Key-dependent Message Security under Active Attacks – BRSIM/UC-Soundness of Symbolic Encryption with Key Cycles –

Michael Backes[1]*, Birgit Pfitzmann[2], and Andre Scedrov[3]**

[1] Saarland University, backes@cs.uni-sb.de
[2] IBM Zurich Research Lab, bpf@zurich.ibm.com
[3] University of Pennsylvania, scedrov@math.upenn.edu

Abstract. Key-dependent message security, short KDM security, was introduced by Black, Rogaway and Shrimpton to address the case where key cycles occur among encryptions, e.g., a key is encrypted with itself. It was mainly motivated by key cycles in Dolev-Yao models, i.e., symbolic abstractions of cryptography by term algebras, and a corresponding soundness result was later shown by Adão et al. However, both the KDM definition and this soundness result do not allow the general active attacks typical for Dolev-Yao models and for security protocols in general.

We extend these definitions so that we can obtain a soundness result under active attacks. We first present a definition AKDM as a KDM equivalent of authenticated symmetric encryption, i.e., it provides chosen-ciphertext security and integrity of ciphertexts even for key cycles. However, this is not yet sufficient for the desired soundness, and thus we give a definition DKDM that additionally allows limited dynamic revelation of keys. We show that this is sufficient for soundness, even in the strong sense of blackbox reactive simulatability (BRSIM)/UC and including joint terms with other operators.

We also present constructions of schemes secure under the new definitions, based on current KDM-secure schemes. Moreover, we explore the relations between the new definitions and existing ones for symmetric encryption in detail, in the sense of implications or separating examples for almost all cases.

1 Introduction

Encryption schemes are the oldest and arguably the most important cryptographic schemes. Their security has been rigorously studied very early, starting with Shannon's work for the information-theoretic case [32]. Computational definitions for public-key encryption were developed over time, in particular in [22, 33, 31, 17]. The strongest of these definitions is security against adaptive chosen-ciphertext attacks, abbreviated IND-CCA2 [9]; it is nowadays strongly believed to cover what one expects from secure encryption on its own. For symmetric encryption, the first real definitions were, to the best of our knowledge, given in [17, 25, 8], using the same basic ideas.

* Michael Backes was partially supported by the German Research Foundation (DFG).

** Andre Scedrov was partially supported by the Office of Naval Research (ONR) and the National Science Foundation (NSF).

However, for the use within larger protocols, additional requirements on encryption schemes are still emerging. One specific additional requirement is the ability to securely encrypt key-dependent messages. The first concrete use case occurred in [12], where encryption of different private keys with one another was used to implement an all-or-nothing property in a credential system to discourage people from transfering individual credentials. Another area that brought up this requirement is the use of formal methods or symbolic cryptography. Here the question is whether simple abstractions of cryptographic primitives exist that can be used by automated proof tools (model checkers or theorem provers) to prove or disprove a wide range of security protocols that use cryptography in a blackbox manner. The best-known abstractions are term algebras constructed from certain base types and cryptographic operators such as E and D for en- and decryption, called Dolev-Yao models after the first such abstraction [18]. As soon as one has a multi-user variant of such a model, the keys are explicit base terms. Hence from the term algebra side it is natural that keys can also be encrypted (i.e., used as leaves in arbitrary terms). Thus most Dolev-Yao models simply assumed that key cycles are allowed. Once the cryptographic justification of such models was started in [2], it was recognized that key cycles had to be excluded from the original models to get cryptographic results. The same holds for later results [1, 23, 6, 24, 28, 4, 16, 15].

Motivated primarily by symbolic cryptography, a definition of key-dependent message security (*KDM security*) was introduced in [11]. It generalizes the definition from [12] by allowing arbitrary functions of the keys (and not just individual keys) as plaintexts, and by considering symmetric encryption schemes. Furthermore, a construction of a KDM-secure scheme in the random oracle model is given in [11]. In [3] it was shown that using the KDM definition one can indeed extend results about the security of Dolev-Yao models in a passive setting to include key cycles. Still, Dolev-Yao models are only of limited interest in a passive setting because their main usage is in protocol proofs, where active attackers are a standard threat.

In this work, we extend the definition of KDM security to active attacks and, as this also turns out to be needed for the desired soundness proof of Dolev-Yao models with key cycles, to a limited dynamic revelation of keys. We speak of *AKDM security* and *DKDM security* for *adaptive KDM security* and *dynamic KDM security*, respectively. While dynamic revelations of keys in cryptography mostly occur as a consequence of adversary models with dynamic corruption of participants, we will see that in the class of protocols typically treated with Dolev-Yao models, they can occur even under static corruptions, which are the standard adversary model in Dolev-Yao models. After the definitions (Section 2), we present the following results:

- In Section 3, we construct symmetric encryption schemes secure under the new definitions. For AKDM security we achieve this by a generic construction from KDM-secure schemes and MACs (message authentication codes). For DKDM security we present a direct construction in the random oracle model. For the time being, we do not lose anything by this need as all currently known KDM-secure schemes are also in the random oracle model. We only need a very weak version of the random oracle idealization: The oracle must output independent random values for different inputs. We do not need reprogramming of the oracle or similar features

that are immediately problematic when the random oracle is replaced by real hash functions. Both constructions are efficient.
- In Section 4, we show that DKDM security is sufficient for the soundness of symbolic symmetric encryption in the strong sense of blackbox reactive simulatability (BRSIM)/UC [29, 30, 14]. This notion entails strong compositionality guarantees and the retention of a variety of security properties.
- In Section 5, we explore the relations between the definition variants, also in combination with normal definitions. For instance, we consider whether KDM-secure schemes that are also IND-CCA2-secure are automatically AKDM-secure. (No, they are not.) The relationship of AKDM and DKDM is particularly important since we can achieve AKDM security by a generic construction based on KDM security, while we need DKDM security in the soundness proof. We show that both definitions coincide provided that, essentially, either only a logarithmically bounded number of keys are used, or only a constant number is revealed. The question whether both definitions coincide for arbitrary leakages of keys remains open in this paper and seems similar to the selective decommitment problem [19], but not directly related to the cases with known answers.
- Our definitions include a variant that we call *polynomial-oracle*; we also define such a variant for KDM security and consider it when exploring the relation between the definitions. We believe this will be needed if one wants to succeed in constructing KDM-secure schemes in the standard model of cryptography in the future.
- We show that for stateful encryption schemes, semantic security does not imply KDM security even if only key cycles of an arbitrary minimum size i are allowed (within Section 5). In [11] all separating results require key cycles of length 1.

Clearly, the constructions that we present in Section 3 are for cases where no simpler equivalence with prior definitions exists in Section 5.

2 Adaptive and Dynamic Security Definitions with Key Cycles

This section contains our new definitions of AKDM and DKDM security. While the first definition constitutes a natural extension of KDM security to active attacks, the second one additionally allows limited dynamic revelation of keys. This is needed for the desired soundness proof of Dolev-Yao models with key cycles.

We start by summarizing basic notation needed for defining and extending the notion of KDM security as introduced in [11].[1] We write ":=" for deterministic and "$\leftarrow$" for probabilistic assignment, and "$\xleftarrow{\mathcal{R}}$" for uniform random choice from a set. An error element $\downarrow$ is available as an addition to the domains and ranges of all functions and algorithms. By $x := {+}{+}y$ for integer variables x, y we mean $y := y + 1; x := y$. The length of a string m is denoted by $|m|$, a string of l zero-bits is written 0^l, and string concatenation $||$.

[1] KDM security was also proposed for asymmetric encryption schemes in [11]. In this paper, we focus on the symmetric setting, but our extended definitions of key-dependent message security can be recast in the asymmetric setting as well.

Like many current cryptographic definitions, the AKDM and DKDM definitions are written in terms of *oracles* performing actions of an honest participant and *adversaries*. Oracles and adversaries can be considered interactive probabilistic Turing machines (except that the original KDM definition corresponds to an interactive infinite-state system) where oracles have one communication tape pair, and adversaries have one communication tape pair (or several if they use several oracles) and an additional output tape. The final content of this output tape is considered the result of a joint computation and can be used in expressions, e.g., equations of the form "$\mathsf{A}^{\mathsf{O}} = x$". That an oracle offers a query type q means that it accepts inputs of the form q and returns one output for each such input.

We always write k for a security parameter, which is a natural number. It is an input of all the following machines, adversaries as well as oracles, but omitted in the notation for readability. For instance, we write A^{O} and not $\mathsf{A}(k)^{\mathsf{O}(k)}$ to denote an adversary A using an oracle O. Thus the resulting expressions are functions of k, and terms like "negligible" are meaningful for them. In particular, a function $g\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is called negligible iff for all positive polynomials Q, there exists k_0 such that $g(k) \leq 1/Q(k)$ for all $k \geq k_0$. Like some underlying papers we use the word "advantage" loosely for measures of adversary success (with precise definitions of those measures, of course).

Definition 1 (Symmetric Encryption). *A* symmetric encryption scheme *is a tuple* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ *of polynomial-time algorithms. Key generation with a security parameter* $k \in \mathbb{N}$ *is written* $sk \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$*. The (probabilistic) encryption of a message* $m \in \{0,1\}^+$ *is denoted by* $c \leftarrow \mathsf{E}(sk, m)$*, and decryption by* $m := \mathsf{D}(sk, c)$*. The result may be* $\downarrow$*; then the ciphertext is called* invalid *for this key. Decryption of a correctly generated ciphertext for a correctly generated key must always yield the original plaintext.*

We assume without loss of generality that keys for a fixed security parameter k *are of a fixed length.* $\diamond$

Typical definitions of encryption security are *semantic security* (*IND-CPA security*) for the passive case and *indistinguishability under adaptive chosen-ciphertext attack* (*IND-CCA2 security*) as well as *integrity of ciphertexts* (*INT-CTXT security*) for the active case. As these definitions are standard, we only present the latter two in Appendix A.1, before the proofs where we need them.

The notion of key-dependent message (KDM) security introduced in [11] as well as our extensions of it use encryption oracles that explicitly deal with messages that depend on keys. Instead of a fixed plaintext m, they accept a function g of the secret keys in encryption queries. For this, the oracle immediately handles several keys, here actually virtually infinitely many. Formally, encryption oracles offer query types (enc, j, g) meaning that the j-th key should be used to encrypt the result of evaluating the function g on the secret keys. In [11] the input g is required to be a RAM program for some fixed RAM machine model, and to represent a function f_g that maps an infinite sequence $\boldsymbol{sk}$ of secret keys to a bitstring, i.e., $f_g : (\{0,1\}^*)^\infty \to \{0,1\}^+$. We assume that the RAM model means that the key numbers used are explicitly visible in the program g. Additionally, every f_g must be a function with fixed-length outputs for a given, fixed security parameter.[2] Let $\mathcal{F}_\infty$ denote all these permitted programs, and let $\mathcal{A}_\infty$ denote the class of polynomial-time adversaries using only permitted programs $g \in \mathcal{F}_\infty$ in encryption queries. The notation in [11] does not distinguish f_g and g, and thus we also always write g in the following. We write π_i to denote the program that computes the projection to the i-th key. Finally, we define "$i \in g$" for a program $g \in \mathcal{F}_\infty$ and a key number $i \in \mathbb{N}$ to mean that the i-th key is addressed by the program g.

In addition to these encryption queries, the new definition of adaptive KDM (AKDM) security allows decryption queries. (This is similar to definitions IND-CCA2 and the integrity of ciphertexts.) Here, decryption queries are of the form (dec, j, c); this means that the ciphertext c should be decrypted with the j-th key. We define AKDM security immediately in a *polynomial-oracle* version besides an unrestricted version closer to the original KDM definition. The problem with the unrestricted version, if one thinks of potential realizations in the standard model of cryptography, is that the adversary can force the oracle to perform non-polynomial-time computations. We exclude this by the polynomial-oracle version. The following p-oracle-bounded adversaries have the appropriate behavior for a particular polynomial p. We immediately include reveal queries so that we can use the same adversary classes for DKDM. (An adversary trying reveal queries on the oracle in the AKDM definition will just get an error result.)

