Proofs of Cryptographic Primitives in CCSA Logic

M1 Internship Report

Jonathan Baumann
Supervisors: Guillaume Scerri, Théo Vignon

August 23, 2024

1. Introduction

Cryptographic protocols are integral to the security of all sorts of communications, e.g. online payments, messaging, or even digital rights management for streaming services. Concretely, a protocol is a distributed program that makes extensive use of cryptographic primitives such as encryption, signatures, and hash functions. These primitives are typically proven to fulfill certain security guarantees which can thus be assumed when designing a protocol. Nonetheless, primitives can only provide security if they are used properly and the protocol does not leak any secrets in any other way, as a flawed protocol can easily nullify any guarantees expected to be provided by the primitives.

As indicated by many real-world vulnerabilities [19, 24, 1], properly designing protocols is a very difficult task. Formal analyses of protocols are therefore essential; however, they are also complex and prone to errors [22, 8]. This has lead to the development of several formal approaches and automated proof techniques.

There are two main approaches to protocol verification, the symbolic model and the computational model. In the symbolic model, attacker capabilities are explicitly stated as an equational theory. This makes it very suitable for automated verification techniques [10, 21], at the risk of under-approximating adversary power. The computational model, on the other hand, models attackers as probabilistic polynomial-time Turing machines. While this ensures stronger guarantees and is more precise, as it can model any polynomial-time attacker, it also means that most security properties can only hold with *overwhelming probability*: in many cases, there will always be some possibility of failure. For example, an attacker could randomly guess a secret key, circumventing any encryption. However, the probability of this happening decreases exponentially with the length of the key, and is thus considered *negligible*.

Proofs in the computational model are done using reductions, where one shows how an efficient attacker against the protocol can be used to construct an efficient attacker against one of the primitives used. As it is assumed that no efficient attacker against the primitive can exist, it follows that there cannot be an efficient attacker against the protocol. Unfortunately, such proofs are hard to automate[1].

Thus, Bana and Comon-Lundh [7] have introduced another approach: the Computationally Complete Symbolic Attacker (CCSA). Based on first-order logic, and thus closer in proof techniques to a symbolic approach, it is nonetheless able to provide computational guarantees. It achieves this via the use of predicates whose semantics are defined with respect to polynomial-time Turing machines (PPTM), most notably the *indistinguishability predicate* $t_1 \sim t_2$. This predicate represents a situation where an attacker is presented with a value and tries to determine whether it was produced by t_1 or t_2. If no efficient adversary can do so, then t_1 and t_2 are considered to be indistinguishable.

[1] For more details, see e.g. the introduction of Dupressoir, Kohbrok, and Oechsner [15]

The CCSA approach requires some basic axioms about the predicates used, which in turn must still be proved using reductions in the computational model. However, with these axioms, one can then perform complex reasoning entirely within CCSA logic. Using first-order logic makes this approach especially suitable for implementation in a proof assistant, as demonstrated by Squirrel[5].

Despite still being relatively recent, CCSA has been successfully used to analyze some protocols [18, 13, 6].

1.1. The Research Problem

While CCSA was originally designed for cryptographic protocol analysis, there is nothing fundamentally preventing it from being applied to cryptographic primitives as well. Indeed, the distinction is not always clear, as some protocols like oblivious transfers or zero-knowledge proofs are often useful to treat as primitives in a larger protocol. On the other hand, encryption schemes can be complex enough to warrant analysis in a proof assistant. Further, it is desirable to analyze both protocols and the primitives they rely on in the same model and using the same tools in order to ensure that the security properties specified in the protocol can be met. Otherwise, there is a potential for errors in translating security properties between models, or even between different tools in the same model. Therefore, the goal of this internship was to investigate whether studying cryptographic primitives in CCSA is feasible in practice.

To this end, we first investigated KEM/DEM hybrid encryption schemes, in which a symmetric encryption scheme is used to encrypt large messages, but the key is then encrypted asymmetrically. This approach is very popular, as symmetric encryption is often much more computationally efficient. We studied how best to represent existing notions of security in CCSA, as well as whether existing results for the security of hybrid encryption schemes [17] translate to CCSA.

We then turned to oblivious transfers, which are small protocols that allow a receiver to receive exactly one message out of a set provided by the sender, without the sender learning which message was received. We investigated how best to represent an abstract oblivious transfer for use in other protocols, as well as how to state the relevant security properties in CCSA and whether they could be proved for some concrete implementations.

1.2. Contributions

Our investigation of hybrid encryption schemes produced an extensive set of security definitions in CCSA, almost all of which are new. In some cases, we provided multiple axioms suitable for different situations, and showed their equivalence. We found that restating game-based definitions as indistinguishability can highlight similarities that were previously obscured, thus providing intuition for how they relate to each other.

Using these security definitions, we were able to reproduce in CCSA the positive results of Herranz, Hofheinz, and Kiltz [17]. This also lead us to generalize some definitions and proof techniques, allowing them to be reused in many different situations.

For oblivious transfers, we provided a way to represent them abstractly, and demonstrated its usability with two examples. One example uses public-key encryption, but the other one builds directly on number-theoretic assumptions, and thus serves to demonstrate that CCSA can be used down to fundamental assumptions.

1.3. Metainformation

The work presented here has been performed in the context of a 5-month internship from February 26, 2024 to July 26, 2024 at the LMF (Laboratoire Méthodes Formelles) at CNRS, ENS-Paris-Saclay, supervised by Guillaume Scerri and his PhD student Théo Vignon.

The first three weeks of the internship were spent familiarizing myself with CCSA logic (I had of course started with this before; however, I realized that I had many questions which I needed answered first before I could properly understand the relevant papers) as well as game-based definitions and proofs. Also, perhaps more importantly, we still needed to decide which exact version of the logic to use (as

CCSA is still very recent, it is constantly evolving, and there are major differences between the logic in [5] and [2]).

After that, it was time to come up with cryptographic axioms for KEM/DEM encryption schemes, which took easily two weeks and many rejected drafts until they were sufficiently expressive, while also being sound and not overly complicated (although this time also included sketches for the proofs I would do later, which uncovered many flaws, and the axioms would also still be refinement later on).

In early April, my supervisors invited me to join them for the Annual Meeting of the WG "Formal Methods in Security"[2], which I was very grateful for, as it allowed me to gain a broader view of the current work in this field.

Also in April, I took a week-long break to be able to attend the ICPC World Finals in Luxor[3].

The end of April and early May were spent looking at all the side conditions in the axioms which had not yet received a formal definition. I found a general construction and ensured that it satisfies the required properties for the proofs. This was followed by an excursion into the work of Baelde et al. [4] and Baelde, Koutsos, and Sauvage [3], which did not make it into this report.

The remaining time was spent investigating oblivious transfers. Unfortunately, I did not have enough time at the end of the internship to use oblivious transfers within a larger protocol, so this report only demonstrates two different implementations, but not how they can be used in a larger context.

In July, my supervisors invited me to attend CSF 2024[4], where Théo was presenting his paper on CCSA logic with concrete bounds [4]. I was clearly the youngest attendee, everyone else being at least a year into their PhD, but I greatly enjoyed the talks and had many interesting discussions with other attendees.

2. Preliminaries

2.1. Cryptographic Games

Cryptographic security properties are commonly defined in terms of an *attack game* between an attacker and a challenger, both of which are probabilistic processes. The notion of security is defined by a particular event S, the probability of which should be very close to some target value t. This target value is commonly 0, $\frac{1}{2}$, or the probability of an event in a different game [23].

As an example, consider the definition of indistinguishability under chosen ciphertext attack for public key encryption [17], which we introduce in detail in Section 3.2. The challenger is described by the following process, where $\mathcal{A}_1$ and $\mathcal{A}_2$ denote two separate phases of the attacker:

$$
\begin{array}{l}
\mathbf{Exp}^{pke-ind-atk-b}_{\mathsf{PKE},\mathcal{A}}(\eta) \\
\hline
(pk, k) \leftarrow\!\$\ \mathsf{PKE.Kg}(1^\eta) \\
(St, m_0, m_1) \leftarrow\!\$\ \mathcal{A}_1^{\mathsf{PKE.Dec}(k,\cdot)}(pk) \\
C^* \leftarrow\!\$\ \mathsf{PKE.Enc}(pk, m_b) \\
b' \leftarrow\!\$\ \mathcal{A}_2(C^*, St) \\
\textbf{return}\ b'
\end{array}
$$

Here, security is defined by the event $\mathbf{Exp}^{pke-ind-atk-1}_{\mathsf{PKE},\mathcal{A}}(\eta) = 1$, while the target value is the probability that $\mathbf{Exp}^{pke-ind-atk-0}_{\mathsf{PKE},\mathcal{A}}(\eta) = 1$.

For clarity, we use $\leftarrow\!\$$ to denote randomized computation and $\leftarrow$ for deterministic computation. $\leftarrow\!\$$ can also be used to draw uniformly at random from a set.

The notation $\mathcal{A}_1^{\mathsf{PKE.Dec}(k,\cdot)}(pk)$ indicates that the attacker is given pk as an input, and has access to an *oracle* that computes $\mathsf{PKE.Dec}(k, c)$ for any attacker-chosen ciphertext c. Games often give the attacker

[2] https://gtmfs2024.sciencesconf.org/
[3] https://news.icpc.global/wf2023/
[4] https://csf2024.ieee-security.org/

access to oracles in order to provide limited access to specific operations that the attacker would otherwise not be able to perform. In this case, it allows the attacker to decrypt without knowing the secret key, but only in the first phase.

Computations performed by an oracle do not count towards the attackers runtime, but querying an oracle and reading its answer do require time. Thus, the attackers runtime puts an upper bound on the number of oracle queries, for example, a polynomial-time attacker can make at most a polynomial (in η) number of queries.

Cryptographic primitives, and therefore also games, are parametric in a *security parameter* η, which regulates e.g. the lengths of keys and hashes. For any game G and attacker $\mathcal{A}$, the difference it can achieve between the probability of event S_η in $G(\eta)$, called its *advantage*, is thus a function of η:

$$\mathsf{Adv}^{\mathrm{G}}_{\mathcal{A}}(\eta) = |\mathtt{Pr}[S_\eta] - t|$$

For the example above, the advantage is defined as

$$\mathsf{Adv}^{\mathrm{pke-ind-atk}}_{\mathsf{PKE},\mathcal{A}}(\eta) = \left|\mathtt{Pr}[\mathbf{Exp}^{pke-ind-atk-1}_{\mathsf{PKE},\mathcal{A}}(\eta) = 1] - \mathtt{Pr}[\mathbf{Exp}^{pke-ind-atk-0}_{\mathsf{PKE},\mathcal{A}} = 1]\right|$$

Commonly, the security property defined by G is satisfied if $\mathsf{Adv}^{\mathrm{G}}_{\mathcal{A}}(\eta)$ is *negligible* for any polynomial-time attacker $\mathcal{A}$.

For a detailed introduction into games and game-based proofs, see Shoup [23].

2.2. Negligibility

A function $f(\eta)$ is *negligible*, written $f(\eta) \in \mathsf{negl}(\eta)$, if it grows asymptotically smaller than the inverse of any polynomial. This is a useful notion in cryptography since, as long as the cryptographic primitives and protocols used have runtime polynomial in η, the adversary's advantage can be decreased *exponentially* with only a *polynomial* increase in runtime.

2.3. CCSA Logic

CCSA logic is a first-order logic on higher-order terms, specifically designed for cryptographic protocols and their security properties. It is designed around a view of protocol executions as their traces, using a folding method to translate protocols to inductively defined sets of terms. This suggests that proofs of primitives at the term level should also be possible, which we explore in this work.

This section aims to give an overview of the parts of the logic necessary to understand our work, but it will not go into detail on other aspects. For a full description, we refer to Baelde, Koutsos, and Lallemand [2].

2.3.1. Terms, Types and Environments

CCSA uses simply-typed λ-terms with a set of variables $\mathcal{X}$ and a set of base types $\mathbb{B}$. The base types include `bool` and `unit` as well as τ_{msg}, which represents bitstrings of arbitrary length, and is used as the type of plain- and ciphertexts as well as of keys. In contrast to Baelde, Koutsos, and Lallemand [2], we allow product types $\tau_1 \times \tau_2$ and option types $\mathtt{opt}\ \tau$[1] in addition to function types $\tau_1 \to \tau_2$.

In addition to the standard constants and operators for each type, we also assume an equality operator $=_\tau : \tau \times \tau \to \mathtt{bool}$ for each type τ. Finally, we introduce $\mathsf{let}\ x = t\ \mathsf{in}\ t'$ as syntactic sugar for $(\lambda x.t')t$.

Names

Terms can access randomness via the use of *names*. These are special functions of type $\tau_0 \to \tau_1$ that provide access to randomness: A name assigns to each value of τ_0 (often called the index) a value of τ_1, sampled randomly according to some specified probability distribution. Importantly, these samples are

[1] Option types will mostly be handled implicitly, i.e. we do not explicitly check whether a value is present. Any operation on $\perp$ (which represents the absence of a value) is assumed to yield $\perp$.

independent of each other and of other names. The index type τ_0 must be finite, but is allowed to grow with η.

Terms are evaluated in environments $\mathcal{E}$, which consist of *definitions* of the form $x : \tau = t$ and *declarations* $x : \tau$. Definitions must be well-typed in the sense that for each Definition $x : \tau = t$, t must itself be well-typed in $\mathcal{E}$. This allows for recursive and mutually recursive definitions.

Interpretation

A *model* of an environment $\mathcal{E}$ is a *term structure* $\mathbb{M} : \mathcal{E}$, which

- associates to each base type τ_b, for all η, a non-empty interpretation domain $\mathbb{M}_{\tau_b}(\eta)$ and an injective mapping to bitstrings. For example, the type `bool` always has the interpretation domain $\{0, 1\}$, with maps to bitstrings in the obvious way: $0 \mapsto 0$ and $1 \mapsto 1$. One could also introduce a type of integers, where the interpretation domain would be $\mathbb{N}$, and each number would be mapped to its binary representation. Other types can be restricted to *fixed* (for all η) or *finite* (but allowed to vary with η) interpretation domains using *labels*.
- provides, for each η, a finite set $\mathbb{T}_{\mathbb{M},\eta}$ of *random tapes* (bitstrings) of equal length. Individual tapes ρ are divided into two parts ρ_h and ρ_a, where ρ_h is used for the evaluation of terms, and ρ_a is available to be used by the attacker.
- provides a partial mapping $\sigma_{\mathbb{M}}$ that associates to each variable in $\mathcal{E}$, for each η and random tape ρ in $\mathbb{T}_{\mathbb{M},\eta}$, an element of the appropriate interpretation domain. In particular, for a name $n : \tau_0 \to \tau_1$, $\sigma_{\mathbb{M}}(\mathcal{E})(\eta)$ maps random tapes ρ to functions f such that for all indices i and j, $f(i)$ and $f(j)$ are independent, and $f(i)$ has the desired probability distribution over ρ. Note that the index type τ_0 must be finite so that $f(i)$ and $f(j)$ can be independent for all i and j.

The semantics of types $[\![\tau]\!]^{\eta}_{\mathbb{M}}$ and of terms $[\![t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$ are defined recursively, with the semantics of base types given by $[\![\tau_b]\!]^{\eta}_{\mathbb{M}} := \mathbb{M}_{\tau_b}(\eta)$, and the semantics of function, product and option types defined as usual. The semantics of variables is $[\![x]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} := \sigma_{\mathbb{M}}(x)(\eta)(\rho)$ with $\rho \in \mathbb{T}_{\mathbb{M},\eta}$. Finally, models must satisfy $[\![x_i]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = [\![\lambda x.t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$ for all η, ρ and $x_i = \lambda x.t \in \mathcal{E}$.

2.3.2. Proof System

Formulas in CCSA consist of conjunction, disjunction, implication and negation ($\tilde{\wedge}, \tilde{\vee}, \tilde{\Longrightarrow}, \tilde{\neg}$), quantification over variables ($\tilde{\forall}(x : \tau).F, \tilde{\exists}(x : \tau).F$) and, most importantly, predicates $p(t_1, \cdots, t_n)$ on terms. The semantics $[\![F]\!]_{\mathbb{M}:\mathcal{E}}$ is defined in a standard way; some important predicates and their semantics will be given below.

CCSA logic features judgements $\mathcal{E}; \Theta \vdash F$, where $\mathcal{E}$ is an environment and Θ a set of formulas (the hypotheses)[2]. Such a judgement is *valid* if all models of $\mathcal{E}$ satisfy $\tilde{\wedge}\Theta \tilde{\Longrightarrow} F$.

Probabilistic Predicates

The most important feature of CCSA logic are its standard predicates, which enable cryptographic reasoning. We present here *indistinguishability*, *overwhelming truth*, and *constancy*.

Indistinguishability of two terms t_1 and t_2, written $t_1 \sim t_2$, states that for any probabilistic polynomial-time Turing machine $\mathcal{A}$, the advantage at distinguishing t_1 and t_2 is negligible:

$$[\![t_1 \sim t_2]\!]_{\mathbb{M}:\mathcal{E}} := \forall \mathcal{A} \in \text{PPTM}, \text{Adv}^{\eta}_{\mathbb{M}:\mathcal{E}}(\mathcal{A} : t_1 \sim t_2) \in \mathsf{negl}(\eta)$$
$$\text{Adv}^{\eta}_{\mathbb{M}:\mathcal{E}}(\mathcal{A} : t_1 \sim t_2) := |\mathtt{Pr}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}[\mathcal{A}1^{\eta}, [\![t_1]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}, \rho_a] - \mathtt{Pr}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}[\mathcal{A}(1^{\eta}, [\![t_2]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}, \rho_a)]|$$

Since indistinguishability is defined up to negligible probability, it is transitive up to a polynomial number of steps. CCSA thus has axioms for transitivity and symmetry (see Appendix A.1), as well as one for the concept of *freshness*, which states that different random samplings using the same probability distribution are indistinguishable (see Appendix A.2).

[2] Baelde, Koutsos, and Lallemand [2] call these "global judgements" and also feature "local judgements", which reason about overwhelming truth of boolean formulas. However, we omit those here – while we do use overwhelming truth, local judgements are not necessary to understand our work.

Overwhelming truth $[\phi]$ of an expression ϕ (of type `bool`) states that ϕ is only false with negligible probability:

$$[\![[\phi]]\!]_{\mathbb{M}:\mathcal{E}} := \mathtt{Pr}_{\rho \in \mathbb{T}_{\mathbb{M},\eta}}[\neg [\![\phi]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}] \in \mathsf{negl}(\eta)$$

This can be used to rewrite terms that are equal with overwhelming probability ($[t_1 = t_2]$), including β-equivalent terms (see Appendix A.3).

