Alternatives to Non-Malleability: Definitions, Constructions and Applications

Philip MacKenzie* Michael K. Reiter† Ke Yang‡

January 2, 2004

Abstract

We explore whether non-malleability is necessary for the applications typically used to motivate it, and propose two alternatives. The first we call weak non-malleability (wnm) and show that it suffices to achieve secure contract bidding (the application for which non-malleability was initially introduced), despite being strictly weaker than non-malleability. The second we call tag-based non-malleability (tnm), and show that it suffices to construct an efficient universally-composable secure message transmission (SMT) protocol, for which the only previous solution was based on a public key encryption functionality whose security is equivalent to non-malleability. We also demonstrate constructions for wnm and tnm encryption schemes that are simpler than known constructions of non-malleable encryption schemes.

1 Introduction

Non-malleability [11] is a security condition for encryption schemes that requires, informally, that an attacker given a challenge ciphertext be unable to produce another, different ciphertext so that the plaintexts underlying the two ciphertexts are "meaningfully related" to each other. Non-malleability is the strongest commonly considered notion of security for encryption, being strictly stronger than indistinguishability [14] under chosen-plaintext or indifferent chosen-ciphertext ("lunchtime") attacks, and being equivalent to indistinguishability under adaptive chosen-ciphertext attacks [1].

In this paper we revisit the definition of non-malleability with an eye toward whether it is *necessary* for applications commonly used to motivate it. Our contributions in this study are twofold. First, we identify alternatives to non-malleability that suffice for applications where previously non-malleability seemed warranted. Second, we identify encryption schemes that implement these variants and that are conceptually simpler than known non-malleable schemes.

The alternative definitions that we propose deviate from non-malleability in different ways. The first notion, which we call *weak non-malleability* (wnm), identifies a point in the space of definitions strictly between non-malleability and indistinguishability (in those cases where there is room between them, i.e., under chosen-plaintext and lunchtime attacks). Informally, wnm allows mauling of a ciphertext c, but such that this mauling does not benefit the adversary. In particular, a mauling that produces a valid ciphertext c' would imply that the adversary has successfully guessed

*Bell Labs, Lucent Technologies, Murray Hill, NJ, USA; philmac@research.bell-labs.com
†Carnegie Mellon University, Pittsburgh, PA, USA; reiter@cmu.edu
‡Carnegie Mellon University, Pittsburgh, PA, USA; yangke@cs.cmu.edu

the plaintext corresponding to c, and thus for many natural applications, this mauling would not be useful. In other words, in such applications, wnm should suffice in place of non-malleability. As an example, we show that a wnm encryption scheme suffices to implement a secure contract bidding auction in the spirit of that originally used to (informally) motivate non-malleability [11]. Still, wnm does allow an adversary to produce a ciphertext c' that has a (very restricted) dependence of a given ciphertext c, and we can in fact show that wnm is a *strictly* weaker property than non-malleability. In addition, we show that this weaker property may be satisfied by very simple encryption schemes similar to those used in Bellare and Rogaway [2] to achieve the (even less stringent) property of indistinguishability under chosen-plaintext attacks [2].[1] These schemes assume p is a prime, H is a hash function (modeled by a random oracle in our security analyses) with range a group X with group operation "·", and f denotes a trapdoor permutation that constitutes the public key (with the trapdoor being the private key):

Mult-Range scheme The encryption of m is $E(m) = \, <f(r), H(r) \cdot m>$ where r is chosen randomly (per encryption) from the domain of f, the plaintext space is an integer range $[a, b]$ satisfying $0 < a < b < p$, $a > (b-a)^2$ and $p > 2b^2$, and $X = \mathbb{Z}_p^*$ with $\cdot$ being multiplication in $\mathbb{Z}_p^*$.

Mult-Adjacent scheme The encryption of m is $E(m) = \, <f(r), H(r) \cdot (m, m+1)>$ where r is chosen randomly (per encryption) from the domain of f, the plaintext space is $\mathbb{Z}_p^* \setminus \{p-1\}$, and $X = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ with group operation $\cdot$ being component-wise multiplication in $\mathbb{Z}_p^*$, i.e., $(x_0, x_1) \cdot (y_0, y_1) = (x_0 y_0, x_1 y_1)$.

Add-Square scheme The encryption of m is $E(m) = \, <f(r), H(r) \cdot (m, m^2)>$, where the plaintext space is $\mathbb{Z}_p^*$, and $X = \mathbb{Z}_p \times \mathbb{Z}_p$ with group operation $\cdot$ being component-wise addition in $\mathbb{Z}_p$, i.e., $(x_0, x_1) \cdot (y_0, y_1) = (x_0 + y_0, x_1 + y_1)$.

For some intuition behind weak non-malleability, consider the Mult-Range scheme above. Without the range restriction on the plaintext space, this scheme would be completely malleable (similar to the first scheme introduced in [2]). However, simply by restricting the range of plaintexts (as opposed to, e.g., adding an additional hash for verification/redundancy, as is done in [2] to achieve non-malleability) we are able to achieve wnm. Informally, this is because any modification of a ciphertext (v, w) to (v, w') implies a multiplying factor w'/w for which there is only a single plaintext in the range that would be transformed into another plaintext in the range.

The second alternative to non-malleability that we propose is called *tag-based non-malleability* (tnm). Here, we structurally modify encryption and decryption to take an additional public string argument called a *tag*. Informally, tnm dictates that an adversary be unable to create a (ciphertext,tag) pair with plaintext related to that of the challenge ciphertext and with the tag being different from the challenge tag, even though it is able to obtain decryptions of (ciphertext,tag) pairs with any tag different from the challenge tag.[2] We demonstrate the utility of tnm by using

[1] While there exist efficient encryption systems that implement indistinguishability under adaptive chosen-ciphertext attacks (and thus non-malleability under these attacks, e.g., [2, 8]), we are unaware of prior constructions that, like those listed here, so *simply* implement a property strictly stronger than indistinguishability (in this case, weak non-malleability) under chosen-plaintext and lunchtime attacks.

[2] Shoup [21] defines something that looks similar — encryption with labels. However, his security property, namely indistinguishability against adaptive chosen ciphertext attacks, allows the adversary to obtain decryptions of (ciphertext,label) pairs as long as either the ciphertext or label is different than the challenge ciphertext or challenge label. The analog to our definition would require that the label be different. This implies a very different security property. In particular, our tag-based non-malleable schemes would not be secure according to his definition.

it to implement the "secure message transmission functionality" in the universal composability framework of [5], replacing the use of non-malleable encryption there, and arguably providing a more natural implementation. tnm also admits exceedingly simple implementations, e.g.:

Tag-based scheme The encryption of m with tag t is $E(m,t) = <f(r), H(r,t) \cdot m>$ where r is chosen randomly (per encryption) from the domain of f. The plaintext space is $\mathbb{Z}_p^*$, and $X = \mathbb{Z}_p^*$ with $\cdot$ being multiplication in $\mathbb{Z}_p^*$.

We also present a tnm construction that is a (simpler) variation of the Cramer-Shoup encryption scheme [8, 9]. The change in structure for encryption and decryption (specifically due to the tag) does not permit us to argue that tnm is definitionally weaker than non-malleability. However, given a non-malleable encryption scheme, it is trivial to implement a tnm scheme using it with no additional assumptions or loss in security. We also show how to implement a non-malleable scheme using a tnm scheme and a strong one-time signature scheme.

2 Preliminaries

Trapdoor Permutations [2, 15] A *permutation generator* G_* is a probabilistic polynomial time algorithm that takes as input 1^k and outputs three polynomial-time algorithms (f, f^{-1}, d), the first two being deterministic, and the last being probabilistic. The range of $d(1^k)$ is required to be a subset of $\{0,1\}^k$, and f, f^{-1} are permutations over the range of $d(1^k)$, and are inverses of each other. G_* is a *trapdoor permutation generator* if it is a permutation generator such that for all non-uniform polynomial-time algorithms $\mathcal{A}$, $\Pr[(f, f^{-1}, d) \leftarrow G_*(1^k); x \leftarrow d(1^k); y \leftarrow f(x) : \mathcal{A}(f, d, y) = x]$ is negligible. It is commonly assumed that, for example, RSA is a trapdoor permutation.

Encryption schemes An *encryption scheme* Π is a triple (G, E, D) of algorithms, the first two being probabilistic, and all running in polynomial time. G takes as input 1^k and outputs a public key pair (pk, sk), i.e., $(pk, sk) \leftarrow G(1^k)$. E takes a public key pk and a message m as input and outputs an encryption c for m; we denote this $c \leftarrow E_{pk}(m)$. D takes a private key sk and a ciphertext c as input and returns either a message m such that c is a valid encryption of m, if such an m exists, and otherwise returns $\perp$; we denote this $m \leftarrow D_{sk}(c)$.

As discussed in Section 1, indistinguishability [14] is the most commonly studied goal for encryption. Here we adopt definitions $\mathsf{ind\text{-}cpa}$, $\mathsf{ind\text{-}cca1}$, and $\mathsf{ind\text{-}cca2}$ from [1]; see Definition B.1 in Appendix B. Below we give the definition of non-malleability from Dolev, Dwork and Naor [11], as written explicitly as the simulator-based non-malleable (snm) definition in Bellare and Sahai [4].[3]

Definition 2.1 **($\mathsf{snm\text{-}cpa}$, $\mathsf{snm\text{-}cca1}$, $\mathsf{snm\text{-}cca2}$)** *Let $\Pi = (G, E, D)$ be an encryption scheme, let R be a relation, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, and let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be an algorithm (the "simulator"). For $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$ and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{snm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{A},\Pi}(R, k) = 1] - \Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{S},\Pi}(R, k) = 1],$$

where

[3] Actually we slightly modify the definition of [4] so as to not require that every element of $\mathbf{y}$ decrypt to a valid plaintext. This is needed for the equivalences stated in [4] to hold.

$\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{A},\Pi}(R,k)$:	$\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)$:
$(pk,sk) \leftarrow G(1^k)$	$(pk,sk) \leftarrow G(1^k)$
$(M,s_1,s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M,s_1,s_2) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$	$x \leftarrow M$
$y \leftarrow E_{pk}(x)$	
$\mathbf{y} \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s_2,y)$	$\mathbf{y} \leftarrow \mathcal{S}_2(s_2)$
$\mathbf{x} \leftarrow D_{sk}(\mathbf{y})$	$\mathbf{x} \leftarrow D_{sk}(\mathbf{y})$
Return 1 iff $y \notin \mathbf{y} \wedge R(x,\mathbf{x},M,s_1)$	Return 1 iff $R(x,\mathbf{x},M,s_1)$

and

$$\begin{aligned}
\mathsf{atk}=\mathsf{cpa} &\;\Rightarrow\; \mathcal{O}_1(\cdot)=\epsilon, \mathcal{O}_2(\cdot)=\epsilon \\
\mathsf{atk}=\mathsf{cca1} &\;\Rightarrow\; \mathcal{O}_1(\cdot)=D_{sk}(\cdot), \mathcal{O}_2(\cdot)=\epsilon \\
\mathsf{atk}=\mathsf{cca2} &\;\Rightarrow\; \mathcal{O}_1(\cdot)=D_{sk}(\cdot), \mathcal{O}_2(\cdot)=D_{sk}(\cdot)
\end{aligned}$$

We say that Π is secure in the sense of snm-atk *for if for every polynomial $q(k)$, every R computable in time $q(k)$, every $\mathcal{A}$ that runs in time $q(k)$ and outputs a valid message space M samplable in time $q(k)$, there exists a polynomial-time algorithm $\mathcal{S}$ such that $\mathsf{Adv}^{\mathsf{snm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$ is negligible.*

Technically, for our definitions to hold with respect to random oracles we would need to explicitly include a random oracle in our experiments. However, this can be done in a standard way, and for readability it is not included.

3 Weak non-malleability

3.1 Definition

Here we propose a definition for weak non-malleable (wnm) encryption schemes. As in Definition 2.1, a wnm-secure encryption scheme requires the existence of a simulator $\mathcal{S}$ (not given a challenge ciphertext y) that has roughly the same probability as an adversary $\mathcal{A}$ (given y) of generating a vector $\mathbf{y}$ of ciphertexts for which the plaintext vector $\mathbf{x}$ bears some relationship R with the plaintext x of y. In the wnm definition, the adversary experiment will take exactly the same form as that in Definition 2.1. The difference lies in the simulator experiment and the form of $\mathcal{S}$. Specifically, $\mathcal{S}$ is permitted to make each element y_i of $\mathbf{y}$ contingent upon a "guess" z_i as to the value of x. That is, relation R tests x against a vector $\mathbf{x}$ where each element x_i is the plaintext of the corresponding y_i in $\mathbf{y}$ if either $\mathcal{S}$ guessed x or offered no guess (i.e., guessed $\perp$), and where x_i is $\perp$ otherwise.

It is easy to see that any snm-secure encryption scheme is also wnm-secure, since the wnm-simulator is simply given more power. It is perhaps not as easy to see that this power is sufficient to allow a wnm-secure scheme that is not snm-secure, but we will show that in fact this is the case. For example, the wnm-schemes presented in the introduction are not snm-secure in the random oracle model.[4]

The precise definition of wnm security is as follows.

[4]Actually, it is much easier to see that they are not comparison-based non-malleable (cnm) [4], and then use the result in [4] that simulation-based non-malleability implies comparison-based non-malleability. Also, note that our separation result in Lemma 3.4 holds not just in the random oracle model, but in the standard model.

Definition 3.1 (wnm-cpa, wnm-cca1, wnm-cca2) *Let $\Pi = (G, E, D)$ be an encryption scheme, let R be a relation, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, and let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be an algorithm ("simulator"). For* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$ *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R, k) = 1] - \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R, k) = 1],$$

where

$\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R, k)$:	$\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R, k)$:
$(pk, sk) \leftarrow G(1^k)$	$(pk, sk) \leftarrow G(1^k)$
$(M, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M, s_1, s_2) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$	$x \leftarrow M$
$y \leftarrow E_{pk}(x)$	
$\mathbf{y} \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s_2, y)$	$(\mathbf{y}, \mathbf{z}) \leftarrow \mathcal{S}_2(s_2)$
$\mathbf{x} \leftarrow D_{sk}(\mathbf{y})$	$\mathbf{x} \leftarrow D'_{sk}(\mathbf{y}, \mathbf{z}, x)$
Return 1 iff $(y \notin \mathbf{y}) \wedge R(x, \mathbf{x}, M, s_1)$	Return 1 iff $R(x, \mathbf{x}, M, s_1)$

and $D'_{sk}(\mathbf{y}, \mathbf{z}, x)$ returns the decryption of each $y_i \in \mathbf{y}$ for which $z_i = x$ or $z_i = \perp$, and returns $\perp$ for each other index, and

$$\begin{aligned}
\mathsf{atk} = \mathsf{cpa} \quad &\Rightarrow \quad \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca1} \quad &\Rightarrow \quad \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca2} \quad &\Rightarrow \quad \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot)
\end{aligned}$$

We say that Π is wnm-atk*-secure if for every polynomial $q(k)$, and every $\mathcal{A}$ that runs in time $q(k)$ and outputs a valid message space M samplable in time $q(k)$, there exists a polynomial-time algorithm $\mathcal{S}$ such that for every R computable in time $q(k)$, $\mathsf{Adv}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k)$ is negligible.*

Lemma 3.2 *For any* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$, snm-atk $\Rightarrow$ wnm-atk $\Rightarrow$ ind-atk.

The proof of Lemma 3.2 is in Appendix A.1.

Lemma 3.3 (ind-cca1 $\not\Rightarrow$ wnm-cpa) *If there exists an* ind-cca1*-secure encryption scheme, then there exists an* ind-cca1*-secure encryption scheme that is not* wnm-cpa*-secure.*

The proof of Lemma 3.3 is essentially identical to the proof that ind-cca1 $\not\Rightarrow$ snm-cpa in [1].

Lemma 3.4 (wnm-cca1 $\not\Rightarrow$ snm-cpa) *If there exists an* snm-cca1*-secure encryption scheme, then there exists a* wnm-cca1*-secure encryption system that is not* snm-cpa*-secure.*

The proof of Lemma 3.4 is in Appendix A.2.

3.2 Constructions

In Section 1, we introduced several constructions for wnm-secure encryption, denoted "Mult-Range", "Mult-Adjacent", and "Add-Square". Our goal in this section will be to prove Lemma 3.5.

Lemma 3.5 *The Mult-Range, Mult-Adjacent, and Add-Square schemes are all* wnm-atk *secure, for* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}\}$.

In fact, we prove a more general result. We show a general construction of weakly non-malleable encryption schemes, of which the three constructions above are special cases. We first introduce a notion called "uniquely identifiable subset," which we will use in our general construction.

We say a sequence of sets $X = \{X_k\}_{k>0}$, $X_k \subseteq \{0,1\}^*$, is *efficient* if there exists a polynomial $p(\cdot)$ such that membership in X_k can be tested in time $p(k)$. For simplicity, we often abuse notations by referring to the sequence $\{X_k\}$ as "the efficient set X" and omitting the subscript k, although it should be understood that X is a sequence of sets. We extend this notation to groups, too, i.e., when we say "X is an efficient finite group," it should be understood that $X = \{X_k\}$ is in fact a sequence of finite groups, whose membership can be efficiently determined. Furthermore, for efficient sets X and S, we use the phrase "S is a subset of X" as shorthand for "for every k, S_k is a subset of X_k."

Definition 3.6 (Unique Identifiability) *Let X be an efficient finite group with identity element e, and let S be an efficient subset of X. We say S is a* uniquely identifiable subset of X*, if for every $\lambda \in X \backslash \{e\}$, there exists at most one $x_\lambda \in S$, such that $\lambda \cdot x_\lambda \in S$ and for any other $x \in S, x \neq x_\lambda$, $\lambda \cdot x \notin S$. Here "$\cdot$" is the group operation. We call x_λ the* soft spot *for λ. If no such x_λ exists, we write this as $x_\lambda = \perp$. We denote the soft spot of λ by $\mathsf{ss}(\lambda)$.*

Furthermore, we say S is an efficient uniquely identifiable subset of X*, if there exists a polynomial-time algorithm A that outputs x_λ on input λ.*

Putting the definition in our context, X is the space of all messages and S is the set of all "valid" messages. The group operation "$\cdot$" corresponds to a "mauling" function the converts an encryption of x to an encryption of $\lambda \cdot x$, and we call λ the "mauling factor." The unique identifiability indicates, therefore, for every mauling factor λ, there is at most one valid message x_λ that can be mauled into another valid one (all other valid messages are mapped to invalid ones). For an efficient uniquely identifiable subset, one can in fact find x_λ efficiently.

Next, we give several examples of efficient uniquely identifiable subsets, which are closely related to the Mult-Range, Mult-Adjacent, and the Add-Square schemes.

Example 1 (Mult-Adjacent) *Let $X = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ with the group operation being component-wise multiplication in $\mathbb{Z}_p^*$, i.e., $(x_0, x_1) \cdot (y_0, y_1) = (x_0 \cdot y_0,\ x_1 \cdot y_1)$. Let $S = \{(x, x+1) \mid x \in \mathbb{Z}_p^*\}$.*

Example 2 (Add-Square) *Let $X = \mathbb{Z}_p \times \mathbb{Z}_p$, with the group operation being component-wise addition in $\mathbb{Z}_p$, i.e., $(x_0, x_1) \cdot (y_0, y_1) = (x_0 + y_0,\ x_1 + y_1)$. Let $S = \{(x, x^2) \mid x \in \mathbb{Z}_p\}$.*

Example 3 (Mult-Range) *Let $X = \mathbb{Z}_p^*$ with multiplication as the group operation. Let $S = \{a, ..., b\}$, where $a > (b-a)^2$ and $p > 2b^2$.*

Lemma 3.7 *All three examples above are efficient uniquely identifiable systems.*

The proof of Lemma 3.7 is straightforward for Mult-Adjacent and Add-Square; Mult-Range is not straightforward, however. See Appendix A.3.

Now we present our general construction of wnm encryption schemes.

Construction 1 *Let X be an efficient finite group. Let S be an efficient uniquely identifiable subset of X, and $H : \{0,1\}^* \to X$ be a hash function. Let G_* be a trapdoor permutation generator. We construct an encryption scheme as follows. G runs G_* to get (f, f^{-1}, d), and sets $pk = <f, d>$, and*

$sk = f^{-1}$. The plaintext space of E_{pk} is S.[5] *To encrypt a message m, $E_{pk}(m)$ generates $r \leftarrow d(1^k)$ and returns $<f(r), H(r) \cdot m>$, where "$\cdot$" is the group operation in X. To decrypt a ciphertext $c = (\alpha, \beta)$, $D_{sk}(c)$ computes $m = \beta \cdot (H(f^{-1}(\alpha))^{-1})$, returns m if $m \in S$, and $\perp$ otherwise.*

Lemma 3.8 *Following the notation in Construction 1, if $H(\cdot)$ is a random oracle, then Construction 1 is* $\mathsf{wnm\text{-}atk}$ *secure, for* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}\}$.

The proof of this result is in Appendix A.4.

3.3 Applications

In this section we show that weak non-malleability suffices to implement a secure contract bidding system between two bidders. Intuitively, in an ideal contract bidding system, each of two bidders would submit its bid to a trusted authority through a secure channel (so that the messages are both secret and authenticated). In a real contract bidding system, however, it may be the case that a dishonest bidder may be able to see the encrypted bid from an honest bidder before it submits its own bid. In either case, we assume there is a public "award" function over these input bids. Depending on the application, the award function varies. For example, the simplest award function can be $\mathsf{Award}((x_0, x_1)) = (y_0, y_1)$, where $y_i = x_i$ if $x_i = \min\{x_0, x_1\}$ and $y_i = 0$ otherwise. This indicates the rule that the lowest bidder wins, with the award being his bid, and the other bidder loses and thus has zero award. (We assume a unique minimum between the bids, otherwise nobody wins.) Other forms of the award function exist. For example, a "Vickrey" award function is $\mathsf{Award}((x_0, x_1)) = (y_0, y_1)$, where $y_i = x_{1-i}$ if $x_i = \min\{x_0, x_1\}$ and $y_i = 0$ otherwise.

We specify our contract bidding system as follows:

Setup: A bidding system consisting of two bidders B_0, B_1 and an award function Award. There is also a bidding upper bound $U > 0$, such that the only valid bids are integers between 0 and U. Both bidders are given U and the award function Award.

Award function: The function $\mathsf{Award} : \{\perp, 0, 1..., U\}^2 \to \{0, 1, ..., U\}^2$ takes the bids from the bidders and computes their awards, respectively.[6] We say an award function is *fair*, if for any $\mathbf{x} = (x_0, x_1)$ and any $i \in \{0, 1\}$, $\mathsf{Award}(\mathbf{x}|_{\perp \to [i]})[i] \le \mathsf{Award}(\mathbf{x})[i]$, and $\mathsf{Award}(\mathbf{x}|_{x_{1-i} \to [i]})[i] \le \mathsf{Award}(\mathbf{x})[i]$. Here we use $\mathbf{x}|_{y \to [i]}$ to indicate the vector obtained by replacing the ith entry of $\mathbf{x}$ by y and we use $\mathbf{x}[i]$ to indicate the ith entry of $\mathbf{x}$. Intuitively, the fairness indicates that a bidder would not gain any advantage in profit by changing his bid to $\perp$ or to the other bidder's bid. We note that fairness is a reasonable requirement for bidding systems to be "useful."

Real Adversary: To model security, we consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that corrupts bidder B_1. $\mathcal{A}_1$ receives the public key pk and U, and outputs a polynomial-time samplable distribution M of bids, from which a bid bid_0 is randomly chosen for B_0. $\mathcal{A}_2$ is then given the ciphertext of bid_0 and outputs encrypted bid ebid_1 for B_1. The profit of the adversary is the award of B_1.

[5] More precisely, we assume a one-to-one correspondence between plaintexts and elements of S, and efficient encoding and decoding functions to map plaintexts to and from elements of S.

[6] We insist that the award function be a positive function. However, this is entirely arbitrary, since one can always "shift" the award function by a constant without changing its nature.

Definition 3.9 (Secure Contract Bidding) *Let* CBS *be a contract bidding system with bidding upper bound U and encryption scheme $\Pi = (G, E, D)$.* CBS *is secure if for every fair award function* Award*, every polynomial $q(k)$, every adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that runs in time $q(k)$, there exists a polynomial-time simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that $\mathsf{Adv}^{\mathsf{profit}}_{\mathcal{A},\mathcal{S},\mathsf{CBS}}(k)$ is negligible,*[7] *where*

$$\mathsf{Adv}^{\mathsf{profit}}_{\mathcal{A},\mathcal{S},\mathsf{CBS}}(k) \stackrel{\text{def}}{=} E[\mathsf{Expt}^{\mathsf{real}}_{\mathcal{A},\mathsf{CBS}}(k) - \mathsf{Expt}^{\mathsf{ideal}}_{\mathcal{S},\mathsf{CBS}}(k)],$$

and

$\mathsf{Expt}^{\mathsf{real}}_{\mathcal{A},\mathsf{CBS}}(k)$:	$\mathsf{Expt}^{\mathsf{ideal}}_{\mathcal{S},\mathsf{CBS}}(k)$:
$(pk, sk) \leftarrow G(1^k)$	
$(M, s) \leftarrow \mathcal{A}_1(pk, U)$	$(M, s) \leftarrow \mathcal{S}_1(U)$
$\mathsf{bid}_0 \leftarrow M$	$\mathsf{bid}_0 \leftarrow M$
$\mathsf{ebid}_0 \leftarrow E_{pk}(\mathsf{bid}_0)$	
$\mathsf{ebid}_1 \leftarrow \mathcal{A}_2(\mathsf{ebid}_0, s)$	$\mathsf{bid}_1 \leftarrow \mathcal{S}_2(s)$
$\mathsf{bid}_1 \leftarrow D_{sk}(\mathsf{ebid}_1)$	
return $\mathsf{Award}((\mathsf{bid}_0, \mathsf{bid}_1))[1]$	return $\mathsf{Award}((\mathsf{bid}_0, \mathsf{bid}_1))[1]$

It is clear that if the encryption scheme Π is malleable, then the system might not be secure. For example, consider the scheme where message m is encrypted as $<f(r), H(r) + m \bmod p>$, where $f(\cdot)$ is a trapdoor permutation and H a random oracle. It is an ind-cpa-secure scheme, but the real bidding system is not secure, since an adversary seeing the bid $<\alpha, \beta>$ from bidder B_0 can submit bid $<\alpha, \beta - 1>$, and underbid B_0 by 1. It is also obvious that if Π is snm-cpa-secure, then the bidding system is secure. The next theorem shows that in fact wnm-cpa-security suffices.

Theorem 3.10 *Let $\Pi = (G, E, D)$ be a* wnm-cpa*-secure encryption scheme, with a domain that includes the integer range $[0, U]$ where U is polynomially bounded by k. Then a contract bidding system* CBS *with bidding upper bound U and encryption scheme Π is secure.*

The proof of this result is in Appendix A.5. We mention that our result only applies to the case of a single auction, and specifically does not claim that repeated auctions will be secure if they use the same encryption scheme. Obviously, for repeated auctions to be secure, we would need some kind of cca2 security for our encryption scheme.

We also mention that the result does not apply to contract bidding schemes with multiple bidders that may collude. Intuitively, this is because they may each make guesses which cover the possible choices of the honest bidder, and a wrong guess for one party does not reduce the award of the party that guesses correctly. To solve the problem with multiple bidders using a wnm-secure cryptosystem, one could either allow *randomization* in the bids (e.g., each bid would be of the form (bid, r), where $r \leftarrow \{0,1\}^k$, which would ensure that the adversary has a negligible chance of guessing the full plaintext), or one could change the model to levy *penalties* for invalid bids.

4 Tag-based non-malleability

In this section, we introduce tag-based non-malleability as an alternative to standard non-malleability. Informally, in a tag-based encryption system, the encryption and decryption operations take an additional "tag." A tag is simply a binary string of appropriate length (i.e., its length has to be

[7] It may be negative, in which case we also consider it to be negligible.

polynomially bounded by the security parameter), and need not have any particular internal structure. We define security for tag-based encryption in manners analogous to security for standard encryption systems. In particular, we define *tag-based indistinguishability* (Definition B.3) and *tag-based non-malleability* (Definition 4.1) with respect to cpa, cca1, and cca2 attacks. The only changes we make to the definitions for standard encryption are: (i) in a cca2 attack, instead of requiring that the adversary $\mathcal{A}$ not query the decryption oracle with the ciphertext y that $\mathcal{A}$ receives as a challenge, we require that $\mathcal{A}$ not query the decryption oracle with a (ciphertext,tag) pair using the same tag with which y was encrypted; (ii) in the non-malleability definition, instead of requiring that $\mathcal{A}_2$ not output the ciphertext y it receives, we require that $\mathcal{A}_2$ not output any (ciphertext,tag) pair for decryption using the tag with which y was encrypted. Informally, one simply changes the "equality of two ciphertexts" in the standard definitions to "equality of *the tags of* two ciphertexts," and we have a tag-based definition.

4.1 Definition

Tag-based encryption schemes A *tag-based encryption scheme* Π is a triple (G, E, D) of algorithms, the first two being probabilistic, and all running in expected polynomial time. G takes as input 1^k and outputs a public key pair (pk, sk), i.e., $(pk, sk) \leftarrow G(1^k)$. E takes a public key pk, a message m, and a tag t as input and outputs an encryption c for m associated with t; we denote this $c \leftarrow E_{pk}(m, t)$. D takes a private key sk, a ciphertext c, and a tag t as input and returns either a message m such that c is a valid encryption of m associated with t, if such an m exists, and otherwise returns $\perp$; we denote this $m \leftarrow D_{sk}(c, t)$.