Definition 2 **(p-oracle-bounded Adversaries).** *For every polynomial p, we define the class $\mathcal{A}_p$ of* p-oracle-bounded adversaries *as the set of adversaries $\mathsf{A} \in \mathcal{A}_\infty$ with the following additional property: For every security parameter k and every query (enc, j, g) or (dec, j, c) that A makes, we have $j \leq p(k)$ and $i \leq p(k)$ for all $i \in g$, and the Turing complexity of g is at most $p(k)$. If A also makes queries (reveal, j), then also $j \leq p(k)$.* $\diamond$

Definition 3 **(Adaptive Key-dependent Message (AKDM) Security).** *Given a symmetric encryption scheme $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ the* adaptive key-dependent message oracle *or* AKDM oracle $\mathsf{SymAKDM}$ *is defined as follows: It has a (virtual) infinite key sequence $\boldsymbol{sk} = (sk_i)_{i \in \mathbb{N}}$ where each key, when first used, is initialized as $sk_i \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$, an initially empty set C of ciphertexts made, a bit b initialized as $b \xleftarrow{\mathcal{R}} \{0,1\}$, and the following query types:*

- *On input (enc, j, g): Let $m_0 := g(\boldsymbol{sk})$ and $m_1 := 0^{|m_0|}$, encrypt $c \leftarrow \mathsf{E}(sk_j, m_b)$, store c as $C := C \cup \{(j, c)\}$, and output c.*
- *On input (dec, j, c) with $j \in \mathbb{N}$ and $c \in \{0,1\}^+$: If $(j, c) \in C$, output $\downarrow$, else output $m := \mathsf{D}(sk_j, c)$ if $b = 0$ and $\downarrow$ if $b = 1$.*

The AKDM advantage *of an adversary* A *that interacts with* $\mathsf{SymAKDM}$ *and finally outputs a bit b^* is defined as $Adv_{\mathsf{AKDM}}(\mathsf{A}) := |\Pr[\mathsf{A}^{\mathsf{SymAKDM}} = 1 \mid b = 0] - \Pr[\mathsf{A}^{\mathsf{SymAKDM}} = 1 \mid b = 1]|$. We say that $\mathcal{SE}$ is* (unrestricted) AKDM-secure *or* p-oracle-bounded AKDM-secure *iff the AKDM advantage is negligible for every adversary $\mathsf{A} \in \mathcal{A}_\infty$ or $\mathsf{A} \in \mathcal{A}_p$, respectively, and that it is* polynomial-oracle AKDM-secure *iff it is p-oracle-bounded AKDM-secure for every polynomial p.* $\diamond$

[2] The definition simply assumes that inputs g are indeed fixed-length. The oracle could also achieve this property algorithmically if encryption queries are augmented by a parameter $l \in \mathbb{N}$ and the oracle cuts or pads the function $g(\boldsymbol{sk})$ of the keys to l bits.

We call SymAKDM with $b = 0$ the *real AKDM oracle* and with $b = 1$ the *fake AKDM oracle*, and similarly for the following oracles that have a bit b. Extending Definition 3 such that proofs can be conducted in the random oracle model can be achieved as usual by allowing both the AKDM oracle and the adversary to query the same random function, called a random oracle. (More algorithmically, the random oracle may choose a random answer when it first gets a query, and otherwise retrieve the prior answer for the same query.)

Our second variant of KDM security is even stronger: keys may dynamically be revealed to the adversary. Dynamic key revelations have been considered in cryptography before (although not for KDM security, of course), in particular where the adversary may dynamically corrupt participants. However, typical Dolev-Yao models do not consider dynamic corruptions and one might therefore expect the desired soundness to hold already for AKDM-secure cryptographic realizations. But even scenarios with static corruptions may lead to dynamic key revelations on the layer of the KDM security: For instance, one can model protocols like group membership services where new participants may be included into a group at any time. Thus an established group key is shared with a new participant who may be an adversary. If one considers this situation for a protocol with key cycles, one immediately sees the need for dynamic KDM security. Hence our DKDM definition adds key revelations to the AKDM oracle. However, in these revelations we must exclude the commitment problem that is otherwise inherent in symmetric encryption, i.e., the problem that if a message m was encrypted with a key sk, and later sk becomes known, this allows the adversary to distinguish the real and fake oracle. For this, the oracle maintains two sets Rev and Enc that denote the keys that were already revealed and those that were already used for en- or decryption, respectively. Keys that are only part of the plaintext in an encryption are not added to Enc. Keys from Enc are not revealed. While this may seem a rather weak form of dynamic revelation, it already turns out to be sufficient for the desired soundness result, cf. Section 4. When deployed in protocols facing strong adversaries, e.g., adaptive corruption of parties, the definition might be further strengthened, e.g., by reflecting that revealing used keys does not leak any information about encryptions computed with unrevealed keys.

Definition 4 (Dynamic Key-dependent Message (DKDM) Security). *Given a symmetric encryption scheme* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$*, the* dynamic key-dependent message oracle *or* DKDM oracle SymDKDM *is defined like the AKDM oracle* SymAKDM *with the following changes:*

- *It maintains two sets Rev and Enc, which are initially empty.*
- *On input* (reveal, j)*: If $j \notin Enc$, it sets $Rev := Rev \cup \{j\}$ and outputs sk_j, else it outputs $\downarrow$.*
- *On input* (enc, j, g) *or* (dec, j, c)*: If $j \in Rev$, it outputs $\downarrow$, else it acts as before and additionally sets $Enc := Enc \cup \{j\}$.*

The DKDM advantage *of an adversary* A *that interacts with* SymDKDM *and finally outputs a bit b^* is defined as* $Adv_{\mathsf{DKDM}}(\mathsf{A}) := |\Pr[\mathsf{A}^{\mathsf{SymDKDM}} = 1 \mid b = 0] - \Pr[\mathsf{A}^{\mathsf{SymDKDM}} = 1 \mid b = 1]|$. *We say that $\mathcal{SE}$ is* (unrestricted) DKDM-secure *or*

p-oracle-bounded DKDM-secure *iff the DKDM advantage is negligible for every adversary* $\mathsf{A} \in \mathcal{A}_\infty$ *or* $\mathsf{A} \in \mathcal{A}_p$*, respectively, and that it is* polynomial-oracle DKDM-secure *iff it is* p*-oracle-bounded AKDM-secure for every polynomial* p*.* $\diamond$

By construction, we always have $Rev \cap Enc = \emptyset$.

3 Constructions of AKDM-secure and DKDM-secure Schemes

In this section we give explicit constructions of AKDM-secure and DKDM-secure schemes. For AKDM-secure schemes, we can present a generic construction based on KDM-secure schemes and strongly unforgeable message authentication codes (MACs); for DKDM-secure schemes, we present an explicit construction in the random-oracle model.

3.1 AKDM-secure Schemes from KDM-secure Schemes and MACs

We show that the generic encrypt-then-MAC construction formalized in [10] yields an AKDM-secure encryption scheme when applied to a KDM-secure encryption scheme and a strongly unforgeable MAC scheme.

Definition 5 (MAC Scheme). *A* MAC scheme*, also called* symmetric authentication scheme*, is a tuple* $\mathcal{MAC} = (\mathsf{gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Test})$ *of polynomial-time algorithms. Key generation with a security parameter* $k \in \mathbb{N}$ *is written* $sk \leftarrow \mathsf{gen}_{\mathsf{MAC}}(0^k)$*. The (potentially probabilistic) authentication of a message* $m \in \{0,1\}^+$ *is denoted by* $t \leftarrow \mathsf{MAC}(sk, m)$ *(where "*t*" stands for "tag"), and testing by* $b := \mathsf{Test}(sk, m, t)$ *with an output* $b \in \{\mathsf{true}, \mathsf{false}\}$*. The tag is called* valid *(for the given message and key) iff* $b = \mathsf{true}$*. Testing a correctly generated MAC for a correctly generated key must always yield* true*.* $\diamond$

Theorem 1 (AKDM Security of Encrypt-then-MAC Construction). *Let* $\mathcal{SE}$ *be a polynomial-oracle or unrestricted KDM-secure symmetric encryption scheme and let* $\mathcal{MAC}$ *be a strongly unforgeable MAC. Then* $\mathsf{Encrypt\_then\_MAC}(\mathcal{SE}, \mathcal{MAC})$ *is a polynomial-oracle or unrestricted AKDM-secure encryption scheme, respectively.* $\square$

This proof is given in Appendix A.2.

3.2 A DKDM-secure Scheme

In the following, we construct a DKDM-secure scheme directly from a random oracle H. The construction extends that of the KDM-secure scheme in [11]. We immediately assume that the random oracle can take inputs of different lengths.

Definition 6 (DKDM-RO Construction). *Let a random oracle* H *be given. We then construct a symmetric encryption scheme* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$*, called* DKDM-RO scheme*, as follows:*

- *Keys are pairs* (sk^e, sk^f)*, generated as* $sk^e \xleftarrow{\mathcal{R}} \{0,1\}^k$ *and* $sk^f \xleftarrow{\mathcal{R}} \{0,1\}^k$*.*

– *The encryption c of a message m with key (sk^e, sk^f) is computed as* $r \xleftarrow{\mathcal{R}} \{0,1\}^k$; $c'' \leftarrow \mathsf{H}(sk^e, r)$; $c' \leftarrow c'' \oplus m$; *and* $c := (r, c', \mathsf{H}(sk^f, r, c'))$.
– *To decrypt a ciphertext c with key (sk^e, sk^f), decompose c into (r, c', t). If this fails or $\mathsf{H}(sk^f, r, c') \neq t$, output $\downarrow$, else output $c' \oplus \mathsf{H}(sk^e, r)$.*

$\diamond$

Theorem 2 (Security of the DKDM-RO Scheme). *In the random-oracle model, the DKDM-RO scheme from Definition 6 is DKDM-secure.* $\square$

Proof. Let H be the random oracle, and at every time let its domain $\mathsf{Dom(H)}$ be the set of queries it already answered. In the interaction of a polynomial-time adversary A and the DKDM oracle $\mathsf{SymDKDM}$, let E_1 be the event that an encryption query (enc, j, g) happens such that the first random oracle call in it is for an old value, i.e., already $(sk_j^e, r) \in \mathsf{Dom(H)}$. Clearly E_1 only happens with exponentially small probability within the polynomially many queries that A can make, because the second input part r is chosen randomly by the oracle $\mathsf{SymDKDM}$ immediately before each query.

If E_1 does not occur, we want to show that $\mathsf{SymDKDM}$ is perfectly indistinguishable from an oracle $\mathsf{SymDKDM}_1$ that chooses each value c'' in an encryption randomly and independently instead of by a random oracle call, and thus also from an oracle $\mathsf{SymDKDM}_2$ that chooses c' randomly for both values of b (because c'' serves as a one-time pad for m or $0^{|m|}$ for $b = 0$ and $b = 1$, respectively). This could only go wrong if sk_j^e became known to A, so that A could verify whether $c'' = \mathsf{H}(sk_j^e, r)$. However, by induction over the steps in the run we can show that A never obtains any information about a key (sk_j^e, sk_j^f) that is or has been used in an en- or decryption, and that the values c'' are random, because there are only two ways how information about such a key could leak: One is in a reveal query, but the use of the sets Rev and Enc ensures that $\mathsf{SymDKDM}$ does not answer these for keys used in en- or decryption. The other is within a message m in the real oracle; however, by the induction hypothesis we know that this has not happened so far, and by the absence of E_1 it does not happen now.

Now let E_2 be the event that the adversary manages to make a decryption query (dec, j, c) where c is a new ciphertext with a correct tag, i.e., we have $(j, c) \notin C$ in $\mathsf{SymDKDM}_2$, but nevertheless c is a triple (r, c', t) with $\mathsf{H}(sk_j^f, r, c') = t$. The condition $(j, c) \notin C$ implies that $\mathsf{SymDKDM}_2$ has not called the random oracle for (sk_j^f, r, c') in an encryption query (enc, j, m) for any m (because this format only fits the second call there, and then $\mathsf{SymDKDM}_2$ would also have obtained the ciphertext c and stored (j, c) in C). Furthermore, the probability is exponentially small that it made this call in an encryption query for a value $j' \neq j$ because that would imply $sk_j^f = sk_{j'}^f$. Even for the non-polynomial DKDM oracle with its infinitely many keys, this only happens with exponentially small probability for two values j, j' that A chooses in its polynomially many queries. If $\mathsf{SymDKDM}_2$ had made this query in a prior decryption call, E_2 would already have been violated earlier.

Hence A has not obtained information about the correct t from $\mathsf{SymDKDM}_2$. Thus it can guess t only with exponentially small probability unless it makes a random oracle call for (sk_j^f, r, c') itself. However, A can only guess this input with exponentially small probability because $\mathsf{SymDKDM}_2$ does not leak any information about sk_j^f. Hence E_2 only has exponentially small probability.

If E_2 does not happen, $\mathsf{SymDKDM}_2$ is perfectly indistinguishable from an oracle $\mathsf{SymDKDM}_3$ that always returns $\downarrow$ for encryption queries (dec, j, c) with $(j, c) \notin C$, independent of b. The reaction on such encryption queries was the only other difference between the real and fake oracle, i.e., between $b = 0$ and $b = 1$. Hence A has the DKDM advantage 0 when interacting with $\mathsf{SymDKDM}_3$, and thus at most an exponentially small one when interacting with the original oracle $\mathsf{SymDKDM}$. ■

4 Sound Symbolic Symmetric Encryption in the Sense of BRSIM/UC

In this section, we show that DKDM security is the right notion to prove the soundness of a Dolev-Yao model (in other words a formal or symbolic model) of symmetric encryption permitting key cycles in the sense of blackbox reactive simulatability/UC, short BRSIM/UC.

We prove soundness for a symbolic model whose terms may contain not only symmetric encryption operators, but also asymmetric encryption, signatures, and message authentication codes, as well as lists (pairing), nonces, and payload messages. We prove BRSIM/UC for the symbolic system as the ideal functionality and the cryptographic realization as the real functionality. By the composition theorems of BRSIM/UC, this implies BRSIM/UC also for all protocols that are designed with the symbolic version and then used with the real version. By the property preservation theorems of BRSIM/UC, this implies that typical security properties are retained between the symbolic and the real version, in particular integrity and typical key and message secrecy [29, 5].