Constancy of a term t, written $\mathrm{const}(t)$, specifies that the value of t is fixed. It is thus neither random, nor can it vary with η. This is often used for indices.

$$[\![\mathrm{const}(t)]\!]_{\mathbb{M}:\mathcal{E}} := \exists c \in (\bigcap_{\eta \in \mathbb{N}} [\![\tau]\!]^{\eta}_{\mathbb{M}}), \forall \eta \in \mathbb{N}, \forall \rho \in \mathbb{T}_{\mathbb{M},\eta}, [\![t]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = c$$

Probabilistic Polynomial-Time Computability

Since this work is concerned with polynomial-time attackers, we commonly require that terms must be computable in polynomial time. This is specified using judgements $\mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} (t)$ with their own derivation rules (see Baelde, Koutsos, and Lallemand [2]). Note, however, that this is not a complete characterization of polynomial-time computability; in fact, it requires computation to follow the same recursive structure as the semantics, as this is necessary for the soundness proofs of cryptographic axioms.

3. KEM/DEM Hybrid Encryption

Encryption schemes can fundamentally be split into two categories: *symmetric* encryption, where the same key is used for encryption and decryption, and *asymmetric* encryption, where the two operations require different keys. Symmetric encryption schemes are often very computationally efficient, but they require that a shared secret must be established between the parties involved in the communication. Asymmetric schemes, on the other hand, allow that the encryption key can be made public, and are thus also called *public-key encryption schemes* (PKE). However, these schemes have much higher computational costs.

The KEM/DEM paradigm, first formalized by Cramer and Shoup [14], is a method to construct efficient public key encryption schemes by combining symmetric and asymmetric schemes. In this paradigm, a *key encapsulation mechanism* (KEM) generates and asymmetrically encrypts a single-use symmetric key, which is then used to encrypt the message using a *data encapsulation mechanism* (DEM).

An important question with such a modular approach is how the security of the resulting hybrid scheme relates to the security of the individual components. This has been studied extensively by Herranz, Hofheinz, and Kiltz [17], whose results we reproduce in CCSA logic here.

3.1. Formal Definition

We first restate the definitions from Herranz, Hofheinz, and Kiltz [17] before making adjustments for CCSA. For this chapter, only definitions in CCSA constitute our work.

3.1.1. Definitions by Herranz, Hofheinz, and Kiltz [17]

A public-key encryption scheme consists of three functions $\mathsf{PKE.Kg}, \mathsf{PKE.Enc}$ and $\mathsf{PKE.Dec}$. Key generation is performed by $(pk, sk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\eta)$ for security parameter η, $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(pk, m)$ denotes randomized encryption with the public key pk and $\{m, \bot\} \leftarrow \mathsf{PKE.Dec}(sk, c)$ denotes decryption with the secret key sk (returning $\bot$ if decryption fails).

Similarly, a key encapsulation mechanism consists of three functions $\mathsf{KEM.Kg}, \mathsf{KEM.Enc}$ and $\mathsf{KEM.Dec}$ and a data encapsulation mechanism consists of $\mathsf{DEM.Kg}, \mathsf{DEM.Enc}$ and $\mathsf{DEM.Dec}$. However, here, $(k, c) \leftarrow\!\!\$\ \mathsf{KEM.Enc}(pk)$ denotes the simultaneous generation and encryption of a symmetric key k. Since DEMs are

symmetric, $k \leftarrow\!\!\$\ \mathsf{DEM.Kg}(1^\eta)$ generates only a single symmetric key, which is then used for both encryption and decryption.

A DEM and a KEM are compatible if, for all η, the key space produced by $\mathsf{KEM.Enc}(pk)$ (with a key pk obtained from $\mathsf{KEM.Kg}(1^\eta)$) and that produced by $\mathsf{DEM.Kg}(1^\eta)$ are equal. In this case, they can be combined into a PKE as follows:

$\mathsf{PKE.Kg}(1^\eta)$	$\mathsf{PKE.Enc}(pk, m)$	$\mathsf{PKE.Dec}(sk, (c_1, c_2))$
$(pk, k) \leftarrow\!\!\$\ \mathsf{KEM.Kg}(1^\eta)$	$(sk, c_1) \leftarrow\!\!\$\ \mathsf{KEM.Enc}(pk)$	$sk \leftarrow \mathsf{KEM.Dec}(k, c_1)$
return (pk, k)	$c_2 \leftarrow\!\!\$\ \mathsf{DEM.Enc}(k, m)$	$m \leftarrow \mathsf{DEM.Dec}(sk, c_2)$
	return (c_1, c_2)	**return** m

3.1.2. Definitions in CCSA

To be able to use these definitions in CCSA, we need to make the following adjustments:

- Terms in CCSA cannot be probabilistic except via names. For randomized encryption, we therefore need to provide explicit access to random samplings via a suitable name, which we will commonly denote r. The return type of this name is τ_{msg}, which allows for bitstrings of arbitrary length, and the necessary length can be specified via the name's probability distribution. Note, however, that this means that our cryptographic axioms only apply if a name of the same probability distribution is used to supply randomness, and that we need to ensure different samplings are used when encrypting different messages.
- The use of a probabilistic function for key generation would be needlessly complicated in CCSA. Instead, we use a name k that is equally distributed over the key space for the secret key, and allow the public key to be derived from the secret key via a function pk.

Note that we generally type everything, including keys and randomness, as τ_{msg}. This may seem imprecise at first, but it is necessary if we want to be able to encrypt and decrypt keys (as is necessary in the KEM/DEM construction) and use the result in further operations.

For clarity, however, we may sometimes add superscripts to τ_{msg} to indicate the role a specific message plays when specifying function types.

We therefore obtain the following signatures for KEM and DEM schemes in CCSA:

$$\text{KEM.Enc} : \tau_{msg}^{\text{pubkey}} \times \tau_{msg}^{\text{rand}} \to \tau_{msg}^{\text{symkey}} \times \tau_{msg}^{\text{enc}} \qquad \text{DEM.Enc} : \tau_{msg}^{\text{symkey}} \times \tau_{msg} \times \tau_{msg}^{\text{rand}} \to \tau_{msg}$$
$$\text{KEM.Dec} : \tau_{msg}^{\text{privkey}} \times \tau_{msg}^{\text{enc}} \to \mathtt{opt}\ \tau_{msg}^{\text{symkey}} \qquad \text{DEM.Dec} : \tau_{msg}^{\text{symkey}} \times \tau_{msg} \to \mathtt{opt}\ \tau_{msg}$$

These can be combined into a PKE as follows:

$\mathrm{PKE.Enc}(\mathrm{pk}(k\ i), m, (r_1\ j, r_2\ j))$	$\mathrm{PKE.Dec}(k\ i, (c_1, c_2))$
$\mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ i), r_1\ j)$	$\mathsf{let}\ sk = \mathrm{KEM.Dec}(k\ i, c_1)$
$\mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ j))$	$\mathsf{in}\ \mathrm{DEM.Dec}(sk, c_2)$

Of course, an important property of any encryption scheme is that it is *correct*, i.e. that encrypting and decrypting a message yields the same message. In CCSA, we can state this the following way (here for the example of KEM):

KEM-CORRECT

$$\mathcal{E}; \Theta \vdash \tilde{\forall}(t_k, t_r).\ [\mathrm{KEM.Dec}(k\ t_k, \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r\ t_r))) = \pi_1(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r\ t_r))]$$

3.2. Security Definitions

We will consider two notions of security: *indistinguishability*, which requires that an attacker cannot distinguish between two encrypted messages, and *nonmalleability*, which enforces that an attacker cannot

modify the ciphertext in any meaningful way. Both notions apply to PKE, KEM and DEM schemes in slightly different ways.

These notions are further refined according to the adversary's capabilities. A CPA attacker (chosen plaintext attack) will have the ability to encrypt arbitrary plaintexts. A CCA1 attacker ([non-adaptive] chosen ciphertext attack) is further able decrypt chosen ciphertexts so long as they do not depend on the challenge ciphertext, and a CCA2 attacker (adaptive chosen ciphertext attak) does not have this last restriction and can decrypt any ciphertexts that are not the challenge ciphertext.

3.2.1. Indistinguishability

The common cryptographic definition of indistinguishability for PKEs uses the following game [17]:

$$\begin{array}{l}\underline{\mathbf{Exp}^{pke-ind-atk-b}_{\mathsf{PKE},\mathcal{A}}(\eta)}\\ (pk,k) \leftarrow\$\ \mathsf{PKE.Kg}(1^\eta)\\ (St, m_0, m_1) \leftarrow\$\ \mathcal{A}_1^{\mathtt{DEC}_1(\cdot)}(pk)\\ C^* \leftarrow\$\ \mathsf{PKE.Enc}(pk, m_b)\\ b' \leftarrow\$\ \mathcal{A}_2^{\mathtt{DEC}_2(\cdot)}(C^*, St)\\ \textbf{return } b'\end{array}$$

C^* is called the *challenge ciphertext*. The oracles $\mathtt{DEC}_1$ and $\mathtt{DEC}_2$ are instantiated differently depending on the type of attacker: For CPA, both oracles do nothing, as the attacker may never decrypt any ciphertexts. For CCA1, however, $\mathtt{DEC}_1$ is instantiated with $\mathsf{PKE.Dec}(k,\cdot)$ while $\mathtt{DEC}_2$ does nothing, and for CCA2, $\mathtt{DEC}_2$ is further instantiated with a modified $\mathsf{PKE.Dec}(k,\cdot)$ that fails when attempting to decrypt C^*.

The advantage of an adversary $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathrm{pke-ind-atk}}_{\mathsf{PKE},\mathcal{A}}(\eta) = \left|\mathtt{Pr}[\mathbf{Exp}^{pke-ind-atk-1}_{\mathsf{PKE},\mathcal{A}}(\eta) = 1] - \mathtt{Pr}[\mathbf{Exp}^{pke-ind-atk-0}_{\mathsf{PKE},\mathcal{A}} = 1]\right|$$

Intuitively, the attacker gets to choose two messages m_0 and m_1 and wins this game if it can correctly guess which one was encrypted. If no attacker can do so with more than a negligible advantage, the PKE is secure.

The definitions for KEMs and DEMs are very similar, but with slightly different games[17]:

$$\begin{array}{l}\underline{\mathbf{Exp}^{kem-ind-atk-b}_{\mathsf{KEM},\mathcal{A}}(\eta)}\\ (pk,k) \leftarrow\$\ \mathsf{KEM.Kg}(1^\eta)\\ St \leftarrow\$\ \mathcal{A}_1^{\mathtt{DEC}_1(\cdot)}(pk)\\ K_0^* \leftarrow\$\ \mathsf{KeySp}(\eta)\\ (K_1^*, C^*) \leftarrow\$\ \mathsf{KEM.Enc}(pk)\\ b' \leftarrow\$\ \mathcal{A}_2^{\mathtt{DEC}_2(\cdot)}(C^*, St, K_b^*)\\ \textbf{return } b'\end{array}$$

$$\begin{array}{l}\underline{\mathbf{Exp}^{dem-ind-atk-b}_{\mathsf{DEM},\mathcal{A}}(\eta)}\\ K \leftarrow\$\ \mathsf{DEM.Kg}(1^\eta)\\ (St, m_0, m_1) \leftarrow\$\ \mathcal{A}_1^{\mathtt{ENC}_1(\cdot),\mathtt{DEC}_1(\cdot)}(1^\eta)\\ C^* \leftarrow\$\ \mathsf{DEM.Enc}(pk, m_b)\\ b' \leftarrow\$\ \mathcal{A}_2^{\mathtt{ENC}_2(\cdot),\mathtt{DEC}_2(\cdot)}(C^*, St)\\ \textbf{return } b'\end{array}$$

For key encapsulation mechanisms, the main difference is that they do not encrypt any messages. Thus, instead of letting the attacker choose, one key is generated by the KEM and another key is randomly sampled from the key space. Then, the attacker is given one of those keys together with the ciphertext and needs to determine whether they match.

For data encapsulation mechanisms, the difference is that the adversary now needs access to an encryption oracle. This is not necessary for KEM and PKE, since the attacker can perform encryption itself using the public key. For CPA, CCA1 and CCA2 attackers, this oracle is just $\mathsf{DEM.Enc}(K,\cdot)$, however, there are actually two more notions which apply specifically for single-use keys: OT (one-time) attackers do not have access to any encryption or decryption oracles and OTCCA (one-time chosen ciphertext) are not able to encrypt, but can attempt to decrypt modified ciphertexts.

CCSA Axioms for PKE Indistinguishability

Since the game-based definitions of indistinguishability essentially gives two situations which no adversary can distinguish with more than a negligible advantage, we can easily state a corresponding axiom in CCSA if we can accurately describe the two experiments in the logic.

A first change we will make is that, instead of letting the adversary choose two messages, we only use one message m and the message $0^{|m|}$, which has the same length, but consists only of zeroes. This will simplify the presentation of the axiom, and we can easily obtain a version with two messages using transitivity.

Another change is that in the CCSA version, it does not make sense to let the attacker choose the message. In the computational model, this is done so that the attacker can choose the messages that give it the greatest advantage. If that advantage is still negligible, however, it must be negligible for all messages, so our axiom will be most flexible if it allows any message to be used.

To translate the experiment into CCSA terms, note that $\mathcal{A}_2$ essentially computes a function whose arguments are the ciphertext and the internal state after $\mathcal{A}_1$. This leads to the following representation:

$$(\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r))$$

where C represents $\mathcal{A}_2$, and $\vec{a}$ captures any terms that may have been computed by $\mathcal{A}_1$[1]. Note that C does not need to be of type `bool`: if an attacker can produce any intermediate results which later allow it to distinguish between the two experiments, this is sufficient, and the indistinguishability predicate will handle the rest. This will make the axiom more flexible.

Thus, the conclusion of our axiom will be

$$\begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r))\end{array}$$

However, we can only guarantee this under the following restrictions:

- $C, \vec{a}$ and m must be polynomial-time computable
- the indices t_k and t_r must be polynomial-time computable and not random
- the random samplings $r_1\ t_r, r_2\ t_r$ must not occur anywhere in $m, \vec{a}$ or C
- the secret key $k\ t_k$ must not occur in $m, \vec{a}$ or C except as $\mathrm{pk}(k\ t_k)$

The first two restrictions can be enforced by requiring $\mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} C, \vec{a}, t_r, m$, $\mathcal{E}, \Theta \vdash \mathrm{const}(t_r)$ and $\mathcal{E}, \Theta \vdash \mathrm{const}(t_k)$. For the remaining conditions, we will use formulas $\phi^{\mathrm{guarded}\ k, t_k}_{\mathrm{pk}(k\ t_k)}(C, \vec{a}, m)$ and $\phi^{r_1, t_r}_{\mathrm{fresh}}(C, \vec{a}, m), \phi^{r_2, t_r}_{\mathrm{fresh}}(C, \vec{a}, m)$, which we will introduce in Appendix C.1.

Thus, the final axiom looks as follows:

PKE-IND-CPA

$$\frac{\mathcal{E}, \Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} C, \vec{a}, t_k, t_r, m \qquad \mathcal{E}, \Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E}, \Theta \vdash [\phi^{r_1, t_r}_{\mathrm{fresh}}(C, \vec{a}, m)] \qquad \mathcal{E}, \Theta \vdash [\phi^{r_2, t_r}_{\mathrm{fresh}}(C, \vec{a}, m)] \qquad \mathcal{E}, \Theta \vdash [\phi^{\mathrm{guarded}\ k, t_k}_{\mathrm{pk}(k\ t_k)}(C, \vec{a}, m)]}{\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r))\end{array}}$$

Note that in this axiom, the same random sampling $(r_1\ t_r, r_2\ t_r)$ is used both in the encryption of m and in the encryption of $0^{|m|}$. This may not be desirable in some use cases, however, the random sampling can easily be changed in another transitive step since $r_1\ t_r$ and $r_2\ t_r$ are fresh.

To obtain a similar axiom for IND-CCA1, we need to allow the attacker to perform decryption operations with the key $k\ t_k$ when computing $\vec{a}$. We could do this by explicitly providing the attacker with a decryption function, however, it will be easier and more flexible to say that $k\ t_k$ may be used in $\vec{a}$ as long as it is

[1] For IND-CPA, this distinction is actually not necessary, since $\mathcal{A}_1$ does not have any more capabilities than $\mathcal{A}_2$. However, this is not the case for CCA1 and CCA2.

guarded by $\mathrm{PKE.Dec}(k\ t_k, \cdot)$:[2]

PKE-IND-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r))\end{array}}$$

Note that here, as well as in all following axioms, we assume that $\vec{a}$ contains only terms of order 0. Thus, it may not contain a term such as $(\lambda\ x.\mathrm{PKE.Dec}(k\ t_k, x))$, which would allow decryption in C.

For CCA2, we can allow decryption in C in a similar way, however, we must ensure that this is not used to decrypt the challenge ciphertext. Therefore, we use the following guarding term: $\mathsf{if}\ t = c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, t)$ (where c is the variable to which the ciphertext is bound in C). This yields the following axiom:

PKE-IND-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,t)}^{\mathrm{guarded}\ k,t_k}(C)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r))\end{array}}$$

This version of the axiom is likely preferred in practice, however, it will not be suitable when we reproduce the results of Herranz, Hofheinz, and Kiltz [17] in Section 3.3, as we will need to rewrite in the guarding expression. Therefore, we also give an alternative version, where guarded decryption is explicitly provided as a function. These two versions of the axiom are in fact equivalent, as we will see in Appendix D.

PKE-IND-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), m, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a} \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = \\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k), 0^{|m|}, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array}}$$

CCSA Axioms for KEM Indistinguishability

The axioms for key encapsulations mechanisms follow largely the same design choices as those for PKE. We will therefore only present the case for CCA1 here; the full set of axioms can be found in Appendix B.1.1.

KEM-IND-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r,t_r}(C,\vec{a})] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a})]\end{array}}{\mathcal{E},\Theta \vdash (\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r\ t_r) \sim \begin{array}{c}(\lambda\ \vec{v}\ (sk,c).C)\ \vec{a} \\ (sk^*(), \pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r\ t_r))\end{array}}$$

A clear difference compared to PKE is that KEM.Enc does not encrypt a message. Instead, it produces a key-ciphertext pair, which is given to the attacker (i.e. passed to C) on the left-hand side. On the right-hand side, the generated key is discarded instead, and we sample a fresh key[3] using the name sk^* (which is assumed to have the same probability distribution and index type `unit`). Since this key must be independent of anything else, we require that sk^* () does not occur anywhere else, which we enforce with $\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})$.