Definition 4.1 (tnm-cpa, tnm-cca1, tnm-cca2) *Let $\Pi = (G, E, D)$ be an encryption scheme, let R be a relation, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary, and let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ be an algorithm (the "simulator"). For* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$ *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\Pi}(R, k) = 1] - \Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{S},\Pi}(R, k) = 1],$$

where

$\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\Pi}(R, k)$:	$\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{S},\Pi}(R, k)$:
$(pk, sk) \leftarrow G(1^k)$	$(pk, sk) \leftarrow G(1^k)$
$(M, t, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M, t, s_1, s_2) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$	$x \leftarrow M$
$y \leftarrow E_{pk}(x, t)$	
$(\mathbf{y}, \mathbf{t}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s_2, y, t)$	$(\mathbf{y}, \mathbf{t}) \leftarrow \mathcal{S}_2(s_2, t)$
$\mathbf{x} \leftarrow D_{sk}(\mathbf{y}, \mathbf{t})$	$\mathbf{x} \leftarrow D_{sk}(\mathbf{y}, \mathbf{t})$
Return 1 iff $(t \notin \mathbf{t}) \wedge R(x, \mathbf{x}, M, s_1)$	Return 1 iff $R(x, \mathbf{x}, M, s_1)$

and

$$\begin{aligned} \mathsf{atk} = \mathsf{cpa} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\ \mathsf{atk} = \mathsf{cca1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\ \mathsf{atk} = \mathsf{cca2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot) \end{aligned}$$

We require that $\mathcal{O}_2$ not be queried with the t given to $\mathcal{A}_2$. We say that Π is secure in the sense of tnm-atk *if for every polynomial $q(k)$, every R computable in time $q(k)$, and every $\mathcal{A}$ that runs in time $q(k)$ and outputs a valid message space M samplable in time $q(k)$, there exists a polynomial-time algorithm $\mathcal{S}$ such that $\mathsf{Adv}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k)$ is negligible.*

4.2 Constructions

We give two constructions of tag-based encryption schemes, both achieving tnm-cca2-security. The first one is based one-way trapdoor permutations in the random oracle model. It is similar to the semantically secure (ind-cpa) encryption scheme from Bellare and Rogaway [2], but enjoys a higher level of security. The second is a modification of the Cramer-Shoup scheme [8, 9], but simpler.

Construction 2 *Let G_* be a trapdoor permutation generator. Let X be a finite group and $H : \{0,1\}^* \to X$ a hash function. We construct an encryption scheme as follows. G runs G_* to get (f, f^{-1}, d), and sets $pk = <f, d>$, and $sk = f^{-1}$. All messages are restricted to be elements in X. To encrypt a message m with tag t, $E_{pk}(m)$ generates $r \leftarrow d(1^k)$ and returns $<f(r), H(r,t) \cdot m>$, where "$\cdot$" is the group operation in X. To decrypt a ciphertext $c = (\alpha, \beta)$, $D_{sk}(c)$ returns $m = \beta \cdot (H(f^{-1}(\alpha), t)^{-1})$.*

Lemma 4.2 *Let H be a random oracle. If f is a trapdoor permutation, then the scheme in Construction 2 is* tnm-cca2*-secure.*

The proof of Lemma 4.2 is in Appendix A.6.

Construction 3 *Let G_q be a finite group in which the DDH assumption holds.*[8] *We define an encryption scheme as follows.*

$G_{CS}(G_q)$**:** *Let g be the generator of G_q (included in the description of G_q). Generate $g_2 \xleftarrow{R} G_q$ and $a, b, c, d, e \xleftarrow{R} \mathbb{Z}_q$, and set $U \leftarrow g^a(g_2)^b$, $V \leftarrow g^c(g_2)^d$, and $W \leftarrow g^e$. Let the public key be $<g, g_2, U, V, W>$ and the secret key be $<a, b, c, d, e>$.*

$E_{<g,g_2,U,V,W>}(m, t)$**:** *Generate $r \xleftarrow{R} \mathbb{Z}_q$ and $x \leftarrow g^r$, $y \leftarrow (g_2)^r$, $w \leftarrow W^r m$, and $v \leftarrow U^r V^{rt}$. Return $<x, y, w, v>$ as the ciphertext.*

$D_{<a,b,c,d,e>}(<x, y, w, v>, t)$**:** *If $v \neq x^{a+ct} y^{b+dt}$, return $\perp$, else return w/x^e.*

Informally, our construction removes the collision-resistant hash function from the original Cramer-Shoup construction, and replaces the hash value $\alpha = H(x, y, w)$ by the tag t.[9]

Lemma 4.3 *The encryption scheme in Construction 3 is* tnm-cca2*-secure.*

The proof of this lemma almost directly follows the proof of security for the original Cramer-Shoup construction; we omit it here.

4.3 Applications

Intuitively, tag-based encryption schemes (and in particular, tnm schemes) are useful in systems that already have authentication, i.e., in systems where Bob cannot impersonate Alice and send messages using her identity. We stress that even with authentication, one still needs non-malleability. For example, in the contract-bidding scenario in both [11] and the previous section, we still need to

[8] Note that one possible group G_q may be found by generating a large prime p such that q divides $p-1$, and letting G_q be the subgroup of order q in $\mathbb{Z}_p^*$.

[9] We assume that $t \in \mathbb{Z}_q$. Otherwise, we would need a collision-resistant hash function to hash the tag.

make sure that Bob cannot underbid Alice by mauling her message. With a tnm system, we can use the sender's identity as the tag to achieve this goal. Suppose Alice sends a encrypted message $c = E_{pk}(m, \mathsf{Alice})$ to Charlie. A malicious Bob may be able to maul c into another ciphertext with the same tag, i.e., Alice — this is allowed in the definition — but this would not be useful for him since he cannot fake Alice's identity. Bob needs to produce some message with the tag Bob, but tnm stipulates that Bob will not have any advantage in doing so. To demonstrate this, we show how to use a tnm-cca2 scheme (in fact, a tind-cca2 scheme) to construct a protocol that realizes the *secure message transmission* functionality in the $\mathcal{F}_{\mathrm{AUTH}}$-hybrid model, in the universal composability framework. Previously, this was done using an ind-cca2 encryption scheme [5].

4.3.1 Universal-composability framework

The universal composability framework was proposed by Canetti [5] for defining the security and composition of protocols. To define security one first specifies an *ideal functionality* using a trusted party that describes the desired behavior of the protocol. Then one proves that a particular protocol operating in a real-life model securely realizes this ideal functionality. Here we briefly summarize the framework.

A (real-life) protocol π is defined as a set of n interactive Turing Machines $P_1, \ldots, P_n$, designating the n parties in the protocol. It operates in the presence of an environment $\mathcal{Z}$ and an adversary $\mathcal{A}$, both of which are also modeled as interactive Turing Machines. The environment $\mathcal{Z}$ provides inputs and receives outputs from honest parties, and may communicate with $\mathcal{A}$. $\mathcal{A}$ controls (and may view) all communication between the parties. (Note that this models asynchronous communication on open point-to-point channels.) We will assume that messages are authenticated, and thus $\mathcal{A}$ may not insert or modify messages between honest parties. (This feature could be added to an unauthenticated model using a message authentication functionality as described in [5].) $\mathcal{A}$ also may corrupt parties, in which case it obtains the internal state of the party. (In the non-erasing model, the internal state would encompass the complete internal history of the party.)

The ideal process with respect to a functionality $\mathcal{F}$, is defined for n parties $P_1, \ldots, P_n$, an environment $\mathcal{Z}$, and an (ideal-process) adversary $\mathcal{S}$. However, $P_1, \ldots, P_n$ are now dummy parties that simply forward (over secure channels) inputs received from $\mathcal{Z}$ to $\mathcal{F}$, and forward (again over secure channels) outputs received from $\mathcal{F}$ to $\mathcal{Z}$. Thus the ideal process is a trivially secure protocol with the input-output behavior of $\mathcal{F}$. More details are given in Appendix E.

4.3.2 UC secure message transmission

The functionality $\mathcal{F}_{\mathrm{M-SMT}}$ is given in Figure 1. Intuitively, this functionality allows multiple parties to send messages securely to a single receiver. Both the secrecy and the integrity of the messages are guaranteed. See [5] for more discussions.

Canetti [5] constructed a protocol that securely realizes this functionality in the $(\mathcal{F}_{\mathrm{AUTH}}, \mathcal{F}_{\mathrm{PKE}})$-hybrid model. He also showed that any ind-cca2 encryption scheme can securely realize the $\mathcal{F}_{\mathrm{PKE}}$ functionality. Therefore, one can construct a protocol using an ind-cca2 encryption scheme to securely realize $\mathcal{F}_{\mathrm{M-SMT}}$ in the $\mathcal{F}_{\mathrm{AUTH}}$-hybrid model. Here, we show that one can instead use a tag-based tind-cca2 encryption scheme.

Given a tind-cca2 encryption scheme $\Pi = (G, E, D)$, the protocol σ runs as follows. In this description, we include the identity of the receiver in the session identifier. (i) When a party P_i receives an input $(\textbf{receiver}, id|P_i)$, it runs $(pk, sk) \leftarrow G(1^k)$, and sends $(\mathsf{key}, id|P_i, pk)$ to all

$\mathcal{F}_{\text{M-SMT}}$ proceeds as follows, running with parties $P_1, \ldots, P_n$, and an adversary $\mathcal{A}$:

- In the first activation, expect to receive a value $(\textbf{receiver}, id)$ from some party P_i. Then send $(\textbf{receiver}, id, P_i)$ to the all parties and the adversary. From now on, ignore all $(\textbf{receiver}, id)$ values.
- Upon receiving a value $(\textbf{send}, id, m)$ from some party P_j, send (id, P_j, m) to P_i and $(id, P_j, |m|)$ to the adversary.

Figure 1: Functionality $\mathcal{F}_{\text{M-SMT}}$

other parties using $\mathcal{F}_{\text{AUTH}}$. Any messages of this type with an identifier not in the correct format are ignored. (ii) On receiving the first message $(P_{i'}, P_j, (\mathsf{key}, id|P_{i'}, pk'))$ from $\mathcal{F}_{\text{AUTH}}$, P_j records $(P_{i'}, id, pk')$ and outputs $(\textbf{receiver}, id|P_{i'}, P_{i'})$. Any messages of this type with an identifier not in the correct format are ignored. Subsequent messages of this type with identifier $id|P_{i'}$ are ignored. (iii) After this, when P_j receives an input $(\textbf{send}, id|P_{i'}, m)$, P_j runs $c \leftarrow E_{pk'}(m, P_j)$, and invokes $\mathcal{F}_{\text{AUTH}}$ to send $(\mathsf{msg}, id|P_{i'}, c)$ to $P_{i'}$. (iv) On receiving a message $(P_j, P_i, (\mathsf{msg}, id|P_i, c))$ from $\mathcal{F}_{\text{AUTH}}$, P_i runs $m \leftarrow D_{sk}(c, P_j)$ and if $m \neq \perp$, outputs $(id|P_i, P_j, m)$. Intuitively, the protocol uses the identity of the senders as the tag for the encryption.

Theorem 4.4 *The protocol σ securely realizes the SMT functionality in the $\mathcal{F}_{\text{AUTH}}$ hybrid model, assuming static corruptions.*

The proof of Theorem 4.4 is in Appendix A.7.

4.4 Relation to standard definitions

We study the relation between the tag-based definitions and the standard ones. First, we note that they are not directly comparable, due to the structural difference in encryption and decryption. However, given a standard encryption scheme $\Pi = (G, E, D)$, it is straightforward to construct a tag-based scheme $\Pi' = (G', E', D')$ with the same security as follows. G' is the same as G; $E'_{pk}(m, t)$ calls $E_{pk}(m \circ t)$, where $x \circ y$ denotes a canonical encoding of the concatenation of two binary strings that can be uniquely parsed; $D'_{sk}(c, t)$ calls $(m, t') \leftarrow D_{sk}(c)$ to and returns m if $t = t'$ and $\perp$ otherwise. It is easy to check that Π' enjoys the same level of security (in the sense of Definition 4.1) as Π (in the sense of Definition 2.1).

Interestingly, the other direction also holds: given a tag-based scheme, one can construct a standard scheme, using a strong one-time signature scheme [20] as defined in Appendix D.

Construction 4 *Let $\Pi = (G, E, D)$ be a tag-based encryption scheme. Let $\mathsf{SIG} = (\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$ be a strong one-time signature scheme. We construct a standard scheme $\Pi' = (G', E', D')$ as follows. $G' = G$. To encrypt massage m using pk, generate a signing/verification key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$; encrypt m using $\mathsf{sig\_vk}$ as the tag, i.e., $c \leftarrow E_{pk}(m, \mathsf{sig\_vk})$; sign c using $\mathsf{sig\_sk}$, i.e., $s \leftarrow \mathsf{sig\_sign}(sk, c)$; and output $(\mathsf{sig\_vk}, c, s)$ as the encryption. To decrypt a ciphertext $(\mathsf{sig\_vk}, c, s)$, verify that s is a valid signature of c with respect to $\mathsf{sig\_vk}$; if not, output $\perp$; if so, return $D_{sk}(c, \mathsf{sig\_vk})$.*

Theorem 4.5 *For* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$*: if* Π *is* $\mathsf{tnm\text{-}atk}$ *secure, then* Π' *is* $\mathsf{snm\text{-}atk}$ *secure; and if* Π *is* $\mathsf{tind\text{-}atk}$ *secure, then* Π' *is* $\mathsf{ind\text{-}atk}$ *secure.*

The proof of this result is in Appendix A.8.

Construction 4 is essentially the construction first shown in [11] and later used in [20, 10, 17, 6] to obtain non-malleable encryption schemes, except that we explicitly separate the underlying tag-based scheme from the "wrapper" that uses the one-time signature scheme. Thus, in each of these papers, there is an implicit tag-based non-malleable encryption scheme.[10] We illustrate this with the scheme of Lindell [17], which we denote as Π_L. In Π_L, an encryption of message m is a tuple $<c_0, c_1, pk_0, pk_1, r, \mathsf{sig\_vk}, \sigma, s>$. Here c_0 and c_1 are two encryptions of m using two $\mathsf{ind\text{-}cpa}$ systems with public keys pk_0 and pk_1, respectively; $\mathsf{sig\_vk}$ is a "fresh" verification key of a strong one-time signature scheme; r is a random string; σ is an NIZK proof that either c_0 and c_1 are the encryption of the same message, or r is the commitment of $\mathsf{sig\_vk}$; s is a signature of the tuple $<c_0, c_1, pk_0, pk_1, \mathsf{sig\_vk}, r, \sigma>$. Then in the underlying tag-based encryption scheme Π, an encryption of message m with tag t is the tuple $<c_0, c_1, pk_0, pk_1, r, t, \sigma>$, where c_0, c_1, pk_0, pk_1, and r are all the same as before, and σ becomes an NIZK proof that either c_0 and c_1 are the encryptions of the same message, or r is the commitment of t. It is easy to verify that Π is $\mathsf{tnm\text{-}cca2}$-secure. In fact, one can prove the security for Π almost exactly the same way as for the security proof of Π_L, observing that the use of the strong one-time signature in Π_L is solely for enforcing that an adversary will not make a query to the decryption oracle with a ciphertext having the same verification key. Since in the tag-based system Π, the verification key is replaced by the tag, by definition, the adversary cannot query the decryption oracle with a ciphertext having the same tag. So in fact the proof for the security of Π is even simpler than the proof for for Π_L. Furthermore, Π_L is exactly the transformed version of protocol Π under Construction 4. Therefore, one could obtain an alternative proof of security for Π_L by pluggin Π into Theorem 4.5.

In the light of Construction 4, the tag-based definitions are in some sense "equivalent" to the standard definitions. It is also straightforward to verify that many known relations among standard security definitions translate to tag-based definitions. In particular, all the results by Bellare et al. [1] are true for the tag-based definitions, as well as the results by Bellare and Sahai [4], for appropriately modified definitions. We summarize these results without proof in Appendix C.

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO '98* (Lecture Notes in Computer Science 1462), pp. 26–45, 1998.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In 1st *ACM Conf. on Comp. and Comm. Security*, pp. 62–73, 1993.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT '94* (Lecture Notes in Computer Science 950), pp. 92–111, 1995.

[10] In the (independent and concurrent) result of [6], there is actually an explicit *identity-based encryption (IBE) scheme* which corresponds to our tag-based non-malleable encryption scheme. They essentially prove the $\mathsf{cca2}$ case of Theorem 4.5. (Note: their cpa-secure IBE scheme corresponds to our $\mathsf{cca2}$-secure tnm scheme.)