As we do not want to make a soundness proof from scratch, but build upon an existing one with the same primitives, only without key cycles, we build upon [4], the extension of the soundness result from [6] by symmetric encryption. This line of work is so far the only one proving BRSIM/UC for the symbolic model as such, and the only one with such a large set of primitives. It turns out that we do not have to change the ideal and real functionality from [4] at all, as the absence of key cycles is modeled by a condition on the users of these functionalities there (typically protocols). Thus we omit this constraint and show that implementing the real system with a polynomial-oracle DKDM-secure encryption scheme gives the desired soundness result. We start by reviewing the notion of BRSIM/UC and by outlining the ideal functionality and the realization from [4]. Readers familiar with these topics can immediately proceed with Section 4.4.

4.1 Dolev-Yao Models in the BRSIM/UC Setting

BRSIM/UC is used for comparing an ideal and a real system with respect to security. We use this joint name for the closely related models of which different pieces were first introduced in [29, 30, 14], building upon secure (one-step) multi-party computation [20, 21, 7, 27, 13]. Reactive simulatability between the real and ideal system essentially means that for all attacks on the real system there exists an equivalent attack on the ideal system. More precisely, blackbox simulatability states that there exists a simulator Sim

that can use an arbitrary real adversary as a blackbox, such that arbitrary honest users H cannot distinguish whether they interact with the real system and the real adversary A, or with the ideal system and the simulator with its blackbox. We always assume that all parties are polynomial-time. The ideal system is often called TH for "trusted host", and the protocol machines of the real system are often called M_u, where u is a user index. In our specific case of symbolic cryptography, the trusted host encapsulates the Dolev-Yao model while the real system is the distributed system that uses actual cryptographic algorithms and exchanges actual bitstrings.

Establishing a BRSIM/UC relation between a Dolev-Yao model and its realization requires some common syntax how higher protocols interact with the ideal Dolev-Yao functionality and the realization; in the underlying model from [4] this is done by letting the protocols operate on terms or real bitstrings, respectively, via local names called handles. Like all Dolev-Yao-style models when actually used for protocol modeling, e.g., using a special-purpose calculus or embedded in CSP or pi-calculus, the model in [4], called the BPW model henceforth after the authors of this paper, has state. An important use of state is to model which participants already know which terms. Here this is given by the handles, i.e., the adversary's knowledge set is the set of terms to which the adversary has a handle. Another use of state is to remember different versions of terms of the same structure for probabilistic operations such as nonce or key generation. In [6], as probably first in [26], the probabilism is abstracted from by counting, i.e., by assigning successive natural numbers to terms, here globally over all types. This *index* of a term allows us (not the participants) to refer to terms unambiguously.

The users can operate on terms in the expected ways, e.g., ask the system to en- or decrypt a term or to generate an additional key. Further, they can input that a term should be sent to another user. In the symbolic representation this only changes the knowledge sets, i.e., in this specific Dolev-Yao model the intended recipient and/or the adversary (depending on the security of the chosen channel) obtains a handle to the term.

4.2 State Representation of the BPW Model

The BPW model of [4], i.e., the ideal functionality of symbolic cryptography, is called $Sys_{n,L}^{\mathsf{cry\_sym,id}}$. (The parameters n and L are of no relevance here; n is the number of participants, L a set of functions enabling one to abstractly compute the leaking length of a term.) Its state is represented as a database D of the existing terms. Each term is characterized by its type (top-level operator) $type$, the list arg of its top-level arguments, the handles hnd_u for different participants u, the index ind mentioned before, and a length len (needed because encryption cannot completely hide the length of messages). The non-atomic subterms of a term are represented by their indices in the list arg.

As we build on [4] we repeat some of their notation: A database D is a set of functions, called entries, each over a finite domain called attributes. For an entry $x \in D$, the value at an attribute att is written $x.att$. For a predicate $pred$ over the attributes, $D[pred]$ is the subset of entries that fulfill $pred$. If $|D[pred]| = 1$, the same notation is used for its one element. An underlying list operation is written $l := (x_1, \ldots, x_j)$, where the arguments are unambiguously retrievable as $l[i]$, with $l[i] = \downarrow$ if $i > j$.

The indices ind come from a set $\mathcal{INDS}$ isomorphic to $\mathbb{N}$, and terms are successively numbered with it in the order of their creation. Index variables sometimes have a superscript "ind" for distinction. One writes $D[i]$ for the i-th term, i.e., short for $D[ind = i]$. We often say "term i" below for this. The types range over a set $typeset$ with $\mathsf{skse}, \mathsf{symenc} \in typeset$ denoting symmetric encryption keys and symmetric encryptions, respectively. Each handle hnd_u comes from a set $\mathcal{HNDS}$, also isomorphic to $\mathbb{N}$. If it has the "undefined" value $\downarrow$, participant u does not know this term (yet). Particularly important is the adversary handle hnd_a (especially whether it is $\downarrow$ or not). Otherwise, u ranges over a set $\mathcal{H}$ of indices of honest users. Handle variables always get a superscript "hnd".

4.3 The Realization of the BPW Model

The realization of the BPW model is called $Sys^{\mathsf{cry\_sym,real}}_{n,\mathcal{S},\mathcal{E},\mathcal{SA},\mathcal{SE},L'}$. (Here n is the number of participants, $\mathcal{S}$, $\mathcal{E}$, $\mathcal{SA}$ and $\mathcal{SE}$ are underlying secure signature, encryption, symmetric message authentication and symmetric encryption schemes and L' is a tuple of functions determining lengths and runtime bounds.) Here each user u has a separate machine M_u which contains a database D_u where real bitstrings $word$ are stored under the handles hnd_u for this user, as well as for convenience the type $type$. The users can use exactly the same commands as to the BPW model, e.g., en- or decrypt a message etc. These commands now trigger real cryptographic operations. The operations essentially use cryptographically secure primitives – a DKDM-secure symmetric encryption scheme as the main scheme in our case – but with certain additional tagging, randomization etc. Send commands now trigger the actual sending of bitstrings between machines and/or to the adversary.

4.4 Soundness Definition with Key Cycles (DY-BRSIM Security)

Our security claim is that the realization of the BPW model with symmetric encryption is as secure as the BPW model with symmetric encryption in the sense of BRSIM/UC even if the surrounding protocol produces key cycles, as long as it avoids the commitment problem, which we already discussed before the DKDM definition. In the context of an BRSIM/UC soundness proof, it is even clearer that the commitment problem must be avoided because it immediately destroys simulatability. (One could resort to the random oracle model but here one needs a strong version where the simulator reprograms the oracle, so that the notion breaks down if the random oracle is implemented with a real hash function.)

In [4] the definitions of key-cycle freeness and avoidance of the commitment problem are joined, unfortunately under the name of commitment-freeness. Essentially we weaken this definition so that encryption cycles are no longer forbidden. We call the new property "pure commitment-freeness". The difference lies in a predicate $\mathsf{wrapped}$ where $\mathsf{wrapped}(j, i)$ denotes that term j occurs in term i only within encryptions that the adversary cannot decrypt. In [4] this predicate also contains a condition that such inner encryptions are consistent with a linear order on the keys and thus cycle-free. We replace it by a predicate $\mathsf{pure\_wrapped}$ that does not contain this second condition.

The actual definition needs some more notation, mostly from [4]: Given a state of the database D of the BPW model and an index i, the *tree of sub-terms* of term i, written $\mathsf{tree}(D[i])$, is defined as follows: Its root is $D[i]$, and $D[j]$ is a child of $D[k]$ iff

$j \in D[k].arg$. An input by user u to encrypt the term that u knows by handle l^{hnd} with the key it knows by handle sk^{hnd} is written $\mathsf{in}_u?.\mathsf{sym\_encrypt}(sk^{\mathsf{hnd}}, l^{\mathsf{hnd}})$. The notation $\mathsf{in}_u?.\mathsf{send\_A}(l^{\mathsf{hnd}})$ means that user u sends the term it knowns by l^{hnd} so that it will be received by the adversary. (The actual input in the original notation is $\mathsf{send\_}x(l^{\mathsf{hnd}}, v)$ with $x \in \{\mathsf{i}, \mathsf{r}\}$ or $v \notin \mathcal{H}$.) The type enc means an asymmetric ciphertext. The wrapping of a term in another one is now defined as follows.

Definition 7 (Wrapping). *Given a state of the database D of the BPW model and indices i, j, the predicate* $\mathsf{pure\_wrapped}(j, i)$ *is true iff for every occurrence of the node $D[j]$ in* $\mathsf{tree}(D[i])$*, there exists a node $D[k]$ in* $\mathsf{tree}(D[i])$ *such that $D[k].type \in$* $\{\mathsf{symenc}, \mathsf{enc}\}$ *and the following holds: For $pkse := D[k].arg[2]$ (the index of a so-called public tag of the used key; these tags are needed for situations where the adversary can distinguish whether several symmetric encryptions have been made with the same key), $sk := pkse + 1$ (the corresponding secret key) and $l := D[k].arg[1]$ (the encrypted message) we have $D[sk].hnd_{\mathsf{a}} = \downarrow$, i.e., the adversary does not know the key, and the given occurrence of $D[j]$ in the tree is a descendant of $D[l]$. We then say that term j is* wrapped *in term i.* ◇

The property Pure_NoComm denoting the absence of the commitment problem is now defined as in [4] except for using pure_wrapped. It states that if some user u encrypts a term l_1 at time t with a secret key sk unknown to the adversary, and a user v sends a term l_2 at a later time t' such that the adversary learns it and sk is contained in this term, then sk is wrapped in l_2.

Definition 8 (Pure Commitment Freeness). *We say that a run of the BPW model $Sys^{\mathsf{cry\_sym,id}}_{n,L}$ (with users and an adversary) fulfills the property* Pure_NoComm *iff the following holds: If there exists $t \in \mathbb{N}$, $sk \in \mathcal{INDS}$, $u \in \mathcal{H}$, and $l_1^{\mathsf{hnd}} \in \mathcal{HNDS}$ such that for $skse_u^{\mathsf{hnd}} := D[sk].hnd_u$, we have at time t*

$$\mathsf{in}_u?.\mathsf{sym\_encrypt}(skse_u^{\mathsf{hnd}}, l_1^{\mathsf{hnd}}) \;\wedge\; D[sk].type = \mathsf{skse} \;\wedge\; D[sk].hnd_{\mathsf{a}} = \downarrow,$$

then the following must hold for every $t' > t$, $v \in \mathcal{H}$, and $l_2^{\mathsf{hnd}} \in \mathcal{HNDS}$:

$$\begin{aligned}&(\mathsf{in}_v?.\mathsf{send\_A}(l_2^{\mathsf{hnd}}) \;\wedge\; D[sk] \in \mathsf{tree}(D[hnd_v = l_2^{\mathsf{hnd}}]))\\ \Longrightarrow\;& \mathsf{pure\_wrapped}(sk, D[hnd_v = l_2^{\mathsf{hnd}}].ind).\end{aligned}$$

◇

Definition 9 (Purely Commitment-free Users). *A machine* H *interacting with the BPW model $Sys^{\mathsf{cry\_sym,id}}_{n,L}$ is called* purely commitment-free *iff the property* Pure_NoComm *from Definition 8 holds in all runs and with all adversaries. The restriction of blackbox reactive simulatability to purely commitment-free users is denoted by $\geq^{\mathsf{Pure\_NoComm}}$.* ◇

Definition 10 (DY-BRSIM Security). *A symmetric encryption scheme $\mathcal{SE}$ is* DY-BRSIM-secure *iff $Sys^{\mathsf{cry\_sym,real}}_{n,\mathcal{S},\mathcal{E},\mathcal{SA},\mathcal{SE},L'} \geq^{\mathsf{Pure\_NoComm}} Sys^{\mathsf{cry\_sym,id}}_{n,L}$ whenever the other underlying cryptographic systems and L, L' fulfil the requirements from [4].* ◇

In reality, a protocol π using $Sys^{\mathsf{cry\_sym,id}}_{n,L}$ or $Sys^{\mathsf{cry\_sym,real}}_{n,\mathcal{S},\mathcal{E},\mathcal{SA},\mathcal{SE},L'}$ will usually determine whether the property $\mathsf{Pure\_NoComm}$ always holds. Thus one first has to analyze π for commitment-freeness; this is accessible to automated tools. If yes, the BRSIM/UC soundness implies that a formal analysis of other properties of π carried out over the ideal Dolev-Yao functionality is also valid for π using the real functionality.

With these definitions, our soundness theorem can now be written as follows:

Theorem 3 (Poly DKDM → DY-BRSIM). *Every polynomial-oracle DKDM-secure symmetric encryption scheme $\mathcal{SE}$ is also DY-BRSIM secure.* □

We do not prove the necessity of DKDM security, but en- and decryptions as in AKDM and the additional key revelations as in DKDM security do occur in such a symbolic model. (We will see how they occur in *our* proof in Section 4.6, where we use a DKDM oracle within the overall systems.) At least for the goal of BRSIM/UC it seems hard to imagine how such a model could be simulated without allowing these capabilities in the underlying definition of encryption security.