[2] The IND-CCA1 axiom is the only axiom presented here that's also commonly found in literature, however, it is often weaker as it does not allow for any operations on the ciphertext (as we do with C). The only version we've found that also achieves this is by Baelde et al. [4]

[3] Note that it would not be possible here to choose a constant key (similar to the message $0^{|m|}$ for PKE), since indistinguishability is defined slightly differently: here, the key changes while the ciphertext remains the same, whereas for PKE, the message remains fixed and the ciphertext changes.

CCSA Axioms for DEM Indistinguishability

Data encapsulation mechanisms also only require a few modifications compared to the PKE axioms. Most importantly, since a symmetric key is used, there is no pk function. Where previously, the key $k\ t_k$ was allowed to appear if guarded by pk, we now need to use $\mathrm{KEM.Enc}(k\ t_k, \cdot, \cdot)$ as the guarding expression instead (except for OT and OTCCA, where the attacker is not allowed to encrypt). However, we also need to be careful with the randomness used for these encryptions. In the computational model, these would be performed by a randomized oracle, which does not allow the attacker to influence its random sampling. Therefore, for a sound axiom, we need to ensure that the attacker is not allowed to perform encryption in a way that would not be possible using the oracle. In particular, this means that the attacker must not be allowed to encrypt different messages with the same randomness, which we enforce using a suitable formula $\phi^{k,t_k}_{\mathrm{dem-rand}}$ introduced in Appendix C.2.

This yields the following axiom for IND-CCA1; the others can be found in Appendix B.2.1.

DEM-IND-CCA1

$$\frac{\begin{array}{c}\mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} C, \vec{a}, t_k, t_r, m \\ \mathcal{E}, \Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E}, \Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E}, \Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C, \vec{a}, m)] \qquad \mathcal{E}, \Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C, \vec{a}, m)] \\ \mathcal{E}, \Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C)] \qquad \mathcal{E}, \Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a}, m)]\end{array}}{\mathcal{E}, \Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k, m, r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k, 0^{|m|}, r\ t_r)}$$

3.2.2. Nonmalleability

Another important security property we investigated is *nonmalleability*. Stronger than indistinguishability, it roughly states that the attacker can not modify the ciphertext in any meaningful way (where a modification is considered meaningful if its decryption is related to the original plaintext in some attacker-known way).

For PKEs, nonmalleability is defined using the following game [17]:

$\mathbf{Exp}^{pke-nm-atk-b}_{\mathsf{PKE},\mathcal{A}}(\eta)$

$(pk, k) \leftarrow\!\$\ \mathsf{PKE.Kg}(1^\eta)$

$(St, \mathcal{M}) \leftarrow\!\$\ \mathcal{A}_1^{\mathtt{DEC}_1(\cdot)}(pk)$

$m_0, m_1 \leftarrow\!\$\ \mathcal{M}$

$C^* \leftarrow\!\$\ \mathsf{PKE.Enc}(pk, m_1)$

$R, \mathbf{C} \leftarrow\!\$\ \mathcal{A}_2^{\mathtt{DEC}_2(\cdot)}(C^*, St)$

$\mathbf{M} \leftarrow \mathsf{PKE.Dec}(k, \mathbf{C})$

return $C^* \notin \mathbb{C} \wedge R(m_b, \mathbb{M})$

Like with indistinguishability, we obtain different versions for CPA, CCA1 and CCA2 depending on the instantiation of the decryption oracles. The advantage of an attacker $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathrm{pke-nm-atk}}_{\mathsf{PKE},\mathcal{A}}(\eta) = \left|\mathtt{Pr}[\mathbf{Exp}^{pke-nm-atk-1}_{\mathsf{PKE},\mathcal{A}}(\eta) = 1] - \mathtt{Pr}[\mathbf{Exp}^{pke-nm-atk-0}_{\mathsf{PKE},\mathcal{A}} = 1]\right|$$

and we consider a PKE to be secure if this advantage is negligible.

We omit here the definitions for KEM and DEM and refer to Herranz, Hofheinz, and Kiltz [17] and Nagao, Manabe, and Okamoto [20]. Corresponding axioms can be found in Appendix B.

To translate this definition into CCSA, we first realize that we can use an equivalent game where the message passed to R is fixed, but the message that is encrypted is randomized. This allows us to once again only use one message m, but vary between its encryption and the encryption of $0^{|m|}$. Another important realization is that $\mathcal{A}_2^{\mathtt{DEC}}$ and R actually have the same capabilities: Even in the CCA2 case, the attacker could precompute any oracle queries it would like to perform in R in $\mathcal{A}_2^{\mathtt{DEC}}$ and hardcode the results (including queries that depend on R's first argument, since there are only two options)[4]. We will thus treat R as a third phase of the attacker, which has the same capabilities of the second. As a final

[4]This is not the case for KEM nonmalleability, see Appendix B.1.2

change, we will refuse to decrypt the challenge ciphertext, instead of checking whether such a query has been made afterwards. This yields the following CCA1 axiom (see Appendix B.3.2 for the others):

PKE-NM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C, C', \vec{a}, \vec{a}', t_k, t_r, m \\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C, C', \vec{a}, \vec{a}', m)] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C, C', \vec{a}, \vec{a}', m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded } k,t_k}(C, C')] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\text{guarded } k,t_k}(\vec{a}, \vec{a}', m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c = \mathrm{PKE.Enc}(k\ t_k, m, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m \\ (\mathsf{let}\ x = (\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x = c\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c = \mathrm{PKE.Enc}(k\ t_k, 0^{|m|}, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m \\ (\mathsf{let}\ x = (\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x = c\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array}}$$

This axiom is surprisingly close to the PKE-IND-CCA1 axiom, with an additional term for the decryption. Like the CCA2 oracle, this additional term allows decryption queries depending on the challenge ciphertext, unlike the CCA2 axiom however, these decryptions can not be nested (the attacker can not make a decryption query based on the result of a previous query)[5].

We find that this perspective helps to build intuition for some results that are somewhat obscured by the game-based definitions. For one, NM-CCA1 and NM-CPA imply IND-CCA1 and IND-CPA respectively, since NM simply gives the attacker access to an additional term. Further, IND-CCA2 and NM-CCA2 are equivalent, since the CCA2 decryption oracle is strictly more powerful than the additional term for non-malleability. Finally, there is no implication between NM-CPA and IND-CCA1, as the CCA1 decryption oracle allows nested queries, but cannot be used for queries depending on the challenge ciphertext. Thus, neither can be used to fully simulate the other.

3.2.3. Soundness of Axioms

Of course, the axioms we presented here need to be proven sound with respect to the computational definitions. We present how to do this for KEM-IND-CCA2 and KEM-sNM-CCA1 in Appendix F. In general, the proofs proceed by assuming an attacker $\mathcal{A}$ against the indistinguishability in the axiom, and using it to construct an attacker against the cryptographic game. This requires specifying a Turing machine that computes the terms in the axioms while querying available oracles for any computations it cannot perform directly, like encryption and decryption. The side conditions of the axioms enforce that all terms that would require knowledge not available to the attacker can be handled by oracles.

Since the computational definitions specify that such attackers must have a negligible advantage, it then follows that the advantage of $\mathcal{A}$ against the indistinguishability must be negligible.

3.3. KEM/DEM Security Results in CCSA

With all definitions in place, we can now reproduce the positive results of Herranz, Hofheinz, and Kiltz [17]. We will show here a quick overview of the proof of IND-CCA2 security of a KEM/DEM hybrid encryption scheme composed of an IND-CCA2 secure KEM and an IND-OTCCA secure DEM, the full proof can be found in Appendix E.1. The security of other combinations can be proved similarly using the same techniques.

This proof uses the versions of the axioms where the oracle is explicitly given as a term, as this enables us to rewrite inside this term. Substituting the definitions of PKE.Enc and PKE.Dec in PKE-IND-CCA2',

[5] While it may seem that this distinction would not be very useful, it is possible to construct encryption schemes that are only secure as long as no nested queries are allowed. Such a scheme is NM-CCA1 secure, but not IND-CCA2.

we have the following proof goal:

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ (\lambda x.\mathsf{if}\ x = \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk = \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk, \pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = \mathsf{let}\ (sk, c_1) = \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r))\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk = \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk, \pi_2(x)))\end{array}$$

As a basic proof idea, the indistinguishability of the two ciphertexts should follow from DEM-IND-OTCCA. However, we cannot apply this directly, as we are currently not using a fresh random key, but one obtained from the KEM – if the KEM is insecure and the attacker can gain information about the key, DEM-IND-OTCCA does not apply. Thus, we first need to apply KEM-IND-CCA2' to exchange the key with a fresh random one. However, before we can do so, we first need to ensure that KEM decryption is properly guarded.

This requires some care, as the PKE-IND-CCA2' oracle permits queries for ciphertexts where the key part is the same as in the challenge ciphertext, and only the message part is changed. Before we can apply KEM-IND-CCA2', we need to handle such cases without querying the KEM decryption oracle. We thus rewrite using β-equivalence and KEM correctness until we obtain the following:

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array}$$

At this point, we can apply transitivity and KEM-IND-CCA2 to exchange the key produced by KEM.Enc with a fresh key $sk^*\ ()$, which will enable us to apply DEM-IND-OTCCA as long as we can properly guard $\mathrm{DEM.Dec}(sk^*\ (), \pi_2(x))$ (note that the other occurrence of DEM.Dec does not need to be guarded, as it uses a key that is not derived from $sk^*\ ()$). Luckily, such a guard can be put in place using β-equivalence, as the existing branches already ensure $\pi_2(x) \neq \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r)$. Thus, the proof concludes using DEM-IND-OTCCA.

However, we have omitted the side conditions here. The most interesting case is that to apply KEM-IND-CCA2', $\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k),\cdot)}(C)$ must imply $\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k),\cdot)}$, which we show in Appendix C.1.1.

4. Oblivious Transfers

Oblivious transfers (OT) are an important building block for secure multiparty computation. Their most direct application are secret database queries, where a client can request exactly one entry from a database without the server learning which entry was queried. However, they can also be used in many other ways, including the secure evaluation of boolean and arithmetic circuits [11].

Many variations of oblivious transfer protocols have been presented over the years (see Yadav et al. [25] for an extensive overview), not only for different security requirements, but also improving on computational efficiency. It thus makes sense to treat oblivious transfers as a primitive when designing a protocol,

specifying only the desired security properties. For CCSA, this means that we will need an abstract representation of oblivious transfers that can describe a reasonable variety of implementations. In this chapter, we present such an abstract representation and the corresponding security properties, and present two implementations of OT protocols that satisfy these properties.

4.1. Definition

A 1-out-of-2 oblivious transfer (1-2 oblivious transfer for short) is a protocol between two parties S (the sender) and R (the receiver), such that:

- S has two messages m_0 and m_1
- R has a secret bit b
- R learns messages m_b, while not gaining any knowledge about m_{1-b}
- S does not learn the secret bit b

4.2. Abstract Representation in CCSA

To represent a reasonable number of oblivious transfer protocols, we choose to represent them with the following types and functions:

$\tau_{internal-sender}$ represents the internal state of the sender after initialization.

$\tau_{internal-receiver}$ represents the internal state of the receiver after making a request.

$\tau_{message-init}$ represents the type of the initialization message from the sender to the receiver.

$\tau_{message-req}$ represents the type of the request message from the receiver to the sender.

$\tau_{message-transfer}$ represents the type of the transfer message from the sender to the receiver.

$S.init : \tau_{index} \to \tau_{message-init} \times \tau_{internal-sender}$ represents the first phase of the sender, which is given an index for its random samplings and generates the initialization message to send to the receiver as well as the internal state for the second phase.

$R.req : \tau_{index} \times \texttt{bool} \times \tau_{message-init} \to \tau_{message-req} \times \tau_{internal-receiver}$ represents the first phase of the receiver, which receives the sender's initialization message, the receiver's secret bit, and an index for its random samplings and produces the request message to send to the sender as well as the internal state for the second phase.

$S.transfer : \tau_{message-req} \times \tau_{internal-sender} \times \tau_{msg} \times \tau_{msg} \to \tau_{message-transfer}$ represents the sender's second phase, which is given the request message from the receiver, its own internal state, and two possible messages to send, and produces a message for the receiver.

$R.recv : \tau_{message-transfer} \times \tau_{internal-receiver} \to \tau_{msg}$ represents the receiver's second phase, which receives the transfer message and the receiver's internal state, and produces an output message.

Note that both parties' initial functions are given an index. We chose this representation because protocols can vary drastically in how they use randomness. With this design, each protocol can specify any number of required names, while the index still serves to make everything deterministic. In the following, we will assume that the names specified in a protocol are never used outside.

4.3. Security Properties

We consider *privacy properties*, which require that one party's secret inputs remain secret for the other parties during the protocol, for *honest-but-curious* attackers. Such attackers take the role of one party in the protocol and may attempt to recover another party's secret, however, they are required to adhere to the protocol.

4.3.1. Receiver Privacy for Honest-but-Curious Attackers

Receiver privacy states that an honest-but-curious sender can not learn the receiver's secret bit (and thus not learn which of the two messages was received). Concretely, this means that a sender can not distinguish between the request messages sent when the secret bit is 0 or 1.

OT-R-PRIV

$$\frac{\mathcal{E}, \Theta \vdash [\phi_{\text{fresh}}^{R.req,j}(\vec{u})]}{\mathcal{E}, \Theta \vdash \ \vec{u}, S.init(i),\ \pi_1(R.req(j, 0, \pi_1(S.init(i)))) \ \sim\ \vec{u}, S.init(i),\ \pi_1(R.req(j, 1, \pi_1(S.init(i))))}$$

Note that receiver privacy assumes that the receiver uses fresh random samplings, so we write $\phi_{\text{fresh}}^{R.req,j}(\vec{u})$ to denote that index j has not been used before[1].

4.3.2. Sender Privacy for Honest-but-Curious Attackers

Sender privacy states that an honest-but-curious receiver can not learn anything about the other message, i.e. the one it chose not to receive. Concretely, this means that if the receiver chose to receive message 0, it can not distinguish between transfer messages where message 0 is fixed, but message 1 varies. Similarly, if it chose to receive message 1, it can not distinguish between transfers where message 0 varies.

OT-S-PRIV-0

$$\frac{\mathcal{E}, \Theta \vdash [\phi_{\text{fresh}}^{S.init,i}(\vec{u}, x_0, x_1)]}{\mathcal{E}, \Theta \vdash \begin{array}{c} \vec{u}, \pi_1(S.init(i)),\ R.req(j, 0, \pi_1(S.init(i))), \\ S.transfer \\ (\pi_1(R.req(j, 0, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0, x_1) \end{array} \sim \begin{array}{c} \vec{u}, \pi_1(S.init(i)),\ R.req(j, 0, \pi_1(S.init(i))), \\ S.transfer \\ (\pi_1(R.req(j, 0, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0, x_1') \end{array}}$$

OT-S-PRIV-1

$$\frac{\mathcal{E}, \Theta \vdash [\phi_{\text{fresh}}^{S.init,i}(\vec{u}, x_0, x_1)]}{\mathcal{E}, \Theta \vdash \begin{array}{c} \vec{u}, \pi_1(S.init(i)),\ R.req(j, 1, \pi_1(S.init(i))), \\ S.transfer \\ (\pi_1(R.req(j, 1, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0, x_1) \end{array} \sim \begin{array}{c} \vec{u}, \pi_1(S.init(i)),\ R.req(j, 1, \pi_1(S.init(i))), \\ S.transfer \\ (\pi_1(R.req(j, 1, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0', x_1) \end{array}}$$

4.4. PKE-based Oblivious Transfer [16]

Even, Goldreich, and Lempel [16] present an algorithm for oblivious transfers that builds upon a full-domain PKE scheme. Here, we will assume a fixed-length scheme, i.e. all plaintexts have the same length and all ciphertexts have the same length.

We can represent this scheme with the following definitions, using the names k_S, m_a, m_b, s and r:

$$\begin{aligned}
\tau_{internal-sender} &:= \tau_{msg} \times \tau_{msg} \times \tau_{msg} \\
\tau_{internal-receiver} &:= \texttt{bool} \times \tau_{msg} \\
\tau_{message-init} &:= \tau_{msg} \times \tau_{msg} \times \tau_{msg} \\
\tau_{message-req} &:= \tau_{msg} \\
\tau_{message-transfer} &:= \tau_{msg} \times \tau_{msg} \\
S.init(i) &:= ((m_a\ i, m_b\ i, \text{pk}(k_S\ i)), (m_a\ i, m_b\ i)) \\
R.req(i, b, (m_0, m_1, pk)) &:= (\text{PKE.Enc}(pk, s_i, r_i) \oplus m_b, (b, s_i)) \\
S.transfer(q, (m_0, m_1), x_0, x_1) &:= (x_0 \oplus \text{PKE.Dec}(k_S\ i, q \oplus m_0), x_1 \oplus \text{PKE.Dec}(k_S\ i, q \oplus m_1)) \\
R.recv((c_0, c_1), (b, r)) &:= c_b \oplus r
\end{aligned}$$

[1] We justify this slight abuse of notation by the fact that the construction of a suitable formula is entirely analogous to that of $\phi_{\text{fresh}}^{n,t_n}$.

