Simplifying Game-Based Definitions

Indistinguishability up to Correctness and Its Application to Stateful AE

Phillip Rogaway and Yusi Zhang

Computer Science Department
University of California Davis, One Shields Avenue, USA

Abstract. Often the simplest way of specifying game-based cryptographic definitions is apparently barred because the adversary would have some trivial win. Disallowing or invalidating these wins can lead to complex or unconvincing definitions. We suggest a generic way around this difficulty. We call it *indistinguishability up to correctness*, or IND|C. Given games G and H and a correctness condition C we define an advantage measure $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}$ wherein G/H distinguishing attacks are effaced to the extent that they are inevitable due to C. We formalize this in the language of *oracle silencing*, an alternative to exclusion-style and penalty-style definitions. We apply our ideas to a domain where game-based definitions have been cumbersome: stateful authenticated-encryption (sAE). We rework existing sAE notions and encompass new ones, like replay-free AE permitting a specified degree of out-of-order message delivery.

Keywords: indistinguishability · oracle silencing · provable security · stateful authenticated encryption

1 Introduction

This paper addresses a common difficulty one encounters in giving game-based cryptographic definitions: the need to ensure that adversaries don't get credit for trivial wins. But what exactly *is* a trivial win? Sometimes answering this is *not* trivial. Our simple but previously unexplored idea is to use a scheme's correctness requirement to automatically determine if a win should or shouldn't count. We believe that this can lead to simpler and more compelling definitions.

Correctness requirements—for example, that a decryption algorithm properly reverses the corresponding encryption algorithm—are normally understood as demands on functionality, not security. Yet we will use correctness to help define security. More specifically, a correctness condition will be used to map a pair of games that an adversary *can* trivially distinguish into a pair of games that it *can't* trivially distinguish. The modified games are identical to the original ones apart from eliminating wins that exploit generic checks on correctness. The adversary's advantage in distinguishing the modified games is elevated to a definition for *indistinguishability up to correctness*, or IND|C. In our main elaboration of this,

responses to oracle queries are *silenced* when the correctness requirement renders a response *fixed*. A response is fixed when the answer depends only on the query history and the correctness constraint. Once silenced, an oracle will stay so.

Besides developing the idea above, this paper is also about an illustrative application of it. The problem we look at, significant in its own right, is how to find a clean and general treatment for stateful authenticated-encryption (sAE). A sender transmits a sequence of encrypted messages to a receiver. The communication channel might be reliable or not, and the parties might or might not maintain state (stateful AE should encompass conventional AE). If the decrypting party does maintain state, it might have a little or a lot. We seek a metaphorical "knob" with which one can specify precise expectations regarding replays, omissions, and out-of-order delivery. Our definition for sAE security does this. Given a set L specifying exactly which message reorderings are considered permissible, we define a matching correctness condition. From it and a pair of simple games, which do not depend on L, one inherits a security notion, courtesy of IND|C. By appropriately setting L we encompass old sAE notions and significant new ones, like sAE permitting reorderings up to a specified lag in message delivery.

Indistinguishability up to correctness. In somewhat greater detail, the methodology we suggest works as follows. To define a cryptographic goal one designs a pair of *utopian* games G and H that an adversary must try to distinguish. Game G surfaces the *real* behavior of some underlying protocol Π, while game H surfaces the *ideal* behavior one might wish for. We call the games *utopian* because there is some simple adversarial attack to distinguish them. For example, if we aim to treat public-key encryption (PKE) secure against chosen-ciphertext attack (CCA), then game G might let the adversary encrypt and decrypt with the underlying encryption scheme Π, while H properly answers decryption queries, but answers encryption queries by encrypting zero bits.

The cryptographer next pins down when a scheme is *correct*. Correctness is a validity requirement, not a security requirement. It captures what needs to happen in the *absence* of an adversary. In our PKE example, correctness for a scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ says that $(pk, sk) \twoheadleftarrow \mathcal{K}(k)$ and $c \twoheadleftarrow \mathcal{E}(pk, m)$ implies $\mathcal{D}(sk, c) = m$. Formally, saying that a scheme Π is correct just means that it belongs to some class C of correct schemes: for us, a correctness condition *is* a class of scheme.

We generalize conventional indistinguishability (IND) to the notion we call *indistinguishability up to correctness* (IND|C). The idea is this. Suppose that the adversary is interacting with a "real" game G that depends on some underlying cryptographic scheme Π. What it wants is to distinguish G from some "ideal" game H (which might also depend on Π). Suppose, at some point in the adversary's attack, it asks an oracle query x_i. It previously asked $x_1, \ldots, x_{i-1}$ and got answers $y_1, \ldots, y_{i-1}$. If given this query history $\boldsymbol{t}$ there is only one possible reply y across all correct schemes $\Pi \in \mathrm{C}$ and all internal coins r that G might use, then we say the oracle's response is *fixed*. The games we denote $\mathrm{G}[\psi]$ and $\mathrm{H}[\psi]$ behave like G and H except that asking a query that is fixed turns

off the oracle: it answers $\Diamond$ from that point on. The symbol ψ in the brackets following G and H denotes the *silencing function*, and we just described defining it by way of fixedness. Correctness-directed *oracle silencing* is the automatic adjustment of games (G, H) to modified games $(\mathrm{G}[\psi], \mathrm{H}[\psi])$. Using this method, we generalize the IND advantage $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G,H}}(A) = \Pr[A^{\mathrm{G}} \to 1] - \Pr[A^{\mathrm{H}} \to 1]$ to the INDC-advantage $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}(A) = \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G}[\psi],\mathrm{H}[\psi]}(A)$.