[4] M. Bellare and A. Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In *Advances in Cryptology—CRYPTO '99* (Lecture Notes in Computer Science 1666), pp. 519–536, 1999.

[5] R. Canetti. Universally Composable Security: A new paradigm for cryptographic protocols. `http://eprint.iacr.org/2000/067`, Extended abstract in 42nd FOCS, 2001.

[6] R. Canetti, S. Halevi and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *ePrint archive*, Report 2003/182.

[7] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally composable two-party computation. In *STOC 02*, pp. 494–503, 2002. The full version in *ePrint archive*, Report 2002/140.

[8] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—CRYPTO '98* (Lecture Notes in Computer Science 1462), pp. 13–25, 1998.

[9] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology—EUROCRYPT 2002* (Lecture Notes in Computer Science 2332), pp. 45–64, 2002.

[10] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust non-interactive zero knowledge. In *Advances in Cryptology – CRYPTO 2001* (LNCS 2139), pp. 566–598, 2001.

[11] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *SIAM J. on Comput.*, 30(2):391–437, 2000. An earlier version appeared in *23rd ACM Symp. on Theory of Computing*, pp. 542–552, 1991.

[12] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. on Info. Theory* 31:469–472, 1985.

[13] S. Even, O. Goldreich, and S. Micali. On-line/Off-line digital signatures. *J. Cryptology* 9(1):35-67 (1996).

[14] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences* 28:270–299, 1984.

[15] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. on Computing* 17(2):281–308, Apr. 1988.

[16] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology—CRYPTO '91*, (Lecture Notes in Computer Science 576), pp. 433–444, 1991.

[17] Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumption. In *Advances in Cryptology—EUROCRYPT 2003*, (LNCS 2656), pp. 241–254, 2003.

[18] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystem. *Comm. of the ACM* 21(2):120–126, Feb. 1978.

[19] A. M. Rockett and P. Szüsz. Continued Fractions. World Scientific Publishing Co. Pte. Ltd. ISBN 981-02-1047-7, 1992.

[20] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th IEEE Symp. on Foundations of Computer Sci.*, pp. 543–553, 1999.

[21] V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1). Manscript. `http://www.shoup.net/papers/iso-2_1.pdf`

[22] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology—EUROCRYPT '98* (Lecture Notes in Computer Science 1403), pp. 1–16, 1998.

A Proofs

A.1 Proof of Lemma 3.2

In light of the discussion in Section 3.1, it would be obvious that snm-atk $\Rightarrow$ wnm-atk, were it not for the fact that the order of quantifiers in Definition 3.1 is slightly changed from that in Definition 2.1. In particular, in wnm-atk, we require the existence of a simulator that works for every relation, i.e., $\exists\mathcal{S}\forall R$, but in snm-atk, security only requires that for every relation there exists a simulator, i.e., $\forall R\exists\mathcal{S}$. However, it is easy to see that either order of quantifiers implies the same security for snm-atk. This is because in the proof in [1] that a different notion of security, namely cnm-atk security (see Definition B.2 in Appendix B), implies snm-atk, the simulator in the proof does not depend on the relation R. Thus the proof actually shows that cnm-atk security implies the stronger snm-atk security, and since the strong snm-atk security implies the weaker snm-atk security, and the weaker snm-atk security implies cnm-atk security, the two definitions must be equivalent. Finally, it is obvious that the stronger snm-atk security implies wnm-atk security.

Now to prove wnm-atk $\Rightarrow$ ind-atk, let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an ind-atk adversary for encryption scheme Π. We construct a wnm-atk adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for Π as follows:

$\mathcal{B}_1^{\mathcal{O}_1}(pk)$:
  $m_0, m_1 \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$
  $M \leftarrow$ uniform distribution on $\{m_0, m_1\}$
  Return $(M, \perp, <pk, m_0, m_1>)$

$\mathcal{B}_2^{\mathcal{O}_2}(<pk, m_0, m_1>, y)$:
  $b \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(y)$
  Return $<E_{pk}(m_b)>$

Now, consider the relation R defined as follows:

Relation $R(x, \mathbf{x}, M, s)$
  $<x_1, \ldots, x_k> \leftarrow \mathbf{x}$
  Return 0 if $k > 1$
  Return 1 if $x_1 = x$
  Return 0

If $\mathcal{A}$ breaks Π (for property ind-atk) with probability $\frac{1}{2}+\epsilon$, then $\Pr[\mathsf{Expt}_{\mathcal{B},\Pi}^{\mathsf{wnm\text{-}atk}}(R, k) = 1] = \frac{1}{2}+\epsilon$. However, for any simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, $\Pr[\mathsf{Expt}_{\mathcal{S},\Pi}^{\mathsf{wnm\text{-}atk}}(R, k) = 1] \leq \frac{1}{2}$ because $R(x, \mathbf{x}, M, s)$ outputs 1 only if $|\mathbf{x}| = 1$ and $x_1 = x$, where x is chosen from a uniform distribution of two elements.

A.2 Proof of Lemma 3.4

As a warm-up, we will show the weaker, but much simpler, result that in the random oracle model, if trapdoor permutations exist, then there exists a wnm-cca1-secure encryption scheme that is not snm-cpa-secure. By Lemma 3.5, the Mult-Range scheme is wnm-cca1-secure in the random oracle model if trapdoor permutations exist. We show that it is not snm-cpa-secure, however. We do this by showing that it is not cnm-cpa-secure (see Definition B.2), and using the fact that

$\mathsf{snm\text{-}cpa} \Rightarrow \mathsf{cnm\text{-}cpa}$ [4]. Consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined as follows. $\mathcal{A}_1(pk)$ returns M equal to the uniform distribution on two distinct elements $x_0, x_1 \in [a, b)$. $\mathcal{A}_2(s, y)$, where $y = <\alpha, \beta>$, returns $\mathbf{y} = (<\alpha, (\beta/x_0) \cdot (x_0 + 1)>)$ and R such that $R(x, \mathbf{x}) = 1$ iff $\mathbf{x} = <x + 1>$. Then, $\Pr[\mathsf{Expt}^{\mathsf{cnm\text{-}cpa}}_{\mathcal{A},\Pi}(k) = 1] = 1/2$ but $\Pr[\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cpa}}_{\mathcal{A},\Pi}(k) = 1] = 1/4$. Our warm-up result follows.

Now we proceed to the proof of the lemma. Let $\Pi = (G, E, D)$ be an $\mathsf{snm\text{-}cca1}$-secure encryption scheme. By [4], since Π is $\mathsf{snm\text{-}cca1}$-secure, it is also $\mathsf{cnm\text{-}cca1}$-secure. Without loss of generality, we will assume Π has a message space of size $s \geq 5$, say with elements $\{w_i : 0 \leq i < s\}$. Now let $\Pi' = (G', E', D')$ be defined as follows, with a message space of $\{0, 1\}$.

- $G'(1^k)$ returns the output of $G(1^k)$.
- $E'_{pk}(x)$ generates a random $r \stackrel{R}{\leftarrow} \{0, 1, 2\}$ and returns $<E_{pk}(w_r), x - r \bmod 3>$. We call the second component the *offset* of the encryption.
- $D'_{sk}(<y, z>)$ checks that $D_{sk}(y) = w_i$ for some $i \in \{0, 1, 2\}$, that $z \in \{0, 1, 2\}$, and that $z + i \bmod 3 \in \{0, 1\}$. If so, it returns $z + i \bmod 3$, and otherwise returns $\perp$.

One can show that Π' is not $\mathsf{snm\text{-}cpa}$-secure using a similar argument to our warm-up result. (Have $\mathcal{A}_2(s, (\alpha, \beta))$ return $\mathbf{y} = (<\alpha, \beta + 1 \bmod 3>)$ and R such that $R(x, \mathbf{x}) = 1$ iff $\mathbf{x} = <x + 1 \bmod 3>$.)

In the remainder of the proof, we will show that Π' is $\mathsf{wnm\text{-}cca1}$-secure. In particular, we will show that if Π' is not $\mathsf{wnm\text{-}cca1}$-secure, then Π is not $\mathsf{cnm\text{-}cca1}$-secure, which is a contradiction.

Let R' be a relation and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary for Π'. Consider the following adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for Π, with U_j defined as the uniform distribution over $\{w_{b-j \bmod 3} : b \in \{0, 1\}\}$, and the BConv function defined below.

$\mathcal{B}_1^{\mathcal{O}_1}(pk)$:	$\mathcal{B}_2((M, s_1, s_2, j), y)$:
$(M, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}'_1}(pk)$	$\mathbf{v} \leftarrow \mathcal{A}_2(s_2, <y, j>)$
$j \stackrel{R}{\leftarrow} \{0, 1, 2\}$	$(R, \mathbf{y}) \leftarrow \mathsf{BConv}(pk, M, s_1, y, j, \mathbf{v})$
Return $(U_j, (M, s_1, s_2, j))$	Return $(R, \mathbf{y})$

Note that for clarity we use the notation $\mathcal{O}'_1$ to denote the decryption oracle for Π' given to $\mathcal{A}_1$. In this experiment, queries to $\mathcal{O}'_1$ are answered by $\mathcal{B}_1$ using $\mathcal{O}_1$.

The BConv function takes the vector $\mathbf{v}$ of Π'-encryptions given by $\mathcal{A}_2$, and returns a vector $\mathbf{y}$ of Π-encryptions and a relation R, such that the real experiments for Π' (with adversary $\mathcal{A}$ and relation R') and Π (with adversary $\mathcal{B}$) are basically equivalent. This is possible because even though encryptions in Π' can be mauled, it is possible to determine how they are being mauled, and set R and $\mathbf{y}$ appropriately. Essential BConv detects attempted mauling and encodes the mauling factor into the $\mathbf{y}$ encryptions, and then sets R to decode the mauling factors and check the guesses. (Note that we use a message space of 5 elements so that we have two extra elements to encode attempted maulings, and these encodings will not interfere with normal encryptions.)

$\mathsf{BConv}(pk, M, s_1, y, j, (<y'_1, j_1>, \ldots, <y'_\ell, j_\ell>))$
  Let $L = \{1, \ldots, \ell\}$
  Let $I = \{i : i \in L \wedge y'_i = y \wedge j_i \in \{0, 1, 2\}\}$
  Compute $\mathbf{y}$
    /* Pass non-mauled or invalid encryptions through */
    For all $i \in L \setminus I$, $y_i \leftarrow y'_i$
    /* Mauled encryptions are marked with the guess plus 3 */
    /* The guess is the mauling factor $j_i - j$ minus 1 */
    For all $i \in I$, $y_i \leftarrow E_{pk}(w_{(j_i - j - 1 \bmod 3)+3})$
  Define R:
    $R(w_{x_0}, (w_{x_1}, \ldots, w_{x_\ell})) = 1$ iff for some $b \in \{0, 1\}$,
    $x_0 = b - j \bmod 3$ and $R'(b, (b_1, \ldots, b_\ell), M, s_1) = 1$, where
      /* For mauled encryptions, check if guess is correct */
      /* (the guess (plus 3) was determined and encrypted above) */
      /* If so, the Π' decryption would return $1 - b$ */
      For all $i \in I$, if $x_i = 3 + b$, $b_i \leftarrow 1 - b$ else $b_i \leftarrow \perp$
      /* All other encryptions are non-mauled or invalid */
      /* so compute the Π' decryption b_i equivalent */
      /* to the Π decryption w_{x_i} with offset j_i */
      For all $i \in L \setminus (J \cup I)$,
        If $x_i \notin \{0, 1, 2\}$, $j_i \notin \{0, 1, 2\}$, or $x_i + j_i \bmod 3 \notin \{0, 1\}$, $b_i \leftarrow \perp$
        Else $b_i \leftarrow x_i + j_i \bmod 3$
  Return $(R, (y_1, \ldots, y_\ell))$

One can verify that $\mathsf{Expt}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)$ performs the same experiment as $\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{A},\Pi'}(R', k)$, but with some of the computation involving the offset moved from the Π'-decryption to the relation R after the Π-decryption. Thus, $\Pr[\mathsf{Expt}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)]$ is exactly the same as $\Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{A},\Pi'}(R', k)]$.

Now we construct a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for Π'. The idea is that $\mathcal{S}$ will run $\mathcal{A}$ on a newly chosen public key for which it knows the decryption key, then produce encryptions under the given (challenge) public key so that the simulator experiment for Π' (with simulator $\mathcal{S}$ and relation R') and the "simulator" experiment for Π (with adversary $\mathcal{B}$) are basically equivalent. This is possible because the simulator in wnm has the ability to make its result depend on the ciphertext, thus matching the ability of $\mathcal{B}$ to do the same, based on its view of the offset.

$\mathcal{S}_1(pk)$:	$\mathcal{S}_2((M, s_2, pk, pk', sk'))$:
$(pk', sk') \leftarrow G'(1^k)$ $(M, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}'_1}(pk')$ return $(M, s_1, (M, s_2, pk, pk', sk'))$	$\tilde{x} \leftarrow M$; $<\tilde{y}, \tilde{j}> \leftarrow E'_{pk'}(\tilde{x})$ $\tilde{\mathbf{v}} \leftarrow \mathcal{A}_2(s_2, <\tilde{y}, \tilde{j}>)$ if $<\tilde{y}, \tilde{j}> \in \tilde{\mathbf{v}}$, abort $(\mathbf{v}, \mathbf{z}) \leftarrow \mathsf{Conv}(pk, sk', \tilde{y}, \tilde{j}, \tilde{\mathbf{v}})$ Return $(\mathbf{v}, \mathbf{z})$

Now we describe Conv.

