each bit of the encoding independently, which is a special case of split-state tampering.

The standard way out is to relax continuous non-malleability, therefore circumventing the above impossibility result, assuming a special "self-destruct" feature: After the first invalid encoding is processed, the system "blows-up" and stops processing further queries.[7]

[6] In fact, note that a persistent continuous attack specified as a sequence of (deterministic) tampering functions $\phi, \phi', \phi'', \cdots$, can always be emulated by a non-persistent continuous attack specified as $\phi(\cdot), \phi'(\phi(\cdot)), \phi''(\phi'(\phi(\cdot))), \cdots$.

[7] In practice self-destruct could be implemented using a single (untamperable) bit of public state, or by having the device overwrite its own memory in case of an invalid encoding.

Message uniqueness. It is not hard to show that any code achieving continuous non-malleability in the split-state model must satisfy a property called *message uniqueness*. Informally, message uniqueness means that if we fix the left share c_0 of an encoding, it should be hard to come up with two distinct right shares $c_1, \bar{c}_1$ such that both (c_0, c_1) and $(c_0, \bar{c}_1)$ are valid codewords decoding to two *distinct* messages, say m and $\bar{m}$ respectively.[8] (An analogous guarantee must hold in case we fix the right share.)

To see why uniqueness is needed, assume it is possible to efficiently find two encodings (c_0, c_1) and $(c_0, \bar{c}_1)$ violating message uniqueness, and let (c_0^*, c_1^*) be the target encoding that we want to maul via a split-state attack. Then, in a continuous attack, we can simply consider the tampering functions $(\phi_0^{(i)}, \phi_1^{(i)})$ that always fix the left share to c_0 (regardless of c_0^*) and, depending on the i-th bit of c_1^* either overwrite c_1^* with c_1 or with $\bar{c}_1$. The sequence of decoded messages produced by such an attack allows an adversary to recover c_1^* without the risk of incurring a self-destruct. After c_1^* is available, an additional tampering query easily allows to encode a related value.[9]

The state of the art: trusted setup and strong computational assumptions. The attack based on uniqueness implies that information-theoretic continuous non-malleability in the split-state model is impossible. This is because message uniqueness in the information-theoretic setting means that each share of a split-state encoding must completely determine the message. So, an unbounded tampering function accessing a single share of the codeword could just recover the underlying message by simply brute forcing all possible values for the missing share, and running the decoding algorithm until a valid message is found. Afterwards, it can complete the attack by setting (along with the other tampering function that hardwires correlated randomness and performs the same steps) an encoding of a related message.

The only known constructions of a CNMC in the split-state model (therefore also achieving message uniqueness) are the codes of [30,28], but unfortunately these constructions rely on both trusted setup and strong computational assumptions. Indeed such codes require: (i) a "common reference string", i.e., the existence of a honestly generated string (with some given distribution) that is assumed to be untamperable, and that is available to the encoding and decoding functions, and to the adversary; (ii) the existence of non-interactive zero-knowledge proofs and either collision-resistant hash functions [30] or public-key encryption resilient to continual leakage [24,28] (which we only know how to obtain under concrete number-theoretic assumptions over bi-linear groups).

[8] Since [30] by default considered continuous super non-malleability, they require an even stronger form of uniqueness called *codeword uniqueness*, which intuitively says that it should be hard to find $(c_0, c_1, \bar{c}_1)$ such that both (c_0, c_1) and $(c_0, \bar{c}_1)$ are valid, and $c_1 \neq \bar{c}_1$, even if the two codewords encode the same message. This flavor of uniqueness is not needed in this paper.

[9] Message uniqueness is, instead, not necessary for the simpler case of continuous non-malleability against *persistent* tampering. Split-state codes achieving such a weaker security guarantee were recently constructed unconditionally in [7].

The open problem. Unfortunately, in practical situations, trusted setup is very difficult to come by (and also expensive to implement). Moreover, one should always try to get the best possible security, limiting or avoiding the trust on other parties, on some setup, and on strong computational assumptions. This leads to the following major open question:

> **Q1:** *Can we construct a split-state CNMC under minimal complexity-theoretic assumptions in the plain model (i.e., without trusted setup)?*

Towards the above main question, one might also be interested in the following natural question:

> **Q2:** *Is any split-state code satisfying both message uniqueness and one-time non-malleability also continuously non-malleable?*

1.2 Our Contribution

In this paper we give definitive answers to the above questions. Our main contribution is a positive answer to question **Q1**, therefore providing the first construction of a split-state CNMC (for non-persistent tampering) without assuming any trusted third party, or strong computational assumption. Indeed, we will show that the sole existence of one-to-one one-way functions already suffices for our purpose.

Theorem 1 (Informal). *If one-to-one one-way functions exists, there is a construction of a split-state code that satisfies continuous non-malleability in the* plain model*.*

In addition, we also give a negative answer to question **Q2**. In particular, we show that there exist (albeit contrived) split-state codes that are one-time non-malleable and satisfy message uniqueness, but can be broken by a simple continuous attack.

Theorem 2 (Informal). *If one-to-one one-way function exists, then there is a construction of a split-state code that satisfies both (perfect) message uniqueness and one-time non-malleability in the* plain model*, but that is insecure for* two *tampering queries.*

We notice that the computational assumption that we use is essentially optimal. In fact, each of the two shares of an encoding of a split-state non-malleable code satisfying message uniqueness implicitly defines a non-interactive commitment, and, as shown in [39], there is no black-box construction of a non-interactive commitment scheme from general one-way functions.

1.3 Positive Result

Our positive result introduces new ideas that make progress towards understanding continuous non-malleability, and shows interesting connections with protocol-design and proof-approach techniques used in other contexts (e.g., look-ahead simulation in zero-knowledge proofs [46], non-malleable commitments [43], and leakage resilience [26]). We highlight some of the challenges below.

Hardness of constructing one-time NMCs with message uniqueness. Before describing our encoding scheme, let us give some intuition why the problem of obtaining both non-malleability and message uniqueness in the plain model might be hard to tackle (even using non-standard assumptions). Let $c = (c_0, c_1)$ be a split-state codeword. Since we want to achieve message uniqueness, the left share must completely determine the encoded message; an analogous property must also hold for the right share. We can thus interpret each of the two shares produced by the encoding as a non-interactive perfectly[10] binding commitment. On the other hand, $c = (c_0, c_1)$ must also be non-malleable.

Now, consider the following natural candidate inspired by the recent construction of [12]. We let $c_0 = (\gamma_0, r_1)$ and $c_1 = (\gamma_1, r_0)$, where γ_0 and γ_1 are perfectly binding non-interactive non-malleable commitments of a message m, using randomness r_0 and r_1 (respectively). In the plain model, such commitments can be based on *adaptive* one-way functions [41], and, as shown by Pass [42], they cannot be constructed under falsifiable assumptions (in a black-box sense).

Although, at least intuitively, the above scheme should satisfy both properties of non-malleability and message uniqueness, we now argue that this might be very hard to prove. Recall that the experiment defining one-time non-malleability in the split-state model proceeds as follows: First the adversary chooses two messages m_0, m_1, and then it is allowed to specify a single pair of tampering functions $\phi = (\phi_0, \phi_1)$ that is applied to an encoding $c = (c_0, c_1)$ of m_b, for hidden bit b that the adversary needs to guess, upon which the attacker receives the decoded value corresponding to the tampered codeword. (Unless such value equals one of m_0, m_1, in which case the adversary obtains a special output $\texttt{same}^*$.) Consider the following pair of split-state functions $\phi = (\phi_0, \phi_1)$.

- Function ϕ_0, by looking at γ_0 recovers some of the bits of r_0; function ϕ_1 acts similarly, i.e. it recovers some of the bits of r_1 by looking at γ_1.[11]
- As a consequence, ϕ_0 and ϕ_1 have some shared randomness ρ (coming from the coins of the commitments). Let us now be generous, and further assume that functions ϕ_0, ϕ_1 can recover the encoded value m_b by looking at γ_0 and γ_1 (respectively).[12] Clearly, any reduction basing one-time non-malleability of the above scheme on the assumption that the commitments are non-malleable must in particular work for such a strong split-state attack.
- Finally, the functions (ϕ_0, ϕ_1) use the shared randomness ρ to coordinate as follows: with probability $1/2$ (with common coins derived from ρ) they replace (c_0, c_1) with an encoding $(\tilde{c}_0, \tilde{c}_1)$ of a value $\tilde{m}$ related to m_b (computed using common randomness derived from ρ), and with probability $1/2$ (again with common coins derived from ρ) they replace (c_0, c_1) with uncorrelated

[10] It is easy to see that, in the plain model, computational uniqueness implies *perfect* uniqueness.

[11] The assumption that the tampering functions can recover some bits of the randomness, is justified by the fact that we do not know of any (even non-adaptive) one-way function that hides all the bits of its input.

[12] Note that both functions recover the same value m_b, because the commitments are perfectly binding.

encodings (therefore decoding to $\perp$) of a value $\tilde{m}$ related to m_{1-b}. The decoded message will be related to m_b with probability $1/2$, and to m_{1-b} with probability $1/2$.

The above attack is clearly successful. Consider now the reduction that, given a target commitment γ, samples a random string r, runs $(\tilde{\gamma}, \tilde{r}) = \phi_0(\gamma, r)$, and returns $\tilde{\gamma}$ as mauled commitment. The advantage of such a reduction is zero, as both ϕ_0 and ϕ_1 return either a commitment to a message related to m_b (with probability $1/2$) or a commitment to a message related to m_{1-b} (still with probability $1/2$). It is, thus, not clear how such functions could help in breaking the non-malleability of the commitment scheme.

Message uniqueness and one-time non-malleability via commitments and leakage resilience. Our first idea is to circumvent the problem that the adversary might be able to coordinate ϕ_0 and ϕ_1 using common randomness coming from the commitment, by hiding such randomness. To this end, we make use of a (non-unique) primitive: an auxiliary split-state non-malleable code which encodes the message m concatenated with the randomness r used to compute the commitment. The reason why we can count on this non-unique tool is that in the security proof we can have a first hybrid experiment where we disconnect the randomness r of the commitment from the input of the auxiliary non-malleable code. Next, non-malleability follows by a reduction to the hiding property of the commitment scheme.

Remarkably, our proof works even if the underlying commitment is *malleable*; hence, we can instantiate our construction based on standard cryptographic assumptions, such as the existence of one-to-one one-way functions (which imply standard perfectly binding and computationally hiding non-interactive commitments [34]). Intuitively, the reason is that mauling the commitment does not help, since the message is also input to the auxiliary non-malleable code. The above trick is inspired by a beautiful idea of Pass and Rosen [44,45]. Indeed, they constructed non-malleable commitments by composing regular (i.e., potentially malleable) commitment schemes and non-malleable zero-knowledge arguments of knowledge. One can see our technique as one more (though completely different) application of the Pass-Rosen trick. We stress that despite the common spirit of their and our technique, our construction has to deal with several difficulties that go much beyond the simple use of the above trick.