4.4.1. Full-Domain Encryption

The PKE scheme used in this protocol has to be NM-CPA secure and full-domain, which means that every bitstring needs to be a valid ciphertext. Otherwise, the sender could receive a request with an invalid ciphertext, trivially allowing it to determine the receiver's chosen bit. In fact, we further need that even with knowledge of the key, it is impossible to distinguish between an encryption of a random sampling and a randomly sampled ciphertext (see Appendix H.1 for an explanation why this is necessary). We represent this condition as follows:

PKE-IND$

$$\frac{\mathcal{E};\Theta;\emptyset \vdash \phi_{\text{fresh}}^{x,t_x}(\vec{u},C) \wedge \phi_{\text{fresh}}^{x',()}(\vec{u},C)}{\mathcal{E};\Theta \vdash \vec{u},C[\text{PKE.Enc}(\text{pk}(k_S\ t_k), x\ t_x, r\ t_r)] \sim \vec{u},C[x'\ ()]}$$

We will also make use of the following property, derived from PKE-IND$ and NM-CPA (see Appendix H.2), which states that to an attacker knowing some challenge ciphertext c, the decryption of $c \oplus a$ is indistinguishable from a randomly sampled message:

PKE-XOR-CPA

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',m,t_r \qquad k \notin_{pk} \vec{u},C,C',m,t_r,a \qquad \mathcal{E};\Theta \vdash [\phi_{\text{fresh}}^{n,()}(\vec{u},C,C',m,t_r,a)] \qquad \mathcal{E};\Theta \vdash [a \neq 0]}{\mathcal{E};\Theta \vdash \begin{array}{c}\vec{u},C[\mathsf{let}\ c \leftarrow \text{PKE.Enc}(\text{pk}(k\ t_k),m,r\ t_r);\\ x \leftarrow c \oplus a\ \mathsf{in}\ c, \text{PKE.Dec}(k\ t_k,x)]\end{array} \sim \begin{array}{c}\vec{u},C[\mathsf{let}\ c \leftarrow \text{PKE.Enc}(\text{pk}(k\ t_k),m,r\ t_r);\\ x \leftarrow c \oplus a\ \mathsf{in}\ c, n\ ()]\end{array}}$$

4.4.2. Sender Privacy

We will only prove one of the two versions here, as the other is entirely analogous. In the abstract version, we wish to prove the following statement:

$$\mathcal{E},\Theta \vdash \begin{array}{c}\pi_1(S.init(i)),\ R.req(j,0,\pi_1(S.init(i))),\\ S.transfer\\ (\pi_1(R.req(j,0,\pi_1(S.init(i)))),\\ \pi_2(S.init(j)),x_0,x_1)\end{array} \sim \begin{array}{c}\pi_1(S.init(i)),\ R.req(j,0,\pi_1(S.init(\\ S.transfer\\ (\pi_1(R.req(j,0,\pi_1(S.init(i)))),\\ \pi_2(S.init(j)),x_0,x_1')\end{array}$$

Substituting the appropriate definitions and simplifying, we get

$$\mathcal{E},\Theta \vdash \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, x_1 \oplus\\ \text{PKE.Dec}(k_S\ i, \text{PKE.Enc}(\text{pk}(k_S\ i),\\ s\ j, r\ j) \oplus m_a\ i \oplus m_b\ i), x_0, x_1\end{array} \sim \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, x_1 \oplus\\ \text{PKE.Dec}(k_S\ i, \text{PKE.Enc}(\text{pk}(k_S\ i),\\ s\ j, r\ j) \oplus m_a\ i \oplus m_b\ i), x_0, x_1'\end{array}$$

We then apply PKE-XOR-CPA to exchange the decryption with a fresh random sampling $n\ ()$. Note that $m_a\ i \oplus m_b\ i \neq 0$ is overwhelmingly true.

$$\mathcal{E},\Theta \vdash \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, x_1 \oplus n\ (), x_0, x_1\end{array} \sim \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, x_1 \oplus n\ (), x_0, x_1'\end{array}$$

Next, we use the fact that exclusive or with random samplings are indistinguishable from random, so $x_1 \oplus n$ is indistinguishable from some fresh $n'\ ()$.

$$\mathcal{E},\Theta \vdash \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, n'\ (), x_0, x_1\end{array} \sim \begin{array}{c}m_a\ i, m_b\ i, \text{pk}(k_S\ i),\\ \text{PKE.Enc}(\text{pk}(k_S\ i), s\ j, r\ j) \oplus m_a\ i, b, s\ j,\\ x_0 \oplus s\ j, n'\ (), x_0, x_1'\end{array}$$

Since both x_1 and x_1' are now fresh, the proof concludes by the indistinguishability of fresh random samplings. This proves sender privacy for this OT protocol, receiver privacy can be found in Appendix H.3.

4.5. Diffie-Hellman based Oblivious Transfer [9]

Bellare and Micali [9] present an algorithm based on the *decisional Diffie-Hellman assumption*, which we introduce below. We can represent their scheme with the following definitions, using the names c, γ_0, γ_1 and r:

$$
\begin{aligned}
\tau_{internal-sender} &:= \tau_{msg} \times \tau_{msg} \times \tau_{msg} \\
\tau_{internal-receiver} &:= \texttt{bool} \times \tau_{msg} \\
\tau_{message-init} &:= \tau_{msg} \\
\tau_{message-req} &:= \tau_{msg} \times \tau_{msg} \\
\tau_{message-transfer} &:= \tau_{msg} \times \tau_{msg} \times \tau_{msg} \times \tau_{msg} \\
S.init(i) &:= (g^{c\ i}, (g^{c\ i}, \gamma_0\ i, \gamma_1\ i)) \\
R.req(j, b, C) &:= \mathsf{let}\ \beta_b = g^{r\ j}, \beta_{1-b} = C / g^{r\ j}\ \mathsf{in}\ ((\beta_0, \beta_1), (b, r\ j)) \\
S.transfer((\beta_0, \beta_1), (C, \gamma_0, \gamma_1), x_0, x_1) &:= \mathsf{let}\ \alpha_0 = g^{\gamma_0}, \alpha_1 = g^{\gamma_1}, y_0 = \beta_0^{\gamma_0}, y_1 = \beta_1^{\gamma_1} \\
&\qquad \mathsf{in}\ (\alpha_0, \alpha_1, x_0 \oplus y_0, x_1 \oplus y_1) \\
R.recv((\alpha_0, \alpha_1, c\ 0, c\ 1), (b, r)) &:= c\ b \oplus \alpha_b^s
\end{aligned}
$$

4.5.1. Axioms for the Diffie-Hellman Assumption

The decisional Diffie-Hellman assumption states that in a cyclic group with generator g, for uniformly and independently chosen a and b, g^{ab} is indistinguishable from a randomly chosen group element g^c[11]. It can be stated in CCSA as follows:

DDH
$$\frac{\mathcal{E}; \Theta; \emptyset \vdash \phi^{\text{guarded}\ x, t_x}_{g^{x\ t_x}}(\vec{u}, C) \wedge \phi^{\text{guarded}\ y, t_y}_{g^{y\ t_y}}(\vec{u}, C) \wedge \phi^{z, t_z}_{\text{fresh}}(\vec{u}, C)}{\mathcal{E}; \Theta \vdash \vec{u}, C[g^{(x\ t_x)(y\ t_y)}] \sim \vec{u}, C[g^{z\ t_z}]}$$

In addition, we will also need that if we perform an exclusive or with a random group element, the result is itself random, and that multiplying a randomly sampled group element with some known value is indistinguishable from just a randomly chosen group element. The former is a basic property of XOR, while the latter is a consequence of modular arithmetic.

XOR-GROUP
$$\frac{\mathcal{E}; \Theta; \emptyset \vdash \phi^{x, t_x}_{\text{fresh}}(\vec{u}, C) \wedge \phi^{y, t_c}_{\text{fresh}}(\vec{u}, C)}{\mathcal{E}; \Theta \vdash \vec{u}, C[m \oplus g^{x\ t_x}] \sim \vec{u}, C[y\ t_c]}$$

FRESH-SHIFT
$$\frac{\mathcal{E}; \Theta; \emptyset \vdash \phi^{a, t_a}_{\text{fresh}}(\vec{u}, C) \wedge \phi^{c, t_c}_{\text{fresh}}(\vec{u}, C)}{\mathcal{E}; \Theta \vdash \vec{u}, C[g^{(a\ t_a)+b}] \sim \vec{u}, C[g^{c\ t_c}]}$$

4.5.2. Sender Privacy

We will only prove one of the two versions here, as the other is entirely analogous. In the abstracted version, we wish to prove the following statement:

$$\mathcal{E}, \Theta \vdash \begin{array}{c} \pi_1(S.init(i)), \\ \pi_2(R.req(j, 0, \pi_1(S.init(i)))),\ S.transfer \\ (\pi_1(R.req(j, 0, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0, x_1) \end{array} \sim \begin{array}{c} \pi_1(S.init(i)), \\ \pi_2(R.req(j, 0, \pi_1(S.init(i)))),\ S.transfer \\ (\pi_1(R.req(j, 0, \pi_1(S.init(i)))), \\ \pi_2(S.init(j)), x_0, x_1') \end{array}$$

Substituting the appropriate definitions and simplifying, we get

$$\mathcal{E}, \Theta \vdash \begin{array}{c} g^{c\ i}, g^{r\ j}, g^{c\ i-r\ j}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j\gamma_0\ i}, x_1 \oplus g^{(c\ i-r\ j)\gamma_1\ i} \end{array} \sim \begin{array}{c} g^{c\ i}, g^{r\ j}, g^{c\ i-r\ j}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j\gamma_0\ i}, x_1' \oplus g^{(c\ i-r\ j)\gamma_1\ i} \end{array}$$

As a first step, we will use the fact that $g^{c\ i-r\ j}$ is indistinguishable from a random group element since $g^{c\ i}$ is just a random group element. We can therefore substitute $c\ i$ with $n\ i + r\ j$ with a fresh name $n\ i$.

$$\mathcal{E}, \Theta \vdash \begin{array}{c} g^{n\ i} \cdot g^{r\ j}, g^{r\ j}, g^{n\ i}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j\gamma_0\ i}, x_1 \oplus g^{n\ i\gamma_1\ i} \end{array} \sim \begin{array}{c} g^{n\ i} \cdot g^{r\ j}, g^{r\ j}, g^{n\ i}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j\gamma_0\ i}, x_1' \oplus g^{n\ i\gamma_1\ i} \end{array}$$

Since $n\ i$ and $\gamma_1\ i$ only occur as exponents of g, we can now apply the decisional Diffie-Hellman assumption, which states that $g^{n_1 \gamma_1\ i}$ is indistinguishable from a random group element $g^{m\ i}$.

$$\mathcal{E}, \Theta \vdash \begin{array}{c} g^{n\ i} \cdot g^{r\ j}, g^{r\ j}, g^{n\ i}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j \gamma_0\ i}, x_1 \oplus g^{m\ i} \end{array} \sim \begin{array}{c} g^{n\ i} \cdot g^{r\ j}, g^{r\ j}, g^{n\ i}, b, r\ j, g^{\gamma_0\ i}, g^{\gamma_1\ i}, \\ x_0 \oplus g^{r\ j \gamma_0\ i}, x_1' \oplus g^{m\ i} \end{array}$$

We can now use XOR-GROUP to replace the last term on both sides with a fresh random sampling. The proof then concludes by the indistinguishability of fresh names.

4.5.3. Receiver Privacy

We wish to prove the following statement:

$$\mathcal{E}, \Theta \vdash S.init(r),\ \pi_1(R.req(r', 0, \pi_1(S.init(r)))) \sim S.init(r),\ \pi_1(R.req(r', 1, \pi_1(S.init(r))))$$

Substituting the definitions and simplifying, we get

$$\mathcal{E}, \Theta \vdash g^{c\ i}, \gamma_0\ i, \gamma_1\ i, g^{r\ j}, g^{c\ i - r\ j} \sim g^{c\ i}, \gamma_0\ i, \gamma_1\ i, g^{c\ i - r\ j}, g^{r\ j}$$

On the right-hand side, we will substitute $r\ j$ with $(c\ i - r')$, yielding

$$\mathcal{E}, \Theta \vdash g^{c\ i}, \gamma_0\ i, \gamma_1\ i, g^{r\ j}, g^{c\ i - r\ j} \sim g^{c\ i}, \gamma_0\ i, \gamma_1\ i, g^{r'}, g^{c\ i - r'}$$

The proof then concludes by the indistinguishability of fresh random samplings.

5. Conclusions

We have demonstrated that CCSA logic is indeed quite suitable for the analysis of cryptographic primitives, whether they are derived from other primitives (Section 3) or directly from number-theoretic assumptions (Section 4.5). As a result, we have produced an extensive set of cryptographic axioms for public key encryption, key encapsulation and data encapsulation mechanisms (Appendix B). This includes a general construction for side conditions (Appendix C.1) that generalizes constructions used by Baelde, Koutsos, and Lallemand [2], as well as general proofs that may be applicable in other situations. Further, we have introduced an abstract representation of 1-2 oblivious transfers in CCSA, which we believe should be suitable for use as a primitive in more complex protocols (Section 4.2). We have demonstrated that it can represent two different protocols (Sections 4.4 to 4.5).

5.1. Future Work

There are many related topics that would be interesting to explore in future work. For KEM/DEM hybrid encryption, for example, we have only reproduced the positive results of Herranz, Hofheinz, and Kiltz [17], not the separation results. Currently, CCSA is not well-equipped to reason about *distinguishability* (i.e. negated indistinguishability), which would be necessary for the counterexamples used in those proofs.

For oblivious transfers, it currently remains open to actually use our abstract representation in a protocol. It is thus possible that our representation could benefit from some changes in order to be more useful for this purpose, but this remains to be investigated. Beyond that, however, it would be interesting to explore mechanisms of protocol composition instead of representing protocols as terms, as the latter is somewhat restricted and will likely not be able to capture all possible implementations. Some work on protocol composition has been done by Comon, Jacomme, and Scerri [12], but we have not investigated whether it can be used for this purpose.

Finally, it is as of yet not clear how this work might eventually be represented in the proof assistant Squirrel, i.e. how to provide definitions of a cryptographic primitive and the proofs of its security properties in a modular way that allows not only using the primitive for many protocols, but also substituting different primitives with the same guarantees.

Bibliography

[1] David Adrian et al. “Imperfect forward secrecy: how Diffie-Hellman fails in practice”. In: *Commun. ACM* 62.1 (2019), pp. 106–114. DOI: 10.1145/3292035. URL: https://doi.org/10.1145/3292035.

[2] David Baelde, Adrien Koutsos, and Joseph Lallemand. “A Higher-Order Indistinguishability Logic for Cryptographic Reasoning”. In: *LICS*. Boston, United States: IEEE, June 2023, pp. 1–13. DOI: 10.1109/LICS56636.2023.10175781. URL: https://inria.hal.science/hal-03981949.

[3] David Baelde, Adrien Koutsos, and Justine Sauvage. “Foundations for Cryptographic Reductions in CCSA Logics”. working paper or preprint. Mar. 2024. URL: https://hal.science/hal-04511718.

[4] David Baelde et al. “A Probabilistic Logic for Concrete Security”. In: *CSF 2024 - 37th IEEE Computer Security Foundations Symposium*. Enschede, Netherlands, July 2024. URL: https://hal.science/hal-04577828.

[5] David Baelde et al. “An Interactive Prover for Protocol Verification in the Computational Model”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 537–554. DOI: 10.1109/SP40001.2021.00078. URL: https://doi.org/10.1109/SP40001.2021.00078.

[6] David Baelde et al. “Cracking the Stateful Nut”. In: *CSF 2022 - 35th IEEE Computer Security Foundations Symposium*. Haifa, Israel, Aug. 2022. URL: https://hal.science/hal-03500056.

[7] Gergei Bana and Hubert Comon-Lundh. “A Computationally Complete Symbolic Attacker for Equivalence Properties”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 609–620. DOI: 10.1145/2660267.2660276. URL: https://doi.org/10.1145/2660267.2660276.

[8] Manuel Barbosa et al. *Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium*. Cryptology ePrint Archive, Paper 2023/246. https://eprint.iacr.org/2023/246. 2023. URL: https://eprint.iacr.org/2023/246.

[9] Mihir Bellare and Silvio Micali. “Non-Interactive Oblivious Transfer and Applications”. In: *Advances in Cryptology - CRYPTO ’89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Springer, 1989, pp. 547–557. DOI: 10.1007/0-387-34805-0_48. URL: https://doi.org/10.1007/0-387-34805-0%5C_48.

[10] Bruno Blanchet. “An Efficient Cryptographic Protocol Verifier Based on Prolog Rules”. In: *14th IEEE Computer Security Foundations Workshop (CSFW-14)*. This paper received a test of time award at the CSF’23 conference. Cape Breton, Nova Scotia, Canada: IEEE Computer Society, June 2001, pp. 82–96.

[11] Dan Boneh and Victor Shoup. *A graduate course in applied cryptography*. 2023.

[12] Hubert Comon, Charlie Jacomme, and Guillaume Scerri. “Oracle simulation: a technique for protocol composition with long term shared secrets”. In: *ACM CCS 2020*. CCS ’20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. Orlando, United States: Association for Computing Machinery, Nov. 2020, pp. 1427–1444. URL: https://inria.hal.science/hal-02913866.

[13] Hubert Comon and Adrien Koutsos. “Formal Computational Unlinkability Proofs of RFID Protocols”. In: *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. 2017, pp. 100–114. DOI: 10.1109/CSF.2017.9.

[14] Ronald Cramer and Victor Shoup. *Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack*. Cryptology ePrint Archive, Paper 2001/108. https://eprint.iacr.org/2001/108. 2001. URL: https://eprint.iacr.org/2001/108.

[15] François Dupressoir, Konrad Kohbrok, and Sabine Oechsner. "Bringing State-Separating Proofs to EasyCrypt - A Security Proof for Cryptobox". In: *IACR Cryptol. ePrint Arch.* (2021), p. 326. URL: https://eprint.iacr.org/2021/326.

[16] Shimon Even, Oded Goldreich, and Abraham Lempel. "A Randomized Protocol for Signing Contracts". In: *Commun. ACM* 28.6 (1985), pp. 637–647. DOI: 10.1145/3812.3818. URL: https://doi.org/10.1145/3812.3818.

[17] Javier Herranz, Dennis Hofheinz, and Eike Kiltz. "Some (in)sufficient conditions for secure hybrid encryption". In: *IACR Cryptol. ePrint Arch.* (2006), p. 265. URL: http://eprint.iacr.org/2006/265.

[18] Adrien Koutsos. "The 5G-AKA Authentication Protocol Privacy". In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2019, pp. 464–479. DOI: 10.1109/EuroSP.2019.00041.

[19] Gavin Lowe. "Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR". In: *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 1055. Lecture Notes in Computer Science. Springer, 1996, pp. 147–166. DOI: 10.1007/3-540-61042-1_43. URL: https://doi.org/10.1007/3-540-61042-1%5C_43.

[20] Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto. "On the Equivalence of Several Security Notions of Key Encapsulation Mechanism". In: *IACR Cryptol. ePrint Arch.* (2006), p. 268. URL: http://eprint.iacr.org/2006/268.

[21] Benedikt Schmidt et al. "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties". In: *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*. Ed. by Stephen Chong. IEEE Computer Society, 2012, pp. 78–94. DOI: 10.1109/CSF.2012.25. URL: https://doi.org/10.1109/CSF.2012.25.

[22] Victor Shoup. *OAEP Reconsidered*. Cryptology ePrint Archive, Paper 2000/060. https://eprint.iacr.org/2000/060. 2000. URL: https://eprint.iacr.org/2000/060.

[23] Victor Shoup. "Sequences of games: a tool for taming complexity in security proofs". In: *IACR Cryptol. ePrint Arch.* (2004), p. 332. URL: http://eprint.iacr.org/2004/332.