There is one more needed element: the adversary needs to *know* if an oracle query is going to be silenced—we need ψ to be efficiently computable. One must show that it is. If it's not, the intuition that the adversary shouldn't ask a question because it trivially knows the answer completely falls apart.

Application: PKE. As a first and simple application of IND|C, we revisit the standard IND-CCA security notion for PKE. We provide a utopian pair of games, G1 and H1, and a correctness class C1, thereby obtaining a security notion PKE.new defined by $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G1,H1,C1}}$. We show, unsurprisingly, that PKE.new is equivalent to PKE.old, the customary definition for IND-CCA secure PKE.

But wait: just what definition is it that we call *customary*? Bellare, Hofheinz, and Kiltz (BHK) describe four variants of IND-CCA secure PKE, which they denote with suffixes SE, SP, BE, and BP [1]. They explain that researchers haven't always been clear as to what version they intend. And they show that it *does* make a difference: while the SE and SP notions are equivalent, all other pairs are inequivalent. BHK suggest that the SE/SP notion is the *right* definitional variant [1, p. 34 & p. 39], implying that the other two notions are *wrong*. We agree. But how can one convincingly justify such a claim? The most convincing response, in our view, is to say that the SE/SP notion coincides with what one gets by invalidating all and only the adversarial wins that one *must* invalidate because of correctness. The BE and BP notions inappropriately invalidate additional wins. This is the response that our work formalizes. Similar reasoning can be used to justify definitional choices that might otherwise seem arbitrary.

Application: sAE. Our second application of IND|C is more involved: we consider the stateful-AE (sAE) problem, first formalized by Bellare, Kohno, and Namprempre (BKN) [2]. BKN adjust the customary definition of AE to make the decryption process stateful. Trying to model the kind of AE achieved by SSL, they want that ciphertext replays, reorderings, and omissions, as well as forgeries, will all be flagged as invalid. Formalizing this requires care.

Building on the above, Kohno, Palacio, and Black (KPB) describe five *types* of sAE [11], these ranging from a version that forgives all replays, omissions, and reorderings, to one that demands authentication to fail if any of these transgressions occur. Boyd, Hale, Mjølsnes, and Stebila (BHMS) [4] rework the KPB taxonomy, defining four *levels* of sAE. While the games they give are not terribly long, it is not easy to understand their technical constraints [4, Fig. 2]. And perhaps it was not easy for the authors, either, who made a technical adjustment in one of the four definitions about a year after their first publication [5, Recv line 4]. And if one wanted to consider some new sAE variant—and we

will explain soon why one might—one would need to start from scratch. The resulting definition might be hard to verify and easy to get wrong.

In our view, sAE is in a muddy state. The BKN, KPB, and BHMS papers use different syntax, making rigorous comparisons problematic. And they live in a sea of disparate and often complex related notions, including UC treatments of secure channels [6,7,12], the ACCE definition of Jager, Kohlar, Schäge, and Schwenk [10], and the notion for stream-based channels from Fischlin, Günther, Marson, and Paterson [9,8].

We go back to the basics for sAE, specifying a scheme's syntax and an extremely simple pair of games for the goal, G2 and H2, which the adversary *will* be able to easily distinguish them. We then "cancel" the trivial wins via IND|C. Given a set L that describes the required level of channel fidelity, we define a corresponding class of correct schemes C2(L). The above induces a security notion sAE[L] via IND|C. The flavors of sAE from BKN, KPB, and BHMS correspond to sAE[L] for specific choices of L. Many further choices are possible. In particular, the set we call L_1^ℓ bans forgeries and replays, but allows omissions and reordering up to some specified lag ℓ. The level we denote L_2^ℓ bans forgeries, replays, and reordering, but allows omissions of up to ℓ messages. The related levels from KPB and BHMS place no limits on ℓ (i.e., $\ell = \infty$). Achieving that aim would normally be impractical, as the decrypting party would need to maintain unlimited state, using it to record every nonce received.

Besides defining sAE[L] security, we show that the natural way to achieve it from nonce-based AE does in fact work. We discuss when this scheme is efficient, and describe efficiency and security improvements that are possible for some L.

Alternatives. The way we have chosen to define IND|C security is not the only way possible: there are a variety of natural variants. For each, one uses the correctness condition C to automatically edit utopian games G and H to new games G′ and H′. Oracle-editing generalizes oracle-silencