Domain-Extension for Public-Key Encryption via Non-Malleable Codes

Sandro Coretti*
ETH Zurich
corettis@inf.ethz.ch

Ueli Maurer
ETH Zurich
maurer@inf.ethz.ch

Björn Tackmann
ETH Zurich
bjoernt@inf.ethz.ch

Daniele Venturi
Sapienza University of Rome
venturi@di.uniroma1.it

October 6, 2014

Abstract

One approach towards basing public-key encryption (PKE) schemes on weak and credible assumptions is to build "stronger" or more general schemes generically from "weaker" or more restricted ones. One particular line of work in this context was initiated by Myers and shelat (FOCS '09) and continued by Hohenberger, Lewko, and Waters (Eurocrypt '12), who provide constructions of multi-bit CCA-secure PKE from single-bit CCA-secure PKE.

It is well-known that encrypting each bit of a plaintext string independently is not CCA-secure—the resulting scheme is *malleable*. We therefore investigate whether this malleability can be dealt with using the conceptually simple approach of applying a suitable non-malleable code (Dziembowski *et al.*, ICS '10) to the plaintext and subsequently encrypting the resulting codeword bit-by-bit. We find that an attacker's ability to ask multiple decryption queries requires that the underlying code be *continuously* non-malleable (Faust *et al.*, TCC '14). Since, as we show, this flavor of non-malleability can only be achieved if the code is allowed to "self-destruct," the resulting scheme inherits this property and therefore only achieves a weaker variant of CCA security.

We formalize this new notion of so-called *self-destruct CCA security (SD-CCA)* as CCA security with the restriction that the decryption oracle stops working once the attacker submits an invalid ciphertext. We first show that the above approach based on non-malleable codes yields a solution to the problem of domain extension for SD-CCA-secure PKE, provided that the underlying code is continuously non-malleable against a *reduced* form of bit-wise tampering. Then, we prove that the code of Dziembowski *et al.* is actually already continuously non-malleable against (even *full*) bit-wise tampering; this constitutes the first *information-theoretically* secure continuously non-malleable code, a technical contribution that we believe is of independent interest. Compared to the previous approaches to PKE domain extension, our scheme is more efficient and intuitive, at the cost of not achieving full CCA security. Our result is also one of the first applications of non-malleable codes in a context other than memory tampering.

*Corresponding author.

1 Introduction

1.1 Overview

A public-key encryption (PKE) scheme enables a sender A to send messages to a receiver B confidentially if B can send a single message, the public key, to A authentically. A encrypts a message with the public key and sends the ciphertext to B via a channel that could be authenticated or insecure, and B decrypts the received ciphertext using the private key. Following the seminal work of Diffie and Hellman [15], the first formal definition of public-key encryption has been provided by Goldwasser and Micali [25], and to date numerous instantiations of this concept have been proposed, e.g., [42, 18, 12, 22, 26, 28, 43, 41], for different security properties and based on various different computational assumptions.

One natural approach towards developing public-key encryption schemes based on weak and credible assumptions is to build "stronger" or more general schemes generically from "weaker" or less general ones. While the "holy grail"—generically building a chosen-ciphertext secure scheme based on any chosen-plaintext secure one—has so far remained out of reach, and despite negative results [24], various interesting positive results have been shown. For instance, Cramer *et al.* [11] build *bounded-query* chosen-ciphertext secure schemes from chosen-plaintext secure ones, Choi *et al.* [6] *non-malleable* schemes from chosen-plaintext secure ones, and Lin and Tessaro [30] show how the security of weakly chosen-ciphertext secure schemes can be amplified. A line of work started by Myers, Sergi, and shelat [39] and continued by Dachman-Soled [13] shows how to obtain chosen-ciphertext secure schemes from plaintext-aware ones. Most relevant for our work, however, are the results of Myers and shelat [40] and Hohenberger, Lewko, and Waters [27], which generically build a multi-bit chosen-ciphertext secure scheme from a single-bit chosen-ciphertext secure one.

A naïve attempt at solving this problem would be to encrypt each bit m_i of a plaintext $m = m_1 \cdots m_k$ under an independent public key pk_i of the single-bit scheme. Unfortunately, this simple approach does not yield chosen-ciphertext security. The reason is that the above scheme is *malleable*: given a ciphertext $e = (e_1, \ldots, e_k)$, where e_i is an encryption of m_i, an attacker can generate a new ciphertext $e' \neq e$ that decrypts to a related message, for instance by copying the first ciphertext component e_1 and replacing the other components by fresh encryptions of, say, 0.

The above malleability issue suggests the following natural "encode-then-encrypt-bit-by-bit" approach: first encode the message using a non-malleable code[1] (a concept introduced by Dziembowski *et al.* [17]) to protect its integrity, obtaining an n-bit codeword $c = c_1 \cdots c_n$; then encrypt each bit c_i of the codeword using public key pk_i as in the naïve protocol from above.

It turns out that non-malleable codes as introduced by [17] are not sufficient: Since they are only secure against a single tampering, the security of the resulting scheme would only hold with respect to a single decryption. *Continuously* non-malleable codes (Faust *et al.* [19]) allow us to extend this guarantee to multiple decryptions. However, such codes "self-destruct" once an attack has been detected, and, therefore, so must any PKE scheme built on top of them. This is a restriction that we prove to be unavoidable for this approach based on non-malleable codes.

The resulting scheme achieves a notion weaker than full CCA, which we term *self-destruct chosen-ciphertext security (SD-CCA)*. Roughly, SD-CCA security is CCA security with the twist that the decryption oracle stops working once the adversary submits an invalid ciphertext.

Our paper consists of two main parts: First, we prove that the above approach allows to build multi-bit SD-CCA-secure PKE from single-bit SD-CCA-secure PKE, provided that the underlying code is continuously non-malleable against a *reduced* form of bit-wise tampering. This proof is greatly facilitated by rephrasing the problem using the paradigm of constructive cryptography [32], since it follows almost immediately from the composition theorem. For comparison we also provide a purely game-based proof. Second, we show that a simplified variant of the code by Dziembowski *et al.* [17] is already continuously non-malleable against the aforementioned reduced bit-wise tampering and that

[1] Roughly, a code is non-malleable w.r.t. a function class $\mathcal{F}$ if the message obtained by decoding a codeword modified via a function in $\mathcal{F}$ is either the original message or a completely unrelated value.

the full variant of said code achieves continuous non-malleability against *full* bit-wise tampering. This constitutes the first *information-theoretically* secure continuously non-malleable code, a contribution that we believe is of independent interest, and forms the technical core of this paper.

1.2 Techniques and Contributions

Constructive cryptography [32]. Security statements for cryptographic schemes can be stated as *constructions* of a "stronger" or more useful desired resource from a "weaker" or more restricted assumed one. Two such construction steps can be composed, i.e., if a protocol π constructs a resource S from an assumed resource R, denoted by $R \overset{\pi}{\Longrightarrow} S$, and, additionally, a protocol ψ assumes resource S and constructs a resource T, then the composition theorem of constructive cryptography (see Appendix A) states that the composed protocol, denoted $\psi \circ \pi$, constructs resource T from R. The resources considered in this work are different types of communication channels between two parties A and B; a channel is a resource that involves three entities: the sender, the receiver, and a (potential) attacker E.

We use and extend the notation by [36], denoting different types of channels by different arrow symbols. A *confidential* channel (later denoted $-\!\diamond\!\!\twoheadrightarrow\!\bullet$) hides the messages sent by A from the attacker E but potentially allows her to inject *independent* messages; an *authenticated* channel (later denoted $\bullet\!-\!\diamond\!\!\twoheadrightarrow$) is dual to the confidential channel in that it potentially leaks the message to the attacker but prevents modifications and injections; an insecure channel (later denoted $-\!\!\twoheadrightarrow$) protects neither the confidentiality nor the authenticity. In all cases, the double arrow head indicates that the channel can be used to transmit multiple messages. A single arrow head, instead, means that channels are single-use. All channels used within this work are described formally in Appendix B.

Warm-up: dealing with the malleability of the one-time pad. To illustrate the intuition behind our approach, consider the following simple example: The one-time pad allows to encrypt an n-bit message m using an n-bit shared key κ by computing the ciphertext $e = m \oplus \kappa$. If e is sent via an insecure channel, an attacker can replace it by a different ciphertext e', in which case the receiver will compute $m' = e' \oplus \kappa = m \oplus (e \oplus e')$. This can be seen, as described in [35], as constructing from an insecure channel and a shared secret n-bit key an "XOR-malleable" channel, denoted $-\!\!\oplus\!\!\rightarrow\!\bullet$, which is confidential but allows the attacker to specify a mask $\delta \in \{0,1\}^n$ $(= e \oplus e')$ to be XORed to the transmitted message.

Non-malleable codes can be used to deal with the XOR-malleability. To transmit a k-bit message m, we encode m with a (k,n)-bit non-malleable code, obtaining an n-bit codeword c, which we transmit via the XOR-malleable channel $-\!\!\oplus\!\!\rightarrow\!\bullet$. Since by XORing a mask δ to a codeword transmitted via $-\!\!\oplus\!\!\rightarrow\!\bullet$ the attacker can influence the value of each bit of the codeword only independently, a code that is non-malleable w.r.t. the function class $\mathcal{F}_{\mathsf{bit}}$, which (in particular) allows to either "keep" or "flip" each bit of a codeword only individually, is sufficient. Indeed, the non-malleability of the code implies that the decoded message will be either the original message or a completely unrelated value, which is the same guarantee as formulated by the single-message confidential channel (denoted $\longrightarrow\!\bullet$), and hence using the code, one achieves the construction

$$-\!\!\oplus\!\!\rightarrow\!\bullet \quad \Longrightarrow \quad \longrightarrow\!\bullet.$$

A more detailed treatment and a formalization of this example appears in Appendix C; suitable non-malleable codes are described in [10, 17, 5].

Dealing with the malleability of multiple single-bit encryptions. Intuitively, CCA encryption guarantees that an attacker, by modifying a particular ciphertext, can either leave the message contained therein intact or replace it by an independently created one. This intuition is formally captured by the confidential channel $-\!\diamond\!\!\twoheadrightarrow\!\bullet$: at the attacker interface E, it allows to either forward messages sent by A or to inject *independent* messages. In [9], it is shown how CCA-secure encryption

can be used to construct a confidential channel $\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet$ from A to B from an authenticated channel $\longleftarrow\!\bullet$ from B to A and an insecure channel $-\!\!\twoheadrightarrow$ from A to B. As shown in Section 3, this and the composition theorem imply that using n independent single-bit PKE schemes, one can construct n (independent) instances of the single-bit confidential channel $\overset{\text{1-bit}}{\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet}$, written $\left[\overset{\text{1-bit}}{\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet}\right]^n$.

The remaining step is showing how to achieve the construction

$$\left[\overset{\text{1-bit}}{\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet}\right]^n \quad \Longrightarrow \quad \overset{k\text{-bit}}{\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet} \tag{1}$$

for some $k > 1$. Then, by the composition theorem, plugging these two steps together yields a protocol $\mathsf{m\text{-}pke}$ that constructs a k-bit confidential channel from an authenticated channel and an insecure channel.

To achieve construction (1), we use non-malleable codes. The fact that the channels are multiple-use leads to two important differences to the one-time-pad example above: First, the attacker can fabricate *multiple* codewords, which are then decoded. Second, each bit of such a codeword can be created by combining *any* of the bits sent by A over the corresponding channel. These capabilities can be formally captured by a particular class $\mathcal{F}_{\mathsf{copy}}$ of tampering functions. We prove in Section 3 that any code that is *continuously non-malleable* w.r.t. $\mathcal{F}_{\mathsf{copy}}$ can be used to achieve (1).

Unfortunately, we show in Appendix F that any code, in order to satisfy the above type of non-malleability, has to "self-destruct" in the event of a decoding error. For the application in the setting of public-key encryption, this means that the decryption algorithm of the receiver B also has to deny processing any further ciphertext once the code self-destructs.

Self-destruct CCA security. In Section 3 we show how the protocol $\mathsf{m\text{-}pke}$ can be seen as a PKE scheme that achieves *self-destruct CCA security (SD-CCA)* and show that the single-bit confidential channel $\overset{\text{1-bit}}{\multimap\!\diamond\!\!\twoheadrightarrow\!\bullet}$ can also be constructed using a single-bit SD-CCA scheme (instead of a CCA-secure one). Thus, overall we obtain a way to transform 1-bit SD-CCA-secure PKE into multi-bit SD-CCA-secure PKE. For comparison we also provide a direct, entirely game-based proof that combining a single-bit SD-CCA PKE scheme with a non-malleable code as above yields a multi-bit SD-CCA scheme (see Appendix D).

SD-CCA is a (weaker) CCA variant that allows the scheme to self-destruct in case it detects an invalid ciphertext. The standard CCA game can easily be extended to include the self-destruct mode of the decryption: the decryption oracle keeps answering decryption queries as long as no invalid ciphertext (i.e., a ciphertext upon which the decryption algorithm outputs an error symbol) is received; after such an event occurs, no further decryption query is answered.

The guarantees of SD-CCA are perhaps best understood if compared to the q-bounded CCA notion by [6]. While q-CCA allows an *a priori determined* number q of decryption queries, SD-CCA allows an *arbitrary* number of *valid* decryption queries and one invalid query. From a practical viewpoint, an attacker can efficiently violate the availability with a scheme of either notion. However, as long as no invalid ciphertexts are received, an SD-CCA scheme can run indefinitely, whereas a q-CCA scheme has to necessarily stop after q decryptions.

Subsequent work [8] shows that SD-CCA security can in fact be achieved from CPA security only, by generalizing a technique by Choi *et al.* [6]. The resulting scheme, however, is considerably less efficient than the one we provide in this paper. In [8], the authors also study the relation between SD-CCA and other standard security notions and discuss possible applications.

Continuous non-malleability w.r.t. $\mathcal{F}_{\mathsf{copy}}$. The class $\mathcal{F}_{\mathsf{copy}}$ can be seen as a multi-encoding version of the function class $\mathcal{F}_{\mathsf{set}}$, which consists of functions that tamper with every bit of an encoding individually and may either leave it unchanged or replace it by a fixed value. In Section 4 we build a continuously non-malleable code w.r.t. $\mathcal{F}_{\mathsf{copy}}$; the code consists of a linear error-correcting secret sharing (LECSS) scheme and can be seen as a simplified version of the code in [17]. The security proof of the code proceeds in two steps: First, we prove that it is continuously non-malleable w.r.t. $\mathcal{F}_{\mathsf{set}}$ against tampering with a single encoding; the main challenge in this proof is showing that by repeatedly

tampering with an encoding, an attacker cannot infer (too much) useful information about it. Then, we show that if a code is continuously non-malleable w.r.t. $\mathcal{F}_{\mathsf{copy}}$ against tampering with a single encoding, then it is also *adaptively* continuously non-malleable w.r.t. $\mathcal{F}_{\mathsf{copy}}$, i.e., against tampering with many encodings simultaneously. In addition, in Appendix G, we also show that the full version of the code by [17] is non-malleable against *full* bit-wise tampering (i.e., when additionally the tamper function is allowed to flip bits of an encoding). These are the main technical contributions of this work.

1.3 More Details on Related Work

The work of Hohenberger *et al.* [27]—building on the work of Myers and shelat [40]—describes a multi-bit CCA-secure encrytion scheme from a single-bit CCA-secure one, a CPA-secure one, and a 1-query-bounded CCA-secure one. Their scheme is rather sophisticated and has a somewhat circular structure, requiring a complex security proof. The public key is of the form $\mathsf{pk} = (\mathsf{pk}_{in}, \mathsf{pk}_A, \mathsf{pk}_B)$, where the "inner" public key pk_{in} is the public key of a DCCA secure PKE scheme, and the "outer" public keys pk_A and pk_B are, respectively, the public key of a 1-bounded CCA and a CPA secure PKE scheme. To encrypt a k-bit message m one first encrypts a tuple (r_A, r_B, m), using the "inner" public key, obtaining a ciphertext e_{in}, where r_A and r_B are thought as being the randomness for the "outer" encryption scheme. Next, one has to encrypt e_{in} under the "outer" public key pk_A (resp. pk_B) using randomness r_A (resp. r_B) and thus obtaining a ciphertext e_A (resp. e_B). The output ciphertext is $e = (e_A, e_B)$.

To use the above scheme, we have to instantiate the DCCA, 1-bounded CCA and CPA components. As argued in [27], all schemes can be instantiated using a single-bit CCA-secure PKE scheme yielding a fully black-box construction of a multi-bit CCA-secure PKE from a single-bit CCA-secure PKE. Let us denote with l_p (resp., l_e) the bit-length of the public key (resp., the ciphertext) for the single-bit CCA-secure PKE scheme. When we refer to the construction of [11] for the 1-bounded CCA component, we get a public key of size roughly $(3 + 16s) \cdot l_p$ for the public key and $(k + 2s) \cdot 4s \cdot l_e^2$ for the ciphertext, for security parameter s.[2]

In contrast, our scheme instantiated with the information-theoretic LECSS scheme of [17] has a ciphertext of length $\approx 5k \cdot l_e$ and a public key of length $k \cdot l_p$. Note that the length of the public key depends on the length of the message, as we need independent public keys for each encrypted bit (whereas the DCCA scheme can use always the same public key). However, we observe that when k is not too large, e.g. in case the PKE scheme is used as a key encapsulation mechanism, we would have $k \approx s$ yielding public keys of comparable size. On the negative side, recall that our construction needs to self-destruct in case an invalid ciphertext is processed, which is not required in [27], and thus our construction only achieves SD-CCA security and not full-blown CCA security.

As shown in [9], the constructive security statement for public-key encryption corresponds to *replayable* CCA security (RCCA), a notion proposed by Canetti *et al.* [3]. Hence, our scheme actually achieves replayable self-destruct CCA security (SD-RCCA)—see Appendix D. We remark, however, that if one is interested in SD-CCA security, this can be achieved generically from SD-RCCA security using the transformation in [3].

Non-malleable codes. Beyond the constructions of [17, 5, 19], non-malleable codes exists against block-wise tampering [7], against split-state tampering—both information-theoretic [16, 1] and computational [31]—and in a setting where the computational complexity of the tampering functions is somewhat limited [4, 21]. We stress that the typical application of non-malleable codes is to protect cryptographic schemes against memory tampering (see, e.g., [17, 14]). A further application of non-malleable codes has been shown by Agrawal et al. [2] (in concurrent and independent work). They show that one can obtain a non-malleable multi-bit commitment scheme from a non-malleable single-bit commitment scheme by encoding the value with a (specific) non-malleable code and then

[2] For simplicity, we assumed that the random strings r_A, r_B are computed by stretching the seed (of length s) of a pseudo-random generator.

committing to the codeword bits. Despite the similarity of the approaches, the techniques applied in their paper differ heavily from ours. The class of tampering functions the code has to protect against is different, and we additionally need continuous non-malleability to handle multiple decryption queries (this is not required for the commitment case).

2 Preliminaries

2.1 Systems: Resources and Converters, Distinguishers, and Reductions

Resources and converters. We use the concepts and terminology of abstract [34] and constructive cryptography [32]. The *resources* we consider are different types of communication channels, which are systems with three interfaces labeled by A, B, and E. A *converter* is a two-interface system which is directed in that it has an *inside* and an *outside* interface. Converters model protocol engines that are used by the parties, and using a protocol is modeled by connecting the party's interface of the resource to the inside interface of the converter (which hides those two interfaces) and using the outside interface of the converter instead. We generally use upper-case, bold-face letters (e.g., $\mathbf{R}$, $\mathbf{S}$) or channel symbols (e.g., $\bullet\!\!-\!\!\diamond\!\!-\!\!\twoheadrightarrow$) to denote resources or single-interface systems and lower-case Greek letters (e.g., α, β) or sans-serif fonts (e.g., enc, dec) for converters. We denote by Φ the set of all resources and by Σ the set of all converters.