$\mathsf{Conv}(pk, sk', y, j, (<y_1, j_1>, \ldots, <y_\ell, j_\ell>))$
  Let $L = \{1, \ldots, \ell\}$
  Let $I = \{i : i \in L \wedge y_i = y \wedge j_i \in \{0, 1, 2\}\}$
  /* Decrypt non-mauled or invalid encryptions, with no guess */
  For all $i \in L \setminus I$, $x_i \leftarrow D'_{sk'}(<y_i, j_i>)$, $z_i \leftarrow \perp$
  /* For mauled encryptions, compute the guess made */
  /* and set result of guess (i.e., 1-guess) */
  For all $i \in I$, $z_i \leftarrow j_i - j - 1 \bmod 3$, $x_i \leftarrow 1 - z_i$
  $\mathbf{v} \leftarrow E'_{pk}(\mathbf{x})$
  Return $(\mathbf{v}, \mathbf{z})$

As in [4], we can expand the definition of $\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(R,k)$, substituting in the definition of S given above, and eliminating superfluous computations, yielding:

$$
\boxed{\begin{array}{l}
\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(R,k):\\
\quad (pk', sk') \leftarrow G'(1^k)\\
\quad (M, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}'_1}(pk')\\
\quad x, \tilde{x} \leftarrow M;\ <\tilde{y}, \tilde{j}> \leftarrow E'_{pk'}(\tilde{x})\\
\quad \tilde{\mathbf{v}} \leftarrow \mathcal{A}_2(s_2, <\tilde{y}, \tilde{j}>)\\
\quad \text{Say } \tilde{\mathbf{v}} = (<y_1, j_1>, \ldots, <y_\ell, j_\ell>)\\
\quad \text{Let } L = \{1, \ldots, \ell\}\\
\quad \text{Let } I = \{i : i \in L \wedge y_i = y \wedge j_i \in \{0,1,2\}\}\\
\quad \text{For all } i \in L \setminus I,\ x_i \leftarrow D'_{sk'}(<y_i, j_i>),\ z_i \leftarrow \perp\\
\quad \text{For all } i \in I,\ z_i \leftarrow j_i - j - 1,\ x_i \leftarrow 1 - z_i\\
\quad \text{For all } i \in L\\
\qquad \text{If } z_i = \perp \text{ or } z_i = x,\ \tilde{x}_i \leftarrow x_i\\
\qquad \text{Else } \tilde{x}_i \leftarrow \perp\\
\quad \text{return 1 iff } (<\tilde{y}, \tilde{j}> \notin \tilde{\mathbf{v}}) \wedge R(x, \tilde{\mathbf{x}}, M, s_1)
\end{array}}
$$

Now examine the second cnm experiment for Π with adversary $\mathcal{B}$.

$$
\boxed{\begin{array}{l}
\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k):\\
\quad (pk, sk) \leftarrow G(1^k)\\
\quad (M, s_1, s_2) \leftarrow \mathcal{A}_1^{\mathcal{O}'_1}(pk')\\
\quad j \stackrel{R}{\leftarrow} \{0,1,2\}\\
\quad w, \tilde{w} \leftarrow U_j\\
\quad \text{Say } w = w_{x-j \bmod 3} \text{ and } \tilde{w} = w_{\tilde{x}-j \bmod 3}\\
\quad \tilde{y} \leftarrow E_{pk}(\tilde{w})\\
\quad \mathbf{v} \leftarrow \mathcal{A}_2(s_2, <\tilde{y}, j>)\\
\quad \text{Say } \mathbf{v} = (<y'_1, j_1>, \ldots, <y'_\ell, j_\ell>)\\
\quad \text{Let } L = \{1, \ldots, \ell\}\\
\quad \text{Let } I = \{i : i \in L \wedge y'_i = y \wedge j_i \in \{0,1,2\}\}\\
\quad \text{Compute } \mathbf{y}\\
\qquad \text{For all } i \in L \setminus I,\ y_i \leftarrow y'_i\\
\qquad \text{For all } i \in I,\ y_i \leftarrow E_{pk}(w_{(j_i - j - 1 \bmod 3)+3})\\
\quad \text{Define } R:\\
\qquad R(w_{x_0}, (w_{x_1}, \ldots, w_{x_\ell})) = 1 \text{ iff for some } b \in \{0,1\},\\
\qquad x_0 = b - j \bmod 3 \text{ and } R'(b, (b_1, \ldots, b_\ell), M, s_1) = 1, \text{ where}\\
\qquad\quad \text{For all } i \in I, \text{ if } x_i = 3 + b,\ b_i \leftarrow 1 - b \text{ else } b_i \leftarrow \perp\\
\qquad\quad \text{For all } i \in L \setminus I,\\
\qquad\qquad \text{If } x_i \notin \{0,1,2\},\ j_i \notin \{0,1,2\}, \text{ or } x_i + j_i \bmod 3 \notin \{0,1\},\ b_i \leftarrow \perp\\
\qquad\qquad \text{Else } b_i \leftarrow x_i + j_i \bmod 3\\
\quad \mathbf{w} \leftarrow D_{sk}(\mathbf{y})\\
\quad \text{Return 1 iff } (<\tilde{y}, j> \notin \mathbf{v}) \wedge R(w, \mathbf{w})
\end{array}}
$$

Similar to above, one can verify that $\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)$ performs the same experiment as $\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(k)$, but with some notational and name changes, and again with some of the computation involving the offset moved from the Π'-decryption to the relation R after the Π-decryption. To facilitate a comparison, we relate the notations in the two experiments here.

- pk' and sk' replaced by pk and sk, respectively.

- $E_{pk'}(\tilde{x})$ is perform explicitly using E_{pk}.
- $\tilde{j}$, $\tilde{\mathbf{v}}$, and $\{y_i\}$ are replaced by j, $\mathbf{v}$, and $\{y_i'\}$, respectively.
- $\{x_i\}$ and $\{y_i\}$ in $\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(k)$ do not have direct counterparts in $\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)$, but x and $\{\tilde{x}_i\}$ in $\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(k)$ are the same as b and $\{b_i\}$, respectively, computed for $R(w, \mathbf{w})$ in $\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)$.

Thus $\Pr[\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)]$ is exactly the same as $\Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{S},\Pi'}(k)]$, and it follows that $\mathsf{Adv}^{\mathsf{wnm\text{-}cca1}}_{\mathcal{A},\mathcal{S},\Pi'}(R,k) = \mathsf{Adv}^{\mathsf{cnm\text{-}cca1}}_{\mathcal{B},\Pi}(k)$.

A.3 Proof of Lemma 3.7

We prove the lemma for the examples one by one.

A.3.1 Mult-Adjacent

Pick an arbitrary $\lambda = (\lambda_0, \lambda_1) \in X \backslash \{(1,1)\}$. We try to find the soft spot for λ. To do this, we are essentially solving the (linear) equation $\lambda_0 \cdot x = y$ and $\lambda_1 \cdot (x+1) = (y+1)$ for unknowns $x, y \in \mathbb{Z}_p^*$. It is easy to see that it has a unique solution $x = \frac{1-\lambda_1}{\lambda_1 - \lambda_0}$, $y = \frac{\lambda_0 - \lambda_0\lambda_1}{\lambda_1 - \lambda_0}$ when $\lambda_0 \neq \lambda_1$. When $\lambda_0 = \lambda_1 \neq 1$, the equation has no solution. In any case, any $\lambda \in X \backslash \{(1,1)\}$ has at most one soft spot that can be easily computed from λ.

A.3.2 Add-Square

Pick an arbitrary $\lambda = (\lambda_0, \lambda_1) \in X \backslash \{(0,0)\}$. We try to find the soft spot for λ. To do this, we are essentially solving the equation $\lambda_0 + x = y$ and $\lambda_1 + x^2 = y^2$ for unknowns $x, y \in \mathbb{Z}_p$. It is easy to see that it has a unique solution $x = \frac{\lambda_1 - \lambda_0^2}{2\lambda_0}$, $y = \frac{\lambda_1 + \lambda_0^2}{2\lambda_0}$ when $\lambda_0 \neq 0$. When $\lambda_0 = 0 \neq \lambda_1$, the equation has no solution. In any case, any $\lambda \in X \backslash \{(0,0)\}$ has at most one soft spot that can be easily computed from λ.

A.3.3 Mult-Range

Pick an arbitrary $\lambda \in X \backslash \{1\}$. We try to find the soft spot for λ. Equivalently, we want to find $x \in \{a, ..., b\}$ and integer y such that

$$\lambda \cdot x - y \cdot p \in \{a, ..., b\}. \tag{1}$$

We shall prove that we can find such (x, y) efficiently and there exists at most one solution.

First, we need to review some some facts about continued fractions. (A good reference can be found at [19].)

Any positive real number t can be written in the form

$$t = a_0 + \cfrac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\cdots}}} \tag{2}$$

where $a_0, a_1, ...$ are positive integers. This is called the *continued fraction* of t, and the sequence $\{a_0, a_1, ..., a_n, ...\}$ is called the *partial quotients*. One normally writes (2) in a more compact form

$$t = [a_0; a_1, a_2, ..., a_n, ...] \tag{3}$$

This sequence is finite if t is rational, and infinite otherwise.

Consider $t_k = [a_0; a_1, a_2, ..., a_k]$ that consists the first $(k+1)$ terms of the continued fraction of t. Obviously t_k is a rational number and we write it as $t_k = A_k/B_k$, where $(A_k, B_k) = 1$. In this way we defined two positive integer sequences $\{A_k\}$ and $\{B_k\}$. The related sequence, $\{A_k/B_k\}$, is known as the *convergents* of t.

We review several known results about these two sequences that would be useful in the paper are (the proofs can be found in, for example, [19]):

Theorem A.1 *Both $\{A_k\}$ and $\{B_k\}$ are strictly increasing sequences, and in particular, $B_k > 2^{\lfloor k/2 \rfloor}$ for all $k \geq 0$.* ∎

Theorem A.2 *For all, $k \geq 0$, we have*

$$|B_k \cdot t - A_k| \cdot (B_k + B_{k+1}) > 1 \tag{4}$$

Theorem A.3 *For positive real number t and positive integers t, α, β, if $|\alpha - \beta \cdot t| < 1/2\beta$, then there exist integers c and k, such that $\alpha = c \cdot A_k$ and $\beta = c \cdot B_k$, where $\{A_k/B_k\}$ is the convergents of t.* ∎

Now, coming back to our problem, notice that for the desired (x, y), we have

$$0 < \frac{a}{p} \leq \frac{\lambda}{p} \cdot x - y \leq \frac{b}{p} < \frac{1}{2b} \leq \frac{1}{2x} \tag{5}$$

Therefore, by Theorem A.3, we know that $x = c \cdot B_k$ and $y = c \cdot A_k$ for some k and c, where $\{A_k/B_k\}$ is the convergent of $t = \lambda/p$.

Next, we need to find k and c. First, we have

$$B_k \leq x \leq b \tag{6}$$

Second, by Theorem A.2, we have that (setting $t = \lambda/p$),

$$2B_{k+1} > B_k + B_{k+1} > \frac{1}{|B_k \cdot t - A_k|} \geq \frac{p}{b} > 2b \tag{7}$$

Thus we know that

$$B_k \leq b < B_{k+1}. \tag{8}$$

There exists a unique k satisfying this condition, and one can easily find it, since B_k increases exponentially fast, by Theorem A.1. Now, fixing this k, we need to prove that there exists at most one c satisfying that

$$a \leq c \cdot B_k \leq b,\ a \leq c \cdot (\lambda \cdot B_k - p \cdot A_k) \leq b \tag{9}$$

Assuming otherwise, then there exists a c such that

$$a \leq c \cdot B_k < (c+1) \cdot B_k \leq b \tag{10}$$

and

$$a \leq c \cdot (\lambda \cdot B_k - p \cdot A_k) < (c+1) \cdot (\lambda \cdot B_k - p \cdot A_k) < b. \tag{11}$$

We define $T = (\lambda - 1) \cdot B_k - p \cdot A_k$. Notice that both $(\lambda - 1)$ and B_k are between 1 and $p-1$ and thus none of them is divisible by p. Therefore $T \neq 0$, and $|T| \geq 1$. Then (11) becomes

$$a \leq c \cdot B_k + c \cdot T < (c+1) \cdot B_k + c \cdot T \leq b \tag{12}$$

By (10), we know that $B_k \leq (b-a)$, and thus $c \geq a/B_k \geq a/(b-a) > (b-a)$.

Now, by combining (10) with (12), we see that both $c \cdot B_k$ and $c \cdot B_k + c \cdot T$ are within the range $\{a, ..., b\}$, yet the distance between them is $|c \cdot T| \geq c > b - a$. This is a contradiction. This proves that there is at most one c satisfying (1), and therefore the S is uniquely identifiable

It is also easy to see that S is in fact efficient uniquely identifiable. An algorithm on input λ, first finds the convergent $\{A_k/B_k\}$ of λ/p by continued fraction, then finds the unique k satisfying (8), and finally finds the unique c satisfying (9). Clearly such an algorithm runs in polynomial time.

A.4 Proof of Lemma 3.8

We prove the lemma for $\mathsf{atk} = \mathsf{cca1}$, which will imply the case $\mathsf{atk} = \mathsf{cpa}$. We use the notation of Definition 3.1 and Construction 1. Additionally, we define a "filter" function $\psi : X \to X \cup \{\perp\}$ as follows.

$$\psi(x) = \begin{cases} x & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

For an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we construct simulators $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ as follows. The simulator $\mathcal{S}_1$ runs $\mathcal{A}_1$ and simulates both the random oracle $H(\cdot)$ and the decryption oracle D_{sk} in a quite standard way. More specifically, $\mathcal{S}_1$ maintains a "query list" L consisting of pairs (α, t), such that $H(f^{-1}(\alpha)) = t$. L is initially $\emptyset$. When $\mathcal{A}_1$ makes a query r to H, $\mathcal{S}_1$ checks if $(f(r), t) \in L$ for some t, and replies with t if so; otherwise, $\mathcal{S}_1$ picks a random $t \leftarrow X$, adds $(f(r), t)$ to L, and replies with t. When $\mathcal{A}_1$ makes a query $y = (\alpha, \beta)$ to D_{sk}, $\mathcal{S}_1$ checks if $(\alpha, t) \in L$ for some t, and replies with $\psi(\beta \cdot t^{-1})$ if so; otherwise, $\mathcal{S}_1$ picks a random $t \leftarrow X$, adds (α, t) to L, and replies with $\psi(\beta \cdot t^{-1})$. Finally, when $\mathcal{A}_1$ outputs (M, s_1, s_2), $\mathcal{S}_1$ outputs $(M, s_1, (s_2, L))$.

Upon invocation, the simulator $\mathcal{S}_2$, generates $r \leftarrow d(1^k)$, $\alpha \leftarrow f(r)$, $\beta \leftarrow X$, $y \leftarrow (\alpha, \beta)$. Then $\mathcal{S}_2$ invokes $\mathcal{A}_2$ with parameters (s_2, y), and simulates the random oracle for $\mathcal{A}_2$ in the same way as $\mathcal{S}_1$ does for $\mathcal{A}_1$, using the list L passed from $\mathcal{S}_1$. When $\mathcal{A}_2$ outputs $\mathbf{y} \leftarrow \mathcal{A}_2(s_2, y)$, $\mathcal{S}_2$ aborts if $y \in \mathbf{y}$. Otherwise, we assume that $\mathbf{y} = (y_1, y_2, .., y_\ell)$, where $y_i = (\alpha_i, \beta_i)$ for $i = 1, 2, ..., \ell$. $\mathcal{S}_2$ generates $\mathbf{z}$ as follows. For each i, if $\alpha_i \neq \alpha$, then set $z_i \leftarrow \perp$; otherwise compute $\lambda_i \leftarrow \beta_i \cdot (\beta)^{-1}$, compute its soft spot $x_i \leftarrow \mathsf{ss}(\lambda_i)$, and then set $z_i \leftarrow x_i$. Finally $\mathcal{S}_2$ sets $\mathbf{z} = (z_1, z_1, ..., z_\ell)$, and outputs $(\mathbf{y}, \mathbf{z})$.

Next, we prove that $\Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R, k) = 1] - \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R, k) = 1]$ is negligible in k.

In order to do so, we introduce a new experiment called Mix, defined as follows.

$\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$:
$(pk, sk) \leftarrow G(1^k)$, $(M, s_1, (s_2, L)) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$, $r \leftarrow d(1^k)$, $\alpha \leftarrow f(r)$, $t \leftarrow X$, $\beta \leftarrow x \cdot t$, $L \leftarrow L \cup \{(\alpha, t)\}$
$y \leftarrow (\alpha, \beta)$
$\mathbf{y} \leftarrow \mathcal{A}_2^{H_L}(s_2, y)$
For every $y_i = (\alpha_i, \beta_i)$:
if $(f(r_i), t_i) \in L$ then $x_i \leftarrow \beta_i \cdot t_i^{-1}$
else $x_i \leftarrow X$
$\mathbf{x} \leftarrow (\psi(x_1), \psi(x_2), ..., \psi(x_\ell))$
Return 1 iff $(y \notin \mathbf{y}) \wedge R(x, \mathbf{x}, M, s_1)$

In the experiment above, H_L is the random oracle for $\mathcal{A}_2$ is simulated by $\mathcal{S}_2$ as in experiment $\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)$.