In order to reduce a successful attack to our code to the security of the inner auxiliary NMC, we need to use the tampering functions (ϕ_0, ϕ_1) chosen by the adversary to define the tampering functions (ϕ'_0, ϕ'_1) against the underlying code. This requires two adjustments: (i) the input to the functions (ϕ'_0, ϕ'_1) must be enriched adding a commitment; (ii) the output of the functions (ϕ_0, ϕ_1) must be shrunk removing the commitment. While the former adjustment is pretty straightforward (indeed it can be accomplished by just hardwiring a commitment and a description of ϕ_0, ϕ_1 in the description of ϕ'_0, ϕ'_1), the latter is more complicated since we can't simply remove the commitment. In fact, the commitments produced by the tampering functions could play an important role for the success of the adversary! This issue will be resolved by additionally assuming

that the inner NMC be a leakage-resilient NMC,[13] which allows us to obtain (via a leakage query) the modified commitment as generated by the tampering functions (ϕ_0, ϕ_1) chosen by the adversary. As we show, this leakage can be used by the distinguisher of the inner auxiliary NMC to simulate consistently the view of the distinguisher attacking the full code, thus reaching a contradiction.

The tough continuous case: we are short on leakage queries! The above technique consisting of using a leakage query to adjust the output of the distinguisher can be applied because the leaked information (i.e., a commitment) is small compared to the size of the codewords, and such a small leakage is tolerated by known constructions.

Consider now a continuous attack, where the adversary picks several tampering functions adaptively. A naive adaptation of the above trick would clearly result in too much leakage, since there is no a-priori fixed bound on the number of tampering queries made by the adversary, and each query requires to leak the corresponding modified commitment. Hence, the proof approach discussed so far fails in the case of a continuous attack.

We overcome this obstacle using two additional ideas: (i) A new proof strategy based on optimistic answers and rewinding simulation exploiting look-ahead threads, and (ii) a special leakage-resilient NMC with unconditional security.

Optimistic answers and simulation through look-ahead threads. Our proof strategy borrows the rewinding simulation used in zero-knowledge proofs, and combines it with optimistic answers in order to save on the overall amount of leakage queries. Recall that the main reason to use a leakage query is to obtain the modified commitments that are part of the tampered codewords produced by the tampering functions chosen by the adversary. Note that, once the commitments are leaked, they can also be decommitted via brute force search, since the goal is now to break *unconditional security* of the underlying leakage-resilient NMC, and therefore the reduction is allowed to run in exponential time.

In order to save on leakage queries, we simulate the answers to the adversary's tampering queries by using an optimistic approach, essentially returning the value that had more chances to be encoded by the tampering functions. Such a value can be computed through brute-force search, by applying the tampering functions to all possible encodings and returning the decoded message that appears more often. This sequence of "simulated" answers can be seen as a look-ahead thread [46], where the reduction tries to understand the correct answers to be played in the main thread of the interaction with the adversary. Indeed, when the adversary stops, the reduction will run a special leakage query in order to learn the first point j of the simulation where the optimistic answer was wrong, and the commitment γ that should have been considered instead. This information implies that all answers up to the j-th query were correct,

[13] Roughly, this means that the code remains non-malleable even given some bounded, independent (yet arbitrary), leakage on the two shares of a target encoding. See §3 for a precise definition. Suitable codes were recently constructed in [6].

therefore the reduction can complete the current lookahead thread, return to the main thread, simulate the answer to the first j queries as before, and decommit γ through brute-force search in order to answer the $(j+1)$-th query. Next, the reduction starts another look-ahead thread, and so on, until all queries have been answered correctly. Through an induction argument, we will show that the reduction can successfully break the underlying one-time NMC by carefully adjusting the last pair of tampering functions chosen by the adversary.

The tricky bit is the following: How do we bound the number of look-ahead threads? Indeed, there is a leakage query for each look-ahead thread, and therefore without bounding the number of threads we can not contradict security of the underlying leakage-resilient NMC.

The small mutual information of [6]. The number of leakage queries is proportional to the number of look-ahead threads, and thus to the number of errors done by the reduction when giving optimistic answers. Hence, it is crucial to study the consequences of a wrong optimistic answer.

Whenever an optimistic answer is wrong, we have that the two tampering functions sent by the adversary modify the target codeword yielding a value that is not the most likely outcome. Intuitively, this means that for each look-ahead thread the adversary risks as decoding the special value $\perp$ (leading to self-destruct) with probability at least $1/2$. In fact, notice that if one tampering function sets a value that is not the most likely one, then with probability at least $1/2$ the other tampering function will set a different value, and therefore the decoding will return $\perp$. Clearly, if the adversary is risking $\perp$ with probability at least $1/2$, the number of such look-ahead threads is at most poly-logarithmic.

While the above argument is intuitively appealing, the difficulty is that the two tampering functions could coordinate their outputs using some correlated information encoded in their inputs. In such a case, they could produce two valid shares that encode a message which is different from the most likely outcome, and still the probability of self-destruct is less than $1/2$. We circumvent this complication by assuming two additional properties of the underlying one-time NMC, namely that the mutual information between the two shares of an encoding is not too high, and further that codewords are uniform over a subset of all possible encodings. Hence, we argue that any such tampering query (yielding a message that is different from the optimistic answer and incurring in a probability of self-destruct less than $1/2$) will cost one bit of correlated information, and thus, after a small number of such queries, the mutual information becomes zero and the probability of $\perp$ for each additional query is at least $1/2$. The latter allows our reduction to succeed.

Finally, we show that the code of [6] satisfies all the properties we need, and moreover, by carefully selecting the parameters, it tolerates enough leakage in order to apply our reduction.

1.4 Negative Result

Since in the split-state model continuous non-malleability implies message uniqueness, a natural question is whether the two properties are actually equivalent. We show that they are not equivalent in a very strong sense, indeed in §5 we describe a code that is one-time non-malleable and satisfies message uniqueness, but that is already insecure for 2 tampering queries. The scheme makes black-box use of any one-time non-malleable code in the split-state model additionally satisfying message uniqueness (such as our scheme from §4). The idea is to encode both the message m and some random pad κ using the underlying non-malleable code. Let us write (c_0^1, c_1^1) and (c_0^2, c_1^2) for the corresponding encodings. The obtained codeword has $c_0^* := (c_0^1, c_0^2, \delta)$ as left share, and $c_1^* := (c_1^1, c_1^2, \delta)$ as right share, where $\delta = m \oplus \kappa$ is a one-time pad encryption of the message m using pad κ. The decoding simply decodes the first component of each share (i.e., the pair (c_0^1, c_1^1)) using the decoding procedure of the underlying non-malleable code (completely ignoring all other elements).

On the one hand, one can show that the modified scheme inherits both message uniqueness and one-time non-malleability from the underlying auxiliary code. Intuitively, this is because successfully mauling a codeword (c_0^*, c_1^*) still requires to maul (c_0^1, c_1^1), which is hard by the one-time non-malleability of the underlying NMC. (We refer the reader to §5 for a detailed proof sketch.) On the other hand, using a first tampering query, a split-state adversary can swap c_0^1 with c_0^2 on the left, and c_1^1 with c_1^2 on the right, thus obtaining the random pad κ in the clear as a response. Once the pad is known, the second tampering query can hard-wire the value κ, recover the message $m = \delta \oplus \kappa$ in the clear (both from the left and the right share), and finally encode a related value.

1.5 Additional Related Work

Non-malleable codes. Only a few constructions of continuously non-malleable codes are known (besides the already mentioned constructions of [30,28]). In particular, continuous non-malleability is known to be achievable in the information-theoretic setting, for the simpler cases of bit-wise independent tampering [20,19] (where each bit of the codeword is tampered independently), and constant-state tampering [5]. Jafargholi and Wichs [36] obtain different flavors of continuous non-malleability for the case of tampering functions with high min-entropy or few fixed points. Aggarwal *et al.* [7] show that split-state continuous non-malleability is achievable in the information-theoretic setting, when tampering is persistent. Finally, Chattopadhyay *et al.* [14] construct *one-many non-malleable codes* that are secure with respect to an adversary that can specify many tampering functions to be applied to the one target codeword; the adversary succeeds if at least one of the tampering functions produces a valid encoding of a related message. Importantly, this notion does not rely on the self-destruct mechanism, but the total number of tampering attempts must be a-priori bounded.

Several other constructions of (one-time) non-malleable codes exist in the literature, achieving security for a plethora of tampering models, including: bit-wise independent tampering and permutations [18,8,9], circuits of polynomial

size [27,17,32], constant-state tampering [16], block-wise tampering [12], space-bounded algorithms [29,11], and bounded-depth circuits [10,15].

Applications. The typical application of non-malleable codes is the protection of cryptographic algorithms from tampering attacks against the memory [27,38,30]. Non-malleable codes were also used to protect arbitrary computations (and not only storage) against tampering [22,31,13].

A recent line of work shows interesting connections between the notions of non-malleable codes and non-malleable commitments. In particular, [12] proves that block-wise non-malleable codes (for two blocks) are equivalent to non-interactive non-malleable commitments (w.r.t. opening). Recently Goyal *et al.* [35] showed how to construct 3-round non-malleable commitments from standard assumptions when the adversary plays left and right sessions in parallel. Their scheme crucially relies on the power of split-state non-malleable codes.

Non-malleable codes can also be used to tackle the question of domain extension for non-malleable public-key encryption [20,19,40] and non-malleable commitments [8].

2 Overview of Techniques

2.1 Description of Our Code

Our code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is formally depicted in Fig. 1 on page 21, and it is based on a non-interactive commitment scheme with message space $\mathcal{M} := \{0,1\}^k$, randomness space $\mathcal{R} := \{0,1\}^\rho$ and commitment space $\Gamma \subseteq \{0,1\}^\ell$, and on an auxiliary split-state code $\Pi' = (\mathsf{Enc}', \mathsf{Dec}')$ mapping bitstrings of length $(k+\rho)$ into bitstrings of length $2n'$; the length of a codeword will be $2n = 2n' + 2\ell$. We denote by Commit the commitment function. (We refer the reader to §3 for the standard definitions of continuously non-malleable codes and non-interactive commitments.)

Intuitively, the encoding algorithm Enc constructs a commitment $\gamma \in \{0,1\}^\ell$ of the message $m \in \{0,1\}^k$ using randomness $r \in \{0,1\}^\rho$. Then it encodes the string $m||r$ via Enc', obtaining (c'_0, c'_1). Finally, it outputs the split-state encoding $((\gamma, c'_0), (\gamma, c'_1))$, of length $2n = 2n' + 2\ell$. The decoding algorithm first checks that the commitment γ on the left and right shares is equal, in which case it decodes (c'_0, c'_1) obtaining a value $m||r$, and outputs m if and only if (m, r) is a valid opening of the commitment.

For the security proof, we need the commitment scheme to be computationally hiding, and the underlying code Π' to be a split-state non-malleable code with unconditional security (under a single tampering query), and that additionally Π' satisfies leakage resilience, and two additional properties on the distribution of the codewords. The first property, which we call *codewords uniformity*, intuitively says that the two shares of an encoding under Π' are uniform over the set of all possible shares when considered in isolation, whereas their joint distribution is uniform over a smaller subset of the codewords space. The second property, which we call *conditional independence*, intuitively says that

the mutual information between the left and right share is bounded. We show how to instantiate our construction in §4.

2.2 Proof Intuition

We next give an overview of the proof of non-malleability. Note that we do not make any assumption on the malleability of the commitment scheme. Let us write $\mathbf{T}(b, q)$ for the random variable corresponding to the tampering experiment defining continuous non-malleability of the above defined encoding scheme Π, with hidden bit b, and where the adversary asks q tampering queries. In this experiment, the adversary can adaptively choose up to q split-state tampering queries that are applied to a target encoding $c = ((\gamma, c'_0), (\gamma, c'_1))$ of message m_b; after each tampering query, the adversary learns the outcome corresponding to decoding the modified codeword. Importantly, both $\mathbf{T}(0, q)$ and $\mathbf{T}(1, q)$ are additionally parameterized by messages m_0, m_1, and moreover the output of the experiments is defined to be $\texttt{same}^*$ in case the tampered codeword decodes to either of m_0, m_1; furthermore, in case the answer to a tampering query is equal to $\perp$ (i.e., the modified codeword is invalid), all future queries are answered with $\perp$ (i.e., the experiment self-destructs).

Our goal is to show $\mathbf{T}(0, q) \approx_c \mathbf{T}(1, q)$, for all polynomials $q(\lambda)$. The main idea is to consider a hybrid experiment $\mathbf{H}(b, q)$ where we decouple the randomness used to define the commitment in the target codeword from the input of the inner encoding scheme Π'. Namely, in experiment $\mathbf{H}(b, q)$ the target codeword has the form $c := ((\gamma, c'_0), (\gamma, c'_1))$ where γ is a commitment to m_b using randomness r (as before), and (c'_0, c'_1) is an encoding of a random uncorrelated value $s' \leftarrow_{\$} \{0,1\}^{k+\rho}$ (instead of the string $m_b||r$). We then argue that $\mathbf{T}(0, q) \approx_s \mathbf{H}(0, q) \approx_c \mathbf{H}(1, q) \approx_s \mathbf{T}(1, q)$, as outlined in the following subsections.

2.3 First Step

We start by showing that $\mathbf{T}(b, q) \approx_s \mathbf{H}(b, q)$, for all $b \in \{0, 1\}$ and for all $q \in \text{poly}(\lambda)$, down to the non-malleability of the underlying encoding scheme Π'. This part of the proof is completely information-theoretic, and moreover it relies on the two additional properties of codewords uniformity and conditional independence discussed above. Fix $b = 0$ (the proof for the other case is analogous). We use induction on the number of tampering queries $q(\lambda)$, as explained below.

Induction Basis The base case of the induction requires to show that $\mathbf{T}(0, 1) \approx_s \mathbf{H}(0, 1)$. We consider a reduction having access to a target encoding $c' = (c'_0, c'_1)$ that is either an encoding of $s'_0 := m_0||r$ or an encoding of a random string $s'_1 := s'$. Note that, since the reduction knows both m_0 and r, it can perfectly

simulate the distribution of the target codeword $c = ((\gamma, c'_0), (\gamma, c'_1))$ for experiments $\mathbf{T}(0,1)$ and $\mathbf{H}(0,1)$ inside the tampering oracle; this is done by computing offline $\gamma = \mathsf{Commit}(m_0; r)$, and by hard-wiring this value into the tampering function.

Thus, the reduction can perfectly simulate the *input* for a tampering query as it would be done in $\mathbf{T}(0,1)$ and $\mathbf{H}(0,1)$. The difficulty, however, is that the reduction only gets to see the decoding of the value $\tilde{s}$ corresponding to the tampered codeword $\tilde{c}' = (\tilde{c}'_0, \tilde{c}'_1)$, which is not directly the same as the output of the experiment in $\mathbf{T}(0,1)$ and $\mathbf{H}(0,1)$. For instance, in case $\tilde{s} \notin \{\mathtt{same}^*, \perp, m_0||\tilde{r}, m_1||\tilde{r}\}$, for any $\tilde{r} \in \{0,1\}^\rho$, the reduction knows that $\tilde{c}'$ is a valid encoding of some string $\tilde{s} := \tilde{m}||\tilde{r} \in \{0,1\}^{k+\rho}$, but the output of experiment $\mathbf{T}(0,1)$ and $\mathbf{H}(0,1)$ is either equal to $\tilde{m}$ or $\perp$ depending on whether $\tilde{m}$ and $\tilde{r}$ are consistent with the modified commitment $\tilde{\gamma}$.

In order to overcome this obstacle, we exploit the leakage resilience property of Π'; in particular, we let the reduction leak the value $\tilde{\gamma}$ (as defined above). Our analysis shows that this is all one needs in order to complete the simulation in a perfect manner (with all but a negligible probability).

Inductive Step Next, we assume that $\mathbf{T}(0,i) \approx_s \mathbf{H}(0,i)$ for some $i \in [q-1]$, and we show that this implies $\mathbf{T}(0,i+1) \approx_s \mathbf{H}(0,i+1)$. This is achieved once again via a reduction to the underlying one-time non-malleable code. Notice that, as before, the reduction can perfectly simulate the distribution of the target codeword $c = ((\gamma, c'_0), (\gamma, c'_1))$ for experiments $\mathbf{T}(0,i+1)$ and $\mathbf{H}(0,i+1)$ inside the tampering oracle. However two new problems arise. First, in the experiments $\mathbf{T}(0,i+1)$ and $\mathbf{H}(0,i+1)$ the adversary can ask up to $i+1$ tampering queries, whereas the reduction can play only one query, and so it needs to simulate the answer to all other i tampering queries on its own (and in a consistent manner). Second, even if the reduction were able to answer all other queries, it is a priori unclear how to choose which of the $i+1$ tampering functions the reduction should use in order to break one-time non-malleability of the code Π'.

The solution to the second problem comes immediately from the induction hypothesis. In fact, we know that, with overwhelming probability, the adversary cannot be successful after just i queries, as this would contradict our assumption that $\mathbf{T}(0,i) \approx_s \mathbf{H}(0,i)$. Using this observation, our strategy will be to simulate the answer to the first i tampering queries in a consistent manner, and later rely on the $(i+1)$-th query in order to violate security of the code Π'.

The solution to the first problem, instead, is more complicated. Essentially our reduction plays the following strategy:

1. At setup, compute all possible encodings $\hat{c} := (\hat{c}_0, \hat{c}_1)$ of the challenge messages $s'_0 = m_0||r$ and $s'_1 = s'$, and store $\hat{c}$ in an initially empty array $\hat{\mathcal{S}}^{(1)} := \hat{\mathcal{S}}_0^{(1)} \times \hat{\mathcal{S}}_1^{(1)}$, where $\hat{\mathcal{S}}_0^{(1)}$ and $\hat{\mathcal{S}}_1^{(1)}$ are the sub-arrays containing, respectively, all the left shares $\hat{c}_0$ and all the right shares $\hat{c}_1$.
2. Upon input a tampering query $(\phi_0^{(j)}, \phi_1^{(j)})$ from the adversary, for any $j \leq i$, answer as follows:
   - For all codewords $\hat{c} = (\hat{c}_0, \hat{c}_1) \in \hat{\mathcal{S}}_0^{(j)} \times \hat{\mathcal{S}}_1^{(j)}$, decode the corresponding tampered codeword $(\phi_0^{(j)}(\gamma, \hat{c}_0)), \phi_1^{(j)}(\gamma, \hat{c}_1))$.

- Let m^* be the most likely outcome, and answer the query with $\tilde{m}^{(j)} = m^*$.
- Define $\hat{\mathcal{S}}_0^{(j+1)}, \hat{\mathcal{S}}_1^{(j+1)}$ to be the sub-arrays of $\hat{\mathcal{S}}_0^{(j)}, \hat{\mathcal{S}}_1^{(j)}$ containing all possible codewords which are compatible with the answer to the j-th query being m^*.

3. Make sure all answers $\tilde{m}^{(1)}, \ldots, \tilde{m}^{(i)}$ are correct whenever the corresponding codewords produced by the tampering functions are valid. This is achieved as follows:
   - Define a leakage query that hardwires all answers $(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(i)})$, as well as the tampering queries $(\phi_0^{(1)}, \ldots, \phi_0^{(i)})$ and the arrays $\hat{\mathcal{S}}_0^{(1)}, \ldots, \hat{\mathcal{S}}_0^{(i)}$, and returns the first index j (if any) such that the target left share (γ, c_0') is contained in the j-th array, but is not contained in the $(j+1)$-th array. (An analogous check is performed on the target right share (γ, c_1'), using $\phi_1^{(1)}, \ldots, \phi_1^{(i)}$ and $\hat{\mathcal{S}}_1^{(1)}, \ldots, \hat{\mathcal{S}}_1^{(i)}$.) In case such an index is found, the leakage query additionally returns the correct answer $\hat{m}$.[14]
   - Rewind the adversary to step 2, at the iteration where it asked the j-th query, and modify the answer using the leaked value. Additionally, update the arrays $\hat{\mathcal{S}}_0^{(j+1)}, \hat{\mathcal{S}}_1^{(j+1)}$ consistently[15] with the answer of the j-th tampering query being $\hat{m}$, and go back to step 2 continuing from the $(j+1)$-th tampering query.
4. Upon input the final tampering query $(\phi_0^{(i)}, \phi_1^{(i)})$ from the adversary, use this query to define the tampering query (ϕ_0', ϕ_1') to be applied to the target encoding; this is done in exactly the same way as discussed above for the base case of the induction.

In order to conclude the proof, we need to show two things. First, we need to argue that the total number of rewinds performed by the reduction is somewhat limited, so that the reduction does not exceed the total leakage bound supported by the underlying non-malleable code. Second, we need to ensure that the simulation performed by the reduction generates a distribution that is indistinguishable from what the adversary would expect in a real execution of experiments $\mathbf{T}(0, i+1)$ and $\mathbf{H}(0, i+1)$. We deal with these issues as follows.

Challenge #1: Bounding the leakage. Let $(\phi_0^{(j)}, \phi_1^{(j)})$ be a tampering query provoking one of the rewinds. Denote by $\tilde{c}_0 := (\tilde{\gamma}, \tilde{c}_0') = \phi_0^{(j)}(\gamma, c_0')$ the corresponding modified left share. By message uniqueness, which for our code easily follows by the perfect binding property of the commitment, $\tilde{c}_0$ is a valid left share of at most one message $\tilde{m} \in \{0,1\}^k$. A counting argument shows that the probability associated to the output of the decoding being $\tilde{m}$ is $\tilde{p} \leq 1/2$. Hence, intuitively, we would like to argue that since $\tilde{m}$ is not the most likely outcome, there is a

[14] This is achieved by leaking the commitment $\tilde{\gamma}$ corresponding to the tampering query $(\phi_0^{(j)}, \phi_1^{(j)})$, and by having the reduction find the corresponding (unique) message via brute force.

[15] The new $\hat{\mathcal{S}}_0^{(j+1)}, \hat{\mathcal{S}}_1^{(j+1)}$ are obtained from $\hat{\mathcal{S}}_0^{(j)}, \hat{\mathcal{S}}_1^{(j)}$ by removing the encodings that are not compatible with the answer of the j-th tampering query being $\hat{m}$.

probability of at least $1/2$ that the modified right share $\tilde{c}_1 := (\tilde{\gamma}, \tilde{c}'_1) = \phi_1^{(j)}(\gamma, c'_1)$ will correspond to a message different from $\tilde{m}$, and thus every such query yields a self-destruct with probability at least $1/2$.

Unfortunately, it is unclear how to complete the above argument using any one-time unconditionally secure non-malleable code. In fact, the left and right shares of the inner encoding $c' = (c'_0, c'_1)$ are correlated, and a tampering query could exploit such correlation in order to generate an output which is not the most likely outcome, and yet the probability of self-destruct is smaller than $1/2$. We solve this problem by relying on the two additional properties of codewords uniformity and conditional independence. In particular, by a careful information-theoretic argument, we can show that codewords uniformity implies that every tampering query evading the above argument decreases the mutual information between the left and right share of c' by at least one bit. By conditional independence, the maximum number of such queries is bounded, after which the mutual information between c'_0 and c'_1 is zero, and any further tampering query causing a rewind will incur a probability of self-destruct of at least $1/2$.