4.5 Overview of the Simulator

For proving a soundness theorem without key cycles, a simulator $\mathsf{Sim}_{\mathcal{H}}$ has been defined in [4] such that the combination of arbitrary polynomial-time users H and adversary A cannot distinguish the combination of the real machines M_u from the combination $\mathsf{TH}_{\mathcal{H}}$ and $\mathsf{Sim}_{\mathcal{H}}$ (for all sets $\mathcal{H}$ indicating the correct machines). We do not need to change this simulator at all, only the later proof of correct simulation. Basically $\mathsf{Sim}_{\mathcal{H}}$ translates real messages from the real adversary A into terms as $\mathsf{TH}_{\mathcal{H}}$ expects them and vice versa. In both directions, $\mathsf{Sim}_{\mathcal{H}}$ has to parse an incoming message completely because it can only construct the other version (abstract or real) bottom-up. This is done by recursive algorithms. The state of $\mathsf{Sim}_{\mathcal{H}}$ mainly consists of a database D_{a}, similar to the databases D_u, but storing the knowledge of the adversary. We omit a detailed description and refer the reader to [4]; our soundness proof can be understood without those details.

4.6 Proof of Correct Simulation

In the overall proof in [4] a system $\mathsf{C}_{\mathcal{H}}$, called initial combined system, is defined that essentially contains all aspects of both the BPW model and its realization. It extends the database D of $\mathsf{TH}_{\mathcal{H}}$ by an attribute $word$ containing real bitstrings as in $\mathsf{M}_{\mathcal{H}}$ or $\mathsf{Sim}_{\mathcal{H}}$. These real words are computed as in $\mathsf{M}_{\mathcal{H}}$ for entries generated by the honest users, and as in $\mathsf{Sim}_{\mathcal{H}}$ for entries resulting from network inputs, i.e., values coming from the adversary. Hence all symmetric encryptions produced by honest users contain a real plaintext message. A second system $\mathsf{C}^*_{\mathcal{H}}$, called final combined system, is equal to $\mathsf{C}_{\mathcal{H}}$ except for symmetric encryptions: For encryptions made by honest users and with keys of honest users and without adversary handle, a simulated message 0^{len^*} of the same length is encrypted instead of a real plaintext message. To distinguish keys generated by honest users from keys generated by the adversary in $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$, entries of type skse have an additional attribute $owner \in \{\mathsf{honest}, \mathsf{adv}\}$. The only part of the overall proof that concerns symmetric encryption and their potential key cycles is the indistinguishability of these two systems $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$. All other proof parts remain exactly the same.

Reduction to DKDM Security of Symmetric Encryption. We now show that the combined systems $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$ are reactively indistinguishable, i.e., black-box simulatable without the need for an additional simulator. The core of this proof is to show how a DKDM oracle $\mathsf{SymDKDM}$ can be used to simulate either $\mathsf{C}_{\mathcal{H}}$ or $\mathsf{C}^*_{\mathcal{H}}$, depending on the bit b in $\mathsf{SymDKDM}$. We call the rest of this simulation $\mathsf{C}'_{\mathcal{H}}$, i.e., the combination of $\mathsf{C}'_{\mathcal{H}}$ and $\mathsf{SymDKDM}$ should yield $\mathsf{C}_{\mathcal{H}}$ or $\mathsf{C}^*_{\mathcal{H}}$ depending on b. Clearly, $\mathsf{C}'_{\mathcal{H}}$ calls $\mathsf{SymDKDM}$ for encryption and decryption with symmetric encryption keys unknown to the adversary. However, the combined systems can also use such a key in operations not supported by $\mathsf{SymDKDM}$, e.g., put the key into a list and send the list over a secure or insecure channel. For these operations we use lazy evaluation, i.e., we leave open as long as possible if the key will remain secret and thus be treated using a reference in a function g, or if it has to be revealed so that $\mathsf{C}'_{\mathcal{H}}$ can use it directly to perform operations without outer encryption. Thus $\mathsf{C}'_{\mathcal{H}}$, in contrast to $\mathsf{C}_{\mathcal{H}}$ and $\mathsf{C}^*_{\mathcal{H}}$, may contain terms for which no real version is yet known, i.e., there are database entries $D[t^{\mathsf{ind}}]$ with $D[t^{\mathsf{ind}}].word = \downarrow$. We call this "uninstantiated". Terms are fully instantiated when they are sent in an insecure way, i.e., if $\mathsf{C}'_{\mathcal{H}}$ has to give the adversary a real message. Then it uses the reveal command of the DKDM oracle to obtain keys about which information is leaked. Additionally, for each symmetric key, $\mathsf{C}'_{\mathcal{H}}$ stores a key number jno that equals the number of this key in $\mathsf{SymDKDM}$. The detailed definition of $\mathsf{C}'_{\mathcal{H}}$ from $\mathsf{C}_{\mathcal{H}}$, which is postponed to Appendix A.3, reconsiders all commands where $\mathsf{C}_{\mathcal{H}}$ constructs a real word corresponding to the symmetric encryption scheme.

The following lemma establishes the indistinguishability of the combined system with DKDM oracle and the initial and final combined systems, respectively, and it thus completes the proof of Theorem 3. The lemma is proved in Appendix A.4.

Lemma 1 (Correct Rewriting of Combined Systems with DKDM Oracle). *The combination of* $\mathsf{C}'_{\mathcal{H}}$ *and* $\mathsf{SymDKDM}$ *with bit* $b = 0$ *is reactively indistinguishable from* $\mathsf{C}_{\mathcal{H}}$*, and with bit* $b = 1$ *it is reactively indistinguishable from* $\mathsf{C}^*_{\mathcal{H}}$*, if the user and/or distinguisher fulfill the property* $\mathsf{Pure\_NoComm}$. □

5 Relations Between the Definitions of Secure Symmetric Encryption

In this section we investigate the relations between our definitions, together with underlying definitions such as IND-CCA2 security and ciphertext integrity. We summarize these results in Figure 1.

Arrows with labels "Th.x" or "L.y" refer to theorems and lemmas in this paper; some other arrows carry citations. Dotted, striked-through arrows show that an implication does not hold. Arrows without a label are clear from the definitions. There are three arrows with a question mark; they all correspond to essentially the same open question whether AKDM security implies DKDM security without a restriction on the number of revealed keys. This question seems similar to the selective decommitment problem [19], but not directly related to the cases with known answers. We believe that all relations between definitions without arrows can be derived from the existing arrows.

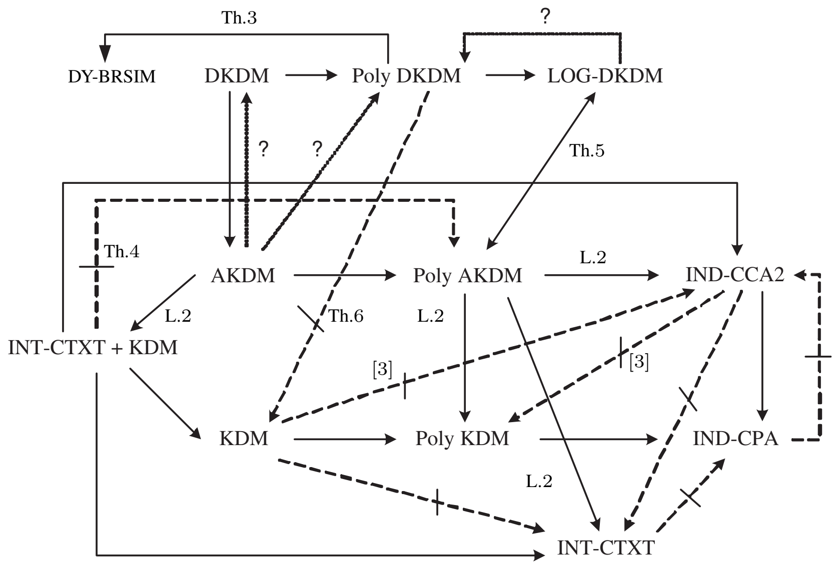

Fig. 1. Summary of the implications between the definitions

5.1 Relations Between AKDM Security and Simpler Definitions

We first show that AKDM security implies the conjunction of the typical simpler definitions, but not vice versa.

Lemma 2 (AKDM → (KDM ∧ INT-CTXT ∧ IND-CCA2)). *A polynomial-oracle or unrestricted AKDM-secure encryption scheme $\mathcal{SE}$ is also polynomial-oracle or unrestricted KDM-secure, respectively, and it provides integrity of ciphertexts and is IND-CCA2-secure.* □

Theorem 4 ((KDM ∧ INT-CTXT ∧ IND-CCA2) $\not\rightarrow$ AKDM). *KDM security, integrity of ciphertexts, and IND-CCA2 security together do not imply polynomial-oracle AKDM security.* □

These proofs are given in Appendix A.5 and A.6, respectively.

As usual, for cryptographic properties P, Q a statement "P does not imply Q" is always interpreted as "If there exists a system $\mathcal{S}$ that fulfills P, then there also exists a system $\mathcal{T}$ that fulfills $P \wedge \neg Q$".

In both these lemmas one could omit mentioning IND-CCA2, because KDM security implies IND-CPA security, and that together with integrity of ciphertexts implies IND-CCA2 as shown in [10].

5.2 Relation of DKDM and AKDM Security

In this section, we consider the relation of AKDM security and DKDM security. We first introduce a variant of DKDM-security where only a bounded number ρ of keys are revealed, because we can prove closer relations with AKDM for it than for the original, stronger DKDM-security.

Definition 11 (Logarithmic Dynamic Key-dependent Message (LOG-DKDM) Security). *An adversary* A *on the DKDM oracle* $\mathsf{SymDKDM}$ *is called* ρ-revealing *for a function* ρ *if it lets at most* $\rho(k)$ *keys be revealed in every run. Let* $\mathcal{A}_{p,\rho}$ *be the class of* p*-oracle-bounded and* ρ*-revealing adversaries from* $\mathcal{A}_\infty$*, where* p *and* ρ *are polynomials or* ∞*. We say that a symmetric encryption scheme* $\mathcal{SE}$ *is* (p, ρ)-bounded DKDM-secure *iff the DKDM advantage is negligible for every adversary* $\mathsf{A} \in \mathcal{A}_{p,\rho}$*, and that it is* LOG-DKDM-secure *iff it is* (p, ρ)*-bounded DKDM-secure for all* p, ρ *where* $\binom{p}{\rho}$ *is polynomial.* ◇

Without loss of generality we can restrict ourselves to classes $\mathcal{A}_{p,\rho}$ witdh $\rho \leq p$. The condition for LOG-DKDM is fulfilled if p^ρ or $p^{p-\rho}$ is polynomially bounded. Two important examples of LOG-DKDM security are that polynomially many keys are generated but only a constant number of them are revealed, or that a linear fraction of the keys is revealed and p^p is polynomial. (Thus p is essentially logarithmic if there is no extra restriction on ρ; this motivates the name "LOG-DKDM".)

Clearly, DKDM security implies AKDM security, and polynomial-oracle DKDM or LOG-DKDM security implies polynomial-oracle AKDM security. We now consider the opposite direction.

Theorem 5 (Polynomial-oracle AKDM → LOG-DKDM). *A polynomial-oracle AKDM-secure symmetric encryption scheme* $\mathcal{SE}$ *is also LOG-DKDM-secure.* □

This proof is given in Appendix A.7.

The question whether AKDM security implies DKDM security without a restriction on the number of revealed keys remains open in this paper. This question bears a strong similarity to the selective decommitment problem [19], but seems not directly related to the cases with known answers.

5.3 The Influence of the Polynomial-Oracle Restriction

We start by showing that the polynomial-oracle restriction on the adversary in the definitions really makes a difference. We first review the original definition of KDM security and augment it with a polynomial-oracle variant.

Definition 12 (KDM Security, with Polynomial-oracle Variant). *Given a symmetric encryption scheme* $\mathcal{SE} = (\mathsf{gen_{SE}}, \mathsf{E}, \mathsf{D})$*, the* key-dependent message oracle *or* KDM oracle SymKDM *is defined as follows: It has a (virtual) infinite sequence of keys* $\boldsymbol{sk} := (sk_i)_{i \in \mathbb{N}}$*, where each key, when first used, is initialized as* $sk_i \leftarrow \mathsf{gen_{SE}}(0^k)$*, a bit* b *initialized as* $b \xleftarrow{\mathcal{R}} \{0, 1\}$*, and the following query type:*

- *On input* (enc, j, g) *with* $j \in \mathbb{N}$ *and* $g \in \mathcal{F}_\infty$*, let* $m_0 := g(\boldsymbol{sk})$ *and* $m_1 := 0^{|m_0|}$ *and output* $c \leftarrow \mathsf{E}(sk_j, m_b)$.

The KDM advantage *of an adversary* A *that interacts with* SymKDM *and finally outputs a bit* b^* *is defined as* $Adv_{\mathsf{KDM}}(\mathsf{A}) := |\Pr[\mathsf{A}^{\mathsf{SymKDM}} = 1 \mid b = 0] - \Pr[\mathsf{A}^{\mathsf{SymKDM}} = 1 \mid b = 1]|$. *We say that* $\mathcal{SE}$ *is* p-oracle-bounded KDM-secure *iff the KDM advantage of every adversary* $\mathsf{A} \in \mathcal{A}_p$ *is negligible, and that it is* polynomial-oracle KDM-secure *iff it is* p*-oracle-bounded KDM-secure for every polynomial* p. ◇

Clearly a KDM-secure encryption system is also polynomial-oracle KDM-secure, and similar for AKDM and DKDM, but we now show that even the strongest of our polynomial-oracle definitions is not sufficient for the weakest unrestricted one.