[24] Mathy Vanhoef and Frank Piessens. "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham et al. ACM, 2017, pp. 1313–1328. DOI: 10.1145/3133956.3134027. URL: https://doi.org/10.1145/3133956.3134027.

[25] Vijay Kumar Yadav et al. "A Survey of Oblivious Transfer Protocol". In: *ACM Comput. Surv.* 54.10s (2022), 211:1–211:37. DOI: 10.1145/3503045. URL: https://doi.org/10.1145/3503045.

A. Basic CCSA Axioms

We list here a few axioms that are relevant to the proofs in this work. All axioms are taken from Baelde, Koutsos, and Lallemand [2].

A.1. Symmetry and Transitivity of Indistinguishability

$$\textsc{Symmetry}\ \frac{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{v}}{\mathcal{E};\Theta \vdash \vec{v} \sim \vec{u}} \qquad \textsc{Transitivity}\ \frac{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{v} \qquad \mathcal{E};\Theta \vdash \vec{v} \sim \vec{w}}{\mathcal{E};\Theta \vdash \vec{u} \sim \vec{w}}$$

A.2. Indistinguishability of Fresh Names

$$\textsc{Fresh}\ \frac{\mathcal{E};\Theta \vdash [\phi^{n,t}_{\text{fresh}}(\vec{u},t)]}{\mathcal{E};\Theta \vdash \vec{u}, n\ t \sim u, \vec{n_{\text{fresh}}}\ ()}$$

Here, n_{fresh} is assumed to be a new, unused name with index type `unit` and the same probability distribution as n. The formula $\phi^{n,t}_{\text{fresh}}(\vec{u},t)$ ensures that the sampling $n\ t$ is not used anywhere in $\vec{u}$ and t (and therefore that $\vec{u}$ is independent of $n\ t$). See either Baelde, Koutsos, and Lallemand [2] for the original definition, or Appendix C.1 for a more general construction.

A.3. Rewriting and β-equivalence

If two terms are equal with overwhelming probability $[t_1 = t_2]$, then this equality can be used to rewrite underneath an indistinguishability predicate, as any adversary can distinguish between the two in at most the cases where t_1 and t_2 are not equal, which is negligible. This is represented by the following axiom:

$$\textsc{Rewrite}\ \frac{\mathcal{E};\Theta \vdash F[s] \qquad \mathcal{E};\Theta \vdash [s = t] \qquad [] \text{ does not appear below const}(\cdot) \text{ or adv}(\cdot) \text{ in } F[]}{\mathcal{E};\Theta \vdash F[t]}$$

Note that we have not introduced the adv$(\cdot)$ predicate, but chose to include it here to match the definition by Baelde, Koutsos, and Lallemand [2]. Similarly, it is possible to rewrite within a boolean formula[1]:

$$\textsc{Rewrite'}\ \frac{\mathcal{E};\Theta \vdash [\phi[s]] \qquad \mathcal{E};\Theta \vdash [s = t]}{\mathcal{E};\Theta \vdash [\phi[t]]}$$

Note that this also gives us transitivity of $[\cdot = \cdot]$.

These axioms are especially useful in conjunction with the following, which allows to derive equality from β-equivalence:

$$\beta\ \frac{}{\mathcal{E};\Theta \vdash [(\lambda(x:\tau).t)\ t_0 = t\{x \mapsto t_0\}]}$$

While this axiom only applies to one single β-reduction, it allows us to obtain equality of any β-equivalent terms since $=$ is an equivalence relation.

[1] adapted from a rule for local judgements by Baelde, Koutsos, and Lallemand [2]

B. KEM/DEM: All Axioms

B.1. KEM

B.1.1. IND

KEM-IND-CPA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r,t_r}(C,\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded }k,t_k}(C,\vec{a})]\end{array}}{\mathcal{E},\Theta\vdash (\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\sim \begin{array}{c}(\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\\ (sk^*(),\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r))\end{array}}$$

KEM-IND-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r,t_r}(C,\vec{a})]\\ \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded }k,t_k}(C)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}^{\text{guarded }k,t_k}(\vec{a})]\end{array}}{\mathcal{E},\Theta\vdash (\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\sim \begin{array}{c}(\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\\ (sk^*(),\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r))\end{array}}$$

KEM-IND-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r,t_r}(C,\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})]\\ \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}^{\text{guarded }k,t_k}(\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\text{if } t=c \text{ then } \bot \text{ else } \mathrm{KEM.Dec}(k\ t_k,t)}^{\text{guarded }k,t_k}(C)]\end{array}}{\mathcal{E},\Theta\vdash (\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\sim \begin{array}{c}(\lambda\ \vec{v}\ (sk,c).C)\ \vec{a}\\ (sk^*(),\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r))\end{array}}$$

KEM-IND-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r,t_r}(C,\vec{a})]\\ \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{sk^*,()}(C,\vec{a})] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded }k,t_k}(C)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}^{\text{guarded }k,t_k}(\vec{a})]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ (sk,c)\ \mathsf{dec}.C)\ \vec{a}\\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ (\lambda c.\mathsf{if}\ c=\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,c))\end{array}\sim \begin{array}{c}(\lambda\ \vec{v}\ (sk,c)\ \mathsf{dec}.C)\ \vec{a}\\ (sk^*(),\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r))\\ (\lambda c.\mathsf{if}\ c=\pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,c))\end{array}}$$

B.1.2. NM

Since a KEM does not encrypt a message, but generates a symmetric key instead, the notion of nonmalleability for PKEs and DEMs does not translate directly. Instead, we provide four different sets of axioms for definitions from Herranz, Hofheinz, and Kiltz [17] and Nagao, Manabe, and Okamoto [20]. Of these, weak NM is strictly weaker than the others, while the others are all equivalent (see Appendix G.1).

wNM

KEM-wNM-CPA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\\ \mathcal{E};\Theta\vdash[\phi_{\mathrm{fresh}}^{r,t_r}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded }k,t_k}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi_{\mathrm{fresh}}^{k^*,()}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}\sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-wNM-CCA1

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}{\mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}$$

For CCA2, note that decryption queries may never depend on k_0. Therefore, the context must be split, where C' corresponds to the second phase of the game, and C'' corresponds to the attacker-chosen relation. We also require that C' is of order 0 for wNM, to prevent passing a decryption function to C''.

KEM-wNM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\\ \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C',C'')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,t)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C'')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C',C'')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda k_0.\ C'')\ ((\lambda\vec{v}\,c\,r.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda k_0.\ C'')\ ((\lambda\vec{v}\,c\,r.\ C')\ \vec{a}'\ c\ (k^*\ ()))\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)))\end{array}}$$

KEM-wNM-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C',C'')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C',C'')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C',C'')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)\ \mathsf{in}\ (\lambda k_0.\ C'')\\ ((\lambda\vec{v}\,c\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec})\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)\ \mathsf{in}\ (\lambda k_0.\ C'')\\ ((\lambda\vec{v}\,c\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ (k^*\ ()))\\ (\mathsf{let}\ x=((\lambda\vec{v}\,c.\ C)\ \vec{a}\ c)\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec})\end{array}}$$

sNM

As mentioned in Section 3.2.2, we can treat the attacker-defined relation as if it had access to the same oracles as the attacker, greatly simplifying the presentation as an axiom. This is justified formally in Appendix G.1.3.

We use $\mathsf{order}(k_0,k^*\ ())$ to give the adversary access to both keys without communicating which one is which. For randomly sampled keys, each order is equally likely.

KEM-sNM-CPA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\\ \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k^*\ ())\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k_0,k^*\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k_0,k^*\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-sNM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k^*\ ())\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k_0,k^*\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k_0,k^*\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

With the standard definition of sNM-CCA2, queries depending on k_0 in C' are not technically allowed, as any oracle queries in the construction of the relation do not yet have access to the symmetric key. However, since the adversary has access to an ordered pair of both keys and can either perform the queries for both keys and choose which result to use later, or it can pick one key at random and negate the relation if it picked incorrectly. The latter version is shown to be equivalent to sNM-CCA2 in Appendix G.1.3.

KEM-sNM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,t)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \qquad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k^*\ ()))\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k_0,k^*\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k_0,k^*\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-sNM-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \qquad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x\ k\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ k_0\\ \mathsf{order}(k_0,k^*\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\ \vec{a}\\ c\ \mathsf{order}(k_0,k^*\ ())\ \mathsf{dec\ in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x\ k\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k_0,k^*\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\ \vec{a}\\ c\ \mathsf{order}(k_0,k^*\ ())\ \mathsf{dec\ in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}}$$

PNM

KEM-PNM-CPA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\\ \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,k_0.\ C)\ \vec{a}\ c\ k_0\ \mathsf{in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,k_0.\ C)\ \vec{a}\ c\ (k^*\ ()))\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-PNM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}')] \qquad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,k_0.\ C)\ \vec{a}\ c\ k_0\ \mathsf{in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,k_0.\ C)\ \vec{a}\ c\ (k^*\ ()))\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-PNM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \qquad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,t)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \qquad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ k_0\ \mathsf{in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ (k^*\ ()))\ \mathsf{in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}}$$

KEM-PNM-CCA2

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}{\mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ k_0\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\ \vec{a}\ c\ k_0\ \mathsf{dec}\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\\ \mathsf{dec}\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\ \vec{a}\ c\ (k^*\ ())\ \mathsf{dec}\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\\ \mathsf{dec}\end{array}$$

CNM

KEM-CNM-CPA

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}{\mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k'\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k'\ ())\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k^*\ (),k'\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k^*\ (),k'\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}$$

KEM-CNM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k'\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k'\ ())\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k^*\ (),k'\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k^*\ (),k'\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}$$

Similar to sNM-CCA2, queries depending on k_0 in C' are not technically allowed. However, they can be allowed for the same reason.

KEM-CNM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,t)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ k_0\ \mathsf{order}(k_0,k'\ ())\\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\ \mathsf{order}(k_0,k'\ ())\\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r)\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k^*\ (),k'\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p.\ C)\ \vec{a}\ c\\ \mathsf{order}(k^*\ (),k'\ ())\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\end{array}$$

KEM-CNM-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C',C,\vec{a}',\vec{a},t_r,t_k\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \quad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \quad \mathcal{E};\Theta\vdash[\phi^{r,t_r}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')]\\ \mathcal{E};\Theta\vdash[\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{KEM.Dec}(k\ t_k,\cdot)}(\vec{a})] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')] \quad \mathcal{E};\Theta\vdash[\phi^{k^*,()}_{\mathrm{fresh}}(\vec{a},\vec{a}',C,C')]\end{array}}{}$$

$$\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x\ k\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ k_0\\ \mathsf{order}(k_0,k'\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\ \vec{a}\ c\\ \mathsf{order}(k_0,k'\ ())\ \mathsf{dec\ in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array} \sim \begin{array}{c}\mathsf{let}\ (k_0,c)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r\ t_r);\\ \mathsf{dec}=\lambda x.\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x\ k\\ \mathsf{in}\ (\lambda\vec{v}\,c\,k_0\,p\,r\,\mathsf{dec}.\ C')\ \vec{a}'\ c\ (k^*\ ())\\ \mathsf{order}(k^*\ (),k'\ ())\ (\mathsf{let}\ x=(\lambda\vec{v}\,c\,p\,\mathsf{dec}.\ C)\\ \vec{a}\ c\ \mathsf{order}(k^*\ (),k'\ ())\ \mathsf{dec\ in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}$$

B.2. DEM

B.2.1. IND

DEM-IND-OT

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)]}{\mathcal{E},\Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)}$$

DEM-IND-CPA

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C,\vec{a},m)]}{\mathcal{E},\Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)}$$

DEM-IND-CCA1

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},m)]}{\mathcal{E},\Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)}$$

DEM-IND-CCA2

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,t)}(C)]}{\mathcal{E},\Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)}$$

DEM-IND-CCA2'

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},m)]}{\mathcal{E},\Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ (\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ (\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-IND-OTCCA

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,t)}(C)]}{\mathcal{E},\Theta \vdash \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r) \quad \sim \quad (\lambda\ \vec{v}\ c.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)}$$

DEM-IND-OTCCA'

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{dem-rand}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{fresh}}(C)] \qquad \mathcal{E},\Theta \vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},m)]}{\mathcal{E},\Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ (\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\ \mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ (\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

B.2.2. NM

DEM-NM-OT

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)]}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-NM-CPA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C,C',\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-NM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m\\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C,C')] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-NM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r)\\ \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,t)}(C,C')]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-NM-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m\\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot)}(C,C')] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Enc}(k\ t_k,\cdot,\cdot),\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r);\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec\ in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r);\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec\ in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}}$$

DEM-NM-OTCCA

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m\\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,t)}(C,C')]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r)\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in\ if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\end{array}}$$

DEM-NM-OTCCA'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k)\\ \mathcal{E},\Theta \vdash [\phi^{r,t_r}_{\mathrm{fresh}}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi^{k,t_k}_{\mathrm{fresh}}(C,C')] \qquad \mathcal{E},\Theta \vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{DEM.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r);\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec\ in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{DEM.Enc}(k\ t_k,0^{|m|},r\ t_r);\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\mathrm{DEM.Enc}(k\ t_k,m,r\ t_r)\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec\ in\ if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}}$$

B.3. PKE

B.3.1. IND

PKE-IND-CPA

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C,\vec{a},m)]}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\end{array}}$$

PKE-IND-CCA1

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)]}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\end{array}}$$

PKE-IND-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)]\\ \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,t)}^{\mathrm{guarded}\ k,t_k}(C)]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\end{array}}$$

PKE-IND-CCA2'

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},m)]}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathrm{dec}.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}}$$

B.3.2. NM

PKE-NM-CPA

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C,C',\vec{a},\vec{a}',m)]}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}}$$

PKE-NM-CCA1

$$\frac{\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_1,t_r}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{fresh}}^{r_2,t_r}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k)}^{\mathrm{guarded}\ k,t_k}(C,C')] \qquad \mathcal{E},\Theta\vdash[\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\mathrm{guarded}\ k,t_k}(\vec{a},\vec{a}',m)]}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}}$$

PKE-NM-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\qquad \mathcal{E},\Theta\vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,C',\vec{a},\vec{a}',m)]\qquad \mathcal{E},\Theta\vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,C',\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta\vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\qquad \mathcal{E},\Theta\vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,t)}(C,C')]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}\sim\begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x=c\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}}$$

PKE-NM-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m\\ \mathcal{E},\Theta\vdash \mathrm{const}(t_r)\qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k)\qquad \mathcal{E},\Theta\vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,C',\vec{a},\vec{a}',m)]\\ \mathcal{E},\Theta\vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,C',\vec{a},\vec{a}',m)]\qquad \mathcal{E},\Theta\vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k)}(C,C')]\qquad \mathcal{E},\Theta\vdash [\phi^{\text{guarded } k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}(\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,m,(r_1\ t_r,r_2\ t_r));\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec}\ \mathsf{in}\ \mathsf{if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}\sim\begin{array}{c}\mathsf{let}\ c=\mathrm{PKE.Enc}(k\ t_k,0^{|m|},(r_1\ t_r,r_2\ t_r));\\ \mathsf{dec}=(\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}\ \mathsf{dec}.C')\ \vec{a}'\ c\ m\\ (\mathsf{let}\ x=(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ c\ \mathsf{dec}\ \mathsf{in}\ \mathsf{if}\ x=c\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\ \mathsf{dec}\end{array}}$$

Note that PKE-NM-CCA2 and PKE-NM-CCA2' are mostly provided for completeness, and illustrative purposes. They would be quite cumbersome to use, and it is easy to see that the decryption oracle in PKE-IND-CCA2 (resp. PKE-IND-CCA2') is sufficiently powerful to handle any decryption occurring in PKE-NM-CCA2 (resp. PKE-NM-CCA2').

C. Side Conditions: Restricting the Usage of Names

C.0.1. Generalized Subterms

The following definitions build on the concept of *generalized subterms*[2] of a term t. Intuitively, this describes all subterms that may be encountered when computing t, including subterms of defined functions in the environment. However, generalized subterms are also equipped with a formula ψ that captures the conditions under which the subterm will be encountered, and a vector of bound variables $\vec{\alpha}$.

Generalized subterms are defined by the following function:

$$\mathcal{ST}_{\mathcal{E}}(x) \coloneqq \begin{cases}\{(\epsilon,\mathtt{true},x)\} & \text{when } (x:\tau)\in\mathcal{E} \text{ or } x\notin\mathcal{E}\\ \mathcal{ST}_{\mathcal{E}}(t) & \text{when } (x:\tau=t)\in\mathcal{E}\end{cases}$$

$$\mathcal{ST}_{\mathcal{E}}(t\ t') \coloneqq \begin{cases}\mathcal{ST}_{\mathcal{E}}(t_0\{y\mapsto t'\}) & \text{when } t=x \text{ and } (x:\tau=\lambda y.t_0)\in\mathcal{E}\\ \{(\epsilon,\mathtt{true},(t\ t'))\}\cup\mathcal{ST}_{\mathcal{E}}(t)\cup\mathcal{ST}_{\mathcal{E}}(t') & \text{otherwise}\end{cases}$$

$$\mathcal{ST}_{\mathcal{E}}(\lambda(x:\tau)\tau) \coloneqq \{(\epsilon,\mathtt{true},\lambda(x:\tau)\tau)\}\cup(x:\tau).\mathcal{ST}_{\mathcal{E}}(t)$$

$$\mathcal{ST}_{\mathcal{E}}(\mathsf{if}\ \phi\ \mathsf{then}\ t_1\ \mathsf{else}\ t_0) \coloneqq \{(\epsilon,\mathtt{true},\mathsf{if}\ \phi\ \mathsf{then}\ t_1\ \mathsf{else}\ t_0)\}\cup\mathcal{ST}_{\mathcal{E}}(\phi)\cup[\phi]\mathcal{ST}_{\mathcal{E}}(t_1)\cup[\phi]\mathcal{ST}_{\mathcal{E}}(t_0)$$

$$[\phi]S \coloneqq \{(\vec{\alpha},\psi\wedge\phi,t)\mid(\vec{\alpha},\psi,t)\in S\}$$

$$(x:\tau).S \coloneqq \{((\vec{\alpha},x:\tau),\psi,t)\mid(\vec{\alpha},\psi,t)\in S\}$$

C.1. Guarded Occurrences

As seen in Section 3, axioms often require that some random sampling $n\ t_n$ may only be used in certain specific contexts. We say that these contexts *guard* the occurrence of the name. Importantly, however, it is not sufficient to check this purely syntactically, as different index terms can have the same value, and thus refer to the same random sampling.