For $I \in \{A, B, E\}$, a resource $\mathbf{R} \in \Phi$, and a converter $\alpha \in \Sigma$, the expression $\alpha^I\mathbf{R}$ denotes the composite system obtained by connecting the inside interface of α to interface I of $\mathbf{R}$; the outside interface of α becomes the I-interface of the composite system. The system $\alpha^I\mathbf{R}$ is again a resource (cf. Figure 2 on page 9). For two resources $\mathbf{R}$ and $\mathbf{S}$, $[\mathbf{R}, \mathbf{S}]$ denotes the parallel composition of $\mathbf{R}$ and $\mathbf{S}$. For each $I \in \{A, B, E\}$, the I-interfaces of $\mathbf{R}$ and $\mathbf{S}$ are merged and become the *sub-interfaces* of the I-interface of $[\mathbf{R}, \mathbf{S}]$.

Two converters α and β can be composed serially by connecting the inside interface of β to the outside interface of α, written $\beta \circ \alpha$, with the effect that $(\beta \circ \alpha)^I\mathbf{R} = \beta^I\alpha^I\mathbf{R}$. Moreover, converters can also be taken in parallel, denoted by $[\alpha, \beta]$, with the effect that $[\alpha, \beta]^I[\mathbf{R}, \mathbf{S}] = [\alpha^I\mathbf{R}, \beta^I\mathbf{S}]$. We assume the existence of an identity converter $\mathsf{id} \in \Sigma$ with $\mathsf{id}^I\mathbf{R} = \mathbf{R}$ for all resources $\mathbf{R} \in \Phi$ and interfaces $I \in \{A, B, E\}$ and of a special converter $\perp \in \Sigma$ with an inactive outside interface.

Distinguishers. A *distinguisher* $\mathbf{D}$ connects to all interfaces of a resource $\mathbf{U}$ and outputs a single bit at the end of its interaction with $\mathbf{U}$. The expression $\mathbf{DU}$ defines a binary random variable corresponding to the output of $\mathbf{D}$ when interacting with $\mathbf{U}$, and the *distinguishing advantage of a distinguisher* $\mathbf{D}$ *on two systems* $\mathbf{U}$ *and* $\mathbf{V}$ is defined as

$$\Delta^{\mathbf{D}}(\mathbf{U}, \mathbf{V}) := |\mathsf{P}[\mathbf{DU} = 1] - \mathsf{P}[\mathbf{DV} = 1]|.$$

The distinguishing advantage measures how much the output distribution of $\mathbf{D}$ differs when it is connected to either $\mathbf{U}$ or $\mathbf{V}$. Note that the distinguishing advantage is a pseudo-metric.[3]

Reductions. When relating two distinguishing problems, it is convenient to use a special type of system $\mathbf{C}$ that translates one setting into the other. Formally, $\mathbf{C}$ is a converter that has an *inside* and an *outside* interface. When it is connected to a system $\mathbf{S}$, which is denoted by $\mathbf{CS}$, the inside interface of $\mathbf{C}$ connects to the (merged) interface(s) of $\mathbf{S}$ and the outside interface of $\mathbf{C}$ is the interface of the composed system. $\mathbf{C}$ is called a *reduction system* (or simply *reduction*).

To reduce distinguishing two systems $\mathbf{S}, \mathbf{T}$ to distinguishing two systems $\mathbf{U}, \mathbf{V}$, one exhibits a reduction $\mathbf{C}$ such that $\mathbf{CS} \equiv \mathbf{U}$ and $\mathbf{CT} \equiv \mathbf{V}$. Then, for all distinguishers $\mathbf{D}$, we have $\Delta^{\mathbf{D}}(\mathbf{U}, \mathbf{V}) = \Delta^{\mathbf{D}}(\mathbf{CS}, \mathbf{CT}) = \Delta^{\mathbf{DC}}(\mathbf{S}, \mathbf{T})$. The last equality follows from the fact that $\mathbf{C}$ can also be thought of as being part of the distinguisher (which follows from the *composition-order independence* [34]).

[3] That is, for any $\mathbf{D}$, it is symmetric, satisfies the triangle inequality, and $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{R}) = 0$ for all $\mathbf{R}$.

2.2 Discrete Systems

The behavior of systems can be formalized by random systems as in [38, 33]: A random system $\mathbf{S}$ is a sequence $(\mathsf{p}^{\mathbf{S}}_{Y^i|X^i})_{i\geq 1}$ of conditional probability distributions, where $\mathsf{p}^{\mathbf{S}}_{Y^i|X^i}(y^i, x^i)$ is the probability of observing the outputs $y^i = (y_1, \ldots, y_i)$ given the inputs $x^i = (x_1, \ldots, x_i)$. If for two systems $\mathbf{R}$ and $\mathbf{S}$,

$$\mathsf{p}^{\mathbf{R}}_{Y^i|X^i} = \mathsf{p}^{\mathbf{S}}_{Y^i|X^i}$$

for all i and for all parameters where both are defined, they are called *equivalent*, denoted by $\mathbf{R} \equiv \mathbf{S}$. In that case, $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) = 0$ for all distinguishers $\mathbf{D}$.

A system $\mathbf{S}$ can be extended by a so-called *monotone binary output* (or *MBO*) $\mathcal{B}$, which is an additional one-bit output $B_1, B_2, \ldots$ with the property that $B_i = 1$ implies $B_{i+1} = 1$ for all i.[4] The enhanced system is denoted by $\hat{\mathbf{S}}$, and its behavior is described by the sequence $(\mathsf{p}^{\hat{\mathbf{S}}}_{Y^i,B_i|X^i})_{i\geq 1}$. If for two systems $\hat{\mathbf{R}}$ and $\hat{\mathbf{S}}$ with MBOs,

$$\mathsf{p}^{\hat{\mathbf{R}}}_{Y^i,B_i=0|X^i} = \mathsf{p}^{\hat{\mathbf{S}}}_{Y^i,B_i=0|X^i}$$

for all i, they are called *game equivalent*, which is denoted by $\hat{\mathbf{R}} \stackrel{g}{\equiv} \hat{\mathbf{S}}$. In such a case, $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \leq \Gamma^{\mathbf{D}}(\hat{\mathbf{R}}) = \Gamma^{\mathbf{D}}(\hat{\mathbf{S}})$, where $\Gamma^{\mathbf{D}}(\hat{\mathbf{R}})$ denotes the probability that $\mathbf{D}$ provokes the MBO. For more details and a proof of this fact, consult [33].[5]

2.3 The Notion of Construction

We formalize the security of protocols via the notion of *construction*, introduced in [32]:

Definition 1. *Let Φ and Σ be as above, and let ε_1 and ε_2 be two functions mapping each distinguisher $\mathbf{D}$ to a real number in $[0, 1]$. A protocol $\pi = (\pi_1, \pi_2) \in \Sigma^2$* constructs resource $\mathbf{S} \in \Phi$ from resource $\mathbf{R} \in \Phi$ with distance $(\varepsilon_1, \varepsilon_2)$ and with respect the simulator $\sigma \in \Sigma$, *denoted*[6]

$$\mathbf{R} \overset{\pi,\sigma,(\varepsilon_1,\varepsilon_2)}{\Longmapsto} \mathbf{S},$$

if for all distinguishers $\mathbf{D}$,

$$\begin{cases} \Delta^{\mathbf{D}}({\pi_1}^A {\pi_2}^B \perp^E \mathbf{R}, \perp^E \mathbf{S}) & \leq \varepsilon_1(\mathbf{D}) \qquad (\textit{availability}) \\ \Delta^{\mathbf{D}}({\pi_1}^A {\pi_2}^B \mathbf{R}, \sigma^E \mathbf{S}) & \leq \varepsilon_2(\mathbf{D}) \qquad (\textit{security}). \end{cases}$$

The availability condition captures that a protocol must correctly implement the functionality of the constructed resource in the absence of the attacker. The security condition models the requirement that everything the attacker can achieve in the setting with the assumed resource and the protocol, she can also accomplish in the setting with the constructed resource (using the simulator to translate the behavior). The notion of construction composes; details can be found in Appendix A.

2.4 Public-Key Encryption Schemes

A public-key encryption (PKE) scheme with message space $\mathcal{M} \subseteq \{0,1\}^*$ and ciphertext space $\mathcal{E}$ is defined as three algorithms $\Pi = (K, E, D)$, where the key-generation algorithm K outputs a key pair $(\mathsf{pk}, \mathsf{sk})$, the (probabilistic) encryption algorithm E takes a message $m \in \mathcal{M}$ and a public key pk and outputs a ciphertext $e \leftarrow E_{\mathsf{pk}}(m)$, and the decryption algorithm takes a ciphertext $e \in \mathcal{E}$ and a secret key sk and outputs a plaintext $m \leftarrow D_{\mathsf{sk}}(e)$. The output of the decryption algorithm can be the special symbol $\diamond$, indicating an invalid ciphertext. A PKE scheme is correct if $m = D_{\mathsf{sk}}(E_{\mathsf{pk}}(m))$ (with probability 1 over the randomness in the encryption algorithm) for all messages m and all key pairs $(\mathsf{pk}, \mathsf{sk})$ generated by K.

We introduce security notions for PKE schemes as we need them.

[4] In other words, once the MBO is 1, it cannot return to 0.

[5] Intuitively, this means that in order to distinguish the two systems, $\mathbf{D}$ has to provoke the MBO.

[6] In less formal contexts, we sometimes drop the superscripts on $\Longmapsto$.

System $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$

init
  $i \leftarrow 0$

on (encode, x)
  $i \leftarrow i+1$
  $c^{(i)} \leftarrow\!\!\$\ \mathrm{Enc}(x)$

on (tamper, f) *with* $f \in \mathcal{F}^{(i)}$
  $c' \leftarrow f(c^{(1)}, \ldots, c^{(i)})$
  $x' \leftarrow \mathrm{Dec}(c')$
  if $x' = \diamond$
    self-destruct
  out x'

System $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$

init
  $i \leftarrow 0$

on (encode, x)
  $i \leftarrow i+1$
  $x^{(i)} \leftarrow\!\!\$\ x$

on (tamper, f) *with* $f \in \mathcal{F}^{(i)}$
  $x' \leftarrow\!\!\$\ \tau(i, f)$
  if $x' = \diamond$
    self-destruct
  if $x' = (\mathsf{same}, j)$
    $x' \leftarrow x^{(j)}$
  out x'

Figure 1: *Systems* $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ *and* $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ *defining adaptive continuous non-malleability of* $(\mathrm{Enc}, \mathrm{Dec})$. *The command* **self-destruct** *has the effect that* $\diamond$ *is output and all future queries are answered by* $\diamond$.

2.5 Continuously Non-Malleable Codes

Non-malleable codes, introduced in [17], are coding schemes that protect the encoded messages against certain classes of adversarially chosen modifications, in the sense that the decoding will result either in the original message or in an unrelated value.

Definition 2 (Coding scheme)**.** *A* (k, n)-coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ *consists of a randomized encoding function* $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ *and a deterministic decoding function* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \{\diamond\}$ *such that* $\mathrm{Dec}(\mathrm{Enc}(x)) = x$ *(with probability* 1 *over the randomness of the encoding function) for each* $x \in \{0,1\}^k$. *The special symbol* $\diamond$ *indicates an* invalid codeword.

In the original definition [17], the adversary is allowed to modify the codeword via a function of a specified class $\mathcal{F}$ only once. Continuous non-malleability, introduced in [19], extends this guarantee to the case where the adversary is allowed to perform multiple such modifications for a fixed target codeword. The notion of *adaptive* continuous non-malleability considered here is an extension of the one in [19] in that the adversary is allowed to adaptively specify messages and the functions may depend on multiple codewords. That is, the class $\mathcal{F}$ is actually a sequence $(\mathcal{F}^{(i)})_{i\geq 1}$ of function families with $\mathcal{F}^{(i)} \subseteq \{f \mid f : (\{0,1\}^n)^i \to \{0,1\}^n\}$, and after encoding i messages, the adversary chooses functions from $\mathcal{F}^{(i)}$. A similar adaptive notion has been already considered for continuous strong non-malleability in the split-state model [20].

Formally, adaptive continuous non-malleability w.r.t. $\mathcal{F}$ is defined by comparing the two random systems $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ defined in Figure 1. Both systems process encode and tamper queries from a distinguisher $\mathbf{D}$, whose objective is to tell the two systems apart.

System $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ produces a random encoding $c^{(i)}$ of each message $x^{(i)}$ specified by $\mathbf{D}$ and allows $\mathbf{D}$ to repeatedly issue tampering functions $f \in \mathcal{F}^{(i)}$. For each such query, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ computes the modified codeword $c' = f(c^{(1)}, \ldots, c^{(i)})$ and outputs $\mathrm{Dec}(c')$. Whenever $\mathrm{Dec}(c') = \diamond$, the system enters a self-destruct mode, in which all further queries are replied to by $\diamond$.

The second random system, $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$, features a simulator τ, which is allowed to keep state. The simulator repeatedly takes a tampering function and outputs either a message x', (same, v) for $v \in \{1, \ldots, i\}$, or $\diamond$, where (same, v) is used by τ to indicate that (it believes that) the tampering results in an n-bit string that decodes to the v^{th} message encoded. System $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ outputs whatever τ outputs, except that (same, v) is replaced by the v^{th} message $x^{(v)}$ specified by $\mathbf{D}$. Moreover, in case of $\diamond$, $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ self-destructs.

For $\ell, q \in \mathbb{N}$, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F},\ell,q}$ is the system that behaves as $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ except that only the first ℓ encode-queries and the first q tamper-queries are handled (and similarly for $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,\ell,q}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$). Note that by setting $\ell = 1$, one recovers continuous non-malleability as defined in [19],[7] and by additionally setting $q = 1$ the original definition of non-malleability.

Definition 3 (Continuous non-malleability)**.** *Consider a sequence* $\mathcal{F} = (\mathcal{F}^{(i)})_{i\geq 1}$ *of function families* $\mathcal{F}^{(i)} \subseteq \{f \mid f : (\{0,1\}^n)^i \to \{0,1\}^n\}$ *and let* $\ell, q \in \mathbb{N}$. *A coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *is* adaptively continuously $(\mathcal{F}, \varepsilon, \ell, q)$-non-malleable *(or simply* $(\mathcal{F}, \varepsilon, \ell, q)$-non-malleable*) if there exists a simulator* τ *such that* $\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F},\ell,q}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,\ell,q}) \leq \varepsilon$ *for all distinguishers* $\mathbf{D}$.

[7] Being based on *strong* non-malleability [17], the notion of [19] is actually stronger than ours.

3 From Single-Bit to Multi-Bit Channels

In this section we examine the question of domain extension for CCA-secure public-key encryption (PKE) via the following intuitive non-malleable code based approach: first encode a k-bit message using a non-malleable (k, n)-code to protect its integrity, obtaining an n-bit codeword c; then encrypt c bit-wise using n independent public keys for a single-bit CCA-secure PKE. We observe that the adversary's ability of asking multiple decryption queries requires to opt for continuously non-malleable codes. The self-destruct property of these codes, however, translates to the resulting PKE scheme, and thus we achieve domain extension only for schemes with so-called *self-destruct CCA security*, a variant of CCA security where the decryption oracle stops working after the attacker submits an invalid ciphertext; this variant is defined more precisely in Section D.1.

We stress that the need for self-destruct is not a limitation of the security proof of our code (cf. Section 4), as continuous non-malleability for the class of tampering functions required for the above transformation to work is impossible without the self-destruct property (cf. Appendix F for details).

As shown below, phrasing PKE domain extension using the paradigm of constructive cryptography allows to decompose the problem into two independent parts: The first part includes a (canonical) reduction to the SD-CCA security of the single-bit PKE scheme, whereas the second part, which involves non-malleable codes, is purely information-theoretic. The two parts can then be combined to obtain a single protocol, whose security follows from the composition theorem. We also show how the resulting protocol can be understood as a PKE scheme and that it achieves SD-CCA security.

All channel resources that appear in this section are formally defined in Section B of the appendix; to understand the statements and explanations below, the informal descriptions given in Section 1.2 are sufficient, however.

3.1 Single-Bit PKE Viewed Constructively

Following the proof of [9, Theorem 2], one can show that a 1-bit SD-CCA-secure PKE scheme can be used to design a protocol that achieves the construction

$$[\longleftarrow\!\bullet, - \twoheadrightarrow] \quad \Longrightarrow \quad \overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\bullet}, \tag{2}$$

where, in a nutshell, the receiver's protocol converter is responsible for key generation, decryption, as well as self-destructing, the sender's protocol converter for encryption, and where the authenticated channel $\longleftarrow\!\bullet$ is used for the transmission of the public key and the insecure channel $- \twoheadrightarrow$ for sending ciphertexts. The constructed single-bit confidential channel $\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\bullet}$ hides all messages sent by the sender from the attacker and allows the attacker to either deliver already sent messages or to inject *independent* messages. This captures the intuitive (SD-)CCA guarantee that an attacker, by modifying a particular ciphertext, can either leave the message contained therein intact or replace it by an independently created one.

Using n independent copies of the single-bit scheme in parallel yields a protocol $\mathsf{1\text{-}pke}$ that achieves:

$$[\longleftarrow\!\bullet, - \twoheadrightarrow] \quad \overset{\mathsf{1\text{-}pke}}{\Longrightarrow} \quad [\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\bullet}]^n, \tag{3}$$

which follows almost directly from the composition theorem. More details can be found in Appendix D.2.

3.2 Tying the Channels Together

We now show how to construct, using an adaptive continuously non-malleable (k, n)-code (cf. Section 2.5), a (single) k-bit confidential channel from the n independent single-bit confidential channels constructed in the previous section. This is achieved by having the sender encode the message with the non-malleable code and sending the resulting codeword over the 1-bit channels (bit-by-bit), while the receiver decodes all n-bit strings received via these channels. Additionally, due to the self-destruct

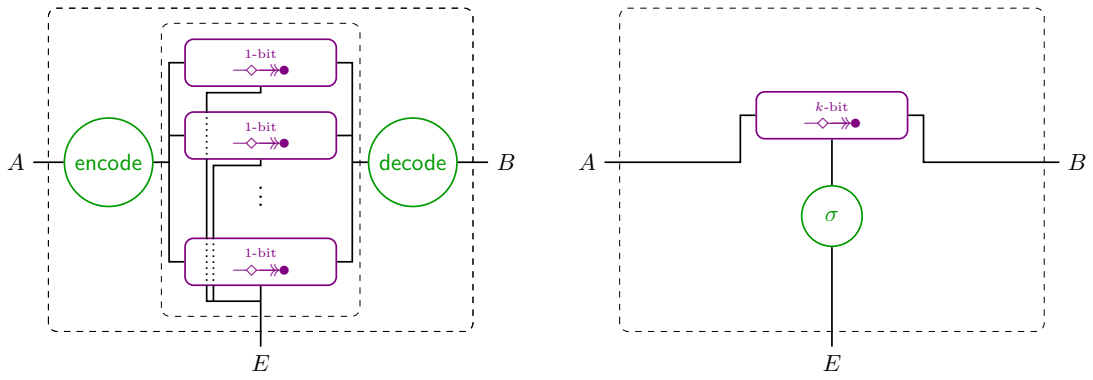

Figure 2: *Left: The assumed resource* $[\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n$ *with protocol converters* encode *and* decode *attached to interfaces A and B, denoted* $\mathsf{encode}^A\mathsf{decode}^B[\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n$. *Right: The constructed resource* $\overset{\text{k-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$ *with simulator* σ *attached to the E-interface, denoted* $\sigma^E \overset{\text{k-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$. *In particular,* σ *must simulate the E-interfaces of* $[\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n$. *The protocol is secure if the two systems are indistinguishable.*

property of continuously non-malleable codes, the receiver must stop decoding once an invalid codeword has been received.

More precisely, let $(\mathrm{Enc}, \mathrm{Dec})$ be a (k, n)-coding scheme and consider the following protocol $\mathsf{nmc} = (\mathsf{encode}, \mathsf{decode})$: Converter encode encodes every message $m \in \{0,1\}^k$ input at its outside interface with fresh randomness, resulting in an n-bit encoding $c = c_1 \cdots c_n \leftarrow \mathrm{Enc}(m)$. Then, for $i = 1, \ldots, n$, it outputs bit c_i to the i^{th} channel at the inside interface. Converter decode, whenever it receives an n-bit string $c' = c'_1 \cdots c'_n$ (where the i^{th} bit c'_i was received on the i^{th} channel), it computes $m' \leftarrow \mathrm{Dec}(c')$ and outputs m' at the outside interface. If $m' = \diamond$, it implements the self-destruct mode, i.e., it answers all future encodings received at the inside interface by outputting $\diamond$ at the outside interface.

The goal is now to show that protocol nmc achieves the construction

$$[\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n \quad \overset{\mathsf{nmc}}{\Longrightarrow} \quad \overset{\text{k-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}\,. \tag{4}$$

The required non-malleability. By inspecting both sides of Figure 2, it becomes immediately apparent why adaptive continuously non-malleable codes are the proper choice to achieve construction (4): On the left-hand side, the distinguisher can repeatedly input messages $m^{(i)}$ at interface A, which results in encodings $c^{(i)}$ being input (bit-by-bit) into the single-bit channels. Using the E-interfaces of these channels, the distinguisher can repeatedly see the decoding of an n-bit string $c' = c'_1 \cdots c'_n$ at interface B, where each bit c'_j results from either forwarding one of the bits already in the j^{th} channel or from injecting a fresh bit that is either 0 or 1.

Put differently, the distinguisher can effectively launch tampering attacks using functions from $\mathcal{F}_{\mathsf{copy}} := (\mathcal{F}^{(i)}_{\mathsf{copy}})_{i \geq 1}$, where $\mathcal{F}^{(i)}_{\mathsf{copy}} \subseteq \{f \mid f : (\{0,1\}^n)^i \to \{0,1\}^n\}$ and each function $f \in \mathcal{F}^{(i)}_{\mathsf{copy}}$ is characterized by a vector $\chi(f) = (f_1, \ldots, f_n)$ where $f_j \in \{\mathsf{zero}, \mathsf{one}, \mathsf{copy}_1, \ldots, \mathsf{copy}_i\}$, with the meaning that f takes as input i codewords $(c^{(1)}, \ldots, c^{(i)})$ and outputs an n-bit string $c' = c'_1 \cdots c'_n$ in which each bit c'_j is either set to 0 (zero), set to 1 (one), or copied from the j^{th} bit in a codeword $c^{(v)}$ (copy_v) for $v \in \{1, \ldots, i\}$.

On the right-hand side, the distinguisher may again input messages $m^{(i)}$ at interface A—to the k-bit confidential channel. At interface E, this channel only allows to either deliver entire k-bit messages already sent by A or to inject independent messages. The simulator σ required to prove (4) needs to simulate the E-interfaces of the single-bit confidential channels at its outside interface and, based solely on what is input at these interfaces, decide whether to forward or inject a message, which corresponds exactly to the task of the simulator τ in the non-malleability experiment (cf. Section 2.5).

Theorem 1 below formalizes this correspondence; its proof is essentially a technicality: one merely needs to "translate" between the channel settings and the non-malleability experiment. For completeness it is provided in full detail in Appendix D.3.

Theorem 1. *For any $\ell, q \in \mathbb{N}$, if* $(\mathrm{Enc}, \mathrm{Dec})$ *is* $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, \ell, q)$*-continuously non-malleable, there exists a simulator σ such that*

$$\left[\text{—}\diamond\twoheadrightarrow\bullet\right]^{n}{}^{1\text{-bit},\ell,q} \quad \overset{(\mathsf{nmc},\sigma,(0,\varepsilon))}{\vDash\!\!\Rightarrow} \quad \text{—}\diamond\twoheadrightarrow\bullet{}^{k\text{-bit},\ell,q},$$

where the additional superscripts ℓ, q on a channel mean that it only processes the first ℓ queries at the A-interface and only the first q queries at the E-interface.

3.3 Plugging It Together

The composition theorem of constructive cryptography (cf. Appendix A) implies that the protocol $\mathsf{m\text{-}pke} = \mathsf{nmc} \circ \mathsf{1\text{-}pke}$ resulting from composing the protocols $\mathsf{1\text{-}pke}$ and nmc for transformations (3) and (4), respectively, achieves

$$[\longleftarrow\bullet, - \twoheadrightarrow] \quad \overset{\mathsf{m\text{-}pke}}{\vDash\!\!\Rightarrow} \quad \text{—}\diamond\twoheadrightarrow\bullet{}^{k\text{-bit}}. \tag{5}$$

Protocol $\mathsf{m\text{-}pke}$ corresponds (in a straight-forward manner) to a PKE scheme Π that achieves SD-CCA security, as shown in Section D.4 of the appendix.[8] Hence, overall, we obtain a domain extension technique for SD-CCA-secure PKE schemes.

Furthermore, in Section D.5, we also provide a direct game-based proof of the fact that combining single-bit SD-CCA-secure PKE with a non-malleable code as shown above yields a multi-bit SD-CCA-secure PKE scheme. That proof is a hybrid argument and is obtained by "unwrapping" the concatenation of the statements in this section. The modular nature and the intuitive simplicity of the proofs are lost, however.

4 Continuous Non-Malleability against Bit-Wise Tampering

In this section, we describe a code that is adaptively continuously non-malleable w.r.t. $\mathcal{F}_{\mathsf{copy}}$. For completeness, in Appendix G, we also provide a code secure w.r.t. to an extension $\mathcal{F}'_{\mathsf{copy}}$ of $\mathcal{F}_{\mathsf{copy}}$ that allows bit-flips as well.

The transition from continuous to *adaptive* continuous non-malleability w.r.t. $\mathcal{F}_{\mathsf{copy}}$ is achieved generically:

Theorem 2. *If a coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *is continuously* $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, 1, q)$*-non-malleable, it is also continuously* $(\mathcal{F}_{\mathsf{copy}}, 2\ell\varepsilon + \frac{q\ell}{2^k}, \ell, q)$*-non-malleable, for all $\ell, q \in \mathbb{N}$.*

The proof of Theorem 2 appears in Appendix E. It remains to construct a continuously non-malleable code that is secure against tampering with a single encoding, which we do below.

Continuous non-malleability for single encoding. The code is based on a linear error-correcting secret-sharing (LECSS). The use of a LECSS is inspired by the work of [17], who proposed a (non-continuous) non-malleable code against bit-wise tampering based on a LECSS and, additionally, an AMD-code (cf. Appendix G), where the AMD-code essentially handles bit-flips. As we do not need to provide non-malleability against bit-flips, using only the LECSS is sufficient for our purposes. The following definition is taken from [17]:[9]

Definition 4 (LECSS code)**.** *A (k, n)-coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *is a* (d, t)-linear error-correcting secret-sharing (LECSS) code *if the following properties hold:*

- Linearity: *For all $c \in \{0,1\}^n$ such that* $\mathrm{Dec}(c) \neq \perp$*, all $\delta \in \{0,1\}^n$, we have*

$$\mathrm{Dec}(c \oplus \delta) = \begin{cases} \perp & \text{if } \mathrm{Dec}(\delta) = \perp \\ \mathrm{Dec}(c) \oplus \mathrm{Dec}(\delta) & \text{otherwise.} \end{cases}$$

[8] Actually, our protocol only achieves *replayable* SD-CCA security, which, however, is not a major issue as explained in Section D.4.

[9] The operator $\oplus$ denotes the bit-wise XOR.

- DISTANCE d: *For all* $c' \in \{0,1\}^n$ *with Hamming weight* $0 < w_H(c') < d$, *we have* $\mathrm{Dec}(c') = \perp$.
- SECRECY t: *For any fixed* $x \in \{0,1\}^k$, *the bits of* $\mathrm{Enc}(x)$ *are individually uniform and* t*-wise independent (over the randomness in the encoding).*

It turns out that a LECSS code is already continuously non-malleable with respect to $\mathcal{F}_{\mathsf{copy}}$:

Theorem 3. *Assume that* $(\mathrm{Enc}, \mathrm{Dec})$ *is a* (t,d)*-LECSS* (k,n)*-code for* $d > n/4$ *and* $d > t$. *Then* $(\mathrm{Enc}, \mathrm{Dec})$ *is* $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, 1, q)$*-continuously non-malleable for all* $q \in \mathbb{N}$ *and*

$$\varepsilon = 2^{-(t-1)} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2}.$$

For brevity, we write $\mathcal{F}_{\mathsf{set}}$ for $\mathcal{F}^{(1)}_{\mathsf{copy}}$ below, with the idea that the tampering functions in $\mathcal{F}^{(1)}_{\mathsf{copy}}$ only allow to keep a bit or to set it to 0 or to 1. More formally, a function $f \in \mathcal{F}_{\mathsf{set}}$ can be characterized by a vector $\chi(f) = (f_1, \ldots, f_n)$ where $f_i \in \{\mathsf{zero}, \mathsf{one}, \mathsf{keep}\}$, with the meaning that f takes as input a codeword c and outputs a codeword $c' = c'_1 \cdots c'_n$ in which each bit is either set to 0 (zero), set to 1 (one), or left unchanged (keep).

For the proof of Theorem 3, fix $q \in \mathbb{N}$ and some distinguisher $\mathbf{D}$. For the remainder of this section, let $\mathcal{F} := \mathcal{F}_{\mathsf{set}}$, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},1,q}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,1,q}$ (for a simulator τ to be determined). For a tamper query $f \in \mathcal{F}$ with $\chi(f) = (f_1, \ldots, f_n)$ issued by $\mathbf{D}$, let $A(f) := \{i \mid f_i \in \{\mathsf{zero}, \mathsf{one}\}\}$, $B(f) := \{i \mid f_i \in \{\mathsf{keep}\}\}$, and $a(f) := |A(f)|$. Moreover, let $\mathsf{val}(\mathsf{zero}) := 0$ and $\mathsf{val}(\mathsf{one}) := 1$. Queries f with $0 \le a(f) \le t$, $t < a(f) < n - t$, and $n - t \le a(f) \le n$ are called *low queries*, *middle queries*, and *high queries*, respectively.

Handling Middle Queries. Consider the hybrid system $\mathbf{H}$ that proceeds as $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$, except that as soon as $\mathbf{D}$ specifies a middle query f, $\mathbf{H}$ self-destructs, i.e., answers f and all subsequent queries by $\diamond$.

Lemma 4. $\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) \le \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2}$.

Proof. Define a *successful* middle query to be a middle query that does not decode to $\diamond$. On both systems $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{H}$, one can define an MBO $\mathcal{B}$ (cf. Section 2.2) that is provoked if and only if the *first* middle query is successful and the self-destruct has not been provoked up to that point.

Clearly, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{H}$ behave identically until MBO $\mathcal{B}$ is provoked, thus $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}} \stackrel{g}{\equiv} \hat{\mathbf{H}}$, and

$$\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) \le \Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}).$$

Towards bounding $\Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}})$, note first that adaptivity does not help in provoking $\mathcal{B}$: For any distinguisher $\mathbf{D}$, there exists a *non-adaptive* distinguisher $\mathbf{D}'$ with

$$\Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}) \le \Gamma^{\mathbf{D}'}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}). \tag{6}$$

$\mathbf{D}'$ proceeds as follows: First, it (internally) interacts with $\mathbf{D}$ only. Initially, it stores the message x output by $\mathbf{D}$ internally. Whenever $\mathbf{D}$ outputs a low query, $\mathbf{D}'$ answers with x. Whenever $\mathbf{D}$ outputs a high query $f = (f_1, \ldots, f_n)$, $\mathbf{D}'$ checks whether there exists a codeword c^* that agrees with f in positions i where $f_i \in \{\mathsf{zero}, \mathsf{one}\}$. If it exists, it answers with $\mathrm{Dec}(c^*)$, otherwise with $\diamond$. As soon as $\mathbf{D}$ specifies a middle query, $\mathbf{D}'$ stops its interaction with $\mathbf{D}$ and sends x and all the queries to $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}$.

To prove (6), fix all randomness in experiment $\mathbf{D}'\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$, i.e., the coins of $\mathbf{D}$ (inside $\mathbf{D}'$) and the randomness of the encoding (inside $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$). Suppose $\mathbf{D}$ would provoke $\mathcal{B}$ in the direct interaction with $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$. In that case all the answers by $\mathbf{D}'$ are equal to the answers by $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$. This is due to the fact that the distance of the LECSS is $d > t$; a successful low query must therefore result in the original message x and a successful high query in $\mathrm{Dec}(c^*)$. Thus, whenever $\mathbf{D}$ provokes $\mathcal{B}$, $\mathbf{D}'$ provokes it as well.

It remains to analyze the success probability of non-adaptive distinguishers $\mathbf{D}'$. Fix the coins of $\mathbf{D}'$; this determines the tamper queries. Suppose there is at least one middle case, as otherwise $\mathcal{B}$ is

trivially not provoked. The middle case's success probability can be analyzed as in [17, Theorem 4.1], which leads to

$$\Gamma^{\mathbf{D}'}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}) \leq \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2}$$

(recall that the MBO cannot be provoked after an unsuccessful first middle query). □

Simulator. The final step of the proof consists of exhibiting a simulator τ such that $\Delta^{\mathbf{D}}(\mathbf{H}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau})$ is small. The indistinguishability proof is facilitated by defining two hardly distinguishable systems $\mathbf{B}$ and $\mathbf{B}'$ and a wrapper system $\mathbf{W}$ such that $\mathbf{WB} \equiv \mathbf{H}$ and $\mathbf{WB}' \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$.

System $\mathbf{B}$ works as follows: Initially, it takes a value $x \in \{0,1\}^k$, computes an encoding $c_1 \cdots c_n \leftarrow\!\!\$\ \mathrm{Enc}(x)$ of it, and outputs λ (where the symbol λ indicates an empty output). Then, it repeatedly accepts guesses $g_i = (j, b)$, where (j, b) is a guess b for c_j. If a guess g_i is correct, $\mathbf{B}$ returns $a_i = 1$. Otherwise, it outputs $a_i = \diamond$ and self-destructs (i.e., all future answers are $\diamond$). The system $\mathbf{B}'$ behaves as $\mathbf{B}$ except that the initial input x is ignored and the $c_1, \ldots, c_n$ are chosen uniformly at random and independently.

The behavior of $\mathbf{B}$ (and similarly that of $\mathbf{B}'$) is described by a sequence $(\mathsf{p}^{\mathbf{B}}_{A^i|G^i})_{i\geq 0}$ of conditional probability distributions (cf. Section 2.2), where $\mathsf{p}^{\mathbf{B}}_{A^i|G^i}(a^i, g^i)$ is the probability of observing the outputs $a^i = (\lambda, a_1, \ldots, a_i)$ given the inputs $g^i = (x, g_1, \ldots, g_i)$. For simplicity, assume below that g^i is such that no position is guessed twice (a generalization is straight-forward) and that a^i is of the form $\{\lambda\}\{1\}^*\{\diamond\}^*$ (as otherwise it has probability 0 anyway).

For system $\mathbf{B}$, all i, and any g^i, $\mathsf{p}^{\mathbf{B}}_{A^i|G^i}(a^i, g^i) = 2^{-(s+1)}$ if a^i has $s < \min(i, t)$ leading 1's; this follows from the t-wise independence of the bits of $\mathrm{Enc}(x)$. All remaining output vectors a^i, i.e., those with at least $\min(i, t)$ preceding 1's, share a probability mass of $2^{-\min(i,t)}$, in a way that depends on the code in use and on x. (It is easily verified that this yields a valid probability distribution.) The behavior of $\mathbf{B}'$ is obvious given the above (simply replace "t" by "n" in the above description).

Lemma 5. $\Delta^{\mathbf{D}}(\mathbf{B}, \mathbf{B}') \leq 2^{-t}$.

Proof. On both systems $\mathbf{B}$ and $\mathbf{B}'$, one can define an MBO $\mathcal{B}$ that is zero as long as *less* than t positions have been guessed correctly. In the following, $\hat{\mathbf{B}}$ and $\hat{\mathbf{B}}'$ denote $\mathbf{B}$ and $\mathbf{B}'$ with the MBO, respectively.

Analogously to the above, the behavior of $\hat{\mathbf{B}}$ (and similarly that of $\hat{\mathbf{B}}'$) is described by a sequence $(\mathsf{p}^{\hat{\mathbf{B}}}_{A^i,B_i=0|G^i})_{i\geq 0}$ of conditional probability distributions, where $\mathsf{p}^{\hat{\mathbf{B}}}_{A^i,B_i=0|G^i}(a^i, g^i)$ is the probability of observing the outputs $a^i = (\lambda, a_1, \ldots, a_i)$ and $b_0 = b_1 = \ldots = b_i = 0$ given the inputs $g^i = (x, g_1, \ldots, g_i)$. One observes that due to the t-wise independence of $\mathrm{Enc}(x)$'s bits, for $i < t$,

$$\mathsf{p}^{\hat{\mathbf{B}}}_{A^i,B_i=0|G^i}(a^i, g^i) = \mathsf{p}^{\hat{\mathbf{B}}'}_{A^i,B_i=0|G^i}(a^i, g^i) = \begin{cases} 2^{-(s+1)} & \text{if } a^i \text{ has } s < i \text{ leading 1's,} \\ 2^{-i} & \text{if } a^i \text{ has } i \text{ leading 1's, and} \\ 0 & \text{otherwise,} \end{cases}$$

and for $i \geq t$,

$$\mathsf{p}^{\hat{\mathbf{B}}}_{A^i,B_i=0|G^i}(a^i, g^i) = \mathsf{p}^{\hat{\mathbf{B}}'}_{A^i,B_i=0|G^i}(a^i, g^i) = \begin{cases} 2^{-(s+1)} & \text{if } a^i \text{ has } s < t \text{ leading 1's,} \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\hat{\mathbf{B}} \overset{g}{\equiv} \hat{\mathbf{B}}'$ and $\Delta^{\mathbf{D}}(\mathbf{B}, \mathbf{B}') \leq \Gamma^{\mathbf{D}}(\hat{\mathbf{B}}')$. Observe that by an argument similar to the one above, adaptivity does not help in provoking the MBO of $\hat{\mathbf{B}}'$. Thus, $\Gamma^{\mathbf{D}}(\hat{\mathbf{B}}') \leq 2^{-t}$, since an optimal non-adaptive strategy simply tries to guess distinct positions. □

Recall that the purpose of the wrapper system $\mathbf{W}$ is to emulate $\mathbf{H}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ using $\mathbf{B}$ and $\mathbf{B}'$, respectively. The key point is to note that low queries f can be answered knowing only the positions

System **W**

init
  $\forall i \in [n] : c_i \leftarrow \varnothing$

on first (encode, x) *at* o
  out x at i

on (tamper, f) *with* $0 \le a(f) \le t$ *at* o
  for i *where* $f_i \in A(f)$
    $g \leftarrow \mathsf{val}(f_i)$
    if $c_i = \varnothing$
      out (i, g) at i
      get $a \in \{\diamond, 1\}$ at i
      if $a = \diamond$
        self-destruct
      $c_i \leftarrow g$
    else
      if $c_i \neq g$
        self-destruct
  out x at out

on (tamper, f) *with* $t < a(f) < n - t$ *at* o
  self-destruct

on (tamper, f) *with* $n - t \le a(f) \le n$ *at* o
  for i *where* $f_i \in A(f)$
    $c'_i \leftarrow \mathsf{val}(f_i)$
  if $\exists$*codeword* $c^* : \forall i \in A(f) : c'_i = c^*_i$
    for i *where* $f_i \in B(f)$
      $g \leftarrow c^*_i$
      if $c_i = \varnothing$
        out (i, g) at i
        get $a \in \{\diamond, 1\}$ at i
        if $a = \diamond$
          self-destruct
        $c_i \leftarrow g$
      else
        if $c_i \neq g$
          self-destruct
  else
    self-destruct
  out $\mathrm{Dec}(c^*)$ at out

Figure 3: *The wrapper system* **W**. *The command* **self-destruct** *causes* **W** *to output* $\diamond$ *at* o *and to answer all future queries by* $\diamond$. *The symbol* $\varnothing$ *stands for "undefined."*

$A(f)$ of $\mathrm{Enc}(x)$, high queries knowing only the positions in $B(f)$, and middle queries can always be rejected. A full description of **W** can be found in Figure 3. It has an outside interface o and an inside interface i; at the latter interface, **W** expects to be connected to either **B** or **B′**.