Challenge #2: Arguing indistinguishability. As for indistinguishability, note that the corrected answers $\tilde{m}^{(1)}, \ldots, \tilde{m}^{(i)}$ might still be inconsistent, due to the fact that the tampered inner codeword $(\tilde{c}'_0, \tilde{c}'_1)$ decodes to $\perp$ for some of the queries. Indeed, such an invalid codeword can not be detected using the above leakage queries since they allow only to read the commitments computed by the tampering function in the two shares. The adversary might notice this inconsistency, and could for instance instruct the distinguisher to flip its output in order to make the reduction fail.

We circumvent this obstacle as follows. First off, let us assume w.l.o.g. that the distinguisher satisfies the following invariant: it outputs 0 (resp. 1) whenever it believes the target codeword is an encoding of m_0 (resp. m_1). Hence, we let the reduction ask an additional leakage query, leaking a single bit, that hard-wires a description of the distinguisher and of the final tampering query $(\phi_0^{(i+1)}, \phi_1^{(i+1)})$, together with all the answers $\tilde{m}^{(1)}, \ldots, \tilde{m}^{(i)}$ to the first i queries, the commitment γ, and the final arrays $\hat{\mathcal{S}}_0^{(i+1)}, \hat{\mathcal{S}}_1^{(i+1)}$. The goal of the leakage query is to allow the reduction to check that the output of the distinguisher on the simulated view satisfies the above invariant. This is achieved as follows. For each $\hat{c}_1 \in \hat{\mathcal{S}}_1^{(i+1)}$,[16] let $\hat{b} \in \{0, 1\}$ be such that $(c'_0, \hat{c}'_1)$ is an encoding of $m_{\hat{b}}$; the leakage query computes the answer $\tilde{m}^*$ to the $(i+1)$-th tampering query by applying $(\phi_0^{(i+1)}, \phi_1^{(i+1)})$ to $((\gamma, c'_0), (\gamma, \hat{c}'_1))$, and then it returns $\hat{\delta} = 1$ iff the output of the distinguisher upon $(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(i)}, \tilde{m}^*)$ is more often equal to $\hat{b}$.

Finally, in case $\hat{\delta} = 0$, the reduction returns a random guess, whereas if $\hat{\delta} = 1$, it uses the output of the distinguisher on the simulated view, the intuition being that the outcome of the distinguisher is used only if no inconsistency was introduced during the simulation of each tampering query. The proof shows that

[16] Without loss of generality we describe the leakage function as a leakage query on the left share.

this allows us to keep the non-negligible advantage of the distinguisher, thus contradicting one-time unconditional non-malleability of the code Π'.

2.4 Second Step

In a second step we show that $\mathbf{H}(0,q) \approx_c \mathbf{H}(1,q)$, down to the hiding property of the commitment scheme. This step is significantly easier, because in both experiments $\mathbf{H}(0,q)$ and $\mathbf{H}(1,q)$ the input of the commitment and of the inner encoding algorithm are completely independent. Hence, in the reduction we can embed a target commitment γ (which is either a commitment to m_0 or a commitment to m_1) and complete the codeword by sampling a fresh encoding (c'_0, c'_1) of a random value $s' \in \{0,1\}^{k+\rho}$. This way, we can easily turn a distinguisher between the two hybrids into an adversary breaking the hiding property of the commitment scheme.

Note that in this case the reduction can perfectly simulate the view of the distinguisher, as it has a perfectly distributed target codeword (either w.r.t. $\mathbf{H}(0,q)$ or w.r.t. $\mathbf{H}(1,q)$) "in its hands."

3 Preliminaries

3.1 Notation

For a string x, we denote its length by $|x|$; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the number of elements in $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\!\$\, \mathcal{X}$. When A is a randomized algorithm, we write $y \leftarrow\!\!\$\, \mathsf{A}(x)$ to denote a run of A on input x and output y; in this case, the value y is a random variable and $\mathsf{A}(x;r)$ denotes a run of A on input x and randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x;r)$ terminates in at most $\mathrm{poly}(|x|)$ steps.

We denote with $\lambda \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ, i.e. $\nu(\lambda) = \lambda^{-\omega(1)}$. We sometimes write $\mathrm{negl}(\lambda)$ (resp., $\mathrm{poly}(\lambda)$) to denote all negligible functions (resp., polynomial functions) in the security parameter. All algorithms are implicitly assumed to take the security parameter as input.

For a random variable $\mathbf{X}$, we write $\mathbb{P}[\mathbf{X} = x]$ for the probability that $\mathbf{X}$ takes on a particular value $x \in \mathcal{X}$ (with $\mathcal{X}$ the set where $\mathbf{X}$ is defined). The statistical distance between two random variables $\mathbf{X}$ and $\mathbf{X}'$ defined over the same set $\mathcal{X}$ is defined as $\Delta(\mathbf{X};\mathbf{X}') = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mathbb{P}[\mathbf{X} = x] - \mathbb{P}[\mathbf{X}' = x]|$. The mutual information between $\mathbf{X}$ and $\mathbf{Y}$ is a measure of their mutual dependence, and it is defined as $\mathbb{I}(\mathbf{X};\mathbf{Y}) = \mathbb{H}(\mathbf{X}) - \mathbb{H}(\mathbf{X}|\mathbf{Y})$, where $\mathbb{H}(\cdot)$ denotes the Shannon's entropy.

Given two ensembles of random variables $\mathbf{X} = \{\mathbf{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathbf{Y} = \{\mathbf{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, we write $\mathbf{X} \equiv \mathbf{Y}$ to denote that the two ensembles are identically distributed, $\mathbf{X} \approx_s \mathbf{Y}$ to denote that the two ensembles are statistically close (i.e., $\Delta(\mathbf{X}_\lambda;\mathbf{Y}_\lambda) \in \mathrm{negl}(\lambda)$), and $\mathbf{X} \approx_c \mathbf{Y}$ to denote that the two ensembles are computationally indistinguishable (i.e., $|\mathbb{P}[\mathsf{D}(\mathbf{X}_\lambda) = 1] - \mathbb{P}[\mathsf{D}(\mathbf{Y}_\lambda) = 1]| \in \mathrm{negl}(\lambda)$ for all PPT distinguishers D).

3.2 Non-Malleable Codes

Introduced by Dziembowski, Pietrzak, and Wichs [27], non-malleable codes allow to encode a message in such a way that the decoding of a tampered codeword (according to a restricted class of modifications) either yields the original message or an unrelated value.

Definition 1 (Encoding scheme). *A (k, n)-code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ consists of a pair of algorithms specified as follows: (i) The (randomized) encoding algorithm Enc takes as input a string $s \in \{0,1\}^k$ and returns a codeword $c \in \{0,1\}^n$; (ii) The (deterministic) decoding algorithm Dec takes as input a codeword $c \in \{0,1\}^n$ and outputs a value in $\{0,1\}^k \cup \{\perp\}$, where $\perp$ denotes an* invalid *codeword. A codeword $c \in \{0,1\}^n$ such that $\mathsf{Dec}(c) \neq \perp$ is called a* valid *codeword.*

The code Π satisfies correctness if, for all $s \in \{0,1\}^k$, we have that $\mathsf{Dec}(\mathsf{Enc}(s)) = s$ with overwhelming probability over the randomness of the encoding algorithm.

Standard non-malleability, as defined in [27], allows an adversary to maul a target encoding only once. Continuous non-malleability [30] extends the basic non-malleability requirement by allowing the adversary to tamper multiple times, where tampering might either be non-persistent (i.e., the adversary always mauls the *same* target encoding) or persistent (i.e., the current tampering function is applied to the encoding resulting from the previous mauling attempt). Throughout this paper, we always assume that tampering is non-persistent (which is the more challenging scenario).

Split-state model. Below we recall the definition of continuous non-malleability in the so-called *split-state model*. Here a codeword $c \in \{0,1\}^{2n}$ consists of two *shares* $c_0 \in \{0,1\}^n$ and $c_1 \in \{0,1\}^n$.[17] We call such codes *split-state* $(k, 2n)$-codes. In the split-state model, the tampering functions $\phi : \{0,1\}^{2n} \to \{0,1\}^{2n}$ can be described as pairs $\phi := (\phi_0, \phi_1)$ of functions with $\phi_0, \phi_1 : \{0,1\}^n \to \{0,1\}^n$. Tampering function ϕ, when applied to codeword c modifies it into $\tilde{c} := \phi(c)$ defined as

$$\tilde{c} := (\phi_0(c_0), \phi_1(c_1)).$$

To define the notion of continuous non-malleability, we introduce experiment $\mathbf{Tamper}^{\Pi,\mathsf{A}}_{s_0,s_1}$ that is parameterized by a split-state code Π, by a PPT adversary A, and by two messages s_0 and s_1, and takes as inputs the security parameter λ, a bit b, and a value $q \in \mathbb{N}$. In this experiment the adversary has access to two leakage oracles $\mathcal{O}^\ell$, and one tampering oracle $\mathcal{O}_{\mathrm{cnm}}$.

Definition 2 (Leakage oracle). *A* leakage oracle *$\mathcal{O}^\ell$ is a stateful oracle that maintains a counter $\mathtt{ct}$ that is initially set to 0. When $\mathcal{O}^\ell$ is invoked for a string c and a leakage function ψ, value $\psi(c)$ is computed, its length is added to $\mathtt{ct}$ and if $\mathtt{ct} \leq \ell$ then $\psi(c)$ is returned; otherwise, $\perp$ is returned.*

[17] More generally, the encoding might not be symmetric in which case $c_0 \in \{0,1\}^{n_0}$ and $c_1 \in \{0,1\}^{n_1}$, for arbitrary values $n_0, n_1 \in \mathbb{N}$ such that $n_0 + n_1 = n$; while this generalization is immediate, it is not needed in this paper.

Definition 3 (Tampering oracle). *A* tampering oracle $\mathcal{O}_{\text{cnm}}^{s_0,s_1}$ *is a stateful oracle (implicitly) parameterized by a split-state code* $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ *and two strings* s_0 *and* s_1*, with state* $\mathtt{st}$ *initialized to* $\mathtt{st} = \mathtt{Active}$*. The oracle takes as input a codeword* $c = (c_0, c_1)$ *and a split-state tampering function* $\phi = (\phi_0, \phi_1)$ *and its output is defined as follows.*

Oracle $\mathcal{O}_{\text{cnm}}^{s_0,s_1}((c_0, c_1), (\phi_0, \phi_1))$*:*
If $\mathtt{state} = \mathtt{SelfDestruct}$*, return* $\perp$
Let $(\tilde{c}_0, \tilde{c}_1) := (\phi_0(c_0), \phi_1(c_1))$
If $\tilde{s} = \mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) \in \{s_0, s_1\}$*, return* $\mathtt{same}^*$
If $\mathsf{Dec}(\tilde{c}_0, \tilde{c}_1) = \perp$*, set* $\mathtt{state} = \mathtt{SelfDestruct}$ *and return* $\perp$
Else, return $\tilde{s}$

Definition 4 (Continuous non-malleability). *Let* $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ *be a split-state* $(k, 2n)$*-code. We say that* Π *is* ℓ-leakage-resilient q-time non-malleable in the split-state model *if for all* $s_0, s_1 \in \{0,1\}^k$ *and for all PPT adversaries* A *asking at most* q *tampering queries, we have that*

$$\left\{\mathbf{Tamper}_{s_0,s_1}^{\Pi,\mathsf{A}}(\lambda, 0, q)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathbf{Tamper}_{s_0,s_1}^{\Pi,\mathsf{A}}(\lambda, 1, q)\right\}_{\lambda\in\mathbb{N}}, \tag{1}$$

where, for $b \in \{0,1\}$*,*

$$\mathbf{Tamper}_{s_0,s_1}^{\Pi,\mathsf{A}}(\lambda, b, q) := \left\{ \begin{array}{c} (c_0, c_1) \leftarrow\!\!\$\ \mathsf{Enc}(s_b); \\ out_{\mathsf{A}} \leftarrow \mathsf{A}^{\mathcal{O}^{\ell}(c_0,\cdot),\mathcal{O}^{\ell}(c_1,\cdot),\mathcal{O}_{\text{cnm}}^{s_0,s_1}((c_0,c_1),(\cdot,\cdot))}(1^{\lambda}) \end{array} \right\}.$$

Without loss of generality, we can assume that the value out_{A} consists of the adversary's view. In case Eq. (1) only holds for $q = 1$, we write that the encoding scheme is *one-time* non-malleable, whereas if Eq. (1) holds for an arbitrary polynomial $q(\cdot)$, we say that encoding scheme is *continuously* non-malleable; it is also worth noting that for $q = 0$ (i.e., no tampering allowed) the above notion collapses to the definition of leakage-resilient codes [23], which have turned useful in several constructions o non-malleable codes [30,32]. Also note that we can cast information-theoretic security by simply requiring that Eq. (1) holds for the statistical distance, for all possibly unbounded distinguishers, where now also the tampering functions ϕ specified by the adversary, as well as the leakage functions ψ, need not be polynomial-time computable.