Informally, $\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$ is the same as $\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R,k)$, except using the simulator $\mathcal{S}$ to simulate the random oracle and the decryption.

We define the following three probabilities.

$$\begin{aligned} p_{\mathcal{A}} &= \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R,k) = 1] \\ p_{\mathcal{S}} &= \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k) = 1] \\ p_{\mathsf{Mix}} &= \Pr[\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k) = 1] \end{aligned}$$

We shall prove that both $p_{\mathcal{A}} - p_{\mathsf{Mix}}$ and $p_{\mathsf{Mix}} - p_{\mathcal{S}}$ are negligible k.

1. **That $p_{\mathcal{A}} - p_{\mathsf{Mix}}$ is negligible.**

   The places that $\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$ differs from $\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\Pi}(R,k)$ are: (1) Mix replaces $\mathcal{A}_1$ by $\mathcal{S}_1$, which in turn runs $\mathcal{A}_1$ while simulating both the random oracle and the decryption oracle, and (2) the Mix replaces the random oracle for $\mathcal{A}_2$ by the one simulated by $\mathcal{S}_2$, as well as the final decryption. It is easy to verify that the simulation of both oracles is valid, except in the case where $\mathcal{A}_1$ has queried H with r or D_{sk} with (α, β') for some β' (in which case L should already contain a pair (α, t') for some t' and the added pair would not be consistent) we call this an "unlucky event."

   However, since r is randomly chosen, the probability that an unlucky event happens is negligible. Therefore $p_{\mathcal{A}} - p_{\mathsf{Mix}}$ is negligible.

2. **That $p_{\mathsf{Mix}} - p_{\mathcal{S}}$ is negligible.**

   We can in fact re-write the experiments $\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$ and $\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)$ so that we can compare them side by side.

$\mathsf{Mix}^{\mathsf{wnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R,k)$:
$(pk, sk) \leftarrow G(1^k)$
$(M, s_1, (s_2, L)) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$
$r \leftarrow d(1^k)$, $\alpha \leftarrow f(r)$, $t \leftarrow X$, $\beta \leftarrow x \cdot t$
$L \leftarrow L \cup \{(\alpha, t)\}$
$y \leftarrow (\alpha, \beta)$
$\mathbf{y} \leftarrow \mathcal{A}_2^{H_L}(s_2, y)$
For every $y_i = (\alpha_i, \beta_i)$:
if $(\alpha_i, t_i) \in L$ then $x_i \leftarrow \beta_i \cdot t_i^{-1}$
else $x_i \leftarrow X$
$\mathbf{x} \leftarrow (\psi(x_1), \psi(x_2), ..., \psi(x_\ell))$
Return 1 iff $(y \notin \mathbf{y}) \wedge R(x, \mathbf{x}, M, s_1)$

$\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)$:
$(pk, sk) \leftarrow G(1^k)$
$(M, s_1, (s_2, L)) \leftarrow \mathcal{S}_1(pk)$
$x \leftarrow M$
$r \leftarrow d(1^k)$, $\alpha \leftarrow f(r)$, $\beta \leftarrow X$
$y \leftarrow (\alpha, \beta)$
$\mathbf{y} \leftarrow \mathcal{A}_2^{H_L}(s_2, y)$
For every $y_i = (\alpha_i, \beta_i)$:
if $\alpha_i = \alpha$, then
$\lambda_i \leftarrow \beta_i \cdot \beta^{-1}$
if $x = \mathsf{ss}(\lambda_i)$ then $x_i = \beta_i \cdot \beta^{-1} \cdot x$
else $x_i \leftarrow \perp$
else if $(\alpha_i, t_i) \in L$ then $x_i \leftarrow \beta_i \cdot t_i^{-1}$
else $x_i \leftarrow X$
$\mathbf{x} \leftarrow (\psi(x_1), \psi(x_2), ..., \psi(x_\ell))$
Return 1 iff $(y \notin \mathbf{y}) \wedge R(x, \mathbf{x}, M, s_1)$

The two experiments differ at $H(r)$. In Mix, $H(r)$ is explicitly defined to be t, and L is set accordingly; in $\mathsf{Expt}_{\mathcal{S},\Pi}$, $H(r)$ is implicitly defined to be $x^{-1} \cdot \beta$ (since the simulator $\mathcal{S}_2$ does not know x), but L is not updated. Notice that in both cases y has the same distribution. Since r is a random element, $\mathcal{A}_2$ never queries $H(\cdot)$ with value r except with negligible probability, because otherwise we can use it to invert $f(\cdot)$. Now assume that $\mathcal{A}_2$ never makes a query with input r. Under this assumption, $\mathcal{A}_2$ behaves identically in the two experiments.

Then the only possible differences between the two experiments are at the decryption of $y_i = (\alpha_i, \beta_i)$ where $\alpha_i = \alpha$. In experiment Mix, we have $x_i = \beta_i \cdot t^{-1} = \beta_i \cdot \beta^{-1} \cdot x$ and y_i decrypts to message $m_i = \psi(x_i)$. Thus, setting $\lambda_i = \beta_i \cdot \beta^{-1}$, we know that if x is the soft spot for λ_i, then y_i decrypts to $x_i = \beta_i \cdot \beta^{-1} \cdot x$, and otherwise to $\perp$. This is exactly what happens in experiment $\mathsf{Expt}_{\mathcal{S}}$. Therefore the two experiments have the same outcome.

Putting things together, we see that $p_{\mathcal{A}} - p_{\mathcal{S}}$ is negligible in k, and thus $\mathcal{S}$ is a valid simulator. This proves the lemma.

A.5 Proof of Theorem 3.10

First the intuition. If CBS were not secure, then there would be an adversary $\mathcal{A}$ that breaks it, meaning that for some fair Award, no simulator could achieve an expected award negligibly less than $\mathcal{A}$. But we will construct an adversary $\mathcal{B}$ for Π out of $\mathcal{A}$, and by the wnm-security of Π, there is a simulator $\mathcal{S}'$ that approximates $\mathcal{B}$ for any relation. Then we will use $\mathcal{S}'$ to build a simulator $\mathcal{S}$ for CBS that does achieve an expected award negligibly less than $\mathcal{A}$, which is a contradiction.

Now the details. Assume CBS is not secure. Then for some $q(k)$ there exists an adversary $\mathcal{A}$ that runs in time $q(k)$ and such that for every simulator $\mathcal{S}$, there exists a non-negligible $r(k)$ and an infinite number of k's in which $\mathsf{Adv}^{\mathsf{profit}}_{\mathcal{A},\mathcal{S},\mathsf{CBS}}(k) \geq r(k)$. Let

$$\mathsf{Edge}_{\mathcal{A},\mathcal{S},\mathsf{CBS}}(k,c) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{real}}_{\mathcal{A},\mathsf{CBS}}(k) \geq c] - \Pr[\mathsf{Expt}^{\mathsf{ideal}}_{\mathcal{S},\mathsf{CBS}}(k) \geq c].$$

Then using the definition of expectation, there is a c such that for an infinite number of k's, $\mathsf{Edge}_{\mathcal{A},\mathcal{S},\mathsf{CBS}}(k, c) \geq r(k)/U$. Without loss of generality, we may assume that $\mathcal{A}_2$ never outputs ebid_0, since by the fairness of the Award function, this cannot increase its advantage.

Define relation $R_c(x, \mathbf{x}, M, s_1)$ to return 1 iff $|\mathbf{x}| = 1$ and $\mathsf{Award}(x, \mathbf{x}[0])[1] \geq c$. Consider the following adversary $\mathcal{B}$ for the wnm-cpa-security of Π.

$\mathcal{B}_1(pk)$:	$\mathcal{B}_2(s, y)$:
$(M, s) \leftarrow \mathcal{A}_1(pk, U)$	$\mathsf{ebid}_1 \leftarrow \mathcal{A}_2(y, s)$
return $(M, \perp, s)$	return ebid_1

Since Π is wnm-cpa-secure, there exists a simulator $\mathcal{S}' = (\mathcal{S}'_1, \mathcal{S}'_2)$ such that $\mathsf{Adv}^{\mathsf{wnm\text{-}cpa}}_{\mathcal{B},\mathcal{S}',\Pi}(R_c, k)$ is negligible for all c. Because $R_c(x, \mathbf{x}, M, s_1)$ returns 1 only if $|\mathbf{x}| = 1$, we assume without loss of generality that $\mathcal{S}'_2$ returns one-element vectors $\mathbf{y}$ and $\mathbf{z}$, i.e., values y and z. Now let a simulator $\mathcal{S}'' = (\mathcal{S}''_1, \mathcal{S}''_2)$ for the contract bidding system be defined as follows.

$\mathcal{S}''_1(U)$:	$\mathcal{S}''_2(s)$:
$(pk, sk) \leftarrow G(1^k)$	
$(M, s_1, s_2) \leftarrow \mathcal{S}'_1(pk)$	$(y, z) \leftarrow \mathcal{S}'_2(s)$
return (M, s_2)	return $D_{sk}(y)$

Note that by the fairness of Award, the award can never decrease when bid_1 is changed from $\perp$ to a valid bid, and so it is easy to see that $\Pr[\mathsf{Expt}^{\mathsf{ideal}}_{\mathcal{S}'',\mathsf{CBS}}(k) \geq c] \geq \Pr[\mathsf{Expt}^{\mathsf{wnm\text{-}atk}}_{\mathcal{S}',\Pi}(R_c, k) = 1]$. Using this fact, one can see that for all c, $\mathsf{Edge}_{\mathcal{A},\mathcal{S}'',\mathsf{CBS}}(k, c) \leq \mathsf{Adv}^{\mathsf{wnm\text{-}cpa}}_{\mathcal{B},\mathcal{S}',\Pi}(R_c, k)$, and thus by the discussion above, for all c, $\mathsf{Edge}_{\mathcal{A},\mathcal{S}'',\mathsf{CBS}}(k, c)$ is negligible. This is a contradiction, so CBS must be secure.

A.6 Proof of Lemma 4.2

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. Informally, the required simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ will work by invoking $\mathcal{A}$ and answering $\mathcal{A}$'s (and E's) oracle queries. Specifically, when $H(r, t)$ is queried, $\mathcal{S}$ first searches a *query list* that it maintains for $(<f(r), t>, z)$ for some z. If such a pair exists, $\mathcal{S}$ returns z; otherwise, it generates $z \stackrel{R}{\leftarrow} \mathbb{Z}_p$, stores $(<f(r), t>, z)$ on its query list, and returns z. When $\mathcal{A}$ queries $D_{sk}(<y_1, y_2>, t)$ (i.e., $\mathsf{atk} \in \{\mathsf{cca1}, \mathsf{cca2}\}$), $\mathcal{S}$ first searches the query list to see if $(<y_1, t>, z)$ appears for some z. If so, $\mathcal{S}$ takes the corresponding z and returns $y_2 \cdot z^{-1} \bmod p$; if not, $\mathcal{S}$ generates $z \stackrel{R}{\leftarrow} \mathbb{Z}_p$, stores $(<y_1, t>, z)$ on the query list, and returns $y_2 \cdot z^{-1} \bmod p$.

To complete the definition of $\mathcal{S}$, we define $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows. $\mathcal{S}_1(pk)$ performs $(M, s_1, s_2, t) \leftarrow \mathcal{A}_1(pk)$ and returns $(M, s_1, <s_2, pk, M, t>)$. $\mathcal{S}_2(<s_2, pk, M, t>)$ generates $x \leftarrow M$ and $y \leftarrow E_{pk}(x, t)$, and then returns $(\mathbf{y}, \mathbf{t}) \leftarrow \mathcal{A}_2(s_2, y, t)$.

We now show that if f is a trapdoor permutation, then $\mathsf{Adv}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k)$ is negligible. To see this, suppose $\mathsf{Adv}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\mathcal{S},\Pi}(R, k)$ is non-negligible. Since $\mathcal{A}_2(s_2, <y_1, y_2>, t)$ is not permitted to query $D_{sk}(<y_1, y_2>, t)$, it must query $H(f^{-1}(y_1), t)$ with non-negligible probability. Consider a simulator $\mathcal{S}' = (\mathcal{S}_1, \mathcal{S}'_2)$ that takes as input some $\hat{y}$ in the range of f and is challenged to return $f^{-1}(\hat{y})$. $\mathcal{S}'$ behaves as $\mathcal{S}$ except that $\mathcal{S}'_2(<s_2, pk, M, t>)$ instead invokes $\mathcal{A}_2(s_2, <\hat{y}, r>, t)$ where $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$. Whenever $\mathcal{A}_2(s_2, <\hat{y}, r>, t)$ queries $H(f^{-1}(\hat{y}), t)$, $\mathcal{S}'_2$ captures $f^{-1}(\hat{y})$ and returns it. Thus, $\mathcal{S}'$ inverts f with non-negligible probability.

A.7 Proof of Theorem 4.4

Let $\mathcal{A}$ be an adversary that interacts with parties running σ in the $\mathcal{F}_{\mathrm{AUTH}}$-hybrid model. We will construct an adversary $\mathcal{S}$ in the ideal process for $\mathcal{F}_{\mathrm{M-SMT}}$ such that no environment $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ and σ in the $\mathcal{F}_{\mathrm{AUTH}}$-hybrid model, or with $\mathcal{S}$ and $\mathcal{F}_{\mathrm{M-SMT}}$ in the ideal process. For simplicity, we assume there exists only one instance of $\mathcal{F}_{\mathrm{M-SMT}}$ with identifier $id|P_i$ for some P_i. It is straightforward to extend the behavior of $\mathcal{S}$ to the case of multiple instances. $\mathcal{S}$ runs a simulated copy of $\mathcal{A}$ and maintains a tuple $(pk^*, sk^*, \mathsf{owner})$, where pk^* is the "session public key", sk^* is the corresponding secret key, and owner is the index of the party who "owns" it. [11] The session key pair (pk^*, sk^*) is initialized to $\perp$. Then $\mathcal{S}$ forwards messages from $\mathcal{Z}$ to $\mathcal{A}$, as well as messages from $\mathcal{A}$ to $\mathcal{Z}$. Furthermore, $\mathcal{S}$ also sees the *public part* (also known as "header" [7]) of all the messages from uncorrupted parties to $\mathcal{F}_{\mathrm{M-SMT}}$ and may decide when and if to forward these messages. We refer the readers to [7] for more detailed discussions. In the case of $\mathcal{F}_{\mathrm{M-SMT}}$, all messages to $\mathcal{F}_{\mathrm{M-SMT}}$ are public, except the "payload message" m in $(\mathbf{send}, id, m)$. $\mathcal{S}$ also simulates the ideal functionality $\mathcal{F}_{\mathrm{AUTH}}$.

Next, we describe the behavior of $\mathcal{S}$ in more detail. Note that $\mathcal{S}$ simulates $\mathcal{F}_{\mathrm{AUTH}}$ as normal except as detailed below.

Simulating Communication with $\mathcal{Z}$: $\mathcal{S}$ directly forwards any messages between $\mathcal{Z}$ and $\mathcal{A}$.

Key Generation: If P_i is uncorrupted and $\mathcal{S}$ sees a message $(\mathbf{receiver}, id|P_i)$ from P_i to $\mathcal{F}_{\mathrm{M-SMT}}$, $\mathcal{S}$ forwards this message to $\mathcal{F}_{\mathrm{M-SMT}}$. If $pk^* \neq \perp$ it does nothing else. Otherwise $\mathcal{S}$ generates $(pk, sk) \leftarrow G(1^k)$, sets $(pk^*, sk^*) \leftarrow (pk, sk)$, and $\mathsf{owner} \leftarrow i$, and simulates $\mathcal{F}_{\mathrm{AUTH}}$ to send $(\mathsf{key}, id|P_i, pk)$ to all other parties.