Instead, we represent such conditions using a formula $\phi_{\mathbb{C}}^{\text{guarded } n,t_n}$, where $\mathbb{C}$ is a set of guarding contexts with no overlap. $\phi_{\mathbb{C}}^{\text{guarded } n,t_n}$ is actually not uniquely defined, but instead deliberately allowed to represent any formula satisfying certain requirement. We will give a suitable construction below, however, this will be an overapproximation, so it may be necessary to use a more precise formula in some cases.

To formally state the formal requirements for $\phi_{\mathbb{C}}^{\text{guarded } n,t_n}$, we need to define *generalized subterms with exclusions*. We thus define a function $\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(t)$, where $\mathcal{E}$ is an environment, n, t_n are the name and index to be guarded, and $\mathbb{C}_1$ and $\mathbb{C}_2$ are two sets of contexts. Contexts in $\mathbb{C}_1$ will only apply to direct subterms, whereas those in $\mathbb{C}_2$ also apply to subterms in the environment. Its definition is analogous to that of generalized subterms, except for the following cases:

$$\begin{aligned}
\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(\underline{t}) &:= \emptyset \\
\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(C[\vec{v}, n\ t_0]) &:= \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(C[\vec{v}, \underline{n\ t_0}]) \\
&\quad \cup \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(t_0) \\
&\quad \cup [t_0 \neq t_n]\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(n\ t_0) \\
&\quad \text{for } C \in \mathbb{C}_0, \mathbb{C}_1 \\
\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(x) &:= \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C}_2,n,t\ n}(t) \quad \text{when } (x : \tau = t \in \mathcal{E})
\end{aligned}$$

This definition makes use of *marked terms* $\underline{t}$ to denote those occurrences of names that should be skipped. These are treated as distinct for the purposes of pattern matching (for example, in the $C[\vec{v}, n\ t_0]$ case, a marking is applied to the subterms $n\ t_0$, which prevents this case from applying again on the recursive call). However, all markings are removed again in the final result.

$\phi_{\mathbb{C}}^{\text{guarded } n,t_n}(t)$ can represent any formula which guarantees that, for every model $\mathbb{M} : \mathcal{E}$ of Θ, for every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$,

$$\begin{aligned}
&[\![[\phi_{\mathbb{C}}^{\text{guarded } n,t_n}(t)]]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \implies \\
&\quad \forall(\vec{\alpha}, \psi, n\ t_0) \in \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}(t), \\
&\quad\quad [\![\forall\vec{\alpha}, \psi \implies t_n \neq t_0]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \qquad (*)
\end{aligned}$$

Note that $\forall\vec{\alpha}$ quantifies over all possible values of the variables in $\vec{a}$ (by syntactically inserting $\vec{a}$ into the formula).

Baelde, Koutsos, and Lallemand [2] provide a suitable construction for such a formula: Let ${\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}}^{0}(\cdot)$ denote the restriction of $\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}(\cdot)$ to only direct subterms, which we obtain by changing the case for defined variables (${\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}}^{0}(x) := \emptyset$ when $(x : \tau = t \in \mathcal{E})$). By applying ${\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}}^{0}$ to t as well as all definitions in the environment, one obtains a finite set of occurrences from which a formula $\phi_{\mathbb{C}}^{\text{guarded } n,t_n}$ can be constructed by taking the conjunction of their corresponding formulas (*). This yields a suitable formula, since no generalized subterm $(\alpha, \psi, t') \in \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}(t)$ can have a weaker condition ψ than a term in this construction.

This construction is a generalized version of constructions used by Baelde, Koutsos, and Lallemand [2] on a case-by-case basis. Thus, $\phi_{\text{fresh}}^{n,t_n}$ can be seen as the special case $\phi_{\emptyset}^{\text{guarded } n,t_n}$ with no guarding expressions.

C.1.1. Subcontexts of Guarding Contexts

Consider a context $C[\vec{v}, n\ t_n] = C_1[\vec{v}_1, C_0[\vec{v}_0, n\ t_n]]$ and a set of contexts $\mathbb{C}$ such that C does not overlap with any contexts in C. We wish to show that a formula $\phi_{\mathbb{C}\cup\{C\}}^{\text{guarded } n,t_n}(\vec{t})$ is also a valid formula $\phi_{\mathbb{C}\cup\{C_0\}}^{\text{guarded } n,t_n}(\vec{t})$.

As a first step, note that

$$\forall t, \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C\},\mathbb{C},n,t_n} = \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C_0\},\mathbb{C},n,t_n}$$

is easily provable by induction on t. The proof requires unfolding all of C_1 in the $C[\vec{v}, n\ t_0]$ case on the left, until the $C_0[\vec{v}_0, n\ t_0]$ case matches on the right, allowing induction to proceed. All subterms produced during this will match.

However, we need to prove the following:

$$\forall t, \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C}\cup\{C\},n,t_n} = \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C}\cup\{C_0\},n,t_n}$$

This can be shown by realizing that the set $\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\mathbb{C}_1,\mathbb{C}_2,n,t_n}(t)$ can be defined inductively. It can then be shown by induction that $\forall t, \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C}\cup\{C\},n,t_n} \subseteq \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C}\cup\{C_0\},n,t_n}$ and vice versa.

C.1.2. Rewriting to Remove Guards

In Section 3.2 and Appendix B, we give two different versions for CCA2 axioms. In order to show that these are equivalent, we need a method of rewriting a term in a way that fully removes one of the guarding expressions and substitutes a variable instead. We thus define a function $\mathsf{extr}_{C,n,t_n,f}(t)$ that extracts a guard C:

$$\mathsf{extr}_{C,n,t_n,f}(C[\vec{v}, n\ t_0]) := \mathsf{if}\ t_0 = t_n\ \mathsf{then}\ f\ (\mathsf{extr}_{C,n,t_n,f}(\vec{v}))\ \mathsf{else}\ C[\mathsf{extr}_{C,n,t_n,f}(\vec{v}), n\ t_0]$$

In all other cases, $\mathsf{extr}_{C,n,t_n,f}(t)$ is simply applied recursively to all subterms.

For this extraction function, we can show that

$$\forall t, \forall(\vec{\alpha}, \psi, n\ t_0) \in \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}(\mathsf{extr}_{C,n,t_n,f}(t)), \exists(\vec{\alpha}', \psi', n\ t'_0) \in \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C\},\mathbb{C},n,t_n}(t),$$
$$\forall \mathbb{M} : \mathcal{E}, \forall \eta \in \mathbb{N}, \forall \rho \in \mathbb{T}_{\mathbb{M},\eta}, [\![\forall \vec{\alpha}.\psi \implies (\exists \vec{\alpha}'.\psi_1 \wedge t_0 = t'_0)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1$$

Intuitively, this states that every occurrence s in $\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\emptyset,\mathbb{C},n,t_n}(\mathsf{extr}_{C,n,t_n,f}(t))$ is subsumed by an occurrence s' in $\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C\},\mathbb{C},n,t_n}(t)$ in the sense that if property (*) holds for s', then it must also hold for s.

This property can be shown by induction on t. In the case of $C[\vec{v}, n\ t_0]$, note that the subterm introduced by the case distinction can be ignored, as it is not of the form $n\ t_0$. Any subterms of $\vec{v}$ occur twice after the extraction, once with additional condition $t_n = t_0$ and once with $t_n \neq t_0$. In either case, the subsumption is trivial, as $\psi \wedge t_n = t_0 \implies \psi$. Finally, the subterms of $n\ t_0$ which occur in the else case after extraction are subsumed by the subterms $[t_0 \neq t_n]\mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C\},\mathbb{C},n,t_n}(n\ t_0) \subset \mathcal{ST}^{\mathsf{excl}}_{\mathcal{E},\{C\},\mathbb{C},n,t_n}(C[\vec{v}, n\ t_0])$.

Note, however, that this result only applies to guarding contexts that only apply within direct subterms, not within the environment. This distinction is necessary since we can not rewrite within the environment. Luckily, the guarding contexts used in our CCA2 axioms include variables that are bound in the term. These contexts can thus not appear in the environment, as an occurrence of unbound variables contradicts the well-formedness of the environment.

C.2. Restricted Reuse

In Section 3.2.1, we saw that we need to restrict the use of random samplings for encryptions in a different way: Since the encryption is performed by an oracle in the computational model, any random sampling can only be used to encrypt one message (although the corresponding subterm may occur multiple times). We will use a similar approach to define this condition, specifying a condition on generalized subterms which a formula $\phi^{k,t_k}_{\mathrm{dem-rand}}(t)$ must guarantee. The main difference to the previous construction is that subterms need to be compared pairwise.

Concretely, $\phi^{k,t_k}_{\mathrm{dem-rand}}(t)$ may be any formula such that, for every model $\mathbb{M} : \mathcal{E}$ of Θ, for every $\eta \in \mathbb{N}$ and $\rho \in \mathbb{T}_{\mathbb{M},\eta}$,

$$[\![[\phi^{k,t_k}_{\mathrm{dem-rand}}(t)]]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \implies$$
$$\forall(\vec{\alpha}, \psi, \mathrm{DEM.Enc}(k\ t_0, m, r\ t_r)), (\vec{\alpha}', \psi', \mathrm{DEM.Enc}(k\ t'_0, m', r\ t'_r)) \in \mathcal{ST}_{\mathcal{E}}(t),$$
$$[\![\forall(\vec{\alpha} \uplus \vec{\alpha}'), \psi \wedge \psi' \implies (t_0 \neq t_k \wedge t'_0 \neq t_k) \vee m = m' \vee t_r \neq t'_r]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}} = 1 \qquad (\dagger)$$

Here, we use $\uplus$ to denote disjoint union, using α-renaming to make the two sets disjoint.

Similar to $\phi_{\mathbb{C}}^{\text{guarded } n,t_n}(\cdot)$, we can construct a suitable formula $\phi_{\text{dem}-\text{rand}}^{k,t_k}(t)$ by taking only the direct generalized subterms of t and the environment, then using the conjunction of the formulae (†) for each pair.

D. Equivalence of CCA2 Axioms

In Section 3.2 and Appendix B, we introduced two versions of each CCA2 axiom, one where the decryption queries are allowed to appear in the context. Here, we demonstrate that these versions are equivalent on the example of PKE-IND-CCA2 and PKE-IND-CCA2'.

PKE-IND-CCA2

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m\\ \mathcal{E},\Theta\vdash \text{const}(t_r)\quad \mathcal{E},\Theta\vdash \text{const}(t_k)\quad \mathcal{E},\Theta\vdash [\phi_{\text{fresh}}^{r_1,t_r}(C,\vec{a},m)]\quad \mathcal{E},\Theta\vdash [\phi_{\text{fresh}}^{r_2,t_r}(C,\vec{a},m)]\\ \mathcal{E},\Theta\vdash [\phi_{\text{pk}(k\ t_k),\text{PKE.Dec}(k\ t_k,\cdot)}^{\text{guarded } k,t_k}(\vec{a},m)]\quad \mathcal{E},\Theta\vdash [\phi_{\text{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t)}^{\text{guarded } k,t_k}(C)]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\end{array}\sim \begin{array}{c}(\lambda\ \vec{v}\ c.C)\ \vec{a}\\ \text{PKE.Enc}(\text{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\end{array}}$$

PKE-IND-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m\quad \mathcal{E},\Theta\vdash \text{const}(t_r)\quad \mathcal{E},\Theta\vdash \text{const}(t_k)\quad \mathcal{E},\Theta\vdash [\phi_{\text{fresh}}^{r_1,t_r}(C,\vec{a},m)]\\ \mathcal{E},\Theta\vdash [\phi_{\text{fresh}}^{r_2,t_r}(C,\vec{a},m)]\quad \mathcal{E},\Theta\vdash [\phi_{\text{pk}(k\ t_k)}^{\text{guarded } k,t_k}(C)]\quad \mathcal{E},\Theta\vdash [\phi_{\text{pk}(k\ t_k),\text{PKE.Dec}(k\ t_k,\cdot)}^{\text{guarded } k,t_k}(\vec{a},m)]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\\ \text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,x))\end{array}\sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\\ \text{PKE.Enc}(\text{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \text{PKE.Enc}(\text{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,x))\end{array}}$$

D.1. PKE-IND-CCA2 $\implies$ PKE-IND-CCA2'

This direction is relatively straightforward, as $\phi_{\text{pk}(k\ t_k)}^{\text{guarded } k,t_k}(C)$ implies

$$\phi_{\text{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t)}^{\text{guarded } k,t_k}\left(\begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{v}\ c\\ (\lambda x.\mathsf{if}\ x=\text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\ \mathsf{then}\ \bot\\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,x))\end{array}\right)$$

Thus, we can simply add the explicit decryption oracle to the context by rewriting with β-equivalence, allowing us to derive PKE-IND-CCA2' from PKE-IND-CCA2.

D.2. PKE-IND-CCA2' $\implies$ PKE-IND-CCA2

For this direction, we make use of the extraction function defined in Appendix C.1.2.

Note that $(\lambda\ \vec{v}\ c.C)\ \vec{a}\ \text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))$ and

$$\begin{array}{r}(\lambda\ \vec{v}\ c\ \mathsf{dec}.\mathsf{extr}_{\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t),k,t_k,\mathsf{dec}}(C))\ \vec{a}\ \text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,x))\end{array}$$

are β-equivalent (since $c=\text{PKE.Enc}(\text{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))$).

Further, the context $\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t)$ contains the variable c, which is not bound in the environment, and can thus only appear in direct subterms of C. Thus, we know that

$$\mathcal{ST}_{\mathcal{E},\emptyset,\{\text{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t)\},k,t_k}^{\mathsf{excl}}(C)=\mathcal{ST}_{\mathcal{E},\{\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \text{PKE.Dec}(k\ t_k,t)\},\{\text{pk}(k\ t_k)\},k,t_k}^{\mathsf{excl}}(C)$$

This allows us to conclude that all subterms of $\mathsf{extr}_{\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,t),k,t_k,\mathsf{dec}}(C)$ satisfy the required condition for $\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathsf{if}\ t=c\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,t)}(C)$ according to Appendix C.1.2. Therefore, we can derive PKE-IND-CCA2 from PKE-IND-CCA2'.

E. KEM/DEM Security Results in CCSA

E.1. PKE-IND-CCA2 from KEM-IND-CCA2 and DEM-IND-OTCCA

In this section, we reproduce a result from Herranz, Hofheinz, and Kiltz [17] for our definitions in CCSA. We will use the versions of the axioms where the oracle is explicitly given as a term. We want to prove the following:

PKE-IND-CCA2'

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset\vdash_{\mathsf{pptm}} C,\vec{a},t_k,t_r,m \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta\vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta\vdash [\phi^{r_1,t_r}_{\mathrm{fresh}}(C,\vec{a},m)]\\ \mathcal{E},\Theta\vdash [\phi^{r_2,t_r}_{\mathrm{fresh}}(C,\vec{a},m)] \qquad \mathcal{E},\Theta\vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k)}(C)] \qquad \mathcal{E},\Theta\vdash [\phi^{\mathrm{guarded}\ k,t_k}_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}(\vec{a},m)]\end{array}}{\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),0^{|m|},(r_1\ t_r,r_2\ t_r))\\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k,x))\end{array}}$$

Unfolding the definitions of PKE.Enc and PKE.Dec, we have the following:

$$\mathcal{E},\Theta\vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\\ \mathsf{let}\ (sk,c_1)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r_1\ t_r)\\ \mathsf{in}\ (c_1,\mathrm{DEM.Enc}(sk,m,r_2\ t_r))\ (\lambda x.\mathsf{if}\ x=\\ \mathsf{let}\ (sk,c_1)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r_1\ t_r)\\ \mathsf{in}\ (c_1,\mathrm{DEM.Enc}(sk,m,r_2\ t_r))\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathsf{let}\ sk=\mathrm{KEM.Dec}(k\ t_k,\pi_1(x))\\ \mathsf{in}\ \mathrm{DEM.Dec}(sk,\pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\\ \mathsf{let}\ (sk,c_1)=\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r_1\ t_r)\\ \mathsf{in}\ (c_1,\mathrm{DEM.Enc}(sk,0^{|m|},r_2\ t_r))\\ (\lambda x.\mathsf{if}\ x=\mathsf{let}\ (sk,c_1)=\\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k),r_1\ t_r)\\ \mathsf{in}\ (c_1,\mathrm{DEM.Enc}(sk,0^{|m|},r_2\ t_r))\ \mathsf{then}\ \bot\\ \mathsf{else}\ \mathsf{let}\ sk=\mathrm{KEM.Dec}(k\ t_k,\pi_1(x))\\ \mathsf{in}\ \mathrm{DEM.Dec}(sk,\pi_2(x)))\end{array}$$

As a basic proof idea, the indistinguishability of the two ciphertexts should follow from DEM-IND-OTCCA. However, we cannot apply this directly, as we are currently not using a fresh random key, but one obtained from the KEM – if the KEM is insecure and the attacker can gain information about the key, DEM-IND-OTCCA does not apply. Thus, we first need to apply KEM-IND-CCA2' to exchange the key with a fresh random one. However, before we can do so, we first need to ensure that KEM decryption is properly guarded.

This requires some care, as the PKE-IND-CCA2' oracle permits queries for ciphertexts where the key part is the same as in the challenge ciphertext, and only the message part is changed. Before we can apply KEM-IND-CCA2', we need to handle such cases without querying the KEM decryption oracle.

Using β-equivalence and KEM correctness, we can rewrite

$$(\lambda x.\mathsf{if}\ x = \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{then}\ \bot \quad \mathsf{else\ let}\ sk = \mathrm{KEM.Dec}(k\ t_k, \pi_1(x))\ \mathsf{in}\ \mathrm{DEM.Dec}(sk, \pi_2(x))) \tag{E.1}$$

$$= (\lambda x.\mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{then}\ \bot \quad \mathsf{else\ if}\ \pi_1(x) = c_1\ \mathsf{then\ let}\ sk' = \mathrm{KEM.Dec}(k\ t_k, c_1)\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)) \quad \mathsf{else\ let}\ sk' = \mathrm{KEM.Dec}(k\ t_k, \pi_1(x))\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x))) \tag{E.2}$$

$$= (\lambda x.\mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{then}\ \bot \quad \mathsf{else\ if}\ \pi_1(x) = c_1\ \mathsf{then\ let}\ sk' = sk\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)) \quad \mathsf{else\ let}\ sk' = \mathrm{KEM.Dec}(k\ t_k, \pi_1(x))\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x))) \tag{E.3}$$

$$= (\lambda x.\mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{then}\ \bot \quad \mathsf{else\ if}\ \pi_1(x) = c_1\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x))\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \quad \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x))\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x))) \tag{E.4}$$

In (E.2), we introduce case distinctions to isolate this case. (E.3) then uses KEM correctness to remove the decryption query. Finally, in (E.4), we can safely introduce the guard for the KEM decryption oracle. Since this branch already ensures $x \neq c_1$, no failure can occur.