As explained in the introduction, our definition of continuous non-malleability is strictly weaker than the one originally considered in [30] (and afterwards also in [32,36]), in that the adversary at the end of each tampering query only obtains the decoding of the tampered codeword (unless this happens to be equal to one of s_0, s_1), and not the tampered codeword itself (as long as $\tilde{c} \neq c$ is valid). We further observe that (continuous) non-malleability can also be stated through the existence of an efficient simulator, however the two formulations are equivalent for messages of super-polynomial size [27]. (This fact was proven for the case of one-time non-malleability, but it holds more generally for the case of continuous non-malleability and when considering leakage.)

Message uniqueness. As shown in [30] continuous non-malleability in the split-state model is impossible to achieve in the information-theoretic setting.[18] In the computational setting, in order to be continuously non-malleable, a split-state code must[19] satisfy a special property called *message uniqueness*. Informally, message uniqueness says that it should be hard to fix one part of an encoding, say $c_0 \in \{0,1\}^n$, and compute two *distinct* other parts $c_1, \bar{c}_1 \in \{0,1\}^n$ such that both (c_0, c_1) and $(c_0, \bar{c}_1)$ are *valid* encodings of two *different* messages.

Definition 5 (Message uniqueness). *Let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a split-state code. We say that Π satisfies* perfect message uniqueness *if, for all $\beta \in \{0,1\}$, there do not exist values $(c_\beta, c_{1-\beta}, \bar{c}_{1-\beta})$ such that $c_{1-\beta} \neq \bar{c}_{1-\beta}$ and, at the same time,*

$$\perp \neq \mathsf{Dec}(c_\beta, c_{1-\beta}) \neq \mathsf{Dec}(c_\beta, \bar{c}_{1-\beta}) \neq \perp.$$

Remark 1 (On perfect uniqueness). One could define a computational or statistical variant of the uniqueness property, where tuples of values violating message uniqueness exist but are hard to find. We note, however, that in the plain model assuming perfect message uniqueness is w.l.o.g. In fact, if a tuple $(c_\beta, c_{1-\beta}, \bar{c}_{1-\beta})$ violating message uniqueness exists (i.e., uniqueness is not perfect), we can always consider the specific PPT adversary that has such a tuple hard-wired in its code (and that thus contradicts computational and statistical message uniqueness).

Remark 2 (On message versus codeword uniqueness). An even stronger flavor of uniqueness, not needed in this paper and known as codeword uniqueness, requires that, for all $\beta \in \{0,1\}$, there do not exist values $(c_\beta, c_{1-\beta}, \bar{c}_{1-\beta})$ such that $c_{1-\beta} \neq \bar{c}_{1-\beta}$ and, at the same time, $\mathsf{Dec}(c_\beta, c_{1-\beta}) \neq \perp$ and $\mathsf{Dec}(c_\beta, \bar{c}_{1-\beta}) \neq \perp$, but eventually $\mathsf{Dec}(c_\beta, c_{1-\beta}) = \mathsf{Dec}(c_\beta, \bar{c}_{1-\beta})$. Codeword uniqueness is a strictly stronger property than message uniqueness, and is known to be necessary for achieving the stronger flavor of continuous super non-malleability [30].

3.3 Non-Interactive Commitments

A non-interactive commitment scheme is a randomized efficient algorithm Commit taking as input a message $m \in \mathcal{M}$ and random coins $r \in \mathcal{R}$, and outputting a commitment $\gamma \in \Gamma$. A decommitment of γ consists simply of revealing m and r. The sets $\mathcal{M}$, $\mathcal{R}$ and Γ are called (respectively) the message space, the randomness space, and the commitment space. A commitment scheme satisfies two properties called *hiding* and *binding*. We recall such properties below.

The binding property says that it is hard to open a given commitment $\gamma \in \Gamma$ in two different ways. Exactly as for the case of uniqueness, the assumption of perfect binding is w.l.o.g. in the plain model.

[18] Information-theoretic security is, instead, possible in other settings, such as bit-wise independent tampering [20,19], constant-state tampering [5], and split-state *persistent* tampering [7].

[19] Otherwise a generic attack is possible; see §1 for an informal description.

Definition 6 (Binding). *We say that a non-interactive commitment* Commit *is* perfectly binding *if there do not exist pairs* $(m_0, r_0), (m_1, r_1)$ *such that* $m_0 \neq m_1$ *and, at the same time,* $\mathsf{Commit}(m_0; r_0) = \mathsf{Commit}(m_1; r_1)$.

The hiding property says that for any pair of messages m_0, m_1 it is hard to tell whether a given commitment γ is for m_0 or for m_1.

Definition 7 (Hiding). *We say that a non-interactive commitment* Commit *is* computationally hiding *if for all messages* $m_0, m_1 \in \mathcal{M}$ *the following holds:*

$$\left\{\gamma : \ \gamma \leftarrow\!\!\$\ \mathsf{Commit}(1^\lambda, m_0)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\gamma : \ \gamma \leftarrow\!\!\$\ \mathsf{Commit}(1^\lambda, m_1)\right\}_{\lambda\in\mathbb{N}}.$$

4 Code Construction

In this section we present a construction of a split-state code that achieves continuous non-malleability. The scheme is in the plain model, and can be based on any (possibly malleable) non-interactive commitment scheme (cf. §3.3), and on an information-theoretic one-time non-malleable and leakage-resilient split-state code (cf. §3.2) satisfying a few additional properties (see below).

Note that the first assumption is necessary, meaning that a continuously non-malleable code in the split-state model implies a non-interactive commitment scheme. In fact, recall that any continuously non-malleable code must satisfy message uniqueness. Given a non-malleable split-state code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ with message uniqueness, consider the non-interactive commitment scheme where, in order to commit to message m, the committer computes a split-state encoding (c_0, c_1) of m using algorithm Enc. The left part c_0 constitutes the commitment, and the right part c_1 is the decommitment. The receiver verifies that c_1 is the correct opening of c_0 as m, by running Dec on input (c_0, c_1) and verifying that the output is indeed m. Binding follows by the fact that Π satisfies message uniqueness, and hiding follows by the non-malleability of Π.

4.1 Additional Properties

For our construction, we will rely on a split-state code meeting two non-standard requirements that we formally define below. The first property intuitively says that, for any message, the encoder outputs codewords that are uniformly random over some subset of all possible codewords.

Definition 8 (Codewords uniformity). *Let* $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ *be a split-state* $(k, 2n)$*-code, and denote by* $\mathbf{C} = (\mathbf{C}_0, \mathbf{C}_1)$ *the random variable corresponding to the output of the encoding algorithm upon input some value* $s \in \{0,1\}^k$*. We say that* Π *satisfies* codewords uniformity *if, for all values* $s \in \{0,1\}^k$*, we have that each of* $\mathbf{C}_0$ *and* $\mathbf{C}_1$ *in isolation is uniform, respectively, over subsets* $\mathcal{C}_0 \subseteq \{0,1\}^n$ *and* $\mathcal{C}_1 \subseteq \{0,1\}^n$*, whereas* $(\mathbf{C}_0, \mathbf{C}_1)$ *is uniformly distributed over some subset* $\overline{\mathcal{C}} := \overline{\mathcal{C}}_0 \times \overline{\mathcal{C}}_1 \subset \mathcal{C}_0 \times \mathcal{C}_1$.

Let Commit be a non-interactive commitment scheme with message space $\mathcal{M} := \{0,1\}^k$, randomness space $\mathcal{R} := \{0,1\}^\rho$, and commitment space $\Gamma \subseteq \{0,1\}^\ell$. Let $\Pi' = (\mathsf{Enc}', \mathsf{Dec}')$ be a split-state $(k+\rho, 2n')$-code. Define the following split-state $(k, 2n)$-code, where $n := n' + \ell$.

Encoding: Upon input a value $m \in \{0,1\}^k$, sample random coins $r \leftarrow\!\!\$\ \{0,1\}^\rho$ and compute $\gamma := \mathsf{Commit}(m; r)$ and $(c'_0, c'_1) \leftarrow\!\!\$\ \mathsf{Enc}(m||r)$. Return the codeword $c = (c_0, c_1) := ((\gamma, c'_0), (\gamma, c'_1))$.

Decoding: Upon input a codeword $c \in \{0,1\}^{2n}$, parse $c := (c_0, c_1) := ((\gamma_0, c'_0), (\gamma_1, c'_1))$. Hence, proceed as follows:

(a) If $\gamma_0 \neq \gamma_1$, return $\perp$; else, let $\gamma = \gamma_0 = \gamma_1$.
(b) Run $s = \mathsf{Dec}'(c'_0, c'_1)$; if $s = \perp$ return $\perp$.
(c) Parse $s := m||r$; if $\gamma = \mathsf{Commit}(m; r)$ return m, else return $\perp$.

Fig. 1: Description of our code.

The second property captures the fact that, for any message, the distribution of the left and right share of a codeword have limited dependence (in terms of their mutual information).

Definition 9 (Conditional independence). *Let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a split-state $(k, 2n)$-code, and denote by $\mathbf{C} = (\mathbf{C}_0, \mathbf{C}_1)$ the random variable corresponding to the output of the encoding algorithm upon input some value $s \in \{0,1\}^k$. We say that Π satisfies* α-conditional independence *if, for all values $s \in \{0,1\}^k$, we have that $\mathbb{I}(\mathbf{C}_0; \mathbf{C}_1) \leq \alpha$.*

4.2 Theorem Statement

Consider the split-state $(k, 2n)$-code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ depicted in Fig. 1, based on a non-interactive commitment scheme Commit with message space $\mathcal{M} := \{0,1\}^k$, randomness space $\mathcal{R} := \{0,1\}^\rho$ and commitment space $\Gamma \subseteq \{0,1\}^\ell$, and on an auxiliary split-state $(k+\rho, 2n')$-code $\Pi' = (\mathsf{Enc}', \mathsf{Dec}')$. The properties we require from each building block are directly stated in Theorem 3 below.