Theorem 6 (Polynomial-oracle DKDM $\not\rightarrow$ KDM). *Polynomial-oracle DKDM security does not imply unrestricted KDM security.* □

This proof is given in Appendix A.8.

5.4 Longer Key Cycles

Finally, to address the question if the key cycle problem is only due to cycles of length one (the case treated by prior counterexamples), we define security if only longer key cycles are allowed. We first define a key-dependency graph.

Definition 13 (Key-dependency Graph). *For a program* $g \in \mathcal{F}_\infty$ *and a key number* $i \in \mathbb{N}$*, we define "*$i \in g$*" to mean that key* sk_i *is addressed by the program* g*. Let* O *denote an encryption oracle, i.e., an oracle that accepts inputs of the form* (enc, j, g)*. Then given a history of calls to* O*, we define a* current key-dependency graph $\mathcal{G}$*. In the initial state, it has no edges. For each input* (enc, j, g)*, an edge* (j, i) *is added for every* $i \in g$*. By* $\mathcal{G}^*$ *we denote the transitive closure of graph* $\mathcal{G}$*.* ◇

Generally, $\mathcal{G}$ could be written with indices O and h for the machine and the history concerned, but this would seem notational overkill for the following simple definition.

Definition 14 (i-KDM Security). *Let a symmetric encryption scheme* $\mathcal{SE}$ *be given. Let* $\mathcal{A}_p^{i-\mathsf{cycles}} \subseteq \mathcal{A}_p$ *with* p *a polynomial or* ∞ *denote the class of adversaries that never produce key cycles of length less than* i*, i.e., in interaction with the KDM oracle the current key-dependency graph* $\mathcal{G}$ *never has cycles of length less than* i*. We say that* $\mathcal{SE}$ *is* i-KDM-secure *iff the KDM advantage of every adversary* $\mathsf{A} \in \mathcal{A}_\infty^{i-\mathsf{cycles}}$ *is negligible, and* polynomial-oracle i-KDM-secure *iff this holds for all* $\mathsf{A} \in \mathcal{A}_p^{i-\mathsf{cycles}}$ *for every polynomial* p*.* ◇

Theorem 7 (IND-CPA $\not\rightarrow$ polynomial-oracle i-KDM). *For all* $i \in \mathbb{N}$*, there exist symmetric encryption schemes that are semantically secure, but not polynomial-oracle* i*-KDM-secure, at least for stateful schemes.* □

Proof. Let a semantically secure symmetric encryption scheme $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ be given. We modify it as follows to a system $\mathcal{SE}^*$: A new key is a string $sk||r$ with $sk \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$ and $r \stackrel{\mathcal{R}}{\leftarrow} \{0,1\}^k$. Let $m_{|k}$ denote the last k bits of m. Encryption becomes

$$\mathsf{E}^*((sk||r), m) := \begin{cases} (\mathsf{E}(sk, m), m_{|k} \oplus r) & \text{if this is the first encryption with } sk \\ \mathsf{E}(sk, m) & \text{otherwise.} \end{cases}$$

Decryption D^* of a ciphertext (c, tag) or c simply decrypts c with D. Note that the system is stateful as it has to record which key was used for encryption already.

To break the i-KDM security of the new scheme, an adversary A asks for the encryption of sk_j with $sk_{j+1 \bmod i}$ for all $j \in \{1, \ldots, i\}$, i.e., it enters queries $(\mathsf{enc}, j{+}1 \bmod i, \pi_j)$. This set of queries produces only one key cycle of length i; hence A is a permitted adversary. If A is interacting with the real KDM oracle SymKDM with $b = 0$, the resulting ciphertexts are of the form (c_j, tag_j) with $tag_j = r_j \oplus r_{j+1 \bmod i}$, and thus the XOR of all these tags is 0. For the fake oracle, $b = 1$, the tags are $tag_j = 0 \oplus r_{j+1 \bmod i} = r_{j+1 \bmod i}$ and thus the XOR of all of them is usually not 0. This allows A to distinguish the oracles.

The semantic security of the new scheme follows from the fact that the tag for each key is random and thus gives an adversary no new abilities to distinguish. ■

6 Conclusion

We have extended the notion of key-dependent message security (KDM security) to active attacks and to dynamic divulging of keys to the adversary, as far as the latter does not introduce a commitment problem; we call this AKDM and DKDM security. We also introduced a polynomial-oracle version of the definitions that should be easier to fulfil in the standard model of cryptography, where currently no KDM-secure encryption scheme is known, and that are sufficient for the soundness result.

We constructed efficient schemes secure under the new definitions. For AKDM security, the construction is generic, i.e., if a KDM-secure scheme in the standard model can be found, then we automatically get an AKDM-secure scheme in the standard model. For DKDM security, the construction relies directly on a random oracle.

We proved that DKDM security is sufficient for proving a symbolic abstraction of symmetric encryption secure in the strong sense of BRSIM/UC. The soundness of such abstractions was a major motivation already for the introduction of KDM security in [11]. It was first shown in [3], but only for passive attacks, which are not sufficient for most usages of symbolic models. We believe that DKDM security is also necessary for a BRSIM/UC result.

We explored the space of old and new definitions related to KDM security and proved or disproved equivalences between individual definitions and combinations of several definitions. In particular, we showed that AKDM security (and consequently DKDM security) is not a consequence of KDM security and standard definitions of security of symmetric encryption schemes under active attacks, IND-CCA2 and INT-CTXT, so that special constructions are indeed necessary. For stateful encryption schemes, we showed that the separation between KDM security and semantic security is not only possible by key cycles of length 1, but of any minimum length i.

The main new open question is whether AKDM security implies DKDM security, or at least DKDM-secure schemes can be constructed from KDM-secure ones in a way that does not involve a random oracle. We can show the former if we restrict the number of keys generated or revealed in DKDM security. The general problem seems very similar to the selective decommitment problem [19], but not directly related to the cases with known answers.

For Dolev-Yao models with key cycles, the overall consequence is that we have put them on a solid basis with the DKDM definition and the soundness result; however, as currently only specially constructed encryption schemes are provably DKDM-secure, and only in the random oracle model, the soundness is not as good as related results for Dolev-Yao models without key cycles, and we recommend that the latter be used unless there is a specific need for key cycles. (The same even holds for the passive case based on KDM security.) On the other hand, while in many cases one might argue that good cryptographic practice is to construct protocols without key cycles in the first place, formal methods are used precisely to analyze protocols automatically that may not follow any specific design principles.

Acknowledgments We thank Dennis Hofheinz for suggesting an improvement of Theorem 7 that allowed us to get rid of the random oracle in our separating example of semantic security and i-KDM security for stateful schemes, and thus to separate both notions in the standard model. We thank Ran Canetti for helpful suggestions on this paper.

References

1. M. Abadi and J. Jürjens. Formal eavesdropping and its computational interpretation. In *Proc. 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, pages 82–94, 2001.
2. M. Abadi and P. Rogaway. Reconciling two views of cryptography: The computational soundness of formal encryption. In *Proc. 1st IFIP International Conference on Theoretical Computer Science*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2000.
3. P. Adão, G. Bana, J. Herzog, and A. Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *Lecture Notes in Computer Science*, pages 374–396. Springer, 2005.
4. M. Backes and B. Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW)*, pages 204–218, 2004.
5. M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. *IEEE Transactions on Dependable and Secure Computing*, 2(2):109–123, 2005.
6. M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations (extended abstract). In *Proc. 10th ACM Conference on Computer and Communications Security*, pages 220–230, 2003. Full version in IACR Cryptology ePrint Archive 2003/015, Jan. 2003, `http://eprint.iacr.org/`.
7. D. Beaver. Secure multiparty protocols and zero knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.
8. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proc. 38th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, 1997.
9. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology: CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

10. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology: ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.
11. J. Black, P. Rogaway, and T. Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Proc. 9th Annual Workshop on Selected Areas in Cryptography (SAC)*, pages 62–75, 2002.
12. J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology: EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001.
13. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 3(1):143–202, 2000.
14. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001. Extended version in Cryptology ePrint Archive, Report 2000/67, `http://eprint.iacr.org/`.
15. R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key exchange protocols. In *Proc. 3rd Theory of Cryptography Conference (TCC)*, volume 3876 of *Lecture Notes in Computer Science*, pages 380–403. Springer, 2006.
16. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proc. 14th European Symposium on Programming (ESOP)*, pages 157–171, 2005.
17. D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.
18. D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
19. C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.
20. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game – or – a completeness theorem for protocols with honest majority. In *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.
21. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology: CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer, 1990.
22. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
23. P. Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP)*, pages 77–91, 2001.
24. P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. 25th IEEE Symposium on Security & Privacy*, pages 71–85, 2004.
25. M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton Computer Society Notes, Princeton, 1996.
26. C. Meadows. Using narrowing in the analysis of key management protocols. In *Proc. 10th IEEE Symposium on Security & Privacy*, pages 138–147, 1989.
27. S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1991.
28. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC)*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.
29. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 245–254, 2000. Extended version (with Matthias Schunter) IBM Research

Report RZ 3206, May 2000, `http://www.semper.org/sirene/publ/PfSW1_00ReactSimulIBM.ps.gz`.
30. B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proc. 22nd IEEE Symposium on Security & Privacy*, pages 184–200, 2001. Extended version of the model (with Michael Backes) IACR Cryptology ePrint Archive 2004/082, `http://eprint.iacr.org/`.
31. C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology: CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1992.
32. C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
33. A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

A Postponed Proofs and Definitions

This section contains the postponed proofs and reviews the common security notions of symmetric encryption as needed in those proofs.

A.1 Standard Security Definitions and Constructions

We repeat two typical definitions of encryption security as well as the notion of strong unforgeability of MACs.

Definition 15 (Indistinguishability under Chosen-ciphertext Attacks). *Given a symmetric encryption scheme* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$*, the* IND-CCA2 oracle SymCCA2 *is defined as follows:*[3] *It has a variable* sk *initialized as* $sk \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$*, a bit* b *initialized as* $b \xleftarrow{\mathcal{R}} \{0,1\}$*, an initially empty set* C *and the following query types:*

- *On input* (enc, m)*: Let* $m_0 := m$ *and* $m_1 := 0^{|m|}$*, set* $c \leftarrow \mathsf{E}(sk, m_b)$ *and* $C := C \cup \{c\}$*, and output* c*.*
- *On input* (dec, c)*: If* $c \notin C$*, return* $\mathsf{D}(sk, c)$*, else* $\downarrow$*.*

The IND-CCA2 advantage *of an adversary* A *that interacts with* SymCCA2 *and finally outputs a bit* b^* *is defined as* $Adv_{\mathsf{CCA2}}(\mathsf{A}) := |\Pr[\mathsf{A}^{\mathsf{SymCCA2}} = 1 \mid b = 0] - \Pr[\mathsf{A}^{\mathsf{SymCCA2}} = 1 \mid b = 1]|$*. We say that* $\mathcal{SE}$ *is* indistinguishable under adaptive chosen-ciphertext attack *or* IND-CCA2-secure *iff the IND-CCA2 advantage of every polynomial-time adversary is negligible. (Recall that the expression is a function of the security parameter* k *by our conventions.)* $\diamond$

We call SymCCA2 with $b = 0$ the *real IND-CCA2 oracle* and with $b = 1$ the *fake IND-CCA2 oracle*, and similarly for the following oracles that have a bit b.

[3] A rigorous notation would be $\mathsf{SymCCA2}_{\mathcal{SE}}$, but usually the encryption scheme is clear from the context; then we omit it. A similar remark holds for the advantage expression and for all the following oracle definitions.

Definition 16 (Integrity of Ciphertexts). *Given a symmetric encryption scheme* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$*, the* ciphertext-integrity oracle *or* INT-CTXT oracle SymInt *is defined as follows: It has a variable* sk *initialized as* $sk \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$*, an initially empty set* C*, and the following query types:*

- *On input* (enc, m)*: Set* $c \leftarrow \mathsf{E}(sk, m)$ *and* $C := C \cup \{c\}$*, and output* c*.*
- *On input* (dec, c)*: Return* $m := \mathsf{D}(sk, c)$*.*

The INT-CTXT advantage $Adv_{\mathsf{INT}}(\mathsf{A})$ *of an adversary* A *that interacts with* SymInt *is defined as the probability that* SymInt *outputs* $m \neq \downarrow$ *on any input* (dec, c) *with* $c \notin C$*. The encryption scheme is said to provide* integrity of ciphertexts *or to be* INT-CTXT-secure *iff the INT-CTXT advantage of every probabilistic polynomial-time adversary* A *is negligible.* ◇

Definition 17 (Strong Unforgeability of MACs). *Given a MAC scheme* $\mathcal{MAC} = (\mathsf{gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Test})$*, the* strong unforgeability oracle *or* SU oracle MAC_SU *is defined as follows: It has a variable* sk *initialized as* $sk \leftarrow \mathsf{gen}_{\mathsf{MAC}}(0^k)$*, an initially empty set* T*, and the following query types:*

- *On input* (auth, m)*: Set* $t \leftarrow \mathsf{MAC}(sk, m)$*;* $T := T \cup \{(m, t)\}$*, and output* t*.*
- *On input* (test, m, t)*: Return* $\mathsf{Test}(sk, m, t)$*.*

The SU advantage $Adv_{\mathsf{SU}}(\mathsf{A})$ *of an adversary* A *that interacts with* MAC_SU *is defined as the probability that* MAC_SU *returns* true *for an input* (test, m, t) *with* $(m, t) \notin T$*. The MAC scheme is called* strongly unforgeable *iff the SU advantage of every probabilistic polynomial-time adversary* A *is negligible.* ◇

In the proof of Theorem 1 in Section A.2, which exploits this definition, we need multiple MAC keys. A standard hybrid argument shows that strong unforgeability of MACs extends to multiple keys.