After rewriting, we now have the following:

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ (sk, c_1) = \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array}$$

We can now rewrite this into the shape required for KEM-IND-CCA2':

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ (sk, c_1)\ \mathsf{dec}'.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{dec}\ \pi_1(x), \pi_2(x)))) \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ (\lambda c.\mathsf{if}\ c = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, c))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ (sk, c_1)\ \mathsf{dec}'.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{dec}\ \pi_1(x), \pi_2(x)))) \\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ (\lambda c.\mathsf{if}\ c = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, c))\end{array}$$

At this point, we can apply transitivity and KEM-IND-CCA2' to exchange the key produced by KEM.Enc for a fresh key sk^* ():

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ (sk, c_1)\ \mathsf{dec}'.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{dec}\ \pi_1(x), \pi_2(x)))) \\ (sk^*\ (), \pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ (\lambda c.\mathsf{if}\ c = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, c))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ (sk, c_1)\ \mathsf{dec}'.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{dec}\ \pi_1(x), \pi_2(x)))) \\ (sk^*\ (), \pi_2\ \mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ (\lambda c.\mathsf{if}\ c = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, c))\end{array}$$

We can now undo the rewriting we performed in order to apply KEM-IND-CCA2'.

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ \mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\ \mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathrm{DEM.Dec}(sk, \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array}$$

We now need to guard the DEM decryption in order to apply DEM-IND-OTCCA'. Note that we only need to guard the first occurrence, as the second occurrence uses the key obtained from $\mathrm{KEM.Dec}(k\ t_k, \pi_1(x))$, which is independent of $sk^*\ ()$. This can be done by rewriting using β-equivalence, as this branch already ensures $\pi_2(x) \neq \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r)$. We thus obtain the following:

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a}\ \mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then\ if}\ \pi_2(x) = \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), \pi_2(x))) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c\mathsf{dec}.C)\ \vec{a}\ \mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ (\lambda x.\mathsf{let}\ (sk, c_1) = \\ (sk^*\ (), \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r))) \\ \mathsf{in\ if}\ x = (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r)) \\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1\ \mathsf{then\ if}\ \pi_2(x) = \\ \mathrm{DEM.Enc}(sk^*\ (), 0^{|m|}, r_2\ t_r)\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), \pi_2(x)) \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', \pi_2(x)))\end{array}$$

With this guard in place, we can rewrite both sides into the shape required for DEM-IND-OTCCA':

$$\mathcal{E}, \Theta \vdash \begin{array}{c}(\lambda\ \vec{v}\ c_2\ \mathsf{dec}.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ c_1 = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ \mathsf{in}\ (c_1, c_2) \\ (\lambda x.\mathsf{let}\ c_1 = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ \mathsf{in\ if}\ x = (c_1, c_2)\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathsf{dec}(\pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)), \pi_2(x)))) \\ \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r) \\ (\lambda x.\mathsf{if}\ x = \mathrm{DEM.Enc}(sk^*(), m, r_2\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), x)\end{array} \sim \begin{array}{c}(\lambda\ \vec{v}\ c_2\ \mathsf{dec}.(\lambda\ \vec{v}\ c\ \mathsf{dec}.C)\ \vec{a} \\ \mathsf{let}\ c_1 = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ \mathsf{in}\ (c_1, c_2) \\ (\lambda x.\mathsf{let}\ c_1 = \pi_2(\mathrm{KEM.Enc}(\mathrm{pk}(k\ t_k), r_1\ t_r)) \\ \mathsf{in\ if}\ x = (c_1, c_2)\ \mathsf{then}\ \bot\ \mathsf{else\ if}\ \pi_1(x) = c_1 \\ \mathsf{then}\ \mathsf{dec}(\pi_2(x)) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathsf{if}\ \pi_1(x) = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, \pi_1(x)), \pi_2(x)))) \\ \mathrm{DEM.Enc}(sk^*\ (), 0^{|m|}, r_2\ t_r) \\ (\lambda x.\mathsf{if}\ x = \mathrm{DEM.Enc}(sk^*(), 0^{|m|}, r_2\ t_r) \\ \mathsf{then}\ \bot\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), x)\end{array}$$

Now, the proof conludes using DEM-IND-OTCCA'.

However, one detail we have thus far neglected are the side conditions $\phi^{\text{guarded } k, t_k}_{\mathrm{pk}(k\ t_k), \mathrm{KEM.Dec}(k\ t_k, \cdot)}(C)$ and $\phi^{\text{guarded } sk^*, ()}_{\mathrm{DEM.Dec}(sk^*\ (), \cdot)}(\vec{a}, m)$ and $\phi^{sk^*, ()}_{\mathrm{dem-rand}}(C, \vec{a}, m)$. The latter two are easy to resolve, as the key $sk^*\ ()$ was chosen to be entirely fresh in $C, \vec{a}$ and m, so we only need to check the occurrences we have introduced in the proof. For the former, however, we need to prove that any formula $\phi^{\text{guarded } k, t_k}_{\mathrm{pk}(k\ t_k), \mathrm{PKE.Dec}(k\ t_k), \cdot)}(C)$ is also a valid formula $\phi^{\text{guarded } k, t_k}_{\mathrm{pk}(k\ t_k), \mathrm{KEM.Dec}(k\ t_k), \cdot)}$. As we show in Appendix C.1.1, this is indeed the case, since $\mathrm{KEM.Dec}(k\ t_k, \cdot)$ is a subcontext of $\mathrm{PKE.Dec}(k\ t_k, \cdot)$.

E.2. PKE-NM-CCA1 from KEM-PNM-CCA1 and DEM-NM-OT

We want to prove the following:

PKE-NM-CCA1

$$\frac{\begin{array}{c}\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',\vec{a},\vec{a}',t_k,t_r,m \\ \mathcal{E},\Theta \vdash \mathrm{const}(t_r) \qquad \mathcal{E},\Theta \vdash \mathrm{const}(t_k) \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_1,t_r}(C,C',\vec{a},\vec{a}',m)] \\ \mathcal{E},\Theta \vdash [\phi_{\mathrm{fresh}}^{r_2,t_r}(C,C',\vec{a},\vec{a}',m)] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k)}^{\text{guarded } k,t_k}(C,C')] \qquad \mathcal{E},\Theta \vdash [\phi_{\mathrm{pk}(k\ t_k),\mathrm{PKE.Dec}(k\ t_k,\cdot)}^{\text{guarded } k,t_k}(\vec{a},\vec{a}',m)]\end{array}}{\mathcal{E},\Theta \vdash \begin{array}{c}\mathsf{let}\ c = \mathrm{PKE.Enc}(k\ t_k, m, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m \\ (\mathsf{let}\ x = (\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x = c\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array} \sim \begin{array}{c}\mathsf{let}\ c = \mathrm{PKE.Enc}(k\ t_k, 0^{|m|}, (r_1\ t_r, r_2\ t_r)) \\ \mathsf{in}\ (\lambda\ \vec{v}\ c\ m\ \mathbf{M}.C')\ \vec{a}'\ c\ m \\ (\mathsf{let}\ x = (\lambda\ \vec{v}\ c.C)\ \vec{a}\ c\ \mathsf{in}\ \mathsf{if}\ x = c\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{PKE.Dec}(k\ t_k, x))\end{array}}$$

Unfolding the definitions of PKE.Enc and PKE.Dec, we have the following:

$$\begin{array}{c}\mathsf{let}\ (c_1,c_2) = \mathsf{let}\ (sk,c_1) = \\ \mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk = \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk, x_2))\end{array} \sim \begin{array}{c}\mathsf{let}\ (c_1,c_2) = \mathsf{let}\ (sk,c_1) = \\ \mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r) \\ \mathsf{in}\ (c_1, \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk = \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk, x_2))\end{array}$$

Similar to the previous proof, we first need to guard the KEM decryption. This can be achieved using β-equivalence and KEM correctness, by introducing special handling for ciphertexts (x_1, x_2) where $x_1 = c_1$.

$$\begin{array}{c}\mathsf{let}\ (sk,c_1) = \mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk, m, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ x_1 = c_1\ \mathsf{then}\ sk \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', x_2))\end{array} \sim \begin{array}{c}\mathsf{let}\ (sk,c_1) = \mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk, 0^{|m|}, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ x_1 = c_1\ \mathsf{then}\ sk \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', x_2))\end{array}$$

We can now apply KEM-PNM-CCA1 to replace the symmetric key with a fresh one $sk^*\ ()$ (omitting a step of using β-equivalence to obtain the proper structure of the term).

$$\begin{array}{c}\mathsf{let}\ c_1 = \pi_2\mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ x_1 = c_1\ \mathsf{then}\ sk^*\ () \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', x_2))\end{array} \sim \begin{array}{c}\mathsf{let}\ c_1 = \pi_2\mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk^*\ (), 0^{|m|}, r_2\ t_r))\ \mathsf{in}\ C' \\ \vec{a}'\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ let}\ sk' = \mathsf{if}\ x_1 = c_1\ \mathsf{then}\ sk^*\ () \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{KEM.Dec}(k\ t_k, x_1) \\ \mathsf{in}\ \mathrm{DEM.Dec}(sk', x_2))\end{array}$$

Now, we need to introduce a guard for DEM decryption. Again, this is possible using β-equivalence, since the existing branches already enforce this condition.

$$\begin{array}{c}\mathsf{let}\ c_1 = \pi_2\mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk^*\ (), m, r_2\ t_r))\ \mathsf{in}\ C'\ \vec{a}' \\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then\ if}\ x_2 = c_2\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), x_2) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathrm{KEM.Dec}(k\ t_k, x_1), x_2))\end{array} \sim \begin{array}{c}\mathsf{let}\ c_1 = \pi_2\mathrm{KEM.Enc}((\mathrm{pk}(k\ t_k), r_1\ t_r); \\ c_2 = \mathrm{DEM.Enc}(sk^*\ (), 0^{|m|}, r_2\ t_r))\ \mathsf{in}\ C' \\ \vec{a}'\ (c_1,c_2)\ m\ (\mathsf{let}\ (x_1,x_2) = C\ \vec{a}\ (c_1,c_2) \\ \mathsf{in\ if}\ (x_1,x_2) = (c_1,c_2)\ \mathsf{then}\ \bot \\ \mathsf{else\ if}\ x_1 = c_1\ \mathsf{then\ if}\ x_2 = c_2\ \mathsf{then}\ \bot \\ \mathsf{else}\ \mathrm{DEM.Dec}(sk^*\ (), x_2) \\ \mathsf{else}\ \mathrm{DEM.Dec}(\mathrm{KEM.Dec}(k\ t_k, x_1), x_2))\end{array}$$

The proof now concludes using DEM-NM-OT. As in the previous proof, the side conditions for DEM-NM-OT are easy to verify since sk^* () was chosen entirely fresh. The side conditions for KEM-PNM-CCA1 follow from the fact that any formula $\phi^{\text{guarded } k,t_k}_{\text{pk}(k\ t_k),\text{PKE.Dec}(k\ t_k),\cdot)}(C)$ is also a valid formula $\phi^{\text{guarded } k,t_k}_{\text{pk}(k\ t_k),\text{KEM.Dec}(k\ t_k),\cdot)}$, as shown in the previous proof.

F. Soundness of KEM/DEM Axioms

F.1. KEM-IND-CCA2

If KEM is not secure in the sense of our axiom, it is not secure in the sense of the IND-CCA2 game.

Assume an adversary $\mathcal{A}$ against the indistinguishability in the axiom. We construct an adversary $\mathcal{A}'$ against the IND-CCA2 game such that the advantage of $\mathcal{A}'$ is exactly the advantage of $\mathcal{A}$.

$\mathcal{A}'$ consists of two phases. The first phase, $\mathcal{A}_1$, is given the public key pk $(k\ t_k)$ and has access to a decryption oracle. Its purpose is to compute $[\![\vec{a}]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$.

The second phase, $\mathcal{A}_2$, is given the result of the first phase, the public key, a symmetric key and the ciphertext. It first computes the result of the term and then simulates $\mathcal{A}$ on it.

Both phases rely on a recursive procedure I_σ which, given a term u with free variables in $\mathcal{E} \cup \vec{\alpha}$ s.t. $\mathcal{E}, \Theta; \vec{\alpha} \vdash_{\mathsf{pptm}} u$ and a semantic assignment σ of domain $\vec{\alpha}$, computes $[\![u]\!]^{\eta,\rho}_{\mathbb{M}[\vec{\alpha}\mapsto\sigma(\alpha)]:(\mathcal{E},\vec{\alpha})}$. For the most part, the procedures proceeds in the obvious way (following the derivation of $\mathcal{E}, \Theta; \vec{\alpha} \vdash_{\mathsf{pptm}} u$, similar to the one described by Baelde, Koutsos, and Lallemand [2]), except for terms which need to be replaced with oracle queries. Note that defined variables in $\mathcal{E}$ must be adversary-computable, since $\mathcal{E}, \Theta; \vec{\alpha} \vdash_{\mathsf{pptm}} u$ is derivable.

The following cases need to be handled differently:

- *Random sampling case*: When u is of the form $r\ t$, compute $I_\sigma(t)$ and $I_\sigma(t_r)$. If they are equal, fail, otherwise sample as normal.
- *Key case*: When u is of the form $k\ t$, compute $I_\sigma(t)$ and $I_\sigma(t_k)$. If they are equal, fail, otherwise sample the tape for $k\ t$ as normal.
- *Public key case*: When u is of the form $pk\ (k\ t)$, compute $I_\sigma(t)$ and $I_\sigma(t_k)$. If they are equal, use the public key that was provided as the input. Otherwise, proceed with calculating the key as normal.
- *Decryption case*: When u is of the form KEM.Dec c $(k\ t)$, compute the indices $I_\sigma(t)$ and $I_\sigma(t_k)$. If they are equal, query the decryption oracle (note that in $\mathcal{A}_2$, this can fail). Otherwise, proceed as normal.

Note that this construction allows abstractions which require oracle queries to evaluate. This is not a problem, since

- within a phase, the oracle is available whenever the abstraction is called
- $\vec{a}$ is of order 0, so the final result of $\mathcal{A}_1$ can not contain any abstractions which would need to be evaluated in $\mathcal{A}_2$
- if C' is an abstraction, it might be called by $\mathcal{A}$. However, this can be simulated by $\mathcal{A}_2$ using the oracle.

The construction thus has two failure conditions: It will fail when trying to compute $r\ t_r$ or $k\ t_k$ or when attempting to decrypt the given ciphertext in $\mathcal{A}_2$.

However, the conditions of the axioms enforce that

- *Random sampling case*: $r\ t_r$ may not occur in the term at all and will thus not be computed.
- *Key case*: $k\ t_k$ may not appear in the term except under pk or KEM.Dec, in which case the other exceptions make sure that $k\ t_k$ is not computed.

- *Decryption case*: In the second phase, decryption fails when attempting to decrypt the original ciphertext. According to our conditions, any occurrence of KEM.Dec in the second phase must be under a guard which enforces that no query is made for the original ciphertext, so no failures can occur.

The axiom requires $\mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} C, \vec{a}, t_r, t_k$, which enforces that these procedures can compute the evaluation of these terms in polynomial time. It therefore follows that $\mathcal{A}'$ is an adversary against the IND-CCA2 game whose advantage is exactly the advantage of $\mathcal{A}$ against the indistinguishability.

If KEM is secure in the sense of the IND-CCA2 game, it follows that this advantage must be negligible; thus, KEM is also secure in the sense of our axiom.

F.2. KEM-sNM-CCA1

If KEM is not secure in the sense of our axiom, it is not secure in the sense of the sNM-CCA1 game.

Assume an adversary $\mathcal{A}$ against the indistinguishability in the axiom. We construct an adversary $\mathcal{A}'$ against the sNM-CCA1 experiment such that the advantage of $\mathcal{A}'$ is exactly that of $\mathcal{A}$.

$\mathcal{A}'$ has two phases, $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1$ computes $\vec{a}$ and $\vec{a}'$ using the decryption oracle if needed. $\mathcal{A}_2$ is given the result of the first phase, the ciphertext and the ordered pair of sk and sk^* and uses them to compute $[\![(\lambda \vec{v}\, c\, p.C)\ \vec{a}\ c\ (\mathsf{order}\ sk\ sk^*)]\!]^{\eta,\rho}_{\mathbb{M}:\mathcal{E}}$ as well as a relation that, given sk or sk^* and the decryption of the previous term, computes the result of C'. Note that all other inputs to C' are available in the second phase and can therefore be "hard-coded".

$\mathcal{A}_1$ proceeds according to the same procedure $I_\sigma(u)$ as in the previous section. $\mathcal{A}_2$ requires some changes, since it does not have access to a decryption oracle. Therefore, the decryption case needs to be changed such that if the key index is equal to t_k, the procedure will just fail. Since the conditions of the axiom do not allow decryption with $k\ t_k$ in C and C', such a failure will not occur.

As in the previous proof, abstractions which require oracle queries to evaluate are allowed in $\mathcal{A}_1$. However, since both $\vec{a}$ and $\vec{a}'$ must be of order 0, the final result passed to $\mathcal{A}_2$ can not contain any abstractions.

By assumption of the axiom, $\mathcal{E}, \Theta; \emptyset \vdash_{\mathsf{pptm}} C', C, \vec{a}', \vec{a}, t_r, t_k$, so the evaluations of these terms are all computable in polynomial time with these procedures. Therefore, $\mathcal{A}'$ requires only polynomial time.

It therefore follows that $\mathcal{A}'$'s advantage is negligible; so $\mathcal{A}$'s advantage is also negligible.

G. PNM, CNM, sNM and Modified sNM-CCA2

G.1. Equivalence of PNM, CNM and sNM

G.1.1. PNM implies CNM

This proof is given by Nagao, Manabe, and Okamoto [20].

G.1.2. CNM implies sNM

We have, for any adversary $\mathcal{A}$,

$$\begin{aligned}
\mathtt{Adv}^{\text{sNM-ATK}}_{\mathcal{A}} =& \mathtt{Pr}[\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] \\
=& \mathtt{Pr}[\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] \\
&+ \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] \\
=& \mathtt{Pr}[\mathtt{Expt}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] \\
&+ \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] \\
=& \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}} + \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1]
\end{aligned}$$

Note that the experiments $\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}}$ and $\mathtt{Expt}^{\text{CNM-ATK}}_{\mathcal{A}}$ are the same.