If P_i is corrupted and $\mathcal{S}$ sees P_i send a message $(\mathsf{key}, id|P_i, pk)$ to $\mathcal{F}_{\mathrm{AUTH}}$, $\mathcal{S}$ simulates $\mathcal{F}_{\mathrm{AUTH}}$. Furthermore, if $pk^* = \perp$ and $pk \neq \perp$, then $\mathcal{S}$ sends message $(\mathbf{receiver}, id)$ to $\mathcal{F}_{\mathrm{M-SMT}}$ on behalf of P_i and sets $\mathsf{owner} \leftarrow i$ and $(pk^*, sk^*) \leftarrow (pk, ?)$. Here "$sk^* = ?$" indicates that $\mathcal{S}$ does not know the corresponding secret key.

Delivery of the public key: When $\mathcal{A}$ delivers a message $(P_i, P_j, (\mathsf{key}, id|P_i, pk))$ from $\mathcal{F}_{\mathrm{AUTH}}$ to an uncorrupted party P_j that has not received such a message previously, $\mathcal{S}$ records the tuple $(P_j, (P_i, pk))$ and delivers $(\mathbf{receiver}, id|P_i, P_i)$ from $\mathcal{F}_{\mathrm{M-SMT}}$ to P_j.

Message transfer from an uncorrupted party: If $\mathcal{S}$ sees an uncorrupted party P_j send a message $(\mathbf{send}, id|P_i, -)$ to $\mathcal{F}_{\mathrm{M-SMT}}$, where "$-$" indicates the "private" part of the message that $\mathcal{S}$ does not see, and if $\mathcal{S}$ has stored a tuple $(P_i, (P_j, pk'))$, $\mathcal{S}$ does the following. First $\mathcal{S}$ forwards the **send** message to $\mathcal{F}_{\mathrm{M-SMT}}$, and receives the length ℓ. Next, if P_i is corrupted, then $\mathcal{S}$ receives the message (id, m, P_i) from $\mathcal{F}_{\mathrm{M-SMT}}$ to the ideal P_i, sets $c \leftarrow E_{pk'}(m, P_i)$. If P_i is uncorrupted, then $\mathcal{S}$ sets $c \leftarrow E_{pk^*}(0^l, P_i)$. Finally, $\mathcal{S}$ simulates $\mathcal{F}_{\mathrm{AUTH}}$ to send $(id|P_i, c)$ to P_i.

Message transfer from a corrupted party If $\mathcal{S}$ sees a corrupted party P_j (controlled by $\mathcal{A}$) send message $(id|P_i, c)$ to P_i through $\mathcal{F}_{\mathrm{AUTH}}$, we may assume that P_i is uncorrupted, since otherwise $\mathcal{S}$ does not need to do anything. In this case, $\mathcal{S}$ sets $m \leftarrow D_{sk^*}(c, P_j)$ and if $m \neq \perp$, sends message $(\mathbf{send}, id, m)$ to $\mathcal{F}_{\mathrm{M-SMT}}$, forwarding the message (id, P_j, m) to the ideal P_i when $\mathcal{A}$ forwards the corresponding message to P_i from $\mathcal{F}_{\mathrm{AUTH}}$.

[11] Since we assume there is only one instance of $\mathcal{F}_{\mathrm{M-SMT}}$ ideal functionality, there is only one instance of protocol σ, and thus there is only one key. Also, in the case of identifier $id|P_i$, $\mathsf{owner} = i$.

Now we show that if any $\mathcal{Z}$ can distinguish whether it is interacting with $\mathcal{A}$ and σ in the $\mathcal{F}_{\text{AUTH}}$-hybrid model, or with $\mathcal{S}$ and $\mathcal{F}_{\text{M-SMT}}$ in the ideal process, then this can be used to construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks the $\mathsf{tind\text{-}cca2}$-security of Π.

Intuitively, this is true because the only possible difference between the ideal process and the real world is in the case when an uncorrupted party P_j sends a message m to another uncorrupted party P_i. In the real world, an encryption of m is sent through $\mathcal{F}_{\text{AUTH}}$; in the ideal process, $\mathcal{S}$ simulates this message using a encryption of 0^ℓ, since $\mathcal{S}$ does not know m. Notice that the tag for this encryption is always P_j, the identity of an uncorrupted party. $\mathcal{S}$ also performs decryptions, but only for messages from corrupted parties. Therefore, $\mathcal{S}$ only decrypts messages with corrupted parties' identities as tags, and in particular, no ciphertexts with tag P_j are decrypted by $\mathcal{S}$. Then, by the $\mathsf{tind\text{-}cca2}$-security of Π, the simulation of $\mathcal{S}$ is indistinguishable from the real world.

We now describe the proof more formally. $\mathcal{B}$ takes a public key pk and decryption oracle, plays the role of $\mathcal{F}_{\text{M-SMT}}$ and runs $\mathcal{S}$ with the following changes. Assume that l messages are sent using $\mathcal{F}_{\text{M-SMT}}$. $\mathcal{B}_1$ choose $h \xleftarrow{R} \{1, \ldots, l\}$. If an uncorrupted party P_i needs to generate a key pair, pk is used as the public key. Let $id|P_i$ be the associated identifier. Then for the first $h-1$ messages to P_i with id from uncorrupted parties, $\mathcal{B}$ has $\mathcal{S}$ encrypt the actual messages, instead of the all zeros message. On the hth message to P_i with $id|P_i$, say from an uncorrupted P_j, $\mathcal{B}_1$ outputs the all zeros message, the real message, the tag P_j, and its internal state. Then $\mathcal{B}_2$ uses the challenge ciphertext in the message to P_i, and continues to run $\mathcal{S}$ as normal, encrypting all zeros messages again. $\mathcal{B}_1$ and $\mathcal{B}_2$ both call the decryption oracle on messages to P_i from a corrupted P_j. Note that the tag in this case is always different from the tag returned by $\mathcal{B}_1$. Finally, $\mathcal{B}_2$ outputs whatever $\mathcal{Z}$ outputs. Note that if $h = 0$ and the bit chosen in the $\mathsf{tind\text{-}cca2}$ experiment is 0, $\mathcal{B}$ runs like $\mathcal{S}$, and if $h = \ell$ and the bit chosen in the $\mathsf{tind\text{-}cca2}$ experiment is 1, $\mathcal{B}$ runs like $\mathcal{A}$ in the real protocol. Then by a standard hybrid argument, if $\mathcal{Z}$ distinguishes whether it is interacting with $\mathcal{A}$ and σ in the $\mathcal{F}_{\text{AUTH}}$-hybrid model, or with $\mathcal{S}$ and $\mathcal{F}_{\text{M-SMT}}$ in the ideal process, $\mathcal{B}$ breaks the $\mathsf{tind\text{-}cca2}$-security of Π.

A.8 Proof of Theorem 4.5

We first consider the case where Π is $\mathsf{tnm\text{-}cca2}$ secure, and we shall prove that Π' is $\mathsf{snm\text{-}cca2}$ secure. All other cases are similar (and simpler).

Given an adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for Π', we shall construct a simulator $\mathcal{S}' = (\mathcal{S}'_1, \mathcal{S}'_2)$ for it. We first construct an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for Π, then since Π is secure, $\mathcal{A}$ has a simulator $\mathcal{S}$. Finally we construct the simulator $\mathcal{S}'$ from $\mathcal{S}$.

Now we describe $\mathcal{A}$. $\mathcal{A}_1$ runs a copy of $\mathcal{A}'_1$ and simulates the decryption oracle D_{sk} in Π. When $\mathcal{A}'_1$ submits a query $(\mathsf{sig\_vk}, c, s)$, $\mathcal{A}_1$ checks if $\mathsf{sig\_verify}(\mathsf{sig\_vk}, c, s) = 1$ and returns $\perp$ if not; otherwise, $\mathcal{A}_1$ forwards $(c, \mathsf{sig\_vk})$ to the decryption oracle D_{sk} for Π, and forwards the reply of D_{sk} back to $\mathcal{A}'_1$. When $\mathcal{A}'_1$ outputs (M, s_1, s_2), $\mathcal{A}_1$ generates a new signature key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$, sets $t = \mathsf{sig\_vk}$, and outputs $(M, \mathsf{sig\_vk}, s_1, (\mathsf{sig\_sk}, s_2))$.

$\mathcal{A}_2((\mathsf{sig\_sk}, s_2), y, t)$ first produces a ciphertext for $\mathcal{A}_2$ by generating a signature for y, $s = \mathsf{sig\_sign}(\mathsf{sig\_sk}, y)$. Then $\mathcal{A}_2$ invokes $\mathcal{A}'_2$ with input (s_2, y'), where $y' = (t, y, s)$. $\mathcal{A}_2$ also simulates the decryption oracle for $\mathcal{A}'_2$. When $\mathcal{A}'_2$ submits a query $c' = (\mathsf{sig\_vk}, c, s')$, $\mathcal{A}_2$ first checks if $c' = y$ and returns $\perp$ if does. Next $\mathcal{A}_2$ checks if s' is a valid signature, and returns $\perp$ if not. Then $\mathcal{A}_2$ checks if $\mathsf{sig\_vk}' = t$. If so, the $\mathcal{A}'_2$ manages to forge a signature for the verification key $\mathsf{sig\_vk}'$, in which case $\mathcal{A}_2$ aborts and report `‘‘SIGNATURE BROKEN.’’` Otherwise $\mathcal{A}_2$ forwards $(c, \mathsf{sig\_vk})$ to the decryption oracle D_{sk} for Π, and then forward the reply from D_{sk} to $\mathcal{A}'_2$. Finally, when $\mathcal{A}'_2$ outputs

$\mathbf{y}' = (y_1, y_2, ..., y_\ell)$, where we write $y_i = (\mathsf{sig\_vk}_i, c_i, s_i)$, $\mathcal{A}_2$ checks if $\mathsf{sig\_vk}_i = t$ and $c_i \neq c$ for any i. If so then aborts and report ``SIGNATURE BROKEN.'' Otherwise $\mathcal{A}_2$ outputs $\mathbf{y} = (c_1, c_2, ..., c_\ell)$.

It is easy to verify that if the signature scheme SIG is strongly one-time unforgeable, then the probability $\mathcal{A}_2$ aborts is negligible. Furthermore, conditioned on $\mathcal{A}_2$ does not abort, it has the same success probability as $\mathcal{A}_2'$. In other words, $\Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{A}',\Pi'}(R,k)] - \Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\Pi}(R,k)]$ is negligible.

Since Π is tnm-cca2 secure, there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that $\Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{A},\Pi}(R,k)] - \Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)]$ is negligible. We can the easily modify $\mathcal{S}$ into a simulator $\mathcal{S}' = (\mathcal{S}_1', \mathcal{S}_2')$ for Π'. Basically $\mathcal{S}_1'$ runs $\mathcal{S}_1$, and when $\mathcal{S}_1$ outputs (M, t, s_1, s_1), $\mathcal{S}_1'$ outputs $(M, s_1, (s_2, t))$. $\mathcal{S}_2'$ on input (s_2, t) simply runs $\mathcal{S}_2(s_2, t)$. It is easy to see that $\Pr[\mathsf{Expt}^{\mathsf{tnm\text{-}atk}}_{\mathcal{S},\Pi}(R,k)] = \Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{S}',\Pi'}(R,k)]$. Therefore, we have $\Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{A}',\Pi'}(R,k)] - \Pr[\mathsf{Expt}^{\mathsf{snm\text{-}atk}}_{\mathcal{S}',\Pi'}(R,k)]$ is negligible, and thus $\mathcal{S}'$ is a valid simulator for $\mathcal{A}'$.

B Other Definitions for Encryption and Tag-Based Encryption

Definition B.1 ([1]; ind-cpa, ind-cca1, ind-cca2) *Let $\Pi = (G, E, D)$ be an encryption scheme, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For* atk $\in$ {cpa, cca1, cca2} *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{ind\text{-}atk}}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{Expt}^{\mathsf{ind\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1] - 1$$

where

$$\boxed{\begin{array}{l} \mathsf{Expt}^{\mathsf{ind\text{-}atk}}_{\mathcal{A},\Pi}(k): \\ \quad (pk, sk) \leftarrow G(1^k) \\ \quad (x_0, x_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk) \\ \quad b \stackrel{R}{\leftarrow} \{0,1\} \\ \quad y \leftarrow E_{pk}(x_b) \\ \quad b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(x_0, x_1, s, y) \\ \quad \text{Return 1 iff } b = b' \end{array}}$$

and

$$\begin{aligned} \mathsf{atk} = \mathsf{cpa} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\ \mathsf{atk} = \mathsf{cca1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\ \mathsf{atk} = \mathsf{cca2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot) \end{aligned}$$

We require that $|x_0| = |x_1|$ and that $\mathcal{O}_2$ is not queried with y. We say that Π is secure in the sense of ind-atk *if for every polynomial $q(k)$ and every $\mathcal{A}$ that runs in time $q(k)$, $\mathsf{Adv}^{\mathsf{ind\text{-}atk}}_{\mathcal{A},\Pi}(k)$ is negligible in k.*

Definition B.2 ([4]; cnm-cpa, cnm-cca1, cnm-cca2) *Let $\Pi = (G, E, D)$ be an encryption scheme, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For* atk $\in$ {cpa, cca1, cca2} *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1] - \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1],$$

where

$\mathsf{Expt}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$:	$\widetilde{\mathsf{Expt}}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$:
$(pk, sk) \leftarrow G(1^k)$	$(pk, sk) \leftarrow G(1^k)$
$(M, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$
$x \leftarrow M$	$x, \tilde{x} \leftarrow M$
$y \leftarrow E_{pk}(x)$	$\tilde{y} \leftarrow E_{pk}(\tilde{x})$
$(R, \mathbf{y}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s, y)$	$(R, \tilde{\mathbf{y}}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s, \tilde{y})$
$\mathbf{x} \leftarrow D_{sk}(\mathbf{y})$	$\tilde{\mathbf{x}} \leftarrow D_{sk}(\tilde{\mathbf{y}})$
Return 1 iff $(y \notin \mathbf{y}) \wedge R(x, \mathbf{x})$	Return 1 iff $(\tilde{y} \notin \tilde{\mathbf{y}}) \wedge R(x, \tilde{\mathbf{x}})$

and

$$\begin{aligned}
\mathsf{atk} = \mathsf{cpa} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot)
\end{aligned}$$

We require that $\mathcal{O}_2$ not be queried with the y given to $\mathcal{A}_2$. We say that Π is secure in the sense of cnm-atk *if for every polynomial $q(k)$: if $\mathcal{A}$ runs in time $q(k)$, outputs a valid message space M samplable in time $q(k)$, and outputs a relation R computable in time $q(k)$, then $\mathsf{Adv}^{\mathsf{cnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$ is negligible.*

Definition B.3 (tind-cpa,tind-cca1,tind-cca2) *Let $\Pi = (G, E, D)$ be a tag-based encryption scheme, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For* atk $\in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$ *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{Expt}^{\mathsf{ind\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1] - 1$$

where

$\mathsf{Expt}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k)$:
$(pk, sk) \leftarrow G(1^k)$
$(x_0, x_1, t, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$
$b \stackrel{R}{\leftarrow} \{0, 1\}$
$y \leftarrow E_{pk}(x_b, t)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(x_0, x_1, t, s, y)$
Return 1 iff $b = b'$

and

$$\begin{aligned}
\mathsf{atk} = \mathsf{cpa} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot)
\end{aligned}$$

We require that $|x_0| = |x_1|$, and that $\mathcal{O}_2$ is not queried with tag t. We say that Π is secure in the sense of tind-atk *if for every polynomial $q(k)$ and every adversary $\mathcal{A}$ that runs in time $q(k)$, $\mathsf{Adv}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k)$ is negligible.*

Below we introduce tag-based analogs of two alternative definitions of non-malleability from [4].