Lemma 6. $\mathbf{WB} \equiv \mathbf{H}$.

Proof. Since the distance of the LECSS is $d > t$, the following holds: A low query results in same if all injected positions match the corresponding bits of the encoding, and in $\diamond$ otherwise. Similarly, for a high query, there can be at most one codeword that matches the injected positions. If such a codeword c^* exists, the outcome is $\mathrm{Dec}(c^*)$ if the bits in the keep-positions match c^*, and otherwise $\diamond$. By inspection, it can be seen that **W** acts accordingly. □

Consider now the system $\mathbf{WB}'$. Due to the nature of $\mathbf{B}'$, the behavior of $\mathbf{WB}'$ is independent of the value x that is initially encoded. This allows to easily design a simulator τ as required by Definition 3. A full description of τ can be found in Figure 4.

Lemma 7. *The simulator* τ *of Figure 4 satisfies* $\mathbf{WB}' \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$.

Proof. Consider the systems $\mathbf{WB}'$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$. Both internally choose uniform and independent bits $c_1, \ldots, c_n$. System $\mathbf{WB}'$ answers low queries with the value x initially encoded if all injected positions match the corresponding random bits and with $\diamond$ otherwise. Simulator τ returns same in the former case, which $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ replaces by x, and $\diamond$ in the latter case.

Observe that the answer by $\mathbf{WB}'$ to a high query f always matches $\mathrm{Dec}(c'_1 \cdots c'_n)$, where for $i \in A(f)$, $c'_i = \mathsf{val}(f_i)$, and for $i \in B(f)$, $c'_i = c_i$: If no codeword c^* matching the injected positions exists, then $\mathrm{Dec}(c'_1 \cdots c'_n) = \diamond$, which is also what $\mathbf{WB}'$ outputs. If such c^* exists and $c^*_i = c_i$ for all

Simulator τ

init
  $\forall i \in [n] : c_i \leftarrow\!\!\$\ \{0,1\}$

on $(1, f)$ *with* $0 \le a(f) \le t$
  if $\forall i \in A(f) : \mathsf{val}(f_i) = c_i$
    return same
  else
    return $\diamond$

on $(1, f)$ *with* $t < a(f) < n - t$
  return $\diamond$

on $(1, f)$ *with* $n - t \le a(f) \le n$
  for i *where* $f_i \in A(f)$
    $c'_i \leftarrow \mathsf{val}(f_i)$
  for i *where* $f_i \in B(f)$
    $c'_i \leftarrow c_i$
  $c' \leftarrow c'_1 \cdots c'_n$
  return $\mathrm{Dec}(c')$

Figure 4: *The simulator τ.*

$i \in B(f)$, the output of $\mathbf{WB'}$ is $\mathrm{Dec}(c'_1 \cdots c'_n)$. If there exists an $i \in B(f)$ with $c^*_i \neq c_i$, $\mathbf{WB'}$ outputs $\diamond$, and in this case $\mathrm{Dec}(c'_1 \cdots c'_n) = \diamond$ since the distance of the LECSS is $d > t$. □

The proof of Theorem 3 now follows from a simple triangle inequality.

Proof (of Theorem 3). From Lemmas 4, 5, 6, and 7, one obtains that for all distinguishers $\mathbf{D}$,

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) &\le \Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) + \underbrace{\Delta^{\mathbf{D}}(\mathbf{H}, \mathbf{WB})}_{=0} + \underbrace{\Delta^{\mathbf{D}}(\mathbf{WB}, \mathbf{WB'})}_{=\Delta^{\mathbf{DW}}(\mathbf{B},\mathbf{B'})} + \underbrace{\Delta^{\mathbf{D}}(\mathbf{WB'}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau})}_{=0} \\ &\le 2^{-t} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2} + 2^{-t} \le 2^{-(t-1)} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2}.\end{aligned}$$

□

Acknowledgments. We thank Yevgeniy Dodis for insightful discussions and for suggesting many improvements to the paper. We thank Joël Alwen and Daniel Tschudi for helpful discussions, in particular on the impossibility proof in Section F. The work was supported by the Swiss National Science Foundation (SNF), project no. 200020-132794.

References

[1] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:81, 2013. To appear in STOC 2014.

[2] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations. Cryptology ePrint Archive, Report 2014/316, May 2014.

[3] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, pages 565–582, 2003.

[4] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In *ITCS*, pages 155–168, 2014.

[5] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *TCC*, pages 440–464, 2014.

[6] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In *TCC*, pages 427–444, 2008.

[7] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In *ASIACRYPT*, pages 740–758, 2011.

[8] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Self-destruct non-malleability, 2014. Manuscript.

[9] Sandro Coretti, Ueli Maurer, and Björn Tackmann. Constructing confidential channels from authenticated channels - public-key encryption revisited. In *ASIACRYPT (1)*, pages 134–153, 2013.

[10] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *EUROCRYPT*, pages 471–488, 2008.

[11] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In *ASIACRYPT*, pages 502–518, 2007.

[12] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25, Heidelberg, 1998. Springer.

[13] Dana Dachman-Soled. A black-box construction of a CCA2 encryption scheme from a plaintext aware encryption scheme. In Hugo Krawczyk, editor, *PKC*, LNCS. Springer, 2014.

[14] Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In *ASIACRYPT (2)*, pages 140–160, 2013.

[15] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[16] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO (2)*, pages 239–257, 2013.

[17] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

[18] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[19] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC*, pages 465–488, 2014.

[20] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient random access machine. *IACR Cryptology ePrint Archive*, 2014:338, 2014.

[21] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *EUROCRYPT*, pages 111–128, 2014.

[22] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 260–274, Heidelberg, 2001. Springer.

[23] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[24] Yael Gertner, Tal Malkin, and Steven Myers. Towards a separation of semantic and CCA security for public key encryption. In *TCC*, pages 434–455, 2007.

[25] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[26] Dennis Hofheinz and Eike Kiltz. Practical chosen ciphertext secure encryption from factoring. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 313–332, Heidelberg, 2009. Springer.

[27] Susan Hohenberger, Allison B. Lewko, and Brent Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In *EUROCRYPT*, pages 663–681, 2012.

[28] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 590–609, Heidelberg, 2009. Springer.

[29] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, and Daniele Venturi. Anonymity-preserving public-key encryption: A constructive approach. In *Privacy Enhancing Technologies*, pages 19–39, 2013.

[30] Huijia Lin and Stefano Tessaro. Amplification of chosen-ciphertext security. In *EUROCRYPT*, pages 503–519, 2013.

[31] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.

[32] Ueli Maurer. Constructive cryptography - a new paradigm for security definitions and proofs. In *TOSCA*, pages 33–56, 2011.

[33] Ueli Maurer. Conditional equivalence of random systems and indistinguishability proofs. In *2013 IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 3150–3154, 2013.

[34] Ueli Maurer and Renato Renner. Abstract cryptography. In *ICS*, pages 1–21, 2011.

[35] Ueli Maurer, Andreas Rüedlinger, and Björn Tackmann. Confidentiality and integrity: A constructive perspective. In *TCC*, pages 209–229, 2012.

[36] Ueli Maurer and Pierre Schmid. A calculus for security bootstrapping in distributed systems. *Journal of Computer Security*, 4(1):55–80, 1996.

[37] Ueli Maurer and Björn Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In *ACM Conference on Computer and Communications Security*, pages 505–515, 2010.

[38] Ueli M. Maurer. Indistinguishability of random systems. In *EUROCRYPT*, pages 110–132, 2002.

[39] Steven Myers, Mona Sergi, and Abhi Shelat. Blackbox construction of a more than non-malleable CCA1 encryption scheme from plaintext awareness. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks*, volume 7485 of *LNCS*, pages 149–165. Springer, 2012.

[40] Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, pages 607–616, 2009.

[41] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011.

[42] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[43] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.

A The Composition Theorem of Constructive Cryptography

The main statement we prove in the main paper shows the security of one protocol step in isolation, i.e., we show for the non-malleable code that it constructs the multi-bit confidential channel from multiple assumed single-bit confidential channels. The composition theorem now states that two such construction steps can be composed: If one (lower-level) protocol constructs the resource that is assumed by the other (higher-level) protocol, then the composition of those two protocols constructs the same resource as the higher-level protocol, but from the resources assumed by the lower-level protocol, under the assumptions that occur in (at least) one of the individual security statements.

The composition theorem was first explicitly stated in [37], but the statement there was restricted to asymptotic settings. Later, in [29], the theorem was stated in a way that also allows to capture concrete security statements. The proof, however, still follows the same steps as the one in [37].

To state the theorem, we make use of a special converter id that behaves transparently (i.e., allows access to the underlying interface of the resource). Furthermore, we assume the operation $[\cdot,\ldots,\cdot]$ to be left-associative; in this way we can simply express multiple resources using the single variable $\mathbf{U}$.

Theorem 8. *Let $\mathbf{R},\mathbf{S},\mathbf{T},\mathbf{U}\in\Phi$ be resources. Let $\pi=(\pi_1,\pi_2)$ and $\psi=(\psi_1,\psi_2)$ be protocols, σ_π and σ_ψ be simulators, and $(\varepsilon_\pi^1,\varepsilon_\pi^2)$, $(\varepsilon_\psi^1,\varepsilon_\psi^2)$ such that*

$$\mathbf{R} \overset{(\pi,\sigma_\pi,(\varepsilon_\pi^1,\varepsilon_\pi^2))}{\Longrightarrow} \mathbf{S} \quad \text{and} \quad \mathbf{S} \overset{(\psi,\sigma_\psi,(\varepsilon_\psi^1,\varepsilon_\psi^2))}{\Longrightarrow} \mathbf{T}.$$

Then

$$\mathbf{R} \overset{(\alpha,\sigma_\alpha,(\varepsilon_\alpha^1,\varepsilon_\alpha^2))}{\Longrightarrow} \mathbf{T}$$

with $\alpha=(\psi_1\circ\pi_1,\psi_2\circ\pi_2)$, $\sigma_\alpha=\sigma_\pi\circ\sigma_\psi$, and $\varepsilon_\alpha^i(\mathbf{D})=\varepsilon_\pi^i(\mathbf{D}\sigma_\psi^E)+\varepsilon_\psi^i(\mathbf{D}\pi_1^A\pi_2^B)$, where $\mathbf{D}\sigma_\psi^E$ and $\mathbf{D}\pi_1^A\pi_2^B$ mean that $\mathbf{D}$ applies the converters at the respective interfaces. Moreover

$$[\mathbf{R},\mathbf{U}] \overset{([\pi,(\mathsf{id},\mathsf{id})],[\sigma_\pi,\mathsf{id}],(\bar\varepsilon_\pi^1,\bar\varepsilon_\pi^2))}{\Longrightarrow} [\mathbf{S},\mathbf{U}],$$

with $\bar\varepsilon_\pi^i(\mathbf{D})=\varepsilon_\pi^i(\mathbf{D}[\cdot,\mathbf{U}])$, where $\mathbf{D}[\cdot,\mathbf{U}]$ means that the distinguisher emulates $\mathbf{U}$ in parallel. (The analogous statement holds with respect to $[\mathbf{U},\mathbf{R}]$ and $[\mathbf{U},\mathbf{S}]$.)

B Channel Resources

From the perspective of constructive cryptography, the purpose of a public-key encryption scheme is to construct a confidential channel from non-confidential channels. A channel is a resource that involves a sender A, a receiver B, and—to model channels with different levels of security—an attacker E. The main types of channels relevant to this work are defined below with respect to interface set $\{A,B,E\}$. All channels are parametrized by a message space $\mathcal{M}\subseteq\{0,1\}^*$, which is only made explicit in the confidential channel (see below), however.

Insecure multiple-use channel. The insecure channel $-\!\!\twoheadrightarrow$ transmits multiple messages $m\in\mathcal{M}$ and corresponds to, for instance, communication via the Internet. If no attacker is present (i.e., in case $\perp^E-\!\!\twoheadrightarrow$), then all messages are transmitted from A to B faithfully. Otherwise (for $-\!\!\twoheadrightarrow$), the communication can be controlled via the E-interface, i.e., the attacker learns all messages input at the A-interface and chooses the messages to be output at the B-interface. The channel is described in more detail in Figure 5.

Authenticated (unreliable) single-use channel. The (single-use) authenticated channel $\bullet\!\!\longrightarrow$, described in Figure 6, allows the sender A to transmit a single message to the receiver B authentically. That means, while the attacker (at the E-interface) can still read the transmitted message, the only influence allowed is delaying the message (arbitrarily, i.e., there is no guarantee that the message will

Channel $\perp^E\!\!- \twoheadrightarrow$	Channel $- \twoheadrightarrow$	
on m *at* A	**on** m *at* A	**on** (inj, m) *at* E
$\quad$ **output** m at B	$\quad$ **output** (msg, m) at E	$\quad$ **output** m at B

Figure 5: *Insecure, multiple-use communication channel from A to B.*

ever be delivered). The channel guarantees that *if* a message is delivered to B, *then* this message was input by A before. There are different constructions that result in the channel $\bullet\!\!\longrightarrow$, based on, for instance, MACs or signature schemes.

Channel $\perp^E\!\!\bullet\!\!\longrightarrow$	Channel $\bullet\!\!\longrightarrow$	
on first m *at* A	**on first** m *at* A	**on first** dlv *at* E
$\quad$ **output** m at B	$\quad$ **output** (msg, m) at E	$\quad$ **output** m at B (if defined)

Figure 6: *Authenticated, single-use communication channel from A to B.*

Confidential multiple-use channel. The k-bit confidential channel $\overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$ allows to transmit multiple messages $m \in \{0,1\}^k$. If no attacker is present (i.e., in case $\perp^E \overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$), then all messages are transmitted from A to B faithfully. Otherwise (for $\overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$), all messages $m \in \{0,1\}^k$ input at the A-interface are stored in a buffer $\mathcal{B}$.[10] The attacker can then choose messages from the buffer $\mathcal{B}$ (by using an index) to be delivered at the B-interface, or inject messages from $\{0,1\}^k$ which are then also output at the B-interface. Note that E cannot inject messages that depend on those in $\mathcal{B}$, i.e., the confidential channel is non-malleable. It is described in more detail in Figure 7.

Channel $\perp^E \overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$	Channel $\overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$	
	init	
	$\quad i \leftarrow 0$	**on** (dlv, i') *at* A
	$\quad \mathcal{B} \leftarrow \emptyset$	$\quad$ **if** $\exists m : (i', m) \in \mathcal{B}$
on $m \in \{0,1\}^k$ *at* A		$\quad\quad$ **output** m at B
$\quad$ **output** m at B	**on** $m \in \{0,1\}^k$ *at* A	
	$\quad i \leftarrow i + 1$	**on** (inj, m') *at* A
	$\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{(i, m)\}$	$\quad$ **output** m' at E
	$\quad$ **output** (msg, i) at E	

Figure 7: *Confidential, multiple-use k-bit channel from A to B.*

C Non-Malleable Codes and the One-Time Pad

C.1 The Malleability of the One-Time Pad

The one-time pad encryption scheme is strongly malleable: If a transmitted ciphertext $e \in \{0,1\}^n$ (corresponding to some message $m \in \{0,1\}^n$) is replaced by a different ciphertext $e' \in \{0,1\}^n$, then the decryption of e' will result in $m \oplus (e \oplus e')$. From the attacker's perspective, the one-time pad is *XOR-malleable*: By replacing the ciphertext e by $e \oplus \delta$ for some $\delta \in \{0,1\}^n$, he can maul the plaintext from m into $m \oplus \delta$.

This circumstance is captured by the XOR-malleable channel (an $\{A, B, E\}$-resource), described in Figure 8. It allows the sender A to input a single message m. If no attacker is present (i.e., in case $\perp^E \overset{n\text{-bit}}{-\!\!\oplus\!\!\rightarrow\!\!\bullet}$), m is simply output at B. Otherwise (for $\overset{n\text{-bit}}{-\!\!\oplus\!\!\rightarrow\!\!\bullet}$), the attacker at interface E can specify a mask δ to be added to the plaintext m.

[10] The $\diamond$ in the symbol $\overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$ is to suggest the presence of $\mathcal{B}$.

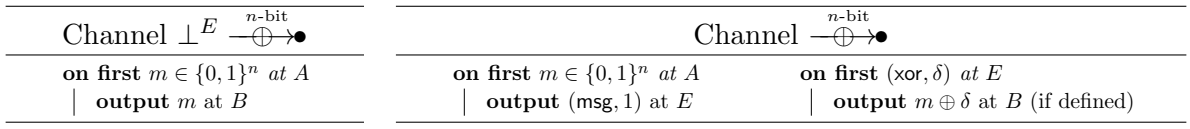

Figure 8: *The XOR-malleable channel from A to B.*

Let $\overset{n\text{-bit}}{\bullet\!=\!\bullet}$ be the resource that outputs a uniformly random n-bit key at A and B and offers no functionality at E. Additionally, let $\longrightarrow$ and $\longrightarrow\!\!\bullet$ the single-use versions of $-\!\!\twoheadrightarrow$ and $-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet$, respectively (cf. Section B). Moreover, consider the (straight-forward) protocol $\mathsf{otp} = (\mathsf{otp\text{-}enc}, \mathsf{otp\text{-}dec})$ that implements one-time pad encryption. Then,

$$[\overset{n\text{-bit}}{\bullet\!=\!\bullet}, \longrightarrow] \quad \overset{\mathsf{otp}}{\Longrightarrow} \quad \xrightarrow{n\text{-bit}}\!\!\bullet\,. \tag{7}$$

The proof of (7) is a restricted case of [37, Lemma 2].

C.2 Getting Rid of the Malleability

One can overcome the malleability described above using a non-malleable code secure against the class $\mathcal{F}_{\mathsf{bit}}$ of tampering functions that modify every bit independently.[11] Thus, assume there exists a (k, n)-coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ that is $(\mathcal{F}_{\mathsf{bit}}, 1, 1, \varepsilon)$-non-malleable for some $\varepsilon > 0$ (according to Definition 3),[12] and consider the following protocol $\mathsf{nmc} = (\mathsf{encode}, \mathsf{decode})$: Converter encode, obtaining a message $m \in \{0,1\}^k$ at its outside interface, computes $c \leftarrow\!\!\$\, \mathrm{Enc}(m)$ and outputs c at its inside interface; converter decode, obtaining a message $c' \in \{0,1\}^n$ at its inside interface, computes $m' \leftarrow \mathrm{Dec}(c')$ and outputs m' at its outside interface.

Theorem 9. *Assume that* $(\mathrm{Enc}, \mathrm{Dec})$ *is a* (k, n)*-coding scheme and* $(\mathcal{F}_{\mathsf{bit}}, \varepsilon)$*-non-malleable. Then, there exists a simulator* σ *such that*

$$\xrightarrow{n\text{-bit}}\!\!\bullet \quad \overset{(\mathsf{nmc},\sigma,(0,\varepsilon))}{\Longrightarrow} \quad \xrightarrow{k\text{-bit}}\!\!\bullet\,.$$

Proof. The availability condition holds by the correctness of the code.

Let $\mathcal{F} := \mathcal{F}_{\mathsf{bit}}$, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},1,1}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,1,1}$, where τ is the simulator guaranteed to exist by Definition 3. Note that a function $f \in \mathcal{F}$ can be characterized by a vector $\chi(f) = (f_1, \ldots, f_n)$ where $f_i \in \{\mathsf{zero}, \mathsf{one}, \mathsf{keep}, \mathsf{flip}\}$, with the meaning that f takes as input a codeword c and outputs a codeword $c' = c'_1 \cdots c'_n$ in which each bit is either set to 0 (zero), set to 1 (one), left unchanged (keep), or flipped (flip).

Consider the following simulator σ (based on τ), which simulates the E-interface of $\xrightarrow{n\text{-bit}}\!\!\bullet$ at its outside interface: When it receives $(\mathsf{msg}, 1)$ at the inside interface, it outputs $(\mathsf{msg}, 1)$ at the outside interface. When it gets (xor, δ) with $\delta = \delta_1 \cdots \delta_n$ at the outside interface, it computes $x' \leftarrow\!\!\$\, \tau(1, f)$, where f is the function such that $\chi(f) = (f_1, \ldots, f_n)$ for

$$f_j := \begin{cases} \mathsf{keep} & \text{if } \delta_j = 0, \text{ and} \\ \mathsf{flip} & \text{if } \delta_j = 1. \end{cases}$$

If $x = (\mathsf{same}, 1)$, σ outputs $(\mathsf{dlv}, 1)$ at the inside interface, and otherwise, it outputs (inj, x).

Consider the following reduction $\mathbf{C}$, which provides interfaces A, B, and E on the outside and expects to connect to either $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ or $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ on the inside: When a message m is input at the A-interface, $\mathbf{C}$ outputs $(\mathsf{msg}, 1)$ at the E interface. When (xor, δ) is input at interface E, it computes f

[11] The proof below makes apparent that one would in fact only need a non-malleable code secure against tampering functions that either keep or flip each bit of the encoding independently.

[12] This is a slight abuse of notation, since $\mathcal{F}_{\mathsf{bit}}$ is just a single family of tamper functions and not a sequence thereof. This is acceptable since only non-adaptive, single-shot non-malleability is considered in this section.

the same way τ does and outputs (tamper, f) at the inside interface. The subsequent response x' is output at interface B.

Consider the systems $\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathsf{encode}^A\mathsf{decode}^B \xrightarrow{n\text{-bit}}\!\!\!\!\bullet$. The output at the E-interface upon input $m \in \{0,1\}^k$ at the A-interface is the same in in both systems, namely $(\mathsf{msg}, 1)$. Moreover, with $\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}}$ the output at the B-interface on input (xor, δ) at the E-interface is computed by applying the tampering function f corresponding to δ to the encoding of the value m; exactly as in $\mathsf{encode}^A\mathsf{decode}^B \xrightarrow{n\text{-bit}}\!\!\!\!\bullet$.

Consider the systems $\mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau}$ and $\sigma^E \xrightarrow{k\text{-bit}}\!\!\!\!\bullet$. Again, when $m \in \{0,1\}^k$ is input at A, $(\mathsf{msg}, 1)$ is output at E in either system. In $\mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau}$, the output at the B-interface on input (xor, δ) at the E-interface is computed by invoking the simulator τ on the tampering function f corresponding to δ; exactly as in $\sigma^E \xrightarrow{k\text{-bit}}\!\!\!\!\bullet$.

Therefore,

$$\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}} \equiv \mathsf{encode}^A\mathsf{decode}^B \xrightarrow{n\text{-bit}}\!\!\!\!\bullet \qquad \text{and} \qquad \mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau} \equiv \sigma^E \xrightarrow{k\text{-bit}}\!\!\!\!\bullet,$$

which implies

$$\Delta^{\mathbf{D}}(\mathsf{encode}^A\mathsf{decode}^B \xrightarrow{n\text{-bit}}\!\!\!\!\bullet, \sigma^E \xrightarrow{k\text{-bit}}\!\!\!\!\bullet) = \Delta^{\mathbf{D}}(\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau}) = \Delta^{\mathbf{DC}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) \leq \varepsilon$$

for all distinguishers $\mathbf{D}$. □

D SD-CCA Security and Deferred Material from Section 3

In Section 3, we provide a protocol (a pair of converters) $\mathsf{m\text{-}pke} = (\mathsf{m\text{-}encrypt}, \mathsf{m\text{-}decrypt})$ that achieves transformation

$$[\longleftarrow\!\!\!\!\bullet, - \twoheadrightarrow] \quad \overset{\mathsf{m\text{-}pke}}{\Longrightarrow} \quad \overset{k\text{-bit}}{\longrightarrow\!\!\!\!\diamond\!\!\!\!\twoheadrightarrow}\!\!\!\!\bullet \tag{5}$$

and results from composing protocol $\mathsf{1\text{-}pke} = (\mathsf{1\text{-}encrypt}, \mathsf{1\text{-}decrypt})$, achieving

$$[\longleftarrow\!\!\!\!\bullet, - \twoheadrightarrow] \quad \overset{\mathsf{1\text{-}pke}}{\Longrightarrow} \quad [\overset{1\text{-bit}}{\longrightarrow\!\!\!\!\diamond\!\!\!\!\twoheadrightarrow}\!\!\!\!\bullet]^n, \tag{3}$$

with protocol $\mathsf{nmc} = (\mathsf{encode}, \mathsf{decode})$, achieving

$$[\overset{1\text{-bit}}{\longrightarrow\!\!\!\!\diamond\!\!\!\!\twoheadrightarrow}\!\!\!\!\bullet]^n \quad \overset{\mathsf{nmc}}{\Longrightarrow} \quad \overset{k\text{-bit}}{\longrightarrow\!\!\!\!\diamond\!\!\!\!\twoheadrightarrow}\!\!\!\!\bullet\ . \tag{4}$$

In this section we fill in the deferred details: First, we formally define the notion of *self-destruct CCA security (SD-CCA)* (Section D.1). Second, we show in detail how protocol $\mathsf{1\text{-}pke}$ is obtained from a 1-bit SD-CCA-secure PKE and prove its security (Section D.2). Third, we formally prove Theorem 1, which states that nmc achieves construction (4) (Section D.3). Fourth, we show that protocol $\mathsf{m\text{-}pke}$ corresponds (in a straight-forward manner) to a PKE scheme Π and prove that Π is SD-CCA secure (Section D.4). Finally, for comparison to our constructive approach we also provide a direct game-based proof of the fact that combining single-bit SD-CCA-secure PKE with a non-malleable code as shown above yields a multi-bit SD-CCA-secure PKE scheme (Section D.5).

D.1 Formal Definition of SD-CCA

In this section we define SD-CCA security and a *replayable* variant thereof called SD-RCCA security. The notion of *replayable* CCA security (RCCA) in general was introduced by Canetti *et al.* [3] to deal with the artificial strictness of full CCA security. Roughly, RCCA security weakens full CCA security by potentially allowing an attacker to maul a ciphertext into one that decrypts to the identical message.

The only difference between the SD-CCA game and the standard game used to define CCA is that the decryption oracle self-destructs, i.e., it stops processing further queries once an invalid ciphertext is queried. Note that the self-destruct feature only affects the decryption oracle; the adversary is still allowed to get the challenge ciphertext after provoking a self-destruct. The game is phrased as a distinguishing problem between the two systems $\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ and $\mathbf{G}_1^{\mathsf{sd\text{-}cca}}$ described in Figure 9.

System $\mathbf{G}_b^{\mathsf{sd\text{-}cca}}$

init
  $(\mathsf{pk}, \mathsf{sk}) \leftarrow K$
  output pk

on $(\mathsf{chall}, m_0, m_1)$ *with* $|m_1| = |m_0|$
  $e \leftarrow E_{\mathsf{pk}}(m_b)$
  output e

on (dec, e')
  $m' \leftarrow D_{\mathsf{sk}}(e')$
  if $m' = \diamond$
    self-destruct
  else if $e' = e$
    output test
  else
    output m'

Figure 9: *System* $\mathbf{G}_b^{\mathsf{sd\text{-}cca}}$*, where* $b \in \{0,1\}$*, defining SD-CCA security of a PKE scheme* $\Pi = (K, E, D)$*. The command* **self-destruct** *causes the system to output* $\diamond$ *and to answer all future decryption queries by* $\diamond$*.*

Definition 5. *A PKE scheme* $\Pi = (K, E, D)$ *is* (t, q, ε)*-SD-CCA secure if*

$$\Delta^{\mathbf{D}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}) \leq \varepsilon$$

for all distinguishers $\mathbf{D}$ *with running time at most* t *and making at most* q *decryption queries.*

For $b \in \{0,1\}$, let $\mathbf{G}_b^{\mathsf{sd\text{-}rcca}}$ be the game that behaves as $\mathbf{G}_b^{\mathsf{sd\text{-}cca}}$, except that it outputs test whenever $D_{\mathsf{sk}}(e') \in \{m_0, m_1\}$ for a decryption query e'.

Definition 6. *A PKE scheme* $\Pi = (K, E, D)$ *is* (t, q, ε)*-SD-RCCA secure if*

$$\Delta^{\mathbf{D}}(\mathbf{G}_0^{\mathsf{sd\text{-}rcca}}, \mathbf{G}_1^{\mathsf{sd\text{-}rcca}}) \leq \varepsilon$$

for all distinguishers $\mathbf{D}$ *with running time at most* t *and making at most* q *decryption queries.*

D.2 Single-bit Channels from Single-bit PKE

Following the proof of [9, Theorem 2], we first show that a 1-bit SD-CCA-secure PKE scheme can be used to design a protocol $\mathsf{1\text{-}pke}'$ that achieves the construction

$$[\longleftarrow\!\bullet, -\!\!\twoheadrightarrow] \quad \overset{\mathsf{1\text{-}pke}'}{\Longrightarrow} \quad \overset{\text{1-bit}}{-\!\diamond\!\!\twoheadrightarrow\!\bullet}, \tag{2}$$

Using the composition theorem, one then obtains

$$[\longleftarrow\!\bullet, -\!\!\twoheadrightarrow]^n \quad \overset{\mathsf{1\text{-}pke}''}{\Longrightarrow} \quad [\overset{\text{1-bit}}{-\!\diamond\!\!\twoheadrightarrow\!\bullet}]^n,$$

where $\mathsf{1\text{-}pke}'' = (\mathsf{1\text{-}encrypt}'', \mathsf{1\text{-}decrypt}'')$ and where $\mathsf{1\text{-}encrypt}''$ and $\mathsf{1\text{-}decrypt}''$ are the n-fold parallel composition of $\mathsf{1\text{-}encrypt}'$ and $\mathsf{1\text{-}decrypt}'$, respectively. A slight modification of protocol $\mathsf{1\text{-}pke}''$ yields the protocol $\mathsf{1\text{-}pke}$ for construction (3). Essentially, all public keys are concatenated and sent via a single $\longleftarrow\!\bullet$. A proof of security is straight-forward.

Towards a proof of (2), let $\Pi = (K, E, D)$ be a PKE scheme and consider the following pair of protocol converters $\mathsf{1\text{-}pke}' = (\mathsf{1\text{-}encrypt}', \mathsf{1\text{-}decrypt}')$: Converter $\mathsf{1\text{-}encrypt}'$ works as follows: It initially expects a public key pk at the inside interface. When a message m is input at the outside interface, $\mathsf{1\text{-}encrypt}'$ outputs $e \leftarrow\!\!\$\ E_{\mathsf{pk}}(m)$ at the inside interface. Converter $\mathsf{1\text{-}decrypt}'$ initially generates a key pair $(\mathsf{pk}, \mathsf{sk})$ using key-generation algorithm K and outputs pk at the inside interface. When $\mathsf{1\text{-}decrypt}'$ receives e' at the inside interface, it computes $m' \leftarrow D_{\mathsf{sk}}(e')$, outputs m' at the outside interface, and if $m' = \diamond$, implements the self-destruct mode, i.e., outputs $\diamond$ on the outside for all future ciphertexts received on the inside.

Theorem 10. *There exists a simulator σ and for any $\ell \in \mathbb{N}$ there exists a (efficient) reduction $\mathbf{C}$ such that for every $\mathbf{D}$,*

$$\Delta^{\mathbf{D}}(\mathsf{1\text{-}encrypt}'^{A}\mathsf{1\text{-}decrypt}'^{B}[\longleftarrow\!\bullet, -\overset{\ell}{\twoheadrightarrow}], \sigma^{E} \overset{\text{1-bit},\ell}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet}) \leq \ell \cdot \Delta^{\mathbf{DC}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}),$$

where the additional superscript ℓ indicates that the channel processes only the first ℓ messages at interface A.

Proof. First, consider the following simulator σ for interface E of $\overset{\text{1-bit},\ell}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet}$: Initially, it generates a key pair $(\mathsf{pk}, \mathsf{sk})$ and outputs pk at the outside interface. When it receives (msg, i) at the inside interface, it generates an encryption $e \leftarrow\!\!\$\, E_{\mathsf{pk}}(\bar{m})$ of some 1-bit message $\bar{m}$, outputs (msg, e) at the outside interface, and records (e, i). When (inj, e') is input at the outside interface, σ proceeds as follows: If (e', i') has been recorded for some i', it outputs (dlv, i') at its inside interface. Otherwise, it computes $m' \leftarrow D_{\mathsf{sk}}(e')$, outputs (inj, m') at the inside interface, and if $m' = \diamond$, it implements the self-destruct mode: for any future (inj, e') input at the outside interface it outputs $(\mathsf{inj}, \diamond)$ at the inside interface.

Consider now the problem of distinguishing the two systems

$$\mathbf{U} := \mathsf{1\text{-}encrypt}^{A}\mathsf{1\text{-}decrypt}^{B}[\longleftarrow\!\bullet, -\overset{1}{\twoheadrightarrow}] \qquad \text{and} \qquad \mathbf{V} := \sigma^{E} \overset{\text{1-bit},1}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet}\,.$$

A distinguisher $\mathbf{D}$ connected to $\mathbf{U}$ initially sees a public key at interface E. If $\mathbf{D}$ inputs a message m at interface A, an encryption of m (created by $\mathsf{1\text{-}encrypt}$) is output at interface E. When $\mathbf{D}$ inputs a ciphertext e' at E, it sees a decryption of e' (by $\mathsf{1\text{-}decrypt}$) at B. The system $\mathbf{V}$ behaves differently: Initially, $\mathbf{D}$ also sees a public key. But when it inputs a message m at A, an encryption e of $\bar{m}$ is output at interface E (by simulator σ). When e is input at interface E, m is output at B (as σ issues a deliver instruction to the channel). When $e' \neq e$ is input at E, a decryption of e' (injected by σ) is output at B.

The translation between the channel setting and the game setting is achieved by the following reduction system $\mathbf{C}'$: Initially, $\mathbf{C}'$ takes a value (which will be the public key pk) from the game and outputs it at the E-interface. When a message m is input at interface A of $\mathbf{C}'$, $(\mathsf{chall}, m, \bar{m})$ is output to the game. The resulting challenge e is output as (msg, e) at interface E. When (inj, e) is input at interface E, $\mathbf{C}'$ outputs m at interface B. When (inj, e') with $e' \neq e$ is input at interface E, $\mathbf{C}'$ passes (dec, e') to the game and outputs the answer m' at interface B. If $m' = \diamond$, $\mathbf{C}'$ implements the self-destruct mode: All future (inj, e') at interface E are handled by outputting $\diamond$ at B. We have

$$\mathbf{C}'\mathbf{G}_0^{\mathsf{sd\text{-}cca}} \equiv \mathbf{U} \qquad \text{and} \qquad \mathbf{C}'\mathbf{G}_1^{\mathsf{sd\text{-}cca}} \equiv \mathbf{V},$$

and thus

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathsf{1\text{-}encrypt}^{A}\mathsf{1\text{-}decrypt}^{B}[\longleftarrow\!\bullet, -\overset{\ell}{\twoheadrightarrow}], \sigma^{E} \overset{\text{1-bit},\ell}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet}) &\leq \ell \cdot \Delta^{\mathbf{DC}''}(\mathbf{U}, \mathbf{V}) \\ &= \ell \cdot \Delta^{\mathbf{DC}''}(\mathbf{C}'\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{C}'\mathbf{G}_1^{\mathsf{sd\text{-}cca}}) \\ &= \ell \cdot \Delta^{\mathbf{DC}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}),\end{aligned}$$

where $\mathbf{C} := \mathbf{C}''\mathbf{C}'$ and the first inequality follows from a standard hybrid argument for a reduction system $\mathbf{C}''$. □

D.3 Tying the Channels Together

Theorem 1. *For any $\ell, q \in \mathbb{N}$, if* (Enc, Dec) *is $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, \ell, q)$-continuously non-malleable, there exists a simulator σ such that*

$$\left[\overset{\text{1-bit},\ell,q}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet}\right]^{n} \qquad \overset{(\mathsf{nmc},\sigma,(0,\varepsilon))}{\Longrightarrow} \qquad \overset{k\text{-bit},\ell,q}{\multimap\!\diamond\!\twoheadrightarrow\!\bullet},$$

where the additional superscripts ℓ, q on a channel mean that it only processes the first ℓ queries at the A-interface and only the first q queries at the E-interface.

Proof. The availability condition holds by the correctness of the code.

Let $\mathcal{F} := \mathcal{F}_{\mathsf{copy}}$, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},\ell,q}$, and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,\ell,q}$ where τ is the simulator guaranteed to exist by Definition 3.

Consider the following simulator σ (based on τ), which simulates the E-sub-interfaces of the 1-bit confidential channels at its outside interface: When (msg, i) is received at the inside interface, it outputs (msg, i) at each outside sub-interface corresponding to a 1-bit confidential channel. Whenever σ receives one instruction to either deliver[13] ((dlv, i') for $i' \in \mathbb{N}$) or inject ((inj, m') for $m' \in \{0,1\}$) a bit at each outside sub-interface corresponding to one of the confidential channels, it assembles these to a function f with $\chi(f) = (f_1, \ldots, f_n)$ as follows: For all $j = 1, \ldots, n$,

$$f_j := \begin{cases} \mathsf{zero} & \text{if the instruction on the } j^{\text{th}} \text{ sub-interface is } (\mathsf{inj}, 0), \\ \mathsf{one} & \text{if the instruction on the } j^{\text{th}} \text{ sub-interface is } (\mathsf{inj}, 1), \\ \mathsf{copy}_{i'} & \text{if the instruction on the } j^{\text{th}} \text{ sub-interface is } (\mathsf{dlv}, i'). \end{cases}$$

Then, σ invokes τ to obtain $x' \leftarrow\!\!\$\ \tau(i, f)$, where i is the number of instructions (msg, i) received at the inside interface so far. If $x' = (\mathsf{same}, j)$, σ outputs (dlv, j) at the inside interface. Otherwise, it outputs (inj, x'). If $x' = \diamond$, σ outputs $(\mathsf{inj}, \diamond)$ at the inside interface and implements the self-destruct mode, i.e., outputs $(\mathsf{inj}, \diamond)$ at the inside interface for all future inputs to the simulated interfaces of the single-bit channels.

Consider the following reduction $\mathbf{C}$, which provides interfaces A, B, and E on the outside and expects to connect to either $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ or $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ on the inside. When a message m is input at the A-interface, $\mathbf{C}$ outputs (encode, m) on the inside. Similarly to σ, it repeatedly collects instructions input at the E-sub-interfaces and uses them to form a tamper function f, which it outputs on the inside as (tamper, f). Then, it outputs the answer x' received on the inside at the B-interface. Additionally, if $x' = \diamond$, $\mathbf{C}$ implements the self-destruct mode, i.e., subsequently only outputs $\diamond$ at interface B.

One observes that

$$\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}} \equiv \mathsf{encode}^A \mathsf{decode}^B [\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n \qquad \text{and} \qquad \mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau} \equiv \sigma^E \overset{k\text{-bit},\ell,q}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}\ .$$

Thus, for all distinguishers $\mathbf{D}$,

$$\Delta^{\mathbf{D}}(\mathsf{encode}^A \mathsf{decode}^B [\overset{\text{1-bit}}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}]^n, \sigma^E \overset{k\text{-bit},\ell,q}{\multimap\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}) = \Delta^{\mathbf{D}}(\mathbf{CS}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{CS}^{\mathsf{simu}}_{\mathcal{F},\tau}) = \Delta^{\mathbf{DC}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) \leq \varepsilon.$$

□

D.4 From Protocols to PKE Schemes

The PKE scheme $\Pi = (K, E, D)$ corresponding to our protocol $\mathsf{m\text{-}pke}$ can be obtained as follows. The key generation algorithm K generates n independent key pairs of the 1-bit scheme. The encryption algorithm E first encodes a message using a non-malleable code and then encrypts each bit of the resulting encoding independently and outputs the n resulting ciphertexts. The decryption algorithm D first decrypts the n ciphertexts, decodes the resulting bitstring, and outputs the decoded message or the symbol $\diamond$, indicating an invalid ciphertext, if any of these steps fails. The scheme is described in more detail in Figure 10.

PKE scheme Π achieves only *replayable* SD-CCA security (SD-RCCA). The reason for this is that given any ciphertext e, an attacker can replace the first component of e by a fresh encryption of a randomly chosen bit and thereby obtain, with probability 1/2, a ciphertext e' that decrypts to the same message as e. In [3], the authors provide generic ways to achieve full CCA security from replayable CCA security. As shown in subsequent work [8] to this paper, these techniques can also be applied in the context of self-destruct CCA security.

[13] For simplicity, assume that no deliver instruction (dlv, i') for some i' greater than the largest number i received via (msg, i) at the inside interface so far is input.

PKE Scheme $\Pi' = (K', E', D')$

Key Generation K'
for $i \leftarrow 1$ **to** n
  $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\!\!\$\ K$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
return $(\mathsf{pk}, \mathsf{sk})$

Encryption $E'_{\mathsf{pk}}(m)$
$c = c_1 \cdots c_n \leftarrow \mathrm{Enc}(m)$
for $i \leftarrow 1$ **to** n
  $e_i \leftarrow\!\!\$\ E_{\mathsf{pk}_i}(c_i)$
return $e = (e_1, \ldots, e_n)$

Decryption $D'_{\mathsf{sk}}(e)$
for $i \leftarrow 1$ **to** n
  $c_i \leftarrow\!\!\$\ D_{\mathsf{sk}_i}(e_i)$
  if $c_i = \diamond$
    return $\diamond$
$m \leftarrow \mathrm{Dec}(c_1 \cdots c_n)$
return m

Figure 10: *The k-bit PKE scheme $\Pi' = (K', E', D')$ built from a 1-bit PKE scheme $\Pi = (K, E, D)$ and a (k, n)-coding scheme* (Enc, Dec).

It remains to prove that our PKE scheme is indeed SD-RCCA secure, based on the security of protocol m-pke; the proof follows that of [9, Theorem 4]. In the following, let

$$\mathbf{U} := \mathsf{m\text{-}encrypt}^A \mathsf{m\text{-}decrypt}^B[\leftarrow\!\!\bullet, - \twoheadrightarrow] \qquad \text{and} \qquad \mathbf{V} := \sigma^E \overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet},$$

where σ is an *arbitrary* simulator.

Theorem 11. *There exist efficient reductions $\mathbf{C}_0$ and $\mathbf{C}_1$ such that, for all adversaries $\mathbf{D}$,*

$$\Delta^{\mathbf{D}}(\mathbf{G}_0^{\mathsf{sd\text{-}rcca}}, \mathbf{G}_1^{\mathsf{sd\text{-}rcca}}) \leq \Delta^{\mathbf{DC}_0}(\mathbf{U}, \mathbf{V}) + \Delta^{\mathbf{DC}_1}(\mathbf{U}, \mathbf{V}).$$

Proof. Consider the following reductions $\mathbf{C}_0$ and $\mathbf{C}_1$. Both connect to an $\{A, B, E\}$-resource on the inside and provide a single interface on the outside: Initially, both obtain $(\mathsf{msg}, \mathsf{pk})$ at the inside E-interface and output pk at the outside interface. When $(\mathsf{chall}, m_0, m_1)$ is received on the outside, $\mathbf{C}_0$ outputs m_0 at the inside A-interface and $\mathbf{C}_1$ outputs m_1. Subsequently, (msg, e) is received at the inside E-interface, and e is output on the outside by both systems. When a decryption query (dec, e') is received on the outside, both systems output (inj, e') at the inside E-interface. A subsequently received message m' at B is output on the outside by both systems (as answer to the decryption query) unless $m' \in \{m_0, m_1\}$, in which case test is returned. Moreover, if $m' = \diamond$, both reduction systems self-destruct, i.e., they answer all future decryption queries by $\diamond$. We have

$$\mathbf{C}_0\mathbf{U} \equiv \mathbf{G}_0^{\mathsf{sd\text{-}rcca}} \qquad \text{and} \qquad \mathbf{C}_1\mathbf{U} \equiv \mathbf{G}_1^{\mathsf{sd\text{-}rcca}} \qquad \text{and} \qquad \mathbf{C}_0\mathbf{V} \equiv \mathbf{C}_1\mathbf{V},$$

where the last equivalence follows from the fact that, in $\mathbf{V}$, the input from $\overset{k\text{-bit}}{-\!\!\diamond\!\!\twoheadrightarrow\!\!\bullet}$ to σ is the same in both systems (the output $(\mathsf{msg}, 1)$) and that decryption queries causing m_0 or m_1 to be output at the B-interface are answered by test. Hence,

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{G}_0^{\mathsf{sd\text{-}rcca}}, \mathbf{G}_1^{\mathsf{sd\text{-}rcca}}) &= \Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{U}, \mathbf{C}_1\mathbf{U}) \leq \Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{U}, \mathbf{C}_0\mathbf{V}) + \Delta^{\mathbf{D}}(\mathbf{C}_0\mathbf{V}, \mathbf{C}_1\mathbf{V}) + \Delta^{\mathbf{D}}(\mathbf{C}_1\mathbf{V}, \mathbf{C}_1\mathbf{U}) \\ &= \Delta^{\mathbf{DC}_0}(\mathbf{U}, \mathbf{V}) + \Delta^{\mathbf{DC}_1}(\mathbf{U}, \mathbf{V}).\end{aligned}$$

□

D.5 Game-Based Proof

This section contains a direct proof that our k-bit PKE scheme Π' is SD-RCCA secure if Π is a SD-CCA-secure 1-bit PKE scheme and (Enc, Dec) a continuously non-malleable coding scheme. The proof is a hybrid argument and is obtained by "unwrapping" the concatenation of the theorems from Sections 3 and D.4. The modular nature and the intuitive simplicity of the proofs in Section 3 are lost, however. Concretely, we prove:

Theorem 12. *If Π is a $(t+\tilde{t}, q, \varepsilon)$-SD-CCA secure 1-bit PKE scheme and* (Enc, Dec) *is a $(\mathcal{F}_{\mathsf{copy}}, \varepsilon_{\mathsf{nmc}}, 1, q)$-non-malleable coding scheme,*[14] *then Π' is a $(t, q, 2n\varepsilon + 2\varepsilon_{\mathsf{nmc}})$-SD-RCCA PKE scheme, where $\tilde{t}$ represents a (very) small overhead.*

[14] The reason one does not need *adaptive* continuous non-malleability is because PKE games are normally formulated as single-challenge games.

In the following, let $\mathcal{F} := \mathcal{F}_{\mathsf{copy}}$. Moreover, let $\mathbf{G}_0^{\mathsf{sd\text{-}rcca}}$ and $\mathbf{G}_1^{\mathsf{sd\text{-}rcca}}$ be the systems capturing the SD-RCCA security for Π', and similarly $\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ and $\mathbf{G}_1^{\mathsf{sd\text{-}cca}}$ the ones for the SD-CCA security of Π. The proof of Theorem 12 follows from the following lemma:

Lemma 13. *For $b \in \{0,1\}$ and $i \in [n]$, there exist reduction systems $\mathbf{C}_b^{(i)}$ and $\mathbf{C}_b b$ such that for all distinguishers $\mathbf{D}$.*

$$\Delta^{\mathbf{D}}(\mathbf{G}_0^{\mathsf{sd\text{-}rcca}}, \mathbf{G}_1^{\mathsf{sd\text{-}rcca}}) \leq \sum_{b,i} \Delta^{\mathbf{DC}_b^{(i)}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}) + \sum_b \Delta^{\mathbf{DC}_b b}(\mathbf{S}_{\mathcal{F}}^{\mathsf{real}}, \mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}}),$$

where τ is the simulator for the non-malleable code. Moreover, all reductions preserve the number of queries q.

Proof (of Theorem 12). Let $\tilde{t}$ be the maximal occurring overhead caused by the reduction systems $\mathbf{C}_b^{(i)}$. Fix a distinguisher $\mathbf{D}$ having running time t and making at most q decryption queries. For all $b \in \{0,1\}$ and $i \in [n]$, system $\mathbf{DC}_b^{(i)}$ makes no more decryption queries than $\mathbf{D}$ and has running time at most $t + \tilde{t}$, and $\mathbf{DC}_b$ makes at most as many tamper queries as $\mathbf{D}$ makes decryption queries. Hence, $\Delta^{\mathbf{DC}_b^{(i)}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}) \leq \varepsilon$ and $\Delta^{\mathbf{DC}_b}(\mathbf{S}_{\mathcal{F},0}^{\mathsf{lor}}, \mathbf{S}_{\mathcal{F},1}^{\mathsf{lor}}) \leq \varepsilon_{\mathsf{nmc}}$, which completes the proof. □

Towards a proof of Lemma 13, consider the following hybrid systems for $b \in \{0,1\}$ and $i \in [n]$: $\mathbf{H}_b^{(i)}$ proceeds as $\mathbf{G}_b^{\mathsf{sd\text{-}rcca}}$ except that the challenge query $(\mathsf{chall}, m_0, m_1)$ and decryption queries (dec, e') are handled differently:

- **Challenge query:** The first i bits of the encoding $c = c_1 \cdots c_n$ of m_b are replaced by uniformly random and independent bits. The resulting n-bit string is then encrypted bit-wise (as done by E). This results in the challenge ciphertext $e = (e_1, \ldots, e_n)$.
- **Decryption query:** Let $e' = (e'_1, \ldots, e'_n)$. System $\mathbf{H}_b^{(i)}$ computes $c' = c'_1 \cdots c'_n$, where

$$c'_j = \begin{cases} c_j & \text{if } e'_j = e_j, \text{ and} \\ D_{\mathsf{sk}_j}(e'_j) & \text{otherwise.} \end{cases}$$

  Then, $\mathbf{H}_b^{(i)}$ outputs $\mathrm{Dec}(c')$ as the answer to the decryption query.[15]

Let $\mathbf{H}_b^{(0)} := \mathbf{G}_b^{\mathsf{sd\text{-}rcca}}$.

Lemma 14. *For all $b \in \{0,1\}$ and $i = 1, \ldots, n$, there exists $\mathbf{C}_b^{(i)}$ such that for all $\mathbf{D}$*

$$\Delta^{\mathbf{D}}(\mathbf{H}_b^{(i-1)}, \mathbf{H}_b^{(i)}) = \Delta^{\mathbf{DC}_b^{(i)}}(\mathbf{G}_0^{\mathsf{sd\text{-}cca}}, \mathbf{G}_1^{\mathsf{sd\text{-}cca}}).$$

Proof. Fix b and i. System $\mathbf{C}_b^{(i)}$ works as follows: Initially, it generates $n-1$ key pairs $(\mathsf{pk}_j, \mathsf{sk}_j)$ for $j \in [n] \setminus \{i\}$, obtains pk_i (but not sk_i) on the inside interface (from $\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ or $\mathbf{G}_1^{\mathsf{sd\text{-}cca}}$), and outputs $\mathsf{pk} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ on the outside. When it receives $(\mathsf{chall}, m_0, m_1)$ on the outside, it computes an encoding $c = c_1 \cdots c_n \leftarrow \mathrm{Enc}(m_b)$. Then, it chooses i random bits $\tilde{c}_1, \ldots, \tilde{c}_i$ and computes

$$e_j = \begin{cases} E_{\mathsf{pk}_j}(\tilde{c}_j) & \text{for } j < i, \text{ and} \\ E_{\mathsf{pk}_j}(c_j) & \text{for } j > i. \end{cases}$$

Moreover, it outputs $(\mathsf{chall}, c_i, \tilde{c}_i)$ at the inside and obtains a ciphertext e_i. It finally outputs $e = (e_1, \ldots, e_n)$ at the outside interface.

When $\mathbf{C}_b^{(i)}$ receives a decryption query (dec, e') for $e' = (e'_1, \ldots, e'_n)$ at its outside interface, it proceeds as follows: For $j \neq i$, it computes c'_j as $\mathbf{H}_b^{(i)}$ does. Moreover, if $e'_i = e_i$, it sets $c'_i \leftarrow c_i$.

[15] Assume here and below that $\mathrm{Dec}(c') = \diamond$ if any of the bits c'_j equal $\diamond$.

Otherwise, it outputs (dec, e'_i) at the inside interface and obtains the answer c'_i. Then, it computes $m' \leftarrow \mathrm{Dec}(c')$. If $m' = \diamond$, $\mathbf{C}_b^{(i)}$ implements the self-destruct mode. Otherwise, it outputs m' at the outside interface unless $m' \in \{m_0, m_1\}$, in which case the output is test.

Consider the systems $\mathbf{C}_b^{(i)}\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ and $\mathbf{H}_b^{(i-1)}$. Both systems generate the public key in the same fashion. As to the challenge ciphertext, the first $i-1$ ciphertext components e_j generated by $\mathbf{C}_b^{(i)}\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ are encryptions of random bits $\tilde{c}_j$, whereas the i^{th} and the remaining components are encryptions of the corresponding bits of an encoding of m_b (generated by $\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ and $\mathbf{C}_b^{(i)}$, respectively). The same is true for $\mathbf{H}_b^{(i-1)}$. The result of a decryption query (dec, e') sent to $\mathbf{C}_b^{(i)}\mathbf{G}_0^{\mathsf{sd\text{-}cca}}$ is $\mathrm{Dec}(c')$ for $c' = c'_1 \cdots c'_n$, where $c'_j = D_{\mathsf{sk}_j}(e'_j)$ unless $j < i$ and $e'_j = e_j$, in which case $c'_j = \tilde{c}_j$. Again, the same holds for system $\mathbf{H}_b^{(i-1)}$. Moreover, both systems answer test if $\mathrm{Dec}(c') \in \{m_0, m_1\}$.

Systems $\mathbf{C}_b^{(i)}\mathbf{G}_1^{\mathsf{sd\text{-}cca}}$ and $\mathbf{H}_b^{(i)}$ are compared similarly. Therefore,

$$\mathbf{C}_b^{(i)}\mathbf{G}_0^{\mathsf{sd\text{-}cca}} \equiv \mathbf{H}_b^{(i-1)} \quad \text{and} \quad \mathbf{C}_b^{(i)}\mathbf{G}_1^{\mathsf{sd\text{-}cca}} \equiv \mathbf{H}_b^{(i)},$$

which concludes the proof. □

Lemma 15. *There exists* $\mathbf{C}_b$ *such that*

1. $\mathbf{C}_b\mathbf{S}_{\mathcal{F}}^{\mathsf{real}} \equiv \mathbf{H}_b^{(n)}$ *for* $b \in \{0,1\}$, *and*
2. $\mathbf{C}_0\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}} \equiv \mathbf{C}_1\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}}$.

Proof. System $\mathbf{C}_b$ works as follows: Initially, it generates n key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ and outputs $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ at the outside interface. When it receives $(\mathsf{chall}, m_0, m_1)$ at the outside interface, it chooses n random values $\tilde{c}_1, \ldots, \tilde{c}_n$, computes $e_i \leftarrow\!\!\$\, E_{\mathsf{pk}}(\tilde{c}_i)$ for $i = 1, \ldots, n$, and outputs $e = (e_1, \ldots, e_n)$ at the outside interface. Additionally, it outputs (encode, m_b) at the inside interface.

When it gets a decryption query (dec, e') with $e' = (e'_1, \ldots, e'_n)$, it proceeds as follows: First, it creates a tamper query f with $\chi(f) = (f_1, \ldots, f_n)$ where

$$f_i = \begin{cases} \mathsf{zero} & \text{if } e'_i \neq e_i \text{ and } D_{\mathsf{sk}_i}(e'_i) = 0, \\ \mathsf{one} & \text{if } e'_i \neq e_i \text{ and } D_{\mathsf{sk}_i}(e'_i) = 1, \text{ and} \\ \mathsf{keep} & \text{if } e'_i = e_i. \end{cases}$$

Then, it outputs (tamper, f) at the inside interface and obtains an answer x'. If $x' = \diamond$, $\mathbf{C}_b$ implements the self-destruct mode. If $x' \in \{m_0, m_1\}$, $\mathbf{C}_b$ outputs test at the outside interface. Otherwise, it outputs x'.

For $b \in \{0,1\}$, consider the systems $\mathbf{C}_b\mathbf{S}_{\mathcal{F}}^{\mathsf{real}}$ and $\mathbf{H}_b^{(n)}$. Both systems generate the public key in the same fashion. Furthermore, in either system, the challenge ciphertext consists of n encryptions of random bits. Finally, both systems answer a decryption query by applying the same tamper function to an encoding of m_b before decoding it. When the decoding of the tampered codeword results in m_0 or m_1, both systems answer test. Thus, $\mathbf{C}_b\mathbf{S}_{\mathcal{F}}^{\mathsf{real}} \equiv \mathbf{H}_b^{(n)}$.

Due to the fact that test is output when a decryption query results in m_0 or m_1, the observable behavior is the same in $\mathbf{C}_0\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}}$ and $\mathbf{C}_1\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}}$.[16] Therefore, $\mathbf{C}_0\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}} \equiv \mathbf{C}_1\mathbf{S}_{\mathcal{F},\tau}^{\mathsf{simu}}$. □

Proof (of Lemma 13). Follows immediately from Lemmas 14 and 15 using a triangle inequality. □

E Achieving Adaptive Continuous Non-Malleability

Theorem 2. *If a coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *is continuously* $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, 1, q)$*-non-malleable, it is also continuously* $(\mathcal{F}_{\mathsf{copy}}, 2\ell\varepsilon + \frac{q\ell}{2^k}, \ell, q)$*-non-malleable, for all* $\ell, q \in \mathbb{N}$.

[16] This is where the proof reflects that Π is only SD-RCCA secure.

Left-or-right non-malleability. The proof of Theorem 2, which uses a hybrid argument, is facilitated by introducing a left-or-right (LOR) variant of non-malleability. The two definitions are equivalent, as shown by Lemmas 16 and 17 below. In the LOR variant,[17] the encode-oracle takes as input pairs of messages and encodes either always the first or always the second message. The goal of the attacker is to find out which is the case. Formally, LOR-non-malleability is defined using the two random systems $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0}$ and $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1}$, shown in Figure 11.[18]

System $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b}$

init
- $i \leftarrow 0$

on $(\mathsf{encode}, x_0, x_1)$
- $i \leftarrow i + 1$
- $x_0^{(i)} \leftarrow\!\!\$\ x_0$
- $x_1^{(i)} \leftarrow\!\!\$\ x_1$
- $c^{(i)} \leftarrow\!\!\$\ \mathrm{Enc}(x_b)$

on (tamper, f) *with* $f \in \mathcal{F}^{(i)}$
- $c' \leftarrow f(c^{(1)}, \ldots, c^{(i)})$
- $x' \leftarrow\!\!\$\ \mathrm{Dec}(c')$
- **if** $x' = \diamond$
  - **self-destruct**
- **if** $\exists j : x' \in \{x_0^{(j)}, x_1^{(j)}\}$
  - $x' \leftarrow (\mathsf{same}, j)$
- **out** x'

Figure 11: *Systems* $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0}$ *and* $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1}$ *defining LOR-non-malleability of* (Enc, Dec). *The* **self-destruct** *command has the effect that* $\diamond$ *is output and all future queries are answered by* $\diamond$.

When processing a tamper query, if there are multiple indices j for which (same, j) could be output, $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b}$ outputs the largest such j. As before, for $b \in \{0,1\}$ and $\ell, q \in \mathbb{N}$, $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b,\ell,q}$ is the system that behaves as $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b}$ except that only the first ℓ encode-queries and the first q tamper-queries are handled.