Intuitively, the encoding algorithm Enc constructs a commitment $\gamma \in \{0,1\}^\ell$ of the message $m \in \{0,1\}^k$ using randomness $r \in \{0,1\}^\rho$. Then it encodes the string $m||r$ via Enc', obtaining (c'_0, c'_1). Finally, it outputs the split-state encoding $((\gamma, c'_0), (\gamma, c'_1))$.

Theorem 3 (Theorem 1, restated). *Assume that* Commit *is a non-interactive perfectly binding and computationally hiding commitment scheme, with message space $\mathcal{M} := \{0,1\}^k$, randomness space $\mathcal{R} := \{0,1\}^\rho$ and commitment space $\Gamma \subseteq \{0,1\}^\ell$. Let Π' be a split-state $(k+\rho, 2n')$-code that is unconditionally ℓ'-leakage-resilient one-time non-malleable, for $\ell' = (2\ell + O(\log \lambda)) \cdot (\alpha + O(\log \lambda))$, and that additionally satisfies the properties of* codewords uniformity *and* α-conditional independence. *Then, the encoding scheme Π described in Fig. 1 is a split-state $(k, 2(n'+\ell))$-code satisfying continuous non-malleability.*

Instantiating the scheme. For the commitment scheme we can rely on the standard construction based on one-to-one one-way functions [34]. If the message m is k-bit long, the resulting commitment will have $\ell \in O(k^2)$ bits.

For the underlying non-malleable code we use a scheme constructed in [6], which we briefly recall below. Let $\mathbb{F}$ be a finite field. The encoder first encodes the underlying message $m' \in \{0,1\}^k$ using an auxiliary one-time split-state non-malleable code with unconditional security, obtaining shares $(c''_0, c''_1) \in [N] \times [N]$, where $[N]$ is a sparse subset of $\mathbb{F}$ with size $N \ll |\mathbb{F}|$. Hence, each share c''_0, c''_1 is processed using a slight variant of the inner-product extractor, i.e. c''_0 (resp. c''_1) is encoded via two additional shares $(c''_{0,0}, c''_{0,1}) \in \mathbb{F}^{2t}$ (resp. $(c''_{1,0}, c''_{1,1}) \in \mathbb{F}^{2t}$) such that $\xi(\langle c''_{0,0}, c''_{0,1} \rangle) = c''_0$ (resp. $\xi(\langle c''_{1,0}, c''_{1,1} \rangle) = c''_1$), where $\xi : \mathbb{F} \to [N]$ is an arbitrary bijection. The final encoding is then defined to be $c' = (c'_0, c'_1) = ((c''_{0,0}, c''_{1,0}), (c''_{0,1}, c''_{1,1})) \in \mathbb{F}^{2t} \times \mathbb{F}^{2t}$.

By plugging in the above construction the split-state non-malleable code of [4,1], which has $\log N \in O(k^7)$, and choosing statistical error $\varepsilon := 2^{-k^2}$, we obtain a leakage-resilient one-time split-state non-malleable code with unconditional security and with leakage parameter $\ell' \approx k^{14}/12$ (cf. [6, Corollary 4.2]). It is important to note that the definition of leakage-resilient non-malleability considered in [6] is simulation based, and not indistinguishability based as our Definition 4. However, the former implies the latter: This was originally proven in [27] without considering leakage, but the same statement holds true, with basically the same proof, for the case of leakage, as long as the indistinguishability between the real and simulated experiment holds for the joint distribution of the leakage and the decoding of the tampered codeword. The latter requirement is fulfilled by the construction in [6], as the outer layer of their encoding is a split-state leakage-resilient code [23].

The above code is also easily seen to satisfy codewords uniformity (as c'_0 and c'_1 are uniform over the entire space of valid codewords when taken in isolation, whereas (c'_0, c'_1) is jointly uniform over a subset of the space of all valid codewords), and α-conditional independence, for $\alpha \in O(k^7)$ (as both $(c''_{0,0}, c''_{0,1})$ and $(c''_{1,0}, c''_{1,1})$ are uniform subject to their inner product being, respectively, c''_0 and c''_1, and moreover $|c''_0|, |c''_1| \in O(k^7)$). Hence, the leakage bound in Theorem 3 is satisfied too, as the required leakage is roughly $4k^9 + 2k^7 + 2k^2 \ll k^{14}/12$ (neglecting constant and logarithmic terms).

4.3 Security Analysis

For simplicity, let us define $\mathbf{T}_{m_0,m_1}(\lambda, b, q) \equiv \mathbf{Tamper}^{\Pi,\mathsf{A}}_{m_0,m_1}(\lambda, b, q)$. We need to show that for all messages $m_0, m_1 \in \{0,1\}^k$ and for all PPT adversaries A asking $q(\lambda) \in \mathrm{poly}(\lambda)$ tampering queries, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that for all PPT distinguishers D:

$$|\mathbb{P}\left[\mathsf{D}(\mathbf{T}_{m_0,m_1}(\lambda, 0, q) = 1\right] - \mathbb{P}\left[\mathsf{D}(\mathbf{T}_{m_0,m_1}(\lambda, 1, q)) = 1\right]| \leq \nu(\lambda).$$

Message Uniqueness We start by showing that our code meets perfect message uniqueness.

Hybrid $\mathbf{H}_{m_0,m_1}(\lambda, b, q)$:

The experiment is parameterized by messages $m_0, m_1 \in \{0,1\}^k$, security parameter $\lambda \in \mathbb{N}$, a secret bit $b \in \{0,1\}$, and the number of tampering queries $q(\lambda) \in \text{poly}(\lambda)$. It proceeds as follows:

- It first computes $\gamma := \mathsf{Commit}(m_b; r)$, for random coins $r \leftarrow\!\!\$\ \{0,1\}^\rho$, and then it sets $(c'_0, c'_1) \leftarrow\!\!\$\ \mathsf{Enc}'(s')$ for random $s' \leftarrow\!\!\$\ \{0,1\}^{k+\rho}$.
- The target encoding is defined to be $(c_0, c_1) := ((\gamma, c'_0), (\gamma, c'_1))$.
- Upon input the i-th tampering query $(\phi_0^{(i)}, \phi_1^{(i)})$, let $(\tilde{c}_0, \tilde{c}_1) = (\phi_0^{(i)}(c_0), \phi_1^{(i)}(c_1))$ be such that $\tilde{c}_0 := (\tilde{\gamma}_0, \tilde{c}'_0)$ and $\tilde{c}_1 := (\tilde{\gamma}_1, \tilde{c}'_1)$. Thus:
  (a) If $\tilde{\gamma}_0 \neq \tilde{\gamma}_1$, return $\perp$; else let $\tilde{\gamma} = \tilde{\gamma}_0 = \tilde{\gamma}_1$ and run $\tilde{s} = \mathsf{Dec}'(\tilde{c}'_0, \tilde{c}'_1)$.
  (b) If $\tilde{s} = \perp$, return $\perp$.
  (c) If $\tilde{s} = s'$ return $\texttt{same}^*$ in case $\tilde{\gamma} = \gamma$, and $\perp$ otherwise.
  (d) Else, parse $\tilde{s} := \tilde{m}||\tilde{r}$. If $\tilde{\gamma} \neq \mathsf{Commit}(\tilde{m}; \tilde{r})$, return $\perp$; otherwise, return $\texttt{same}^*$ if $\tilde{m} \in \{m_0, m_1\}$, and else return $\tilde{m}$.

Fig. 2: Hybrid experiment in the proof of Theorem 3.

Lemma 1. *The code of Fig. 1 satisfies perfect message uniqueness.*

Proof. Since our code is symmetric, it suffices to prove message uniqueness for the case $\beta = 0$. Assume there exist values $(c_0, c_1, \bar{c}_1)$ such that both (c_0, c_1) and $(c_0, \bar{c}_1)$ are *valid* codewords satisfying

$$\mathsf{Dec}(c_0, c_1) = m_0 \neq m_1 = \mathsf{Dec}(c_0, \bar{c}_1).$$

Write $c_0 = (\gamma, c'_0)$, $c_1 = (\gamma, c'_1)$, and $\bar{c}_1 = (\bar{\gamma}, \bar{c}'_1)$. By the fact that $(c_0, \bar{c}_1)$ is valid, it follows that $\bar{\gamma} = \gamma$. Let $s_0 := m_0||r_0 = \mathsf{Dec}'(c'_0, c'_1)$ and $s_1 := m_1||r_1 = \mathsf{Dec}'(c'_0, \bar{c}'_1)$ be obtained, respectively, by decoding (c'_0, c'_1) and $(c'_0, \bar{c}'_1)$. Note that both s_0 and s_1 are different from $\perp$, as (c_0, c_1) and $(c_0, \bar{c}_1)$ are valid codewords.

We conclude that

$$\mathsf{Commit}(m_0; r_0) = \gamma = \mathsf{Commit}(m_1; r_1)$$

with $m_0 \neq m_1$, which contradicts the fact that Commit is perfectly binding.

First Hybrid Step Consider the hybrid experiment $\mathbf{H}_{m_0,m_1}(\lambda, b, q)$ that is identical to $\mathbf{T}_{m_0,m_1}(\lambda, b, q)$, except that we let the auxiliary code $(\mathsf{Enc}', \mathsf{Dec}')$ encode a random string $s' \in \{0,1\}^{k+\rho}$ (instead of the string $m||r$). The experiment is described formally in Fig. 2.

We will now prove that, as long as the number of tampering queries is polynomial, the above hybrid experiment is statistically close to the original experiment. The proof is by induction on the number of tampering queries $q(\lambda) \in \text{poly}(\lambda)$. The lemma below constitutes the induction basis.

Lemma 2. *For all messages $m_0, m_1 \in \{0,1\}^k$, for all values $b \in \{0,1\}$, and for all unbounded adversaries A, we have that*

$$\{\mathbf{T}_{m_0,m_1}(\lambda, b, 1)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{H}_{m_0,m_1}(\lambda, b, 1)\}_{\lambda\in\mathbb{N}}.$$

Proof. We show the proof for the case $b = 0$, the proof for the other case being analogous. Assume that there exist a pair of messages $m_0, m_1 \in \{0,1\}^k$, an unbounded adversary A, an unbounded distinguisher D, and a polynomial $p(\cdot)$ such that, for infinitely many values of $\lambda \in \mathbb{N}$, we have

$$|\mathbb{P}\left[\mathsf{D}(\mathbf{T}_{m_0,m_1}(\lambda,0,1)) = 1\right] - \mathbb{P}\left[\mathsf{D}(\mathbf{H}_{m_0,m_1}(\lambda,0,1)) = 1\right]| \geq 1/p(\lambda).$$

Note that the probabilities in the above equation are taken over the random coin tosses of (A,D), over the choice of $r \leftarrow\!\!\$\, \{0,1\}^\rho$ and $s' \leftarrow\!\!\$\, \{0,1\}^{k+\rho}$, and over the randomness of algorithm Enc'. By an averaging argument, this means that there must exist at least two values $r \in \{0,1\}^\rho$ and $s' \in \{0,1\}^{k+\rho}$ such that the above equation holds when we fix these particular values of r and s'. We build an unbounded adversary A' and an unbounded distinguisher D' such that

$$\left|\mathbb{P}\left[\mathsf{D}'(\mathbf{T}'_{s'_0,s'_1}(\lambda,0,1)) = 1\right] - \mathbb{P}\left[\mathsf{D}'(\mathbf{T}'_{s'_0,s'_1}(\lambda,1,1)) = 1\right]\right| \geq 1/p(\lambda) - \nu(\lambda),$$

where $s'_0 := m_0||r$ and $s'_1 := s'$, $\nu(\lambda) \in \mathrm{negl}(\lambda)$ is a negligible function, and where we wrote $\mathbf{T}'_{s'_0,s'_1}(\lambda,b,1)$ as a shorthand for $\mathbf{Tamper}^{\Pi',\mathsf{A}'}_{s'_0,s'_1}(\lambda,b,1)$. This will contradict the one-time unconditional non-malleability of $(\mathsf{Enc}',\mathsf{Dec}')$, and thus will conclude the proof of the lemma.