Definition 18 (Encrypt-then-MAC Construction). *Let an encryption scheme* $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ *and a MAC scheme* $\mathcal{MAC} = (\mathsf{gen}_{\mathsf{MAC}}, \mathsf{MAC}, \mathsf{Test})$ *be given. Then the* encrypt-then-MAC encryption scheme $\mathsf{Encrypt\_then\_MAC}(\mathcal{SE}, \mathcal{MAC})$ *is defined as follows:*

- *Keys are pairs* (sk^e, sk^f)*, generated as* $sk^e \leftarrow \mathsf{gen}_{\mathsf{SE}}(0^k)$ *and* $sk^f \leftarrow \mathsf{gen}_{\mathsf{MAC}}(0^k)$*.*
- *The encryption* c *of a message* m *with key* (sk^e, sk^f) *is computed as* $c' \leftarrow \mathsf{E}(sk^e, m)$ *and* $c := (c', \mathsf{MAC}(sk^f, c'))$*.*
- *To decrypt a ciphertext* c *with key* (sk^e, sk^f)*, decompose* c *into* (c', t)*. If this fails or* $\mathsf{Test}(sk^f, c', t) \neq \mathsf{true}$*, output* $\downarrow$*, else output* $\mathsf{D}(sk^e, c')$*.*

◇

A.2 Proof of Theorem 1 (AKDM Security of Encrypt-then-MAC Construction)

Assume that an adversary $\mathsf{A} \in \mathcal{A}_p$ has a not negligible AKDM advantage against $\mathcal{SE}^* := \mathsf{Encrypt\_then\_MAC}(\mathcal{SE}, \mathcal{MAC})$, where p is a polynomial or ∞. We construct two adversaries $\mathsf{A}_{\mathcal{SE},1,p}$ and $\mathsf{A}_{\mathcal{SE},2,p}$ against the KDM security of $\mathcal{SE}$ and an adversary

$\mathsf{A}_{\mathcal{MAC},p}$ against the strong unforgeability of $\mathcal{MAC}$, all using A as a blackbox, and we prove that at least one of them succeeds in its attack with not negligible probability.

The adversary $\mathsf{A}_{\mathcal{SE},1,p}$ for a polynomial p is defined as follows. It generates $p(k)$ MAC keys, which we call sk_i^f for $i = 1, \ldots, p(k)$ and maintains an initially empty set C of ciphertexts.

- Whenever A makes an encryption query (enc, j, g), then $\mathsf{A}_{\mathcal{SE},1,p}$ for every $i \in g$ inserts the MAC key sk_i^f into all positions in g where the AKDM oracle for $\mathcal{SE}^*$ would use sk_i^f. This is clearly possible in polynomial-time since g was constructed by the polynomial-time adversary A and, as a program, is therefore of polynomial length. Furthermore, by definition of $\mathcal{A}_p$, only keys with indices $i \leq p(k)$ are used. Call the resulting function g^*. $\mathsf{A}_{\mathcal{SE},1,p}$ then inputs (enc, j, g^*) to the KDM oracle, gets a ciphertext c', computes $t \leftarrow \mathsf{MAC}(sk_j^f, c')$ and $c := (c', t)$, sets $C := C \cup \{(j, c)\}$, and outputs c to A.
- Whenever A makes a decryption query (dec, j, c), then $\mathsf{A}_{\mathcal{SE},1,p}$ decomposes c into (c', t). If this fails or $(j, c) \in C$ or $\mathsf{Test}(sk_j^f, c', t) = \mathsf{false}$, it outputs $\downarrow$ to A. Otherwise it aborts the simulation; let E denote the event that this happens.
- When A outputs a bit b^* (meant to distinguish the real and fake oracle), $\mathsf{A}_{\mathcal{SE},1,p}$ outputs $b' := b^*$.

The definition of $\mathsf{A}_{\mathcal{SE},1,\infty}$ is the same except for the generation and retrieval of the appropriate MAC keys, because A may now use super-polynomial key indices j or $i \in g$. However, still only a polynomial number of keys is used overall, and hence we can use lazy generation. For this, $\mathsf{A}_{\mathcal{SE},1,\infty}$ maintains an initially empty set $\hat{sk}$ of pairs (i, sk_i^f) of a key number and a MAC key instead of $\boldsymbol{sk}$. When needing a MAC key with index i during an encryption query, it checks whether there is an entry $(i, sk_i^f) \in \hat{sk}$. If yes, it reuses this key. Otherwise it generates the key and stores (i, sk_i^f).

The adversary $\mathsf{A}_{\mathcal{SE},2,p}$ for every p is defined like $\mathsf{A}_{\mathcal{SE},1,p}$ except that it outputs $b' := 1$ if the event E occurs, and $b' := 0$ otherwise.

Clearly the new adversaries $\mathsf{A}_{\mathcal{SE},1,p}$ and $\mathsf{A}_{\mathcal{SE},2,p}$ for a polynomial p belong to the class of permitted adversaries $\mathcal{A}_{p'}$ for some polynomial p'. (Typically $p' = p$ because the only change is the substitution of constants for variables in the programs g, which should not increase the runtime, but the programming model is not so precisely fixed that this can be stated.)

The adversary $\mathsf{A}_{\mathcal{MAC},p}$ for a polynomial p is defined as follows. It initially generates $p(k)$ encryption keys sk_i^e and lets the multi-key MAC oracle generate $p(k)$ MAC keys internally. It maintains an initially empty set C of ciphertexts of the encrypt-then-MAC scheme.

- Whenever A makes an encryption query (enc, j, g), then $\mathsf{A}_{\mathcal{MAC},p}$ computes $c' \leftarrow \mathsf{E}(sk_j^e, 0^{|g(\cdot)|})$, i.e., it always encrypts messages like the fake KDM oracle. It then inputs (auth, j, c') to the multi-key SU oracle yielding a tag t, sets $c := (c', t)$ and $C := C \cup \{(j, c)\}$, and outputs c to A.
- Whenever A makes a decryption query (dec, j, c), then $\mathsf{A}_{\mathcal{MAC},p}$ decomposes c into (c', t). If this fails or $(j, c) \in C$, it outputs $\downarrow$ to A. Otherwise it asks the multi-key SU oracle the query $(\mathsf{test}, j, c', t)$. If the result is false, $\mathsf{A}_{\mathcal{MAC},p}$ outputs $\downarrow$ to A,

otherwise it stops the simulation (since t has proved to be a MAC forgery). With respect to the behavior of A, this is exactly the event E again.

For the unrestricted case, $\mathsf{A}_{\mathcal{MAC},\infty}$ is again defined like $\mathsf{A}_{\mathcal{MAC},p}$ for polynomials p except that it uses lazy generation of the encryption keys it really needs.

As long as the event E does not occur, both $\mathsf{A}_{\mathcal{SE},1,p}$ and $\mathsf{A}_{\mathcal{SE},2,p}$ together with the KDM oracle with a bit b perfectly simulate the AKDM oracle with the same bit b for every p.

Furthermore, $\mathsf{A}_{\mathcal{MAC},p}$ together with the multi-key SU oracle perfectly simulates the fake AKDM oracle, i.e., SymAKDM with $b = 1$, until it stops. Let real and fake denote the events $b = 0$ and $b = 1$, respectively. By construction, the probability that $\mathsf{A}_{\mathcal{MAC},p}$ breaks the strong unforgeability of the MAC is $Adv_{\mathsf{SU}}(\mathsf{A}_{\mathcal{MAC},p}) := \mathsf{Pr}[E \mid \mathsf{fake}]$. We distinguish three cases:

- Case 1: If $\mathsf{Pr}[E \mid \mathsf{fake}]$ is not negligible, then $\mathsf{A}_{\mathcal{MAC},p}$ succeeds because $\mathsf{Pr}[E \mid \mathsf{fake}]$ is exactly the success probability $Adv_{\mathsf{SU}}(\mathsf{A}_{\mathcal{MAC},p})$.
- Case 2: If $\mathsf{Pr}[E]$ is negligible, then $\mathsf{A}_{\mathcal{SE},1,p}$ succeeds: The advantage of both $\mathsf{A}_{\mathcal{SE},1,p}$ and $\mathsf{A}_{\mathcal{SE},2,p}$ is

$$\begin{aligned}
&Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},i,p})\\
&= |\mathsf{Pr}[b' = 1 \mid \mathsf{real}] - \mathsf{Pr}[b' = 1 \mid \mathsf{fake}]|\\
&= |\mathsf{Pr}[b' = 1 \mid \mathsf{real} \wedge E] \cdot \mathsf{Pr}[E \mid \mathsf{real}] + \mathsf{Pr}[b' = 1 \mid \mathsf{real} \wedge \neg E] \cdot \mathsf{Pr}[\neg E \mid \mathsf{real}]\\
&- \mathsf{Pr}[b' = 1 \mid \mathsf{fake} \wedge E] \cdot \mathsf{Pr}[E \mid \mathsf{fake}] - \mathsf{Pr}[b' = 1 \mid \mathsf{fake} \wedge \neg E] \cdot \mathsf{Pr}[\neg E \mid \mathsf{fake}]|.
\end{aligned}$$

  For $\mathsf{A}_{\mathcal{SE},1,p}$ and if $\neg E$ holds, i.e., in the second and fourth term of the sum, we have $b' = b^*$ by construction. As $\mathsf{Pr}[E]$ is negligible, the first and third term are negligible. This also holds if we replace b' by b^* in these terms. The resulting formula, where all b''s are replaced by b^*'s, is precisely the advantage $Adv_{\mathsf{AKDM}}(\mathsf{A})$ against the AKDM security of $\mathcal{SE}^*$.
  Hence $Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},1,p}) \geq Adv_{\mathsf{AKDM}}(\mathsf{A}) - \epsilon(k)$ for some negligible $\epsilon(k)$. By assumption, $Adv_{\mathsf{AKDM}}(\mathsf{A})$ is not negligible. Thus $Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},1,p})$ is also not negligible.
- Case 3: If $\mathsf{Pr}[E \mid \mathsf{fake}]$ is negligible and $\mathsf{Pr}[E]$ is not negligible, then $\mathsf{A}_{\mathcal{SE},2,p}$ succeeds: We use the fomula for $Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},2,p})$ from Case 2. Since $\mathsf{A}_{\mathcal{SE},2,p}$ outputs 1 if the event E occurs and 0 otherwise, the second and fourth term of the sum are zero and $\mathsf{Pr}[b' = 1 \mid \mathsf{real} \wedge E] = \mathsf{Pr}[b' = 1 \mid \mathsf{fake} \wedge E] = 1$. Thus $Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},2,p}) = |\mathsf{Pr}[E \mid \mathsf{real}] - \mathsf{Pr}[E \mid \mathsf{fake}]|$. As $\mathsf{Pr}[E \mid \mathsf{fake}]$ is negligible and $\mathsf{Pr}[E]$ is not negligible in this case, $\mathsf{Pr}[E \mid \mathsf{real}]$ is not negligible. Therefore $Adv_{\mathsf{KDM}}(\mathsf{A}_{\mathcal{SE},2,p})$ is not negligible either.

As one of these cases must be true, one of our three adversaries succeeds with not negligible probability. This is the desired contraction and finishes the proof.