Definition B.4 (tcnm-cpa, tcnm-cca1, tcnm-cca2) *Let $\Pi = (G, E, D)$ be a tag-based encryption scheme, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For* atk $\in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$ *and $k \in \mathbb{N}$ define*

$$\mathsf{Adv}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} \Pr[\mathsf{Expt}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1] - \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{S},\Pi}(k) = 1],$$

where

$\mathsf{Expt}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$:	$\widetilde{\mathsf{Expt}}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$:
$(pk, sk) \leftarrow G(1^k)$	$(pk, sk) \leftarrow G(1^k)$
$(M, t, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$	$(M, t, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$
$x \leftarrow M$	$x, \tilde{x} \leftarrow M$
$y \leftarrow E_{pk}(x, t)$	$\tilde{y} \leftarrow E_{pk}(\tilde{x}, t)$
$(R, \mathbf{y}, \mathbf{t}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s, y, t)$	$(R, \tilde{\mathbf{y}}, \mathbf{t}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(s, \tilde{y}, t)$
$\mathbf{x} \leftarrow D_{sk}(\mathbf{y}, \mathbf{t})$	$\tilde{\mathbf{x}} \leftarrow D_{sk}(\tilde{\mathbf{y}}, \mathbf{t})$
Return 1 iff $(t \notin \mathbf{t}) \wedge R(x, \mathbf{x})$	Return 1 iff $(t \notin \mathbf{t}) \wedge R(x, \tilde{\mathbf{x}})$

where

$$\begin{aligned}
\mathsf{atk} = \mathsf{cpa} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{cca2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot)
\end{aligned}$$

We require that $\mathcal{O}_2$ not be queried with the t given to $\mathcal{A}_2$. We say that Π is secure in the sense of tcnm-atk *if for every polynomial $q(k)$: if $\mathcal{A}$ runs in time $q(k)$, outputs a valid message space M samplable in time $q(k)$, and outputs a relation R computable in time $q(k)$, then $\mathsf{Adv}^{\mathsf{tcnm\text{-}atk}}_{\mathcal{A},\Pi}(k)$ is negligible.*

Below, we consider additional types of attack called *parallel query* atacks pa0, pa1, and pa2 [4]. In these attacks, a decryption oracle can be queried only once, but in that one invocation will accept and decrypt any polynomial number (in k) of inputs.

Definition B.5 (tind-pa0, tind-pa1, tind-pa2) *Let $\Pi = (G, E, D)$ be a tag-based encryption scheme and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For* $\mathsf{atk} \in \{\mathsf{pa0}, \mathsf{pa1}, \mathsf{pa2}\}$*, and $k \in \mathbb{N}$, let*

$$\mathsf{Adv}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k) \stackrel{\text{def}}{=} 2 \cdot \Pr[\mathsf{Expt}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k) = 1] - 1$$

where

$$\begin{array}{|l|}
\hline
\mathsf{Expt}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(R, k) : \\
\quad (pk, sk) \leftarrow G(1^k) \\
\quad (x_0, x_1, t, s_1) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk) \\
\quad b \stackrel{R}{\leftarrow} \{0, 1\} \\
\quad y \leftarrow E_{pk}(x_b, t) \\
\quad (\mathbf{y}, \mathbf{t}, s_2) \leftarrow \mathcal{A}_{2,q}^{\mathcal{O}_2}(x_0, x_1, t, s_1, y) \\
\quad \mathbf{x} \leftarrow D_{sk}(\mathbf{y}, \mathbf{t}) \\
\quad g \leftarrow \mathcal{A}_{2,g}^{\mathcal{O}_2}(\mathbf{x}, s_2) \\
\quad \text{Return 1 iff } (t \notin \mathbf{t}) \wedge (g = b) \\
\hline
\end{array}$$

and

$$\begin{aligned}
\mathsf{atk} = \mathsf{pa0} &\Rightarrow \mathcal{O}_1(\cdot) = \epsilon, \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{pa1} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = \epsilon \\
\mathsf{atk} = \mathsf{pa2} &\Rightarrow \mathcal{O}_1(\cdot) = D_{sk}(\cdot), \mathcal{O}_2(\cdot) = D_{sk}(\cdot)
\end{aligned}$$

We require that $|x_0| = |x_1|$ and that $\mathcal{O}_2$ not be queried with tag t. We say that Π is secure in the sense of tind-atk *if for any polynomial $q(k)$ and any adversary $\mathcal{A}$ that runs in time $q(k)$, $\mathsf{Adv}^{\mathsf{tind\text{-}atk}}_{\mathcal{A},\Pi}(k)$ is negligible.*

C Relations among Tag-based Encryption Schemes

We state a relations among notions of tag-based encryption security. These results parallel those from [1, 4] and are proved using similar techniques. We omit the proofs here.

Theorem C.1 ($\mathsf{tind\text{-}cca1} \not\Rightarrow \mathsf{tnm\text{-}cpa}$) *If there exists an* $\mathsf{tind\text{-}cca1}$*-secure encryption scheme, then there exists an* $\mathsf{tind\text{-}cca1}$*-secure encryption scheme that is not* $\mathsf{tnm\text{-}cpa}$*-secure.*

Theorem C.2 ($\mathsf{tnm\text{-}cpa} \not\Rightarrow \mathsf{tind\text{-}cca1}$) *If there exists an* $\mathsf{tnm\text{-}cpa}$*-secure encryption scheme, then there exists an* $\mathsf{tnm\text{-}cpa}$*-secure encryption scheme that is not* $\mathsf{tind\text{-}cca1}$*-secure.*

Theorem C.3 ($\mathsf{tnm\text{-}cca1} \not\Rightarrow \mathsf{tnm\text{-}cca2}$) *If there exists an* $\mathsf{tnm\text{-}cca1}$*-secure encryption scheme, then there exists an* $\mathsf{tnm\text{-}cca1}$*-secure encryption scheme that is not* $\mathsf{tnm\text{-}cca2}$*-secure.*

Theorem C.4 ($\mathsf{tnm\text{-}atk} \Rightarrow \mathsf{tind\text{-}atk}$) *If an encryption scheme is* $\mathsf{tnm\text{-}atk}$*-secure, then it is* $\mathsf{tind\text{-}atk}$*-secure, for* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$*.*

Theorem C.5 ($\mathsf{tind\text{-}cca2} \Rightarrow \mathsf{tnm\text{-}cca2}$) *If an encryption scheme is* $\mathsf{tind\text{-}cca2}$*-secure, then it is* $\mathsf{tnm\text{-}cca2}$*-secure.*

Theorem C.6 ($\mathsf{tcnm\text{-}atk} \Rightarrow \mathsf{tnm\text{-}atk}$) *If an encryption scheme is* $\mathsf{tcnm\text{-}atk}$*-secure, then it is* $\mathsf{tnm\text{-}atk}$*-secure, for* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$*.*

Theorem C.7 ($\mathsf{tnm\text{-}atk} \Rightarrow \mathsf{tind\text{-}pxx}$) *For any* $\mathsf{atk} \in \{\mathsf{cpa}, \mathsf{cca1}, \mathsf{cca2}\}$*, if an encryption scheme is* $\mathsf{tnm\text{-}atk}$*-secure, then it is* $\mathsf{tind\text{-}pxx}$*-secure, where*

$$\begin{aligned} \mathsf{atk} = \mathsf{cpa} &\quad\Rightarrow\quad \mathsf{pxx} = \mathsf{pa0} \\ \mathsf{atk} = \mathsf{cca1} &\quad\Rightarrow\quad \mathsf{pxx} = \mathsf{pa1} \\ \mathsf{atk} = \mathsf{cca2} &\quad\Rightarrow\quad \mathsf{pxx} = \mathsf{pa2} \end{aligned}$$

Theorem C.8 ($\mathsf{tind\text{-}pxx} \Rightarrow \mathsf{tcnm\text{-}atk}$) *For any* $\mathsf{pxx} \in \{\mathsf{pa0}, \mathsf{pa1}, \mathsf{pa2}\}$*, if an encryption scheme is* $\mathsf{tind\text{-}pxx}$*-secure, then it is* $\mathsf{tcnm\text{-}atk}$*-secure, where*

$$\begin{aligned} \mathsf{pxx} = \mathsf{pa0} &\quad\Rightarrow\quad \mathsf{atk} = \mathsf{cpa} \\ \mathsf{pxx} = \mathsf{pa1} &\quad\Rightarrow\quad \mathsf{atk} = \mathsf{cca1} \\ \mathsf{pxx} = \mathsf{pa2} &\quad\Rightarrow\quad \mathsf{atk} = \mathsf{cca2} \end{aligned}$$

D Digital Signature Schemes

A *signature scheme* SIG is a triple $(\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$ of algorithms, the first two being probabilistic, and all running in polynomial time (with a negligible probability of failing). $\mathsf{sig\_gen}$ takes as input 1^k and outputs a public key pair $(\mathsf{sig\_vk}, \mathsf{sig\_sk})$, i.e., $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$. $\mathsf{sig\_sign}$ takes a message m and a secret key sk as input and outputs a signature σ for m, i.e., $\sigma \leftarrow \mathsf{sig\_sign}(\mathsf{sig\_sk}, m)$. $\mathsf{sig\_verify}$ takes a message m, a public key $\mathsf{sig\_vk}$, and a candidate signature σ' for m as input and returns the bit $b = 1$ if σ' is a valid signature for m for the corresponding private key, and otherwise returns the bit $b = 0$. That is, $b \leftarrow \mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma')$. Naturally, if $\sigma \leftarrow \mathsf{sig\_sign}(\mathsf{sig\_sk}, m)$, then $\mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma) = 1$.

Security for signature schemes We specify existential unforgeability against adaptive chosen-message attacks [15] for a signature scheme $\mathsf{SIG} = (\mathsf{sig\_gen}, \mathsf{sig\_sign}, \mathsf{sig\_verify})$. A forger is given $\mathsf{sig\_vk}$, where $(\mathsf{sig\_vk}, \mathsf{sig\_sk}) \leftarrow \mathsf{sig\_gen}(1^k)$, and tries to forge signatures with respect to $\mathsf{sig\_vk}$. It is allowed to query a signature oracle (with respect to $\mathsf{sig\_sk}$) on messages of its choice. It succeeds if after this it can output a valid forgery (m, σ), where $\mathsf{sig\_verify}(\mathsf{sig\_vk}, m, \sigma) = 1$, but m was not one of the messages signed by the signature oracle. We say a forger (t, q, ϵ)-breaks a scheme if the forger runs in time $t(k)$ makes $q(k)$ queries to the signature oracle, and succeeds with probability at least $\epsilon(k)$. A signature scheme SIG is existentially unforgeable against adaptive chosen-message attacks if for all t and q polynomial in k, if a forger (t, q, ϵ)-breaks SIG, then ϵ is negligible in k.

In a *one-time* signature scheme, security is formulated as above except that the adversary may only query the signature oracle once, and we call it "existential unforgeability against chosen-message attacks," since the term "adaptive" only makes sense with multiple queries. We note that one-time signatures scheme can be made very efficient since they don't need public-key cryptographic operations [13]. In a *strong* one-time signature scheme [20], we require that a forger is not even able to produce a different valid signature on a message that was signed by the signature oracle. A strong one-time signature scheme can be constructed from any one-way function [20].

E The Universal Composability Framework

In more detail, the execution in the real-life model and the ideal process proceeds basically as follows. The environment $\mathcal{Z}$ drives the execution. It can provide input to a party P_i or to the adversary, $\mathcal{A}$ or $\mathcal{S}$. If P_i is given an input, P_i is activated. In the ideal process P_i simply forwards the input directly to $\mathcal{F}$ (this is the "direct forwarding" that we discussed in the introduction), which is then activated, possibly writing messages on its outgoing communication tape, and then handing activation back to P_i. In the real-life model, P_i follows its protocol, either writing messages on its outgoing communication tape or giving an output to $\mathcal{Z}$. Once P_i is finished, $\mathcal{Z}$ is activated again. If the adversary is activated, it follows its protocol, possibly giving output to $\mathcal{Z}$, and also either corrupting a party, or performing one of the following activities. If the adversary is $\mathcal{A}$ in the real-life model, it may deliver a message from the output communication tape of one honest party to another, or send a message on behalf of a corrupted party. If the adversary is $\mathcal{S}$ in the ideal process, it may deliver a message from $\mathcal{F}$ to a party, or send a message to $\mathcal{F}$. If a party or $\mathcal{F}$ receives a message, it is activated, and once it finishes, $\mathcal{Z}$ is activated

At the beginning of the execution, all participating entities are given the security parameter $k \in \mathbb{N}$ and random bits. The environment is also given an auxiliary input $z \in \{0,1\}^*$. At the end of the execution, the environment outputs a single bit. Let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing $\mathcal{Z}$'s output when interacting in the real-life model with adversary $\mathcal{A}$ and players running protocol π, with input z, security parameter k, and uniformly-chosen random tapes for all participating entities. Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble of random variables describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{S}$ and ideal functionality $\mathcal{F}$, with input z, security parameter k, and uniformly-chosen random tapes for all participating entities.

A protocol π *securely realizes* an ideal functionality $\mathcal{F}$ if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any auxiliary input, can tell with non-negligible advantage whether it is interacting with $\mathcal{A}$ and players running π in the real-life model, or with $\mathcal{S}$ and $\mathcal{F}$ in the ideal-process. More precisely, $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, where $\stackrel{c}{\approx}$ denotes *computational indistinguishability.* (In particular, this means that for any $d \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and for all inputs z, $|\Pr \mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z) - \Pr \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)| < k^{-d}$).

To formulate the composition theorem, one must introduce a hybrid model, a real-life model with access to an ideal functionality $\mathcal{F}$. In particular, this $\mathcal{F}$-hybrid model functions like the real-life model, but where the parties may also exchange messages with an unbounded number of copies of $\mathcal{F}$, each copy identified via a unique *session identifier* (*sid*). The communication between the parties and each one of these copies mimics the ideal process, and in particular the hybrid adversary does not have access to the contents of the messages. Let $\mathrm{HYB}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble of random variables describing the output of $\mathcal{Z}$, after interacting in the $\mathcal{F}$-hybrid model with protocol π. Let π be a protocol in the $\mathcal{F}$-hybrid model, and ρ a protocol that secures realizes $\mathcal{F}$. The composed protocol π^ρ is now constructed by replacing the first message to $\mathcal{F}$ in π by an invocation of a new copy of ρ, with fresh random input, the same *sid*, and with the contents of that message as input; each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of ρ, with the contents of that message as new input to ρ.

Canetti [5] proves the following composition theorem.

Theorem E.1 ([5]) *Let $\mathcal{F}$, $\mathcal{G}$ be ideal functionalities. Let π be an n-party protocol that securely realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model, and let ρ be an n-party protocol that securely realizes $\mathcal{F}$. Then protocol π^ρ securely realizes $\mathcal{G}$.*