Let $c' := (c'_0, c'_1)$ be the target encoding in the tampering experiment relative to $(\mathsf{Enc}',\mathsf{Dec}')$. Here, c' is either an encoding of s'_0 or an encoding of s'_1. Adversary A', on input $(1^\lambda, m_0, s'_0, s'_1)$, proceeds as follows:

- Parse $s'_0 := m_0||r$ and compute $\gamma := \mathsf{Commit}(m_0; r)$.
- Run $\mathsf{A}(1^\lambda)$, obtaining a pair of polynomial-time computable functions (ϕ_0, ϕ_1), where $\phi_0, \phi_1 : \{0,1\}^{n'+\ell} \to \{0,1\}^{n'+\ell}$.
- Define the polynomial-time computable leakage function ψ'_0 (resp. ψ'_1) that hardwires γ and ϕ_0 (resp. ϕ_1), and, upon input c'_0 (resp. c'_1) returns the value $\tilde{\gamma}_0$ (resp. $\tilde{\gamma}_1$) defined by $\phi_0(\gamma, c'_0) := (\tilde{\gamma}_0, \tilde{c}'_0)$ (resp. $\phi_1(\gamma, c'_1) := (\tilde{\gamma}_1, \tilde{c}'_1)$).
- Forward ψ'_0 to $\mathcal{O}^\ell(c'_0)$ and ψ'_1 to $\mathcal{O}^\ell(c'_1)$, obtaining values $\tilde{\gamma}_0, \tilde{\gamma}_1$.
- Define the polynomial-time computable tampering function ϕ'_0 (resp. ϕ'_1) that hardwires γ and ϕ_0 (resp. ϕ_1), and, upon input c'_0 (resp. c'_1), returns the value $\tilde{c}'_0$ (resp. $\tilde{c}'_1$) defined by $\phi'_0(\gamma, c'_0) := (\tilde{\gamma}_0, \tilde{c}'_0)$ (resp. $\phi'_1(\gamma, c'_1) := (\tilde{\gamma}_1, \tilde{c}'_1)$).
- Forward (ϕ_0, ϕ_1) to $\mathcal{O}^{s'_0,s'_1}_{\mathrm{cnm}}$, obtaining a value $\tilde{s} \in \{0,1\}^k \cup \{\bot, \mathtt{same}^*\}$.

Notice that attacker A' asks a single (split-state) leakage query yielding exactly 2ℓ bits, and a single (split-state) tampering query, as required. Distinguisher D', upon input $(1^\lambda, m_0, m_1, r, s')$, and upon receiving a pair $(\tilde{\gamma}_0, \tilde{\gamma}_1)$ in response of A's leakage query, and a value $\tilde{s} \in \{0,1\}^{k+\rho} \cup \{\mathtt{same}^*, \bot\}$ in response of A's tampering query, proceeds as follows.

- If $\tilde{s} = \bot$, return $\mathsf{D}(\bot)$.
- If $\tilde{s} = \mathtt{same}^*$:
  - In case $\tilde{\gamma}_0 = \tilde{\gamma}_1 = \mathsf{Commit}(m_0; r)$, return $\mathsf{D}(\mathtt{same}^*)$;
  - Else return $\mathsf{D}(\bot)$.
- If $\tilde{s} \notin \{\mathtt{same}^*, \bot\}$:

- Parse $\tilde{s} := \tilde{m}||\tilde{r}$;
- In case $\tilde{\gamma}_0 \neq \mathsf{Commit}(\tilde{m}; \tilde{r})$ or $\tilde{\gamma}_1 \neq \mathsf{Commit}(\tilde{m}; \tilde{r})$, return $\mathsf{D}(\perp)$;
- In case $\tilde{m} \in \{m_0, m_1\}$ return $\mathsf{D}(\mathtt{same}^*)$;
- Else return $\mathsf{D}(\tilde{m})$.

For the analysis, we next prove that the simulation performed by $(\mathsf{A}', \mathsf{D}')$ is perfect with overwhelming probability. First, depending on the target encoding (c'_0, c'_1) being either an encoding of s'_0 or an encoding of s'_1, the view of A's tampering functions is identical to the distribution of the target codeword in either experiment $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ or $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$, with our fixed choice of r and s'. Second, the view of D is simulated correctly, with all but a negligible probability. Indeed:

- If $(\tilde{c}'_0, \tilde{c}'_1)$ yields $\perp$, both $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ and $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$ would return $\perp$, which is perfectly emulated by the reduction.
- If $(\tilde{c}'_0, \tilde{c}'_1)$ yields $\mathtt{same}^*$, it means that the inner codeword decodes to either $s'_0 = m_0||r$ or to $s'_1 = s' := m'||r'$. Without loss of generality, assume further that the commitments in the tampered share satisfy $\tilde{\gamma}_0 = \tilde{\gamma}_1 := \tilde{\gamma}$. (In fact, if this is not the case, both experiments return $\perp$, which is once again perfectly emulated by the reduction.) There are 4 possible cases: either both experiments output s'_0, or both experiments output s'_1, or one experiment outputs s'_0 while the other outputs s'_1. However, since the view in the real experiment is independent of the value m', we can condition on the event that the real experiment does not output s'. Thus, there are only two cases to consider:
  (i) Experiment $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ and $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$ both return $s'_0 = m_0||r$.
  (ii) Experiment $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ returns $s'_0 = m_0||r$, but $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$ returns $s'_1 = m'||r'$.

  In both cases, the output of experiments $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ and $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$ is equal to $\mathtt{same}^*$ if $\tilde{\gamma} = \gamma$, and else both experiments return $\perp$. This is exactly what the reduction does. So, depending on the target codeword being either an encoding of s'_0 or an encoding of s'_1, the reduction simulates, except with negligible probability 2^{-k}, the outcome of either experiment $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ or $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$.
- If $(\tilde{c}'_0, \tilde{c}'_1)$ yields some value $\tilde{s} = \tilde{m}||\tilde{r} \notin \{\mathtt{same}^*, \perp\}$, it means in particular that $\tilde{s} \notin \{s'_0, s'_1\}$. In such a case both experiments $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ and $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$ would return $\perp$ in case the modified commitments $\tilde{\gamma}_0, \tilde{\gamma}_1$ do not match the opening $\tilde{m}, \tilde{r}$. Otherwise, it means that the modified codeword produced by A leads to a valid encoding of some message $\tilde{m} \in \{0,1\}^k$. Hence, the output of both experiments would either be $\mathtt{same}^*$ or $\tilde{m}$ (depending on $\tilde{m}$ being equal to one of the two messages m_0, m_1 or not).

To summarize, depending on the target encoding (c'_0, c'_1) being either an encoding of $s'_0 := m_0||r$ or an encoding of $s'_1 := s'$, the view of (A, D) is identical, except with negligible probability, to the view in either experiment $\mathbf{T}_{m_0,m_1}(\lambda, 0, 1)$ or $\mathbf{H}_{m_0,m_1}(\lambda, 0, 1)$, for our fixed choice of r and s'. Thus, the advantage of $(\mathsf{A}', \mathsf{D}')$ is negligibly close to that of (A, D). This concludes the proof of the lemma.

The next lemma constitutes the inductive step. The proof appears in the full version.

Lemma 3. *Assume that for all messages $m_0, m_1 \in \{0,1\}^k$, for all $b \in \{0,1\}$, and for all unbounded adversaries A, it holds that*

$$\{\mathbf{T}_{m_0,m_1}(\lambda, b, i)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{H}_{m_0,m_1}(\lambda, b, i)\}_{\lambda\in\mathbb{N}},$$

where $i \in [q-1]$ and $q \in \mathrm{poly}(\lambda)$. Then, for all messages $m_0, m_1 \in \{0,1\}^k$, for all $b \in \{0,1\}$, and for all unbounded adversaries A, we have that

$$\{\mathbf{T}_{m_0,m_1}(\lambda, b, i+1)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{H}_{m_0,m_1}(\lambda, b, i+1)\}_{\lambda\in\mathbb{N}},$$

By combining Lemma 2 and Lemma 3, we have shown that the hybrid experiment of Fig. 2 is statistically indistinguishable from the original tampering experiment:

Lemma 4. *For all messages $m_0, m_1 \in \{0,1\}^k$, for all values $b \in \{0,1\}$, for all $q(\lambda) \in \mathrm{poly}(\lambda)$, and for all unbounded adversaries A, we have that*

$$\{\mathbf{T}_{m_0,m_1}(\lambda, b, q)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{H}_{m_0,m_1}(\lambda, b, q)\}_{\lambda\in\mathbb{N}}.$$

Second Hybrid Step Finally, we show that the view in experiment $\mathbf{H}_{m_0,m_1}(\lambda, b, q)$ is (computationally) independent of the hidden bit $b \in \{0,1\}$. The proof appears in the full version.

Lemma 5. *For all messages $m_0, m_1 \in \{0,1\}^k$, for all $q(\lambda) \in \mathrm{poly}(\lambda)$, and for all PPT adversaries A, we have that*

$$\{\mathbf{H}_{m_0,m_1}(\lambda, 0, q)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_{m_0,m_1}(\lambda, 1, q)\}_{\lambda\in\mathbb{N}}.$$

Putting it Together By combining Lemma 4 and Lemma 5, we obtain that for all $m_0, m_1 \in \{0,1\}^k$, for all $q(\lambda) \in \mathrm{poly}(\lambda)$, and for all PPT adversaries A:

$$\begin{aligned}\{\mathbf{T}_{m_0,m_1}(\lambda, 0, q)\}_{\lambda\in\mathbb{N}} &\approx_s \{\mathbf{H}_{m_0,m_1}(\lambda, 0, q)\}_{\lambda\in\mathbb{N}} \\ &\approx_c \{\mathbf{H}_{m_0,m_1}(\lambda, 1, q)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{T}_{m_0,m_1}(\lambda, 1, q)\}_{\lambda\in\mathbb{N}},\end{aligned}$$

which concludes the proof of the theorem.

5 Uniqueness $\not\Rightarrow$ Continuous Non-Malleability

As mentioned earlier, the property of message uniqueness is necessary for constructing continuously non-malleable codes in the split-state model. It is a natural question whether message uniqueness is also sufficient, namely any split-state code that satisfies message uniqueness and one-time non-malleability is also continuously non-malleable.

Here, we give a negative answer to the above question, by exhibiting a contrived split-state code that satisfies both message uniqueness and one-time non-malleability, but can be broken with a simple continuous attack. The constructed code makes black-box use of any split-state code satisfying both (perfect) message uniqueness and computational one-time non-malleability (as, e.g., our encoding scheme from §4). Our counter-example is "tight", in the sense that the attack breaking continuous non-malleability requires only two tampering queries.

The code. Consider the following split-state $(k, 4n+2k)$-code $\Pi^* = (\mathsf{Enc}^*, \mathsf{Dec}^*)$, based on an auxiliary split-state (k, n)-code $\Pi = (\mathsf{Enc}, \mathsf{Dec})$. The properties we require from each building block are directly stated in Theorem 4 below.

Encoding: Upon input a value $m \in \{0,1\}^k$, sample a random string $\kappa \leftarrow\!\!\$\ \{0,1\}^k$, compute $\delta := m \oplus \kappa$, and return the codeword

$$c^* = (c_0^*, c_1^*) := ((c_0^1, c_0^2, \delta), (c_1^1, c_1^2, \delta)), \tag{2}$$

where $(c_0^1, c_1^1) \leftarrow\!\!\$\ \mathsf{Enc}(m)$ and $(c_0^2, c_1^2) \leftarrow\!\!\$\ \mathsf{Enc}(\kappa)$.

Decoding: Upon input a codeword $c^* \in \{0,1\}^{4n+2k}$, parse $c^* := (c_0^*, c_1^*)$ as defined in Eq. (2) and return the same as $\mathsf{Dec}(c_0^1, c_1^1)$.

Note that the decoding process simply decodes the first encoding (c_0^1, c_1^1) contained in c^*, completely ignoring the rest of the codeword.

Theorem 4 (Theorem 2, restated). *Assume that $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is a split-state (k, n)-code satisfying (perfect) message uniqueness and (computational) one-time non-malleability. Then the encoding scheme $\Pi^* = (\mathsf{Enc}^*, \mathsf{Dec}^*)$ described above is a split-state $(k, 4n + 2k)$-code meeting the following conditions:*

(i) Π^* satisfies (perfect) message uniqueness;
(ii) Π^* satisfies (computational) one-time non-malleability;
(iii) Π^* is not 2-non-malleable.

Proof overview. The proof of Theorem 4 can be found in the full version. Here, we give the main intuition. The proof of property (i) follows almost directly by message uniqueness of Π. As for the proof of property (iii), it is sufficient to consider the tampering function that simply swaps c_0^1 with c_0^2 and c_1^1 with c_1^2. Note that the decoded message corresponding to such a query is equal to the value κ; hence, we can hard-wire κ in the second tampering query which allows to unmask the message computing $m = \delta \oplus \kappa$ and thus encode a related value.

To prove property (ii) we consider two hybrid experiments $\mathbf{H}_1^*$ and $\mathbf{H}_2^*$, and show that $\mathbf{T}_0^* \approx_c \mathbf{H}_1^* \approx_c \mathbf{H}_2^* \approx_c \mathbf{T}_1^*$ where $\mathbf{T}_b^*$ denotes the random variable corresponding to the non-malleability experiment with Π^* using hidden bit $b \in \{0,1\}$. Here, the difference between $\mathbf{T}_0^*$ and $\mathbf{H}_1^*$ is that in the latter we replace the codeword (c_0^1, c_1^1) with an encoding of m_1 (instead of m_0); in $\mathbf{H}_2^*$, instead, we change the distribution of δ to $\delta := m_1 \oplus \kappa$ and additionally we now let (c_0^2, c_1^2) be an encoding of $\kappa' := \kappa \oplus m_0 \oplus m_1$. To argue the indistinguishability of the hybrids, we then proceed as follows:

– In a first step we show that $\mathbf{T}_0^* \approx_c \mathbf{H}_1^*$, down to the non-malleability of the underlying encoding scheme Π. The reduction has access to a target codeword $c^1 = (c_0^1, c_1^1)$ that is either an encoding of m_0 or an encoding of m_1, and, given m_0, it can perfectly simulate the distribution of a target codeword for either experiment $\mathbf{T}_0^*$ or $\mathbf{H}_1^*$ inside the tampering oracle $\mathcal{O}_{\text{cnm}}^{m_0,m_1}(c^1, \cdot)$. To do so, the reduction can sample offline a random κ, define $\delta = m_0 \oplus \kappa$, and set $c^2 := (c_0^2, c_1^2)$ to be an encoding of κ.
  Notice that the reduction gets to see the output of the decoding corresponding to the modified pair $(\tilde{c}_0^1, \tilde{c}_1^1)$, which is a perfect simulation for the output of either experiment $\mathbf{T}_0^*$ or $\mathbf{H}_1^*$.
– In a second step we show that $\mathbf{H}_1^* \equiv \mathbf{H}_2^*$; this is because if κ is random so is κ', and moreover $\kappa' \oplus m_0 = m_1 \oplus \kappa$; thus the two distributions are identical.
– In a third step we show that $\mathbf{H}_2^* \approx_c \mathbf{T}_1^*$, down to the non-malleability of the underlying encoding scheme Π. The reduction has access to a target codeword $c^2 = (c_0^2, c_1^2)$ that is either an encoding of $\kappa' := \kappa \oplus m_0 \oplus m_1$ or an encoding of κ, and, as before, it can perfectly simulate the distribution of the target codeword in either experiment $\mathbf{H}_2^*$ or $\mathbf{T}_1^*$ inside the tampering oracle $\mathcal{O}_{\text{cnm}}^{\kappa,\kappa'}(c^2, \cdot)$. After computing the codeword $((\tilde{c}_0^1, \tilde{c}_0^2, \tilde{\delta}_0), (\tilde{c}_1^1, \tilde{c}_1^2, \tilde{\delta}_1))$, the tampering function defined by the reduction swaps $\tilde{c}_0^1$ with $\tilde{c}_0^2$ and $\tilde{c}_1^1$ with $\tilde{c}_1^2$; this way it obtains the decoding of $(\tilde{c}_0^1, \tilde{c}_1^1)$, which is what one needs in order to simulate the output of the two experiments.
  An additional difficulty is that the experiment in which the reduction runs is parameterized by messages (κ, κ'), whereas the emulated experiments $\mathbf{H}_2^*$ or $\mathbf{T}_1^*$ are parameterized by (m_0, m_1). This means, for instance, that if the reduction obtains $\texttt{same}^*$ it cannot directly conclude that the simulated output should also be $\texttt{same}^*$, but it needs to carefully adjust the received output in order to make the simulation go through.

6 Conclusion and Open Problems

We have shown a construction of a split-state continuously non-malleable code in the plain model. Our construction can be instantiated under the assumption that one-to-one one-way functions exist. Additionally, we have clarified that message uniqueness, albeit being necessary for obtaining continuous non-malleability in the split-state model, is not sufficient for constructing such codes.

Interesting open questions related to our work are, for instance, whether continuous non-malleability can be achieved, under minimal assumptions, together with additional properties, such as strong non-malleability [27], super-non-malleability [32], augmented non-malleability [2], and locality [22,13,21], or whether the rate of our code construction can be improved.

References

1. Aggarwal, D.: Affine-evasive sets modulo a prime. Inf. Process. Lett. **115**(2), 382–385 (2015)

2. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: TCC. pp. 393–417 (2016)
3. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: ACM STOC. pp. 459–468 (2015)
4. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: ACM STOC. pp. 774–783 (2014)
5. Aggarwal, D., Dottling, N., Nielsen, J.B., Obremski, M., Purwanto, E.: Continuous non-malleable codes in the 8-split-state model. Cryptology ePrint Archive, Report 2017/357 (2017), https://eprint.iacr.org/2017/357
6. Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. In: TCC. pp. 398–426 (2015)
7. Aggarwal, D., Kazana, T., Obremski, M.: Inception makes non-malleable codes stronger. In: TCC. pp. 319–343 (2017)
8. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In: CRYPTO. pp. 538–557 (2015)
9. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In: TCC. pp. 375–397 (2015)
10. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes for bounded depth, bounded fan-in circuits. In: EUROCRYPT. pp. 881–908 (2016)
11. Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes from average-case hardness: AC0, decision trees, and streaming space-bounded tampering. In: EUROCRYPT. pp. 618–650 (2018)
12. Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In: ICALP. pp. 31:1–31:14 (2016)
13. Chandran, N., Kanukurthi, B., Raghuraman, S.: Information-theoretic local non-malleable codes and their applications. In: TCC. pp. 367–392 (2016)
14. Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. In: ACM STOC. pp. 285–298 (2016)
15. Chattopadhyay, E., Li, X.: Non-malleable codes and extractors for small-depth circuits, and affine functions. In: ACM STOC. pp. 1171–1184 (2017)
16. Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: IEEE FOCS. pp. 306–315 (2014)
17. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In: Innovations in Theoretical Computer Science. pp. 155–168 (2014)
18. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. In: TCC. pp. 440–464 (2014)
19. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. In: TCC. pp. 306–335 (2016)
20. Coretti, S., Maurer, U., Tackmann, B., Venturi, D.: From single-bit to multi-bit public-key encryption via non-malleable codes. In: TCC. pp. 532–560 (2015)
21. Dachman-Soled, D., Kulkarni, M., Shahverdi, A.: Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In: PKC. pp. 310–332 (2017)
22. Dachman-Soled, D., Liu, F., Shi, E., Zhou, H.: Locally decodable and updatable non-malleable codes and their applications. In: TCC. pp. 427–450 (2015)
23. Davì, F., Dziembowski, S., Venturi, D.: Leakage-resilient storage. In: SCN. pp. 121–137 (2010)

24. Dodis, Y., Lewko, A.B., Waters, B., Wichs, D.: Storing secrets on continually leaky devices. In: IEEE FOCS. pp. 688–697 (2011)
25. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: CRYPTO. pp. 239–257 (2013)
26. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: IEEE FOCS. pp. 293–302 (2008)
27. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Innovations in Computer Science. pp. 434–452 (2010)
28. Faonio, A., Nielsen, J.B., Simkin, M., Venturi, D.: Continuously non-malleable codes with split-state refresh. In: ACNS. pp. 1–19 (2018)
29. Faust, S., Hostáková, K., Mukherjee, P., Venturi, D.: Non-malleable codes for space-bounded tampering. In: CRYPTO. pp. 95–126 (2017)
30. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: TCC. pp. 465–488 (2014)
31. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: A tamper and leakage resilient von Neumann architecture. In: PKC. pp. 579–603 (2015)
32. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In: EUROCRYPT. pp. 111–128 (2014)
33. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: TCC. pp. 258–277 (2004)
34. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991)
35. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: ACM STOC. pp. 1128–1141 (2016)
36. Jafargholi, Z., Wichs, D.: Tamper detection and continuous non-malleable codes. In: TCC. pp. 451–480 (2015)
37. Li, X.: Improved non-malleable extractors, non-malleable codes and independent source extractors. In: ACM STOC. pp. 1144–1156 (2017)
38. Liu, F., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: CRYPTO. pp. 517–532 (2012)
39. Mahmoody, M., Pass, R.: The curious case of non-interactive commitments - on the power of black-box vs. non-black-box use of primitives. In: CRYPTO. pp. 701–718 (2012)
40. Matsuda, T., Hanaoka, G.: An asymptotically optimal method for converting bit encryption to multi-bit encryption. In: ASIACRYPT. pp. 415–442 (2015)
41. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: CRYPTO. pp. 57–74 (2008)
42. Pass, R.: Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In: TCC. pp. 334–354 (2013)
43. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: IEEE FOCS. pp. 563–572 (2005)
44. Pass, R., Rosen, A.: Concurrent nonmalleable commitments. SIAM J. Comput. **37**(6), 1891–1925 (2008)
45. Pass, R., Rosen, A.: New and improved constructions of nonmalleable cryptographic protocols. SIAM J. Comput. **38**(2), 702–752 (2008)
46. Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: EUROCRYPT. pp. 415–431 (1999)