A.3 Definition of Combined Systems with DKDM Oracle

Definition 19 (Combined Systems with DKDM Oracle). *The* combined system with DKDM oracle $\mathsf{C}'_{\mathcal{H}}$ *is defined like* $\mathsf{C}_{\mathcal{H}}$ *with the following exceptions: The database entries for secret keys are extended by an attribute* jno*, and there is a counter* cnt *for* *secret keys, initialized as* 0*. The following changes are made for inputs at every port* $\mathsf{in}_u?$ *with* $u \in \mathcal{H}$*:*

- *In key generation,* $\mathsf{C}'_{\mathcal{H}}$ *sets* $D[sk^{\mathsf{ind}}].jno := {+}{+}cnt$ *for the new database entry* $D[sk^{\mathsf{ind}}]$*. The word attribute implicitly remains undefined, i.e.,* $D[sk^{\mathsf{ind}}].word = \downarrow$*.*
- *In an encryption command* $c^{\mathsf{hnd}} \leftarrow \mathsf{sym\_encrypt}(skse^{\mathsf{hnd}}, l^{\mathsf{hnd}})$*, let* $sk^{\mathsf{ind}} := D[hnd_u = skse^{\mathsf{hnd}}].ind$ *and* $c^{\mathsf{ind}} := D[hnd_u = c^{\mathsf{hnd}}].ind$*. If* $D[sk^{\mathsf{ind}}].owner = \mathsf{adv}$*, then* $\mathsf{C}'_{\mathcal{H}}$ *acts like* $\mathsf{C}_{\mathcal{H}}$*, else it leaves* $D[c^{\mathsf{ind}}]$ *uninstantiated.*
- *In all other commands except sending, i.e., list construction, signatures etc., the potential new entry is instantiated as before if all its (direct) arguments are instantiated, otherwise it is left undefined. (Note that a decryption command in* $\mathsf{C}_{\mathcal{H}}$ *never computes a new attribute* $word$*, i.e., we need not consider it here.)*
- *In an operation* $\mathsf{send\_A}(l^{\mathsf{hnd}})$ *let* $l^{\mathsf{ind}} := D[hnd_u = l^{\mathsf{hnd}}].ind$*. (Recall that this denotes send operations where the sent term becomes known to the adversary.) Let* $T := \mathsf{tree}(D[l^{\mathsf{ind}}])$*, and let* T_{enc} *be the tree of symmetric ciphertexts within* T*: Its root is* $D[l^{\mathsf{ind}}]$ *(even if this is not a symmetric ciphertext) and* $D[c^{\mathsf{ind}}]$ *is a child of* $D[c'^{\mathsf{ind}}]$ *in* T_{enc} *if it is a symmetric ciphertext, i.e.,* $D[c^{\mathsf{ind}}].type = \mathsf{symenc}$ *and a descendent of* $D[c'^{\mathsf{ind}}]$ *in* T *and no other entry on the path between them is a symmetric ciphertext.*

  Furthermore, for every t^{ind} *let* $\mathsf{tree\_top}(D[t^{\mathsf{ind}}])$ *denote the tree derived from* $\mathsf{tree}(D[t^{\mathsf{ind}}])$ *by treating inner symmetric ciphertexts as leaves.*

  First $\mathsf{C}'_{\mathcal{H}}$ *recursively instantiates all symmetric encryption keys in* T *that leak during this sending by applying a recursive procedure* $\mathsf{inst\_keys}(l^{\mathsf{ind}})$*. Generally,* $\mathsf{inst\_keys}(t^{\mathsf{ind}})$ *acts as follows:*

  - *Reveal all uninstantiated secret keys in* $T' := \mathsf{tree\_top}(D[t^{\mathsf{ind}}])$*, i.e., for all* $D[sk^{\mathsf{ind}}] \in T'$ *with* $D[sk^{\mathsf{ind}}].type = \mathsf{skse}$ *and* $D[sk^{\mathsf{ind}}].word = \downarrow$*, input* $(\mathsf{reveal}, D[sk^{\mathsf{ind}}].jno)$ *to* $\mathsf{SymDKDM}$*. For the resulting output* sk*, set* $D[sk^{\mathsf{ind}}].word := sk$*.*
  - *Call* $\mathsf{inst\_keys}(c^{\mathsf{ind}})$ *for all directly enclosed ciphertexts that the adversary can decrypt, i.e., for every child* $D[c^{\mathsf{ind}}]$ *of* $D[t^{\mathsf{ind}}]$ *in* T_{enc} *where* $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} \neq \downarrow$ *for* $sk^{\mathsf{ind}} := D[c^{\mathsf{ind}}].arg[1]$*.*

  Secondly, $\mathsf{C}'_{\mathcal{H}}$ *instantiates the uninstantiated ciphertexts* $D[c^{\mathsf{ind}}] \in T_{enc}$ *bottom-up, and for each such ciphertext also* $\mathsf{tree\_top}(D[c^{\mathsf{ind}}])$*, i.e., the directly enclosed terms. We call the procedure for one such ciphertext* $\mathsf{inst\_enc}(c^{\mathsf{ind}})$*. Bottom-up implies that within this procedure, all children of* $D[c^{\mathsf{ind}}]$ *in* T_{enc}*, the directly enclosed ciphertexts, are instantiated.*

  In $\mathsf{inst\_enc}(c^{\mathsf{ind}})$ *we first distinguish whether the key is known: Let* $sk^{\mathsf{ind}} := D[c^{\mathsf{ind}}].arg[1]$*. If* $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} \neq \downarrow$*, then the first recursion, i.e., the procedure* $\mathsf{inst\_keys}$*, ensured that all secret keys in* $T' := \mathsf{tree\_top}(D[c^{\mathsf{ind}}])$ *are instantiated. Hence all leaves in* T' *are instantiated and* $\mathsf{C}'_{\mathcal{H}}$ *can instantiate the rest of* T' *like* $\mathsf{C}_{\mathcal{H}}$*, i.e., compute the attributes* $word$ *for all the sub-term entries.*

  Otherwise, $\mathsf{C}'_{\mathcal{H}}$ *sets* $j := D[sk^{\mathsf{ind}}].jno)$ *and* $p^{\mathsf{ind}} := D[c^{\mathsf{ind}}].arg[2]$*. It then obtains the ciphertext by a call* $c \leftarrow (\mathsf{enc}, j, g_{l^{\mathsf{ind}}})$ *to* $\mathsf{SymDKDM}$*, and sets* $D[c^{\mathsf{ind}}].word := c$*. Here the function* $g_{p^{\mathsf{ind}}}$ *corresponding to evaluating the plaintext term is constructed as follows: Let and* $T'' := \mathsf{tree\_top}(D[p^{\mathsf{ind}}])$*. Then we translate* T'' *into a function bottom-up:*

- *Leaves other than secret keys are instantiated, i.e., constants in the function.*
- *A leaf that is a secret key is instantiated by the RAM program $\pi_{j'}$ where j' is the attribute jnr of this secret key entry.*
- *Lists are translated in the canonical way.*
- *For a signature or a MAC, which are probabilistic operations, choose a random string r of sufficient length for the computation and construct the deterministic function that computes the signature or MAC with this randomness. This randomness must be stored with this entry and reused if this signature or MAC occurs again.*
- *For an asymmetric encryption, the combined systems always encrypt strings of zeros, i.e., an asymmetric encryption term is always fully instantiated.*

Finally, when a message is received from the adversary A*, the recursive procedure for converting terms into corresponding bitstrings (called* $\mathsf{id2real}$ *in [4]), like* $\mathsf{Sim}_{\mathcal{H}}$*, tries to decrypt received ciphertexts with all keys known to the adversary, i.e., keys with* $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} \neq \downarrow$*. Then we have that* $D[sk^{\mathsf{ind}}].word \neq \downarrow$ *in* $\mathsf{C}'_{\mathcal{H}}$ *and thus* $\mathsf{C}'_{\mathcal{H}}$ *can decrypt like* $\mathsf{C}_{\mathcal{H}}$*.* ◇

A.4 Proof of Lemma 1 (Correct Rewriting of Combined Systems with DKDM Oracle)

We first prove the following additional lemma:

Lemma 3. *The combination of* $\mathsf{C}'_{\mathcal{H}}$ *and* $\mathsf{SymDKDM}$ *(independent of bit* $b = 0$*) always fulfils the following invariant: For all* sk^{ind} *with* $D[sk^{\mathsf{ind}}].type = \mathsf{skse}$ *and* $D[sk^{\mathsf{ind}}].owner = \mathsf{honest}$*, we have*

$$D[sk^{\mathsf{ind}}].jno \in Rev \iff D[sk^{\mathsf{ind}}].word \neq \downarrow \iff D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} \neq \downarrow .$$

Furthermore, if these conditions are true, then $D[sk^{\mathsf{ind}}].word := sk_j$ *for* $j := D[sk^{\mathsf{ind}}].jno$*.* □

Proof. The first equivalence is clear because $\mathsf{C}'_{\mathcal{H}}$ instantiates a secret key, i.e., assigns a value to its attribute $word$, exactly in all situations after it made an input $(\mathsf{reveal}, D[sk^{\mathsf{ind}}].jno)$ to $\mathsf{SymDKDM}$. For the resulting output sk, it sets $D[sk^{\mathsf{ind}}].word := sk$, which proves the last statement.

The second equivalence is true because assigning adversary handles follows exactly the same recursion pattern as the first recursion in Definition 19. More precisely, a symmetric key with the attribute $D[sk^{\mathsf{ind}}].owner = \mathsf{honest}$ was originally created by an honest user and thus with $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} = \downarrow$. An adversary handle is later only assigned in the evaluation of a send command, and there in the procedure $\mathsf{id2real}$ that $\mathsf{C}'_{\mathcal{H}}$ executes like TH and Sim, at least as far as handles go. This procedure also follows the tree of the sent term, assigning adversary handles to all subterms outside encryptions, and within encryptions with keys that already have adversary handles. ■

We now proceed with the proof of Lemma 1. First, the last sentence of the construction contained a claim. This is always true by Lemma 3.

The main statement of the lemma looks quite clear for $b = 0$ because $\mathsf{C}'_{\mathcal{H}}$ essentially acts like $\mathsf{C}_{\mathcal{H}}$. It only uses $\mathsf{SymDKDM}$ for some operations with symmetric encryption keys, and when it performs other operations on the same encryption keys itself, it uses the same actual bitstring $D[sk^{\mathsf{ind}}].word := sk_j$ by Lemma 3.

We only have to show that $\mathsf{SymDKDM}$ performs operations with encryption keys like $\mathsf{C}_{\mathcal{H}}$ would. This is clear by construction except in cases where $\mathsf{SymDKDM}$ refuses an operation. These cases are an encryption when already $j \in Rev$, and a revelation if already $j \in Enc$. Thus we show that $\mathsf{C}'_{\mathcal{H}}$ does not make such requests.

A query of the form (enc, j, g) is only made in a send operation, and only if $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} = \downarrow$ for the (one) key with $D[sk^{\mathsf{ind}}].jno = j$. Then $j \notin Rev$ by Lemma 3.

A query (reveal, j) is also only made in a send operation. Assume that this happens at time t_2, and for contradiction that $j \in Enc$ at this time. Then a query $(\mathsf{enc}, j, \cdot)$ must have been made at a time $t_1 < t_2$. As in the previous paragraph, this implies $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} = \downarrow$ at time t_1 for the key with $D[sk^{\mathsf{ind}}].jno = j$. The query $(\mathsf{enc}, j, \cdot)$ is only made by $\mathsf{C}'_{\mathcal{H}}$ if the term sent contains an encryption with the j-th key, i.e., there exists a term $D[c^{\mathsf{ind}}]$ with $D[c^{\mathsf{ind}}].type = \mathsf{symenc}$ and $D[c^{\mathsf{ind}}].arg[1] = sk^{\mathsf{ind}}$. Such a term must have been constructed with a command $\mathsf{sym\_encrypt}(sk^{\mathsf{hnd}}, l^{\mathsf{hnd}})$ by a user u with $D[hnd_u = sk^{\mathsf{hnd}}].ind = sk^{\mathsf{ind}}$ at a time $t_0 < t_1$. Furthermore, we have $u \in \mathcal{H}$ because $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} = \downarrow$ also at time t_0. This is the precondition for the property $\mathsf{Pure\_NoComm}$. Thus the implication of this property must hold for our considered sending operation at time t_2, i.e., the key must be wrapped in the term t that is actually sent, formally $\mathsf{pure\_wrapped}(sk^{\mathsf{ind}}, t^{\mathsf{ind}})$. We now show that this contradicts the precondition that a query (reveal, j) was made in this operation. By definition, $\mathsf{pure\_wrapped}(sk^{\mathsf{ind}}, t^{\mathsf{ind}})$ means that for every occurrence of $D[sk^{\mathsf{ind}}]$ in $\mathsf{tree}(D[t^{\mathsf{ind}}])$, there exists an intermediate encryption $D[k^{\mathsf{ind}}]$ with $D[k^{\mathsf{ind}}] = \mathsf{symenc}$ (the alternative enc is not possible here because $\mathsf{C}_{\mathcal{H}}$ and thus $\mathsf{C}'_{\mathcal{H}}$ contains only fake asymmetric encryptions) and $D[k^{\mathsf{ind}}].hnd_{\mathsf{a}} = \downarrow$. Then, however, the recursive procedure $\mathsf{inst\_keys}(t^{\mathsf{ind}})$ does not reach $D[sk^{\mathsf{ind}}]$ and thus no query (reveal, j) is made. This finishes the proof that $\mathsf{C}'_{\mathcal{H}}$ together with $\mathsf{SymDKDM}$ with bit $b = 0$ perfectly simulates $\mathsf{C}_{\mathcal{H}}$.

Now we consider $b = 1$, i.e., the fake DKDM oracle. The only difference is that in certain encryptions, a zero-string is now encrypted instead of a real message. This happens iff $\mathsf{SymDKDM}$ is called for an encryption, and thus if the ciphertext was initially uninstantiated and therefore not obtained from the adversary, and if $D[sk^{\mathsf{ind}}].owner = \mathsf{honest}$ and $D[sk^{\mathsf{ind}}].hnd_{\mathsf{a}} \neq \downarrow$.

Similarly, $\mathsf{C}^*_{\mathcal{H}}$ only differs from $\mathsf{C}_{\mathcal{H}}$ in encrypting zero-strings if encryptions are made by honest users and with keys of honest users that have no adversary handle. This is the same condition. Furthermore, in both cases this replacement is applied immediately before the operation $\mathsf{sym\_encrypt}$.

A.5 Proof of Lemma 2 (AKDM → (KDM ∧ INT-CTXT ∧ IND-CCA2))

That AKDM security implies KDM security is clear since for an adversary making only encryption queries, an AKDM oracle acts exactly like a KDM oracle with the same bit b.

Once we also showed integrity of ciphertexts, IND-CCA2 security follows because KDM security implies IND-CPA security, and with the same proof this holds for polynomial-oracle KDM security. Then IND-CPA security together integrity of ciphertexts implies IND-CCA2 security, see Figure 1.