Definition 7 (Adaptive continuous left-or-right non-malleability)**.** *Let* $\mathcal{F} = (\mathcal{F}^{(i)})_{i\geq 1}$ *be a sequence of function families* $\mathcal{F}^{(i)} \subseteq \{f \mid f : (\{0,1\}^n)^i \to \{0,1\}^n\}$ *and let* $\ell, q \in \mathbb{N}$. *A coding scheme* (Enc, Dec) *is* adaptively continuously $(\mathcal{F}, \varepsilon, \ell, q)$-LOR-non-malleable *(or simply* $(\mathcal{F}, \varepsilon, \ell, q)$-LOR-non-malleable*) if there exists a simulator* τ *such that* $\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0,\ell,q}, \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1,\ell,q}) \leq \varepsilon$ *for all distinguishers* $\mathbf{D}$.

Lemma 16. *If* (Enc, Dec) *is* $(\mathcal{F}, \varepsilon, \ell, q)$*-non-malleable, it is also* $(\mathcal{F}, 2\varepsilon, \ell, q)$*-LOR-non-malleable.*

Proof. Fix ℓ, q, and a simulator τ, and let $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},\ell,q}$, $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,\ell,q}$, $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0} := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0,\ell,q}$, and $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1} := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1,\ell,q}$. For $b \in \{0,1\}$, consider the following reduction $\mathbf{C}_b$: Upon the i^{th} query $(\mathsf{encode}, x_0, x_1)$ at the outside interface, it stores $x_0^{(i)} := x_0$ and $x_1^{(i)} := x_1$ internally and outputs (encode, x_b) at the inside interface. Upon a query (tamper, f) at the outside interface, $\mathbf{C}_b$ outputs (tamper, f) at the inside interface and subsequently receives a value x' at the inside interface. If there exist indices i' such that $x' \in \{x_0^{(i')}, x_1^{(i')}\}$, $\mathbf{C}_b$ outputs (same, i') for the largest such index at the outside interface. Otherwise, it outputs x'.

One observers that

$$\mathbf{C}_0\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} \equiv \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0} \qquad \text{and} \qquad \mathbf{C}_1\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} \equiv \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1} \qquad \text{and} \qquad \mathbf{C}_0\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} \equiv \mathbf{C}_1\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau},$$

where the third equivalence follows from the fact that the observable behavior of $\mathbf{C}_b\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ is independent of the messages $\mathbf{C}_b$ outputs to $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$. Hence, for all attackers $\mathbf{A}$,

$$\begin{aligned}
\Delta^{\mathbf{A}}(\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0}, \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1}) &= \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{C}_1\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}) \\
&\leq \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{C}_0\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) + \Delta^{\mathbf{A}}(\mathbf{C}_0\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}, \mathbf{C}_1\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) + \Delta^{\mathbf{A}}(\mathbf{C}_1\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}, \mathbf{C}_1\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}) \\
&\leq \Delta^{\mathbf{A}\mathbf{C}_0}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) + \Delta^{\mathbf{A}\mathbf{C}_1}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) \\
&\leq 2\varepsilon.
\end{aligned}$$

□

Lemma 17. *If* (Enc, Dec) *is* $(\mathcal{F}, \varepsilon, \ell, q)$*-LOR-non-malleable, it is also* $(\mathcal{F}, \varepsilon + \frac{q\ell}{2^k}, \ell, q)$*-non-malleable.*

[17] One should not confuse the above LOR variant with *strong* non-malleability, the difference being that for strong non-malleability $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b}$ would output (same, j) iff $c' = c^{(j)}$. In fact, being equivalent to non-malleability, our LOR variant is strictly weaker.

[18] The same LOR variant was already considered in [17, Definition A.1] (and referred to as "alternative" non-malleability). In this sense Lemma 16 and 17 below are a generalization of [17, Theorem A.1] to the adaptive and continuous case.

Proof. Fix ℓ and q, and let $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},\ell,q}$, $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,\ell,q}$ (for a simulator τ to be defined next), $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0} := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0,\ell,q}$, and $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1} := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1,\ell,q}$. Consider the following simulator τ: It internally keeps a counter $i \leftarrow 0$. When invoked on (i', f) with $f \in \mathcal{F}^{(i')}$, if $i' > i$, it samples $x_1^{(j)} \leftarrow\!\!\$\ \{0,1\}^k \setminus \{x_1^{(1)}, \ldots, x_1^{(j-1)}\}$ and computes $c_1^{(j)} \leftarrow\!\!\$\ \mathrm{Enc}(x_1^{(j)})$ for all $i < j \le i'$ and sets $i \leftarrow i'$. Then, it computes the tampered codeword $c' \leftarrow \mathrm{Dec}(f(c_1^{(1)}, \ldots, c_1^{(i)}))$ and decodes it to $x' \leftarrow \mathrm{Dec}(c')$. If $x' = x_1^{(j)}$ for some indices j, τ returns (same, j) for the largest such j. Otherwise, it returns x'.

Consider the following reduction $\mathbf{C}$: Upon the i^{th} query (encode, x) at the outside interface, it chooses $x_1^{(i)} \leftarrow\!\!\$\ \{0,1\}^k \setminus \{x_1^{(1)}, \ldots, x_1^{(i-1)}\}$, stores $x_0^{(i)} := x$ internally, and outputs $(\mathsf{encode}, x_0^{(i)}, x_1^{(i)})$ at the inside interface. Upon a query (tamper, f) at the outside interface, $\mathbf{C}$ outputs (tamper, f) at the inside interface and subsequently receives a value x' at the inside interface. If $x' = (\mathsf{same}, j)$ for some j, $\mathbf{C}$ outputs $x_0^{(j)}$ at the outside interface. Otherwise, it outputs x'.

Observe that $\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},1} \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$. In both cases, the i^{th} query of the type (encode, x) is treated by sampling fresh values $x_1^{(i)}$ distinct from all $x_1^{(1)}, \ldots, x_1^{(i-1)}$ and computing $c_1^{(i)}$ as an encoding of $x_1^{(i)}$. (This is delayed in $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$, but that does not change the distribution.) A query (tamper, f) with some function $f \in \mathcal{F}^{(i)}$ is answered by evaluating $f(c_1^{(1)}, \ldots, c_1^{(i)})$, decoding the resulting codeword to obtain a message x', and if $x' = x_1^{(j)}$ for some $j \in \{1, \ldots, i\}$, returning $x_0^{(j)}$ and x' otherwise.

The systems $\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}$ and $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ are, however, not equivalent. The reason is that if, in $\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}$, $\mathrm{Dec}(f(c_0^{(1)}, \ldots, c_0^{(i)})) = x_1^{(j)}$ for some $j \in \{1, \ldots, i\}$, then $\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0}$ returns (same, j), which $\mathbf{C}$ replaces by $x_0^{(j)}$. There is no comparable behavior in $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$. Provoking this event, however, corresponds to "non-adaptively guessing" one of the values $x_1^{(j)}$, which occurs with probability at most $\frac{i}{2^k}$ in each query.

Formally, one can define a monotone binary output (MBO, see Section 2.1) on $\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}$; $\widehat{\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}}$ (the system extended by this additional output) and $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ are now conditionally equivalent, and by [38, Theorem 1], the distinguishing advantage $\Delta^{\mathbf{A}}(\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}, \mathbf{S}^{\mathsf{real}}_{\mathcal{F}})$ is upper-bounded by the probability of provoking this event, which for at most ℓ encode- and at most q tamper-queries can be bounded by $\frac{q\ell}{2^k}$.

Hence, for all attackers $\mathbf{A}$,

$$\begin{aligned}
\Delta^{\mathbf{A}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) &= \Delta^{\mathbf{A}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},1}) \\
&\le \Delta^{\mathbf{A}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}) + \Delta^{\mathbf{A}}(\mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},0}, \mathbf{CS}^{\mathsf{lor}}_{\mathcal{F},1}) \\
&\le \tfrac{q\ell}{2^k} + \Delta^{\mathbf{AC}}(\mathbf{S}^{\mathsf{lor}}_{\mathcal{F},0}, \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},1}) \\
&\le \tfrac{q\ell}{2^k} + \varepsilon.
\end{aligned}$$

□

Lemma 18. *If* $(\mathrm{Enc}, \mathrm{Dec})$ *is continuously* $(\mathcal{F}_{\mathsf{copy}}, \varepsilon, 1, q)$*-LOR-non-malleable, it is also continuously* $(\mathcal{F}_{\mathsf{copy}}, \ell \cdot \varepsilon, \ell, q)$*-LOR-non-malleable, for all* $\ell \in \mathbb{N}$.

Proof. Fix ℓ and q, let $\mathcal{F} := \mathcal{F}_{\mathsf{copy}}$, and set $\mathbf{S}'_b := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b,\ell,q}$ and $\mathbf{S}_b := \mathbf{S}^{\mathsf{lor}}_{\mathcal{F},b,1,q}$ for $b \in \{0,1\}$.

The distinguishing advantage between $\mathbf{S}'_0$ and $\mathbf{S}'_1$ is bounded via a hybrid argument, where the i^{th} hybrid $\mathbf{H}^{(i)}$ picks x_0 when processing the first i encode queries $(\mathsf{encode}, x_0, x_1)$ and x_1 afterwards. For each i, the distinguishing advantage between successive hybrids $\mathbf{H}^{(i-1)}$ and $\mathbf{H}^{(i)}$ is bounded by exhibiting a system $\mathbf{C}_i$ that reduces distinguishing $\mathbf{S}_0$ and $\mathbf{S}_1$ to distinguishing the hybrids.

For $i = 0, 1, \ldots, \ell$, hybrid $\mathbf{H}^{(i)}$ works as follows: Initialization and (tamper, f) are defined as with $\mathbf{S}'_0$ and $\mathbf{S}'_1$. The first i queries $(\mathsf{encode}, x_0, x_1)$ are handled by encoding x_0, i.e., $c^{(j)} \leftarrow \mathrm{Enc}(x_0)$ for the j^{th} encoding. For all later queries, x_1 is encoded, i.e., $c^{(j)} \leftarrow \mathrm{Enc}(x_1)$.

One observes that

$$\mathbf{H}^{(\ell)} \equiv \mathbf{S}'_0 \qquad \text{and} \qquad \mathbf{H}^{(0)} \equiv \mathbf{S}'_1.$$

For $i = 1, \ldots, n$, reduction $\mathbf{C}_i$ works as follows: For the first $i-1$ encode queries $(\mathsf{encode}, x_0, x_1)$ (at the outside interface), it computes and stores an encoding of x_0, i.e., $c^{(j)} \leftarrow \mathrm{Enc}(x_0)$ for the

j^{th} encoding. Upon the i^{th} query $(\mathsf{encode}, x_0, x_1)$, it outputs $(\mathsf{encode}, x_0, x_1)$ at the inside interface. (Note that as a consequence, a target encoding $c \leftarrow\!\!\$\, \mathrm{Enc}(x_b)$ is generated, depending on whether $\mathbf{C}_i$ is connected to $\mathbf{S}_0$ or $\mathbf{S}_1$.) The remaining encode queries are handled by encoding the second message x_1, i.e., $c^{(j)} \leftarrow \mathrm{Enc}(x_1)$.

System $\mathbf{C}_i$ maintains a counter j that keeps track of the number of encode queries it has encountered. When a tamper query (tamper, f) with $f \in \mathcal{F}^{(j)}_{\mathsf{copy}}$ and $\chi(f) = (f_1, \ldots, f_n)$ is received at the outside interface, it computes $f'_1, \ldots, f'_n$, where

$$f'_v := \begin{cases} f_v & \text{if } f_v \in \{\mathsf{zero}, \mathsf{one}\}, \\ \mathsf{zero} & \text{if } f_v = \mathsf{copy}_w \text{ for } w \neq i, \text{ and } c_v^{(w)} = 0, \\ \mathsf{one} & \text{if } f_v = \mathsf{copy}_w \text{ for } w \neq i, \text{ and } c_v^{(w)} = 1, \\ \mathsf{copy}_1 & \text{if } f_v = \mathsf{copy}_i. \end{cases}$$

Then, it outputs (tamper, f') at the inside interface, where f' is the function in $\mathcal{F}^{(1)}_{\mathsf{copy}}$ with $\chi(f)' = (f'_1, \ldots, f'_n)$.[19] Let x' be the answer to the tamper query at the inside interface. $\mathbf{C}_i$ computes the set of indices j for which x' matches one of the two messages of the j^{th} encode query. Moreover, if $x' = \mathsf{same}$, index i is added to that set as well. Then, it outputs (same, j) for the largest index j in the set. If the set is empty, x' is output.

One observes that

$$\mathbf{C}_i\mathbf{S}_0 = \mathbf{H}^{(i)} \qquad \text{and} \qquad \mathbf{C}_i\mathbf{S}_1 = \mathbf{H}^{(i-1)}.$$

Thus, for all adversaries $\mathbf{A}$,

$$\begin{aligned} \Delta^{\mathbf{A}}(\mathbf{S}'_0, \mathbf{S}'_1) = \Delta^{\mathbf{A}}(\mathbf{H}^{(\ell)}, \mathbf{H}^{(0)}) &\leq \sum_{i=1}^{\ell} \Delta^{\mathbf{A}}(\mathbf{H}^{(i)}, \mathbf{H}^{(i-1)}) \\ &\leq \sum_{i=1}^{\ell} \Delta^{\mathbf{A}}(\mathbf{C}_i\mathbf{S}_0, \mathbf{C}_i\mathbf{S}_1) \leq \sum_{i=1}^{\ell} \Delta^{\mathbf{AC}_i}(\mathbf{S}_0, \mathbf{S}_1) \leq \ell \cdot \varepsilon. \end{aligned}$$

□

Proof (of Theorem 2). Follows immediately from Lemmas 16, 17, and 18. □

F On the Necessity of Self-Destruct

In this section we show that no (k, n)-coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ can achieve (even non-adaptive, i.e. for $\ell = 1$) continuous non-malleability against $\mathcal{F}_{\mathsf{copy}}$ without self-destruct. This fact is reminiscent of the negative result by Gennaro *et al.* [23], and was already observed by Faust *et al.* [19] (without a proof) for the easier case of *strong* continuous non-malleability. The impossibility proof in this section assumes that Dec is deterministic and that $\mathrm{Dec}(\mathrm{Enc}(x)) = x$ with probability 1 for all $x \in \{0,1\}^k$ (cf. Definition 2). The distinguisher $\mathbf{D}$ provided by Theorem 19 is universal, i.e., it breaks any coding scheme (if given oracle access to its decoding algorithm).

For the remainder of this section, let $\mathcal{F} := \mathcal{F}_{\mathsf{set}}$ (as defined in Section 4), $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},1,n}$, and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,1,n}$ (with some simulator τ). Moreover, both $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ are stripped of the self-destruct mode.

Theorem 19. *There exists a distinguisher* $\mathbf{D}$ *such that for all coding schemes* $(\mathrm{Enc}, \mathrm{Dec})$ *and all simulators* τ*,*

$$\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) \geq 1 - \frac{n+1}{2^k}.$$

[19] For simplicity, we assume here that $\mathbf{S}_0$ and $\mathbf{S}_1$ answer tamper queries consisting of zero and one instructions only even before a message has been encoded.

The corollary below states no pair of converters $(\mathsf{encode}, \mathsf{decode})$ can achieve the constructive statement corresponding to Theorem 1 without relying on the self-destruct feature.

Corollary 20. *For any protocol* $\mathsf{nmc} := (\mathsf{encode}, \mathsf{decode})$ *and all simulators* σ*, if both converters are stateless and*

$$\left[\text{—}\diamond\twoheadrightarrow\bullet\right]^{n} \quad \overset{((\mathsf{encode},\mathsf{decode}),\sigma,(0,\varepsilon))}{\Longmapsto} \quad \text{—}\diamond\twoheadrightarrow\bullet,$$

then,

$$\varepsilon \geq 1 - \frac{n+1}{2^k}.$$

Proof. Note that the protocol achieves perfect availability and thus constitutes a perfectly correct (k, n)-coding scheme (since the converters are stateless and with perfect correctness, decode can w.l.o.g. be assumed to be deterministic). Consider an arbitrary simulator σ. It can be converted into a simulator τ as required by Definition 3 in a straight-forward manner. Similarly, there exists a straight-forward reduction $\mathbf{C}$ such that

$$\mathbf{C}(\mathsf{encode}^A \mathsf{decode}^B [\text{—}\diamond\twoheadrightarrow\bullet^{\text{1-bit},1,n}]^n) \equiv \mathbf{S}^{\mathsf{real}}_{\mathcal{F}} \qquad \text{and} \qquad \mathbf{C}(\sigma^E \text{—}\diamond\twoheadrightarrow\bullet^{k\text{-bit},1,n}) \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}.$$

Thus, $\mathbf{DC}$ achieves advantage $1 - \frac{n+1}{2^k}$. □

F.1 Proof of Theorem 19

Distinguisher $\mathbf{D} := \mathbf{D}_{\mathsf{Ext}}$ uses an algorithm Ext that always extracts the encoded message when interacting with system $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and does so with small probability only when interacting with system $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ (for any simulator).

The Extraction Algorithm. Consider the following algorithm Ext, which repeatedly issues tamper queries (tamper, f) with $f \in \mathcal{F}_{\mathsf{set}}$, expects an answer in $\{0,1\}^k \cup \{\diamond, \mathsf{same}\}$, and eventually outputs a value $x' \in \{0,1\}^k$: Initially, it initializes variables $f_1, \ldots, f_n \leftarrow \varnothing$ (where the value $\varnothing$ stands for "undefined"). Then, for $i = 1, \ldots, n$ it proceeds as follows: It queries (tamper, f) with $\chi(f) = (f_1, \ldots, f_{i-1}, \mathsf{zero}, \mathsf{keep}, \ldots, \mathsf{keep})$. If the answer is same, it sets $f_i \leftarrow \mathsf{zero}$ and otherwise $f_i \leftarrow \mathsf{one}$. In the end Ext outputs $x' \leftarrow \mathrm{Dec}(\mathsf{val}(f_1) \cdots \mathsf{val}(f_n))$.

The Distinguisher. Consider the following distinguisher $\mathbf{D}_{\mathsf{Ext}}$: Initially, it chooses $x \leftarrow \{0,1\}^k$ and outputs (encode, x) to the system it is connected to. Then, it lets Ext interact with that system, replacing an answer by same whenever it is x. When Ext terminates and outputs a value x', $\mathbf{D}_{\mathsf{Ext}}$ outputs 1 if $x' = x$ and 0 otherwise.

Lemma 21. $\mathsf{P}[\mathbf{D}_{\mathsf{Ext}} \mathbf{S}^{\mathsf{real}}_{\mathcal{F}} = 1] = 1$.

Proof. Assume that before the i^{th} iteration of Ext, asking the query (tamper, f) with $\chi(f) = (f_1, \ldots, f_{i-1}, \mathsf{keep}, \mathsf{keep}, \ldots, \mathsf{keep})$ to $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ yields the answer x. From this it follows that either $(f_1, \ldots, f_{i-1}, \mathsf{zero}, \mathsf{keep}, \ldots, \mathsf{keep})$ or $(f_1, \ldots, f_{i-1}, \mathsf{one}, \mathsf{keep}, \ldots, \mathsf{keep})$ leads to the answer x; Ext sets f_i appropriately (the fact that the answer x is replaced by same plays no role here). Thus, in the end, computing $\mathrm{Dec}(\mathsf{val}(f_1) \cdots \mathsf{val}(f_n))$ yields x. □

In other words, Lemma 21 means that Ext always succeeds at recovering the value x chosen by $\mathbf{D}$. Showing that this happens only with small probability when $\mathbf{D}_{\mathsf{Ext}}$ interacts with $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ completes the proof.

Lemma 22. $\mathsf{P}[\mathbf{D}_{\mathsf{Ext}} \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} = 1] \leq \frac{n+1}{2^k}$.