We now define another adversary $\mathcal{A}'$:

$$\begin{aligned}
\mathcal{A}_1'^{\mathcal{O}_1}(pk) \coloneqq& \mathcal{A}_1^{\mathcal{O}_1}(pk) \\
\mathcal{A}_2'^{\mathcal{O}_2}(st, X, C^*) \coloneqq& (Rel, \mathbb{C}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(st, X, C^*) \\
& (Rel', \mathbb{C}) \leftarrow \lambda K^*\, \mathbb{K}. \\
& \quad (X_1, X_2) \leftarrow X \\
& \quad \text{if } K^* = X_1 \text{ then } 1 - Rel(X_2, \mathbb{K}) \\
& \quad\quad \text{else } 1 - Rel(X_1, \mathbb{K}) \\
& \text{return } (Rel', \mathbb{C})
\end{aligned}$$

This adversary only changes the relation used. Given either the real or a fresh key, it calls $\mathcal{A}$'s relation on the *other* key and then negates the result.

This is helpful, as it relates

$$\begin{aligned}
\mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] &= \mathtt{Pr}[\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}'} = 0] \\
&= 1 - \mathtt{Pr}[\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}'} = 1] \\
&= 1 - \mathtt{Pr}[\mathtt{Expt}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1]
\end{aligned}$$

Further,

$$\begin{aligned}
\mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] &= \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}'} = 0] \\
&= 1 - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1]
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\mathtt{Adv}^{\text{sNM}}_{\mathcal{A}} &= \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}} + \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] \\
&= \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}} + 1 - (\mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1]) - (1 - \mathtt{Pr}[\mathtt{Expt}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1]) \\
&= \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}} + \mathtt{Pr}[\mathtt{Expt}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{CNM-ATK}}_{\mathcal{A}'} = 1] \\
&= \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}} + \mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{A}'}
\end{aligned}$$

If the key encapsulation mechanism is CNM-ATK secure, $\mathtt{Adv}^{\text{CNM-ATK}}_{\mathcal{B}}(\eta)$ is negligible in η for any adversary $\mathcal{B}$ and thus in particular for $\mathcal{A}$ and $\mathcal{A}'$. It therefore follows that $\mathtt{Adv}^{\text{sNM-ATK}}_{\mathcal{A}}(\eta)$ is also negligible in η.

G.1.3. sNM implies PNM

Assume an adversary $\mathcal{A}$ such that $\mathtt{Adv}^{\text{PNM-ATK}}_{\mathcal{A}}(\eta)$ is not negligible in η, i.e. $\mathtt{Pr}[\mathtt{Expt}^{\text{PNM-ATK}}_{\mathcal{A}}(\eta)=1]-\frac{1}{2} > \mu(\eta)$ for some non-negligible function μ. We construct a sNM-adversary $\mathcal{B}$ as follows:

$$\mathcal{B}_1^{\mathcal{O}_1}(pk) \coloneqq \mathcal{A}_1^{\mathcal{O}_1}(pk)$$

$\mathcal{B}_2^{\mathcal{O}_2}(C^*, X, st) \coloneqq X_1, X_2 \leftarrow X$ — X is a pair of two keys in unknown order

$st_1, C_1 \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(X_1, st, C^*)$

$st_2, C_2 \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(X_2, st, C^*)$

$\mathbb{C} \leftarrow C_1 + + C_2$

$R \leftarrow \lambda K_b\, \mathbb{K}.$

$\quad \mathbb{K}_1, \mathbb{K}_2 \leftarrow \mathbb{K}$ — Splitting can be done based on the length of C_1

$\quad$ if $K_b = X_1$ then $1 - \mathcal{A}_3(st_1, \mathbb{K}_1)$

$\quad\quad$ else $1 - \mathcal{A}_3(st_2, \mathbb{K}_2)$

return $(R, \mathbb{C})$

We have

$$\begin{aligned}
\mathtt{Adv}^{\text{sNM-ATK}}_{\mathcal{B}} &= \mathtt{Pr}[\mathtt{Expt}^{\text{sNM-ATK}}_{\mathcal{B}}=1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}^{\text{sNM-ATK}}_{\mathcal{B}}=1] \\
&= \mathtt{Pr}[g=0|b=0] - \mathtt{Pr}[g=0|b=1] \\
&= \mathtt{Pr}[g=0|b=0] - (1-\mathtt{Pr}[g=1|b=1]) \\
&= \mathtt{Pr}[g=0|b=0] + \mathtt{Pr}[g=1|b=1] - 1 \\
&= \frac{\mathtt{Pr}[g=0\wedge b=0]}{\mathtt{Pr}[b=0]} + \frac{\mathtt{Pr}[g=1\wedge b=1]}{\mathtt{Pr}[b=1]} - 1 \\
&= 2(\mathtt{Pr}[g=0\wedge b=0] + \mathtt{Pr}[g=1+b=1]) - 1 \\
&= 2\mathtt{Pr}[g=b] - 1 \\
&\geq 1 + 2\mu(\eta) - 1 = 2\mu(\eta)
\end{aligned}$$

where g and b refer to the variables in the PNM-ATK experiment. The first line is justified by the fact that from the point of view of $\mathcal{A}$, the two experiments correspond exactly to the options in the PNM-ATK game.

sNM implies PNM (without rewinding)

The previous proof involves duplicating $\mathcal{A}$'s internal state, and executing the same attack phase multiple times. This is called *rewinding* (since the attacker is reset to some previous point in time, and run again on different input). If possible, it is often avoided, since proof techniques including rewinding are not applicable to quantum adversaries. Thus, we present a modified version of the proof:

Assume an adversary $\mathcal{A}$ such that $\mathtt{Adv}^{\text{PNM-ATK}}_{\mathcal{A}}(\eta)$ is not negligible in η, i.e. $\mathtt{Pr}[\mathtt{Expt}^{\text{PNM-ATK}}_{\mathcal{A}}(\eta) = 1] - \frac{1}{2} > \mu(\eta)$ for some non-negligible function μ. We construct a sNM-adversary $\mathcal{B}$ as follows:

$$\mathcal{B}_1^{\mathcal{O}_1}(pk) \coloneqq \mathcal{A}_1^{\mathcal{O}_1}(pk)$$

$$
\begin{aligned}
\mathcal{B}_2^{\mathcal{O}_2}(C^*, X, st) \coloneqq & X_1, X_2 \leftarrow X && X \text{ is a pair of two keys in unknown order} \\
& b' \leftarrow^{\$} \{0,1\} \\
& st', \mathbb{C} \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(X_{b'}, st, C^*) \\
& R \leftarrow \lambda K_b\, \mathbb{K}. \\
& \quad \text{if } K_b = X_{b'} \text{ then } 1 - \mathcal{A}_3(st, \mathbb{K}) \\
& \quad\quad \text{else } 0 \\
& \text{return } (R, \mathbb{C})
\end{aligned}
$$

Note that we could also construct the adversary by negating the answer instead of just returning 0 if the guessed key is not the correct one. This attacker's advantage would be better by a factor of 2, and we will use this construction in the next proof (Appendix G.1.3).

We have

$$
\begin{aligned}
\mathtt{Adv}_{\mathcal{B}}^{\text{sNM-ATK}} &= \mathtt{Pr}[\mathtt{Expt}_{\mathcal{B}}^{\text{sNM-ATK}} = 1] - \mathtt{Pr}[\widetilde{\mathtt{Expt}}_{\mathcal{B}}^{\text{sNM-ATK}} = 1] \\
&= (\mathtt{Pr}[\mathtt{Expt}_{\mathcal{B}}^{\text{sNM-ATK}} = 1 | K_b = X_{b'}] + \mathtt{Pr}[\mathtt{Expt}_{\mathcal{B}}^{\text{sNM-ATK}} = 1 | K_b \neq X_{b'}]) \\
&\quad - (\mathtt{Pr}[\widetilde{\mathtt{Expt}}_{\mathcal{B}}^{\text{sNM-ATK}} = 1 | K_b = X_{b'}] + \mathtt{Pr}[\widetilde{\mathtt{Expt}}_{\mathcal{B}}^{\text{sNM-ATK}} = 1 | K_b \neq X_{b'}]) \\
&= \frac{1}{2} \cdot \mathtt{Pr}[g = 0 | b = 0] - \frac{1}{2} \cdot \mathtt{Pr}[g = 0 | b = 1] \\
&= \frac{1}{2} \cdot (\mathtt{Pr}[g = 0 | b = 0] - (1 - \mathtt{Pr}[g = 1 | b = 1])) \\
&= \frac{1}{2} \cdot (\mathtt{Pr}[g = 0 | b = 0] + \mathtt{Pr}[g = 1 | b = 1] - 1) \\
&= \frac{1}{2} \cdot (\frac{\mathtt{Pr}[g = 0 \wedge b = 0]}{\mathtt{Pr}[b = 0]} + \frac{\mathtt{Pr}[g = 1 \wedge b = 1]}{\mathtt{Pr}[b = 1]} - 1) \\
&= (\mathtt{Pr}[g = 0 \wedge b = 0] + \mathtt{Pr}[g = 1 + b = 1]) - \frac{1}{2} \\
&= \mathtt{Pr}[g = b] - \frac{1}{2} \\
&\geq \frac{1}{2} + \mu(\eta) - \frac{1}{2} = \mu(\eta)
\end{aligned}
$$

where g and b refer to the variables in the PNM-ATK experiment. The first line is justified by the fact that from the point of view of $\mathcal{A}$, the two experiments correspond exactly to the options in the PNM-ATK game.

Modified sNM-CCA2 We define a modified version of sNM, in which the adversary-defined relation also has access to the decryption oracle.

sNM′-CCA2 $\implies$ sNM-CCA2 is trivial: Any adversary against sNM-CCA2 is also an adversary against sNM′-CCA2, the relation just does not make use of the oracle.

On the other hand, we have sNM-CCA2 $\implies$ sNM′-CCA2: Given an sNM′-CCA2 adversary $\mathcal{A}$, we construct an adversary $\mathcal{A}'$ against sNM-CCA2 as follows.

$$
\begin{aligned}
\mathcal{A}_1'^{\mathcal{O}_1}(pk) \coloneqq & \mathcal{A}_1^{\mathcal{O}_1}(pk) \\
\mathcal{A}_2'^{\mathcal{O}_2}(C^*, X, St) \coloneqq & (R, \mathbb{C}) \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(C^*, X, St) \\
& \mathbb{K} \leftarrow \mathcal{O}_2(\mathbb{C}) \\
& r \leftarrow R^{\mathcal{O}_2}(\pi_1 X, \mathbb{K}) \\
& R'(K_b, -) \leftarrow C^* \in \mathbb{C} ? 0 : (K_b = X_1 ? r : \neg r) \\
& \text{return } (R', [])
\end{aligned}
$$

Note that $\mathcal{A}'$ does not make use of decryption via $\mathbb{C}$ at all, since the decryption oracle can handle the same queries. Note also that R is always 0 if $\mathbb{C}$ contains the ciphertext, matching the check in $\texttt{Expt}^{\text{sNM-ATK}}$.

$$\begin{aligned}
\texttt{Adv}^{\text{sNM}'-\text{ATK}}_{\mathcal{A}'} =& \texttt{Pr}[\texttt{Expt}^{\text{sNM}'\text{-CCA2}}_{\mathcal{A}'} = 1] - \texttt{Pr}[\widetilde{\texttt{Expt}}^{\text{sNM}'\text{-ATK}}_{\mathcal{A}'} = 1] \\
=& \frac{1}{2}(\texttt{Pr}[\texttt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] - \texttt{Pr}[\widetilde{\texttt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1]) \\
&+ \frac{1}{2}(\texttt{Pr}[\widetilde{\texttt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 0] - \texttt{Pr}[\texttt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 0]) \\
=& \frac{1}{2}\texttt{Adv}^{\text{sNM-ATK}}_{\mathcal{A}} + \frac{1}{2}((1 - \texttt{Pr}[\widetilde{\texttt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1]) \\
&- (1 - \texttt{Pr}[\texttt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 1])) \\
=& \frac{1}{2}\texttt{Adv}^{\text{sNM-ATK}}_{\mathcal{A}} + \frac{1}{2}(\texttt{Pr}[\texttt{Expt}^{\text{sNM-ATK}}_{\mathcal{A}} = 1] - \texttt{Pr}[\widetilde{\texttt{Expt}}^{\text{sNM-ATK}}_{\mathcal{A}} = 1]) \\
=& \texttt{Adv}^{\text{sNM-ATK}}_{\mathcal{A}}
\end{aligned}$$

The above follows from the fact that if K_b happens to be the first key in X, then the experiments correspond exactly to the sNM experiments. In the other case, the result of the experiment is negated.

The factor $\frac{1}{2}$ comes from the order of X being chosen at random, although any other distribution would work as well.

H. PKE-based Oblivious Transfer

H.1. Insufficient Characterization of Full-Domain Encryption

We first attempted to state the full-domain assumption as follows:

PKE-FD

$$\frac{}{\begin{array}{c}\mathcal{E}; \Theta \vdash \tilde{\forall}((m, i) : \tau_{msg} \times \tau_{index}).\ \tilde{\exists}((m', s') : \tau_{msg} \times \tau_{msg}). \\ [\text{PKE.Enc}(\text{pk}(k_S\ i), m', s') = m] \tilde{\wedge} [\phi^{k_S,i}_{\text{fresh}}(m) \implies \phi^{k_S,i}_{\text{fresh}}(m')]\end{array}}$$

This states that for every message m and secret key k_S i, there is a message m' that can be encrypted with $\text{pk}(k_S\ i)$ to produce the ciphertext m. Importantly, if m does not depend on k in any way, then neither does m'.

However, it is not quite sufficient, as there could by encryption schemes where a uniform distribution of ciphertexts does not yield a uniform distribution of plaintexts. As a most extreme example, consider an encryption scheme where all but one message m are effectively encrypted without randomization (i.e. they always produce the same ciphertext), but encryption of m makes full use of the available randomness and can produce a vast number of different ciphertexts. Such a scheme could still be NM-CPA secure (as required for the protocol) as long as m is different for each key, and thus not known to the attacker.

However, such a scheme would not yield receiver privacy when used in this protocol, as the receiver would have a low probability of choosing message m to encrypt, but the ciphertext XOR m_a i and m_b i produces a ciphertext of m with high likelihood. To the sender, which knows the secret key, these are distinguishable.

H.2. Derived Property: Indistinguishability of Modified Ciphertext from Random Sampling

From PKE-IND$ and PKE-NM-CPA, we can derive the following property, which states that an attacker modifying the ciphertext using XOR can not distinguish between the decryption of the result and a

randomly sampled message:

PKE-XOR-CPA

$$\frac{\mathcal{E},\Theta;\emptyset \vdash_{\mathsf{pptm}} C,C',m,t_r \qquad k \notin_{pk} \vec{u},C,C',m,t_r,a \qquad \mathcal{E};\Theta \vdash [\phi^{n,()}_{\mathrm{fresh}}(\vec{u},C,C',m,t_r,a)] \qquad \mathcal{E};\Theta \vdash [a \neq 0]}{\mathcal{E};\Theta \vdash \begin{array}{c}\vec{u},C[\mathsf{let}\ c \leftarrow \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,r\ t_r);\\ x \leftarrow c \oplus a\ \mathsf{in}\ c,\mathrm{PKE.Dec}(k\ t_k,x)]\end{array} \sim \begin{array}{c}\vec{u},C[\mathsf{let}\ c \leftarrow \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m,r\ t_r);\\ x \leftarrow c \oplus a\ \mathsf{in}\ c,n\ ()]\end{array}}$$

We prove this property by reducing to PKE-IND$ and PKE-NM-CPA. Assume an adversary $\mathcal{A}$ against PKE-XOR-CPA for some message m. If $\mathcal{A}$ has a non-negligible advantage, then $\mathrm{PKE.Dec}(k\ t_k,\ \mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),\ m,r\ t_r) \oplus a)$ must have a non-uniform distribution (and be distinguishable from a uniform distribution). Note, however, that $\bigcup_m \{\mathrm{PKE.Dec}(k\ t_k,\mathrm{PKE.Enc}(\mathrm{pk}(k),m,r) \oplus a) | \text{for all values of } k,r\}$ is a partition of the set of all possible ciphertexts. This leaves two options: either, the distribution of $\mathrm{PKE.Dec}(k\ t_k,\mathrm{PKE.Enc}(\mathrm{pk}(k\ t_k),m \oplus a,r\ \ t_r) \oplus a)$ is equal (or at least indistinguishable) for all m, or there are two messages m and m' such that the corresponding distributions are distinguishable.

In the first case, this means that there is an attacker with non-negligible advantage against PKE-IND$, since the distributions of ciphertexts obtained via encryption is not uniform. In the second case, however, there is an attacker with non-negligible advantage against PKE-NM-CPA, since it is possible to distinguish between the encryptions of m and m' by querying the decryption oracle for $c \oplus a$ (where c is the challenge ciphertext).

Unfortunately, we have not been able to do this proof in CCSA, since CCSA (currently) offers no way of reasoning about the probability distributions of terms other than names.

H.3. Receiver Privacy

The abstract statement we wish to prove is the following:

$$\mathcal{E},\Theta \vdash \ S.init(r),\ \pi_1(R.req(r',0,\pi_1(S.init(r)))) \ \sim \ S.init(r),\ \pi_1(R.req(r',1,\pi_1(S.init(r))))$$

Substituting the definitions and simplifying, we get

$$\mathcal{E},\Theta \vdash \begin{array}{c} m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\\ \mathrm{PKE.Enc}(\mathrm{pk}(k_S\ i), s_j, r_j) \oplus m_a\ i\end{array} \sim \begin{array}{c} m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\\ \mathrm{PKE.Enc}(\mathrm{pk}(k_S\ i), s_j, r_j) \oplus m_b\ i\end{array}$$

Using PKE-IND$ as defined above, we can treat the encryption as a random sampling (denoted by the fresh name $n\ ()$):

$$\mathcal{E},\Theta \vdash \quad m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\ n\ () \oplus m_a\ i \quad \sim \quad m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\ n\ () \oplus m_b\ i$$

Now, we again use the fact that the result of exclusive or is random when one of the operands is.

$$\mathcal{E},\Theta \vdash \quad m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\ n'\ () \quad \sim \quad m_a\ i, m_b\ i, \mathrm{pk}(k_S\ i),\ n''\ ()$$

The proof concludes using the indistinguishability of fresh names.