For proving integrity of ciphertexts, assume that an adversary A has a not negligible INT-CTXT advantage δ. We construct an adversary $\mathsf{A}_{\mathsf{AKDM}}$ against the AKDM oracle $\mathsf{SymAKDM}$ as follows, using A as a blackbox:

- It translates each encryption query (enc, m) from A into a query $(\mathsf{enc}, 1, g_m)$ to $\mathsf{SymAKDM}$, where g_m denotes the RAM program that maps everything to the constant m. For the resulting ciphertext c, it stores (m, c) in a set C and returns c to A.
- For each decryption query (dec, c) from A it first checks whether there exists a pair $(m, c) \in C$. If yes, it returns m to A. If not, it inputs $(\mathsf{dec}, 1, c)$ into $\mathsf{SymAKDM}$. If the result is $\downarrow$, it returns that to A. Otherwise it outputs $b^* = 1$ and aborts the simulation. Let E denote this event.
- If A finishes its attack and $\mathsf{A}_{\mathsf{AKDM}}$ has not aborted, then $\mathsf{A}_{\mathsf{AKDM}}$ outputs $b^* = 0$.

Clearly $\mathsf{A}_{\mathsf{AKDM}} \in \mathcal{A}_p$ already for a polynomial p of degree 1.

If $b = 0$ in $\mathsf{SymAKDM}$, i.e., the oracle is real, then $\mathsf{A}_{\mathsf{AKDM}}$ perfectly simulates the ciphertext-integrity oracle SymInt until a potential occurrence of E. This event, a new valid ciphertext from A, happens exactly if A is successful. Thus $\mathsf{Pr}[b^* = 1 \mid b = 0] = \delta$. If $b = 1$, the oracle is fake and never decrypts a new ciphertext c. Thus $\mathsf{Pr}[b^* = 1 \mid b = 1] = 0$. Hence the AKDM advantage of $\mathsf{A}_{\mathsf{AKDM}}$ is not negligible.

A.6 Proof of Theorem 4 ((KDM $\wedge$ INT-CTXT $\wedge$ IND-CCA2) $\not\rightarrow$ AKDM)

Let an encryption scheme $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ be given that is KDM secure and provides integrity of ciphertexts. (Recall that IND-CCA2 security is a consequence of these properties.) We construct an encryption scheme $\mathcal{SE}^* = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}^*, \mathsf{D}^*)$ that is not polynomial-oracle AKDM-secure, but still provides KDM security and integrity of ciphertexts. Let

- $\mathsf{E}^*(sk, m) \leftarrow \mathsf{E}(sk, m)||0$;
- $\mathsf{D}^*(sk, c||0) := \mathsf{D}(sk, c)$;
- For $\mathsf{D}^*(sk, c||1)$, let $m := \mathsf{D}(sk, c)$. If $m = sk$, output m, else $\downarrow$.

To show that $\mathcal{SE}^*$ is not AKDM-secure, we define an adversary $\mathsf{A}_{\mathsf{AKDM}}$ that makes two queries to the ADKM oracle:

- It first inputs $(\mathsf{enc}, 1, \pi_1)$ and decomposes the resulting ciphertext c into $c'||0$.
- It then inputs $(\mathsf{dec}, 1, c'||1)$ and expects a message m.

As $c' = \mathsf{E}(sk, sk)$ after the first query, the second query succeeds with the output $m = sk$ when $\mathsf{A}_{\mathsf{AKDM}}$ interacts with the real AKDM oracle, while the fake AKDM oracle always outputs $\downarrow$ on decryption queries. Thus $\mathsf{A}_{\mathsf{AKDM}}$ can distinguish the two oracles.

KDM security follows directly from the KDM security of $\mathcal{SE}$ because appending 0 to ciphertexts does not change anything when only encryption queries can be made.

Finally we show integrity of ciphertexts based on the integrity of ciphertexts of $\mathcal{SE}$. Assume we have a successful adversary $\mathsf{A}_{\mathsf{INT}^*}$ against the oracle $\mathsf{SymInt}_{\mathcal{SE}^*}$. (Here we once need the fully indexed notation for the oracles.) We then construct an adversary $\mathsf{A}_{\mathsf{INT}}$ against the oracle $\mathsf{SymInt}_{\mathcal{SE}}$ as follows, using $\mathsf{A}_{\mathsf{INT}^*}$ as a blackbox:

- It forwards an encryption query (enc, m) to SymInt. If the resulting ciphertext is c, it returns $c||0$ to $\mathsf{A}_{\mathsf{INT}^*}$ and stores c.
- For a decryption query $(\mathsf{dec}, c'||0)$, it forwards (dec, c') to SymInt and forwards the result m.
- For a decryption query $(\mathsf{dec}, c'||1)$, it returns $\downarrow$.

This simulation is only incorrect if $\mathsf{A}_{\mathsf{INT}^*}$ can produce a ciphertext $c'||1$ for which $\mathsf{SymInt}_{\mathcal{SE}^*}$ would not return $\downarrow$, i.e., where $\mathsf{D}(sk, c') = sk$. This only happens with negligible probability.

A.7 Proof of Theorem 5 (Polynomial-oracle AKDM → LOG-DKDM)

Let $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ be a polynomial-oracle AKDM-secure symmetric encryption scheme. Assume that it is not LOG-DKDM-secure, i.e., there exist polynomials p, ρ and an adversary $\mathsf{A} \in \mathcal{A}_{p,\rho}$ such that $\binom{p}{\rho}$ is polynomial and A has a not negligible DKDM advantage against $\mathcal{SE}$.

We construct an adversary $\mathsf{A}_{\mathsf{AKDM}}$ against the AKDM oracle $\mathsf{SymAKDM}$, using A as a blackbox. Initially, $\mathsf{A}_{\mathsf{AKDM}}$ randomly selects a subset $S \subseteq \{1, \ldots, p(k)\}$ of size at most $\rho(k)$. Essentially, S is a guess at the set Rev of the keys that will be revealed throughout the run. $\mathsf{A}_{\mathsf{AKDM}}$ therefore generates keys sk_i^* for all $i \in S$ itself. Furthermore, it maintains an initially empty set C of ciphertexts made and sets Rev and Enc of keys already revealed or used for encryption, respectively. Then it handles queries from A as follows:

- On input (enc, j, g): If $j \in Rev$, it returns $\downarrow$ to A. Else if $j \in S$ (i.e., the key is assumed to be revealed later), it aborts. We call this event E_1.
  Otherwise it sets $Enc := Enc \cup \{j\}$ and modifies g to a function g^* by replacing every reference to sk_i in g with $i \in S$ by its own actual key sk_i^*. It inputs (enc, j, g^*) into $\mathsf{SymAKDM}$, obtains a ciphertext c, stores (j, c) in C and outputs c to A.
- On input (dec, j, c): If $j \in Rev$ or $(j, c) \in C$, it returns $\downarrow$ to A. Else if $j \in S$, it aborts. We call this event E_2. Otherwise it sets $Enc := Enc \cup \{j\}$, inputs (dec, j, c) to $\mathsf{SymDKDM}$, and forwards the resulting output m to A.
- On input (reveal, j): If $j \in Enc$, it returns $\downarrow$ to A. Else if $j \in S$, it returns sk_j^* and sets $Rev := Rev \cup \{j\}$; otherwise it aborts. We call this abort event E_3.
- If A outputs a bit b^*, then $\mathsf{A}_{\mathsf{AKDM}}$ outputs this as its own bit.

We have $\mathsf{A}_{\mathsf{AKDM}} \in \mathcal{A}_p$ because it only gets polynomially many queries, and the functions g it obtains are of polynomial length and only address key indices up to $p(k)$ by the precondition $\mathsf{A} \in \mathcal{A}_{p,\rho}$.

Clearly, $\mathsf{A}_{\mathsf{AKDM}}$ together with $\mathsf{SymAKDM}$ with bit $b = 0$ (the real oracle) perfectly simulates $\mathsf{SymDKDM}$ with bit $b = 0$ until a potential event E_1, E_2, or $E3$, because for the indices $i \in S$ it consistently uses its own keys sk_i^* and for $i \notin S$ it consistently uses the keys sk_i in $\mathsf{SymAKDM}$. This also holds for $b = 1$ because no encryptions and decryptions are made with the keys sk_i^* (where the fake oracle would deviate).

We now show that if S is a correct guess at the final value of the set Rev, short *is correct*, then no event E_1, E_2, or E_3 can occur: A query (enc, j, g) or (dec, j, g) with $j \notin Rev$ causes j to be put into Enc, and thus $j \in Rev$ cannot become true later in the run. (Recall that always $Rev \cap Enc = \emptyset$ and that the sets only grow.) Thus the condition $j \in S$ of E_1 and E_2 cannot be true. A query (reveal, j) with $j \notin Enc$ causes j to be put into Rev. Thus the condition $j \notin S$ of E_3 cannot be true.

Hence if S is correct, the simulation is perfect until its end, and the output of $\mathsf{A}_{\mathsf{AKDM}}$ is correct iff that of A is correct.

Furthermore, the perfect simulation implies that the view of A is independent of S until a potential event E_1, E_2, or E_3. Thus the probability that S is correct is $\binom{p}{\rho}^{-1}$. Hence the advantage of $\mathsf{A}_{\mathsf{AKDM}}$ is at least $Adv_{\mathsf{AKDM}}(\mathsf{A}_{\mathsf{AKDM}}) \geq \binom{p}{\rho}^{-1} Adv_{\mathsf{DKDM}}(\mathsf{A})$. This is not negligible, and thus the desired contradition to AKDM security.

A.8 Proof of Theorem 6 (Polynomial-oracle DKDM $\not\to$ KDM)

Let $\mathcal{SE} = (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}, \mathsf{D})$ be a polynomial-oracle DKDM-secure symmetric encryption scheme. Without loss of generality, let key generation be the uniformly random choice from $\{0,1\}^k$. (Typically this will be true anyway; otherwise one can treat the original randomness in $\mathsf{gen}_{\mathsf{SE}}$ as the key.) Let f denote a one-way permutation with domains and ranges $\{0,1\}^k$. Let $\mathcal{SE}^* := (\mathsf{gen}_{\mathsf{SE}}, \mathsf{E}^*, \mathsf{D})$ be the symmetric encryption scheme where $\mathsf{E}^*(sk, m) = 0$ if $f(m) = sk$, else $\mathsf{E}^*(sk, m) = \mathsf{E}(sk, m)$.

We first show that $\mathcal{SE}^*$ is not unrestricted KDM-secure. An adversary A^* constructs the deterministic function g that breaks f by brute-force search for the pre-image of the key sk_1 among strings of length k and inputs $(\mathsf{enc}, 1, g)$ to the oracle SymKDM. For $b = 0$ (the real oracle) this always yields 0 by construction, whereas for $b = 1$ (the fake oracle) the result is $\mathsf{E}(sk_1, 0^k)$, which is not equal to 0 with overwhelming probability. Thus A^* can easily distinguish real and fake oracle.

We now show that $\mathcal{SE}^*$ is polynomial-oracle DKDM-secure. Assume for contradiction that an adversary $\mathsf{A} \in \mathcal{A}_p$ has a not negligible DKDM advantage for a certain polynomial p. We then construct an adversary A_f that breaks the underlying one-way permutation f using A as a blackbox. Initially A_f gets an input sk^*, which was randomly chosen from $\{0,1\}^k$. It selects $b \xleftarrow{\mathcal{R}} \{0,1\}$ and $i \xleftarrow{\mathcal{R}} \{1, \ldots, p(k)\}$ and sets $sk_i := sk^*$ and $sk_j \xleftarrow{\mathcal{R}} \{0,1\}^k$ for all $j \in \{1, \ldots, p(k)\} \setminus \{i\}$. Now A_f simulates the oracle $\mathsf{SymDKDM}$ for bit b; this is clearly possible as it knows all the keys. For every encryption query (enc, i, g) from A, i.e., with the chosen key index, it additionally checks if $f(g(\boldsymbol{sk})) = sk_i$. If yes, it outputs $g(\boldsymbol{sk})$ as a pre-image of sk^* and stops the simulation.

Since A only makes polynomially many queries, and the time complexity of every function g that A_f has to evaluate in encryption queries is at most $p(k)$, the adversary A_f runs in polynomial time.

Let E_j denote the event that A in interaction with oracle $\mathsf{SymDKDM}_p$ makes a query (enc, j, g) such that $f(g(\boldsymbol{sk})) = sk_j$, and let $E := E_1 \cup \ldots \cup E_{p(k)}$. If the probability $\mathsf{Pr}[E]$ were negligible, then the same A would also have a not negligible DKDM advantage for the underlying system $\mathcal{SE}$, contradicting the polynomial-oracle DKDM security of $\mathcal{SE}$. Hence $\mathsf{Pr}[E_i]$ is not negligible. Until a potential occurrence of E_i, the adversary A_f perfectly simulates $\mathsf{SymDKDM}$. Thus the probability that event E_i occurs in the simulation is the same value $\mathsf{Pr}[E_i]$ as in the original attack, and given the uniformly random choice of i, the probability of E_i in the simulation is at least $\mathsf{Pr}[E]/p(k)$. This is still not negligible, and thus the desired contradiction to the security of the one-way permutation.