Proof. Consider the following modified distinguisher $\hat{\mathbf{D}}_{\mathsf{Ext}}$ that works as $\mathbf{D}_{\mathsf{Ext}}$ except that it does *not* modify the answers received by the system it is connected to. Moreover, let $\hat{\mathbf{S}}^{\mathsf{simu}}_{\mathcal{F},\tau}$ be the the system that ignores all encode-queries and handles queries (tamper, f) by invoking $\tau(1, f)$ and outputting τ's answer.

Note that in both experiments, Ext's view is identical unless it causes τ to output x (the value encoded by $\mathbf{D}$), which happens with probability at most $\frac{n}{2^k}$. Thus,

$$|\mathsf{P}^{\mathbf{D}_{\mathsf{Ext}}\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}}[\mathsf{Ext}\text{ outputs }x] - \mathsf{P}^{\hat{\mathbf{D}}_{\mathsf{Ext}}\hat{\mathbf{S}}^{\mathsf{simu}}_{\mathcal{F},\tau}}[\mathsf{Ext}\text{ outputs }x]| \leq \frac{n}{2^k}.$$

Furthermore, in experiment $\hat{\mathbf{D}}_{\mathsf{Ext}}\hat{\mathbf{S}}^{\mathsf{simu}}_{\mathcal{F},\tau}$, Ext's view is independent of x, and therefore, x is output by Ext with probability $\frac{1}{2^k}$. The claim follows. □

G Continuous Non-Malleability against Full Bit-Wise Tampering

In this section we show that the coding scheme by [17] is continuously non-malleable against $\mathcal{F}_{\mathsf{copy}}$ extended with bit flips. The scheme relies on a LECSS (E, D) (cf. Definition 4 in Section 4) and a so-called AMD code (A, V); the latter concept was introduced by [10].

Definition 8 (AMD code)**.** *A (k, n)-coding scheme (A, V) is a ρ-secure algebraic manipulation detection (AMD) code if for all $x \in \{0,1\}^n$ and non-zero $\Delta \in \{0,1\}^n$, $\mathsf{P}[\mathsf{V}(\mathsf{A}(x) + \Delta) \neq \diamond] \leq \rho$.*

The scheme (Enc, Dec) by [17] is the concatenation of an AMD code and a LECSS, i.e., $\mathsf{Enc} := \mathsf{E} \circ \mathsf{A}$ and $\mathsf{Dec} := \mathsf{V} \circ \mathsf{D}$, where $\mathsf{V}(\diamond) = \diamond$.

The tampering class $\mathcal{F}_{\mathsf{copy}}$ can be extended to account for bit flips: Let $\mathcal{F}'_{\mathsf{copy}} := (\mathcal{F}'^{(i)}_{\mathsf{copy}})_{i \geq 1}$ where $\mathcal{F}'^{(i)}_{\mathsf{copy}} \subseteq \{f \mid f : (\{0,1\}^n)^i \to \{0,1\}^n\}$ and each function $f \in \mathcal{F}'^{(i)}_{\mathsf{copy}}$ is characterized by a vector $\chi(f) = (f_1, \ldots, f_n)$ where $f_i \in \{\mathsf{zero}, \mathsf{one}, \mathsf{copy}_1, \ldots, \mathsf{copy}_i, \mathsf{flip}_1, \ldots, \mathsf{flip}_i\}$, with the meaning that f takes as input i codewords $(c^{(1)}, \ldots, c^{(i)})$ and outputs a codeword $c' = c'_1 \cdots c'_n$ in which each bit is either set to 0 (zero), set to 1 (one), copied from the *corresponding* bit in a codeword $c^{(j)}$ (copy_j), or copied and flipped from the corresponding bit in a codeword $c^{(j)}$ (flip_j).

Theorem 23. *Let* (Enc, Dec) *as defined above with a (t, d)-LECSS (k, n)-code for $d > n/4$ and $d > t$ and a ρ-secure AMD code. Then* (Enc, Dec) *is $(\mathcal{F}'_{\mathsf{copy}}, \varepsilon, 1, q)$-continuously non-malleable for all $q \in \mathbb{N}$ and*

$$\varepsilon = 2^{-(t-1)} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2} + \rho.$$

For brevity, we write $\mathcal{F}_{\mathsf{bit}}$ for $\mathcal{F}'^{(1)}_{\mathsf{copy}}$ below, with the idea that the tampering functions in $\mathcal{F}'^{(1)}_{\mathsf{copy}}$ only allow to keep or flip a bit or to set it to 0 or to 1. More formally, a function $f \in \mathcal{F}_{\mathsf{bit}}$ can be characterized by a vector $\chi(f) = (f_1, \ldots, f_n)$ where $f_i \in \{\mathsf{zero}, \mathsf{one}, \mathsf{keep}, \mathsf{flip}\}$, with the meaning that f takes as input a codeword c and outputs a codeword $c' = c'_1 \cdots c'_n$ in which each bit is either set to 0 (zero), set to 1 (one), left unchanged (keep), or flipped (flip).

For the proof of Theorem 23, fix $q \in \mathbb{N}$ and some distinguisher $\mathbf{D}$. For the remainder of this section, let $\mathcal{F} := \mathcal{F}_{\mathsf{bit}}$, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}} := \mathbf{S}^{\mathsf{real}}_{\mathcal{F},1,q}$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau} := \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau,1,q}$ (for a simulator τ to be determined). For a tamper query $f \in \mathcal{F}$ with $\chi(f) = (f_1, \ldots, f_n)$ issued by $\mathbf{D}$, let $A(f) := \{i \mid f_i \in \{\mathsf{zero}, \mathsf{one}\}\}$, $B(f) := \{i \mid f_i \in \{\mathsf{keep}, \mathsf{flip}\}\}$, and $a(f) := |A(f)|$. Moreover, let $\mathsf{val}(\mathsf{zero}) := \mathsf{val}(\mathsf{keep}) := 0$ and $\mathsf{val}(\mathsf{one}) := \mathsf{val}(\mathsf{flip}) := 1$. Queries f with $0 \leq a(f) \leq t$, $t < a(f) < n - t$, and $n - t \leq a(f) \leq n$ are called *low queries*, *middle queries*, and *high queries*, respectively.

Dangerous queries. A tamper query is *dangerous* if it is

- a middle query or
- a low query such that there exists a codeword δ^* of the LECSS with $\forall i \in B(f) : \delta^*_i = \mathsf{val}(f_i)$ and $\mathsf{D}(\delta^*) \neq 0$.

Consider the hybrid system $\mathbf{H}$ that proceeds as $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$, except that as soon as $\mathbf{D}$ specifies a dangerous query f, $\mathbf{H}$ self-destructs, i.e., answers f and all subsequent queries with $\diamond$.

Lemma 24. $\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) \leq \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2} + \rho$.

Proof. Define a *successful* dangerous query to be a dangerous query that does not decode to $\diamond$. On both systems $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{H}$, one can define an MBO $\mathcal{B}$ (cf. Section 2.1) that is provoked if and only if the *first* dangerous query is successful and the self-destruct has not been provoked up to that point.

Clearly, $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$ and $\mathbf{H}$ behave identically until MBO $\mathcal{B}$ is provoked, thus $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}} \stackrel{g}{\equiv} \hat{\mathbf{H}}$, and

$$\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) \leq \Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}).$$

Towards bounding $\Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}})$, note first that adaptivity does not help in provoking $\mathcal{B}$: For any distinguisher $\mathbf{D}$, there exists a *non-adaptive* distinguisher $\mathbf{D}'$ with

$$\Gamma^{\mathbf{D}}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}) \leq \Gamma^{\mathbf{D}'}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}). \tag{8}$$

$\mathbf{D}'$ proceeds as follows: First, it (internally) interacts with $\mathbf{D}$ only. Initially, it stores the message x output by $\mathbf{D}$ internally. Then, it handles the tamper queries f by $\mathbf{D}$ as follows:

- *Low query:* If there exists a codeword δ^* of the LECSS with $\forall i \in B(f) : \delta^*_i = \mathsf{val}(f_i)$ and $\mathsf{D}(\delta^*) = 0$, $\mathbf{D}'$ answers with x. Otherwise, $\mathbf{D}'$ stops its interaction with $\mathbf{D}$ and sends x and all the queries to $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}$.
- *Middle query:* $\mathbf{D}'$ stops its interaction with $\mathbf{D}$ and sends x and all the queries to $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}$.
- *High query:* If there exists a codeword c^* that agrees with f in positions i where $f_i \in \{\mathsf{zero}, \mathsf{one}\}$, $\mathbf{D}'$ answers with $\mathrm{Dec}(c^*)$. Otherwise, $\mathbf{D}'$ stops its interaction with $\mathbf{D}$ and sends x and all the queries to $\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}$.

To prove (8), fix all randomness in experiment $\mathbf{D}'\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$, i.e., the coins of $\mathbf{D}$ (inside $\mathbf{D}'$) and the randomness of the encoding (inside $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$). Suppose $\mathbf{D}$ would provoke $\mathcal{B}$ in the direct interaction with $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$. In that case all the answers by $\mathbf{D}'$ are equal to the answers by $\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}$. This is due to the fact that the distance of the LECSS is $d > t$; a successful non-dangerous low query must result in the original message x and a successful high query in $\mathrm{Dec}(c^*)$. Thus, whenever $\mathbf{D}$ provokes $\mathcal{B}$, $\mathbf{D}'$ provokes it as well.

It remains to analyze the success probability of non-adaptive distinguishers $\mathbf{D}'$. Fix the coins of $\mathbf{D}'$; this determines the tamper queries. Suppose there is at least one dangerous query, as otherwise $\mathcal{B}$ is trivially not provoked. The query's success probability can be analyzed as in [17], depending on whether it is a low or a high query, which leads to $\Gamma^{\mathbf{D}'}(\hat{\mathbf{S}}^{\mathsf{real}}_{\mathcal{F}}) \leq \frac{1}{2^t} + \left(\frac{t}{n(d/n-1/4)^2}\right)^{t/2} + \rho$ (recall that the MBO cannot be provoked after an unsuccessful first dangerous query). □

Simulator. The final step of the proof consists of exhibiting a simulator τ such that $\Delta^{\mathbf{D}}(\mathbf{H}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau})$ is small. The indistinguishability proof is facilitated by reusing the two (hardly distinguishable) systems $\mathbf{B}$ and $\mathbf{B}'$ from Section 4 and the wrapper system $\mathbf{W}$ defined in Figure 12, such that $\mathbf{WB} \equiv \mathbf{H}$ and $\mathbf{WB}' \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$. System $\mathbf{W}$ has an outside interface $\mathtt{o}$ and an inside interface $\mathtt{i}$; at the latter interface, $\mathbf{W}$ expects to be connected to either $\mathbf{B}$ or $\mathbf{B}'$.

Lemma 25. $\mathbf{WB} \equiv \mathbf{H}$.

Proof. Fix a message x. Consider a low query $f = (f_1, \ldots, f_n)$. Let $c = \mathsf{E}(\mathsf{A}(x))$ be an encoding of x, set $c' := f(c)$, and let $\delta' := c + c'$. Using the linearity of the LECSS,

$$\mathsf{D}(c') = \mathsf{D}(\mathsf{E}(\mathsf{A}(x)) + \delta') = \mathsf{A}(x) + \mathsf{D}(\delta').$$

System **W**

init
- $\forall i \in [n] : c_i \leftarrow \varnothing$

on first (encode, x) *at* $\mathtt{o}$
- **output** x at $\mathtt{i}$

on (tamper, f) *with* $0 \le a(f) \le t$ *at* $\mathtt{o}$
- **for** i *where* $f_i \in B(f)$
  - $\delta'_i \leftarrow \mathsf{val}(f_i)$
- **if** $\exists$*codeword* δ^*: $\forall i \in B(f) : \delta'_i = \delta^*_i$
  - **for** i *where* $f_i \in A(f)$
    - $g \leftarrow \mathsf{val}(f_i) \oplus \delta^*_i$
    - **if** $c_i = \varnothing$
      - **output** (i, g) at $\mathtt{i}$
      - **get** $a \in \{\diamond, 1\}$ at $\mathtt{i}$
      - **if** $a = \diamond$
        - **self-destruct**
      - $c_i \leftarrow g$
    - **else**
      - **if** $c_i \neq g$
        - **self-destruct**
  - **if** $\mathsf{D}(\delta^*) \neq 0$
    - **self-destruct**
  - **else**
    - **output** x at $\mathtt{o}$
- **else**
  - **self-destruct**

on (tamper, f) *with* $t < a(f) < n - t$ *at* $\mathtt{o}$
- **self-destruct**

on (tamper, f) *with* $n - t \le a(f) \le n$ *at* $\mathtt{o}$
- **for** i *where* $f_i \in A(f)$
  - $c'_i \leftarrow \mathsf{val}(f_i)$
- **if** $\exists$*codeword* $c^* : \forall i \in A(f) : c'_i = c^*_i$
  - **for** i *where* $f_i \in B(f)$
    - $g \leftarrow c^*_i \oplus \mathsf{val}(f_i)$
    - **if** $c_i = \varnothing$
      - **output** (i, g) at $\mathtt{i}$
      - **get** $a \in \{\diamond, 1\}$ at $\mathtt{i}$
      - **if** $a = \diamond$
        - **self-destruct**
      - $c_i \leftarrow g$
    - **else**
      - **if** $c_i \neq g$
        - **self-destruct**
  - **if** $\mathrm{Dec}(c^*) = \diamond$
    - **self-destruct**
  - **else**
    - **output** $\mathrm{Dec}(c^*)$ at $\mathtt{o}$
- **else**
  - **self-destruct**

Figure 12: *The wrapper system* **W**. *The command* **self-destruct** *causes* **W** *to output* $\diamond$ *at* $\mathtt{o}$ *and to answer all future queries by* $\diamond$.

Therefore, **H** answers tamper query f by x if $\mathsf{D}(\delta') = 0$ and by $\diamond$ otherwise. In order for δ' to be equal to some codeword δ^* of the LECSS, it is necessary that $\mathsf{val}(f_i) = \delta^*_i$ for all $i \in B(f)$ and that

$$c_i + \underbrace{c'_i}_{\mathsf{val}(f_i)} = \delta^*_i$$

for all $i \in A(f)$. Note that δ^*, if existent, is unique due to the fact that f is a low query and that the distance of the LECSS is $d > t$.

Similarly, for a high query f, there can be at most one codeword that matches the injected positions. If such a codeword c^* exists, the outcome is $\mathrm{Dec}(c^*)$ if the bits in the keep-positions match c^*, and otherwise $\diamond$.

By inspection, it can be seen that **W** acts accordingly. □

Consider now the system $\mathbf{WB}'$. Due to the nature of $\mathbf{B}'$, the behavior of $\mathbf{WB}'$ is independent of the value x that is initially encoded. This allows to easily design a simulator τ as required by Definition 3. The description of τ is given in Figure 13.

Lemma 26. *The simulator* τ *of Figure 13 satisfies* $\mathbf{WB}' \equiv \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$.

Proof. Consider the systems $\mathbf{WB}'$ and $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$. Both internally choose a vector of n uniform and independent bits $c = c_1 \cdots c_n$. Set $c' := f(c)$, and let $\delta' := c + c'$. System $\mathbf{WB}'$ answers low queries with the value x initially encoded if and only if $\mathsf{D}(\delta') = 0$ and with $\diamond$ otherwise. Simulator τ returns same in the former case, which $\mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}$ replaces by x, and $\diamond$ in the latter case.

Simulator τ

init
- $\forall i \in [n] : c_i \leftarrow\!\!\$\ \{0,1\}$

on (tamper, f) *with* $0 \le a(f) \le t$
- **for** i *where* $f_i \in A(f)$
  - $\delta'_i \leftarrow \mathsf{val}(f_i) \oplus c_i$
- **for** i *where* $f_i \in B(f)$
  - $\delta'_i \leftarrow \mathsf{val}(f_i)$
- $\delta' \leftarrow \delta'_1 \cdots \delta'_n$
- **if** $\mathsf{D}(\delta') \neq 0$
  - **return** $\diamond$
- **else**
  - **return** same

on (tamper, f) *with* $t < a(f) < n - t$
- **return** $\diamond$

on (tamper, f) *with* $n - t \le a(f) \le n$
- **for** i *where* $f_i \in A(f)$
  - $c'_i \leftarrow \mathsf{val}(f_i)$
- **for** i *where* $f_i \in B(f)$
  - $c'_i \leftarrow c_i \oplus \mathsf{val}(f_i)$
- $c' \leftarrow c'_1 \cdots c'_n$
- **return** $\mathrm{Dec}(c')$

Figure 13: *Simulator τ.*

Observe that the answer by $\mathbf{WB}'$ to a high query f always matches $\mathrm{Dec}(c'_1 \cdots c'_n)$, where for $i \in A(f)$, $c'_i = \mathsf{val}(f_i)$, and for $i \in B(f)$, $c'_i = c_i \oplus \mathsf{val}(f_i)$: If no codeword c^* matching the injected positions exists, then $\mathrm{Dec}(c'_1 \cdots c'_n) = \diamond$, which is also what $\mathbf{WB}'$ outputs. If such c^* exists and $c^*_i = c_i \oplus \mathsf{val}(f_i)$ for all $i \in B(f)$, the output of $\mathbf{WB}'$ is $\mathrm{Dec}(c'_1 \cdots c'_n)$. If there exists an $i \in B(f)$ with $c^*_i \neq c_i \oplus \mathsf{val}(f_i)$, $\mathbf{WB}'$ outputs $\diamond$, and in this case $\mathrm{Dec}(c'_1 \cdots c'_n) = \diamond$ since the distance of the LECSS is $d > t$. □

The proof of Theorem 23 now follows from a simple triangle inequality.

Proof (of Theorem 23). From Lemmas 24, 5, 25, and 26, one obtains that for all distinguishers $\mathbf{D}$,

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau}) &\le \Delta^{\mathbf{D}}(\mathbf{S}^{\mathsf{real}}_{\mathcal{F}}, \mathbf{H}) + \underbrace{\Delta^{\mathbf{D}}(\mathbf{H}, \mathbf{WB})}_{=0} + \underbrace{\Delta^{\mathbf{D}}(\mathbf{WB}, \mathbf{WB}')}_{=\Delta^{\mathbf{DW}}(\mathbf{B},\mathbf{B}')} + \underbrace{\Delta^{\mathbf{D}}(\mathbf{WB}', \mathbf{S}^{\mathsf{simu}}_{\mathcal{F},\tau})}_{=0} \\ &\le 2^{-t} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2} + \rho + 2^{-t} \\ &\le 2^{-(t-1)} + \left(\frac{t}{n(d/n - 1/4)^2}\right)^{t/2} + \rho.\end{aligned}$$

□

Lemma 27. *If* $(\mathrm{Enc}, \mathrm{Dec})$ *is continuously* $(\mathcal{F}'_{\mathsf{copy}}, \varepsilon, 1, q)$*-LOR-non-malleable, it is also continuously* $(\mathcal{F}'_{\mathsf{copy}}, \ell \cdot \varepsilon, \ell, q)$*-LOR-non-malleable, for all* $\ell \in \mathbb{N}$*.*

Proof. The proof is analogous to the proof of Lemma 18, except that the reduction system $\mathbf{C}_i$ computes f'_v as follows:

$$f'_v := \begin{cases} f_v & \text{if } f_v \in \{\mathsf{zero}, \mathsf{one}\}, \\ \mathsf{zero} & \text{if } f_v = \mathsf{copy}_w \text{ for } w \neq i, \text{ and } c^{(w)}[v] = 0, \\ \mathsf{one} & \text{if } f_v = \mathsf{copy}_w \text{ for } w \neq i, \text{ and } c^{(w)}[v] = 1, \\ \mathsf{copy}_1 & \text{if } f_v = \mathsf{copy}_i, \\ \mathsf{one} & \text{if } f_v = \mathsf{flip}_w \text{ for } w \neq i, \text{ and } c^{(w)}[v] = 0, \\ \mathsf{zero} & \text{if } f_v = \mathsf{flip}_w \text{ for } w \neq i, \text{ and } c^{(w)}[v] = 1, \\ \mathsf{flip}_1 & \text{if } f_v = \mathsf{flip}_i. \end{cases}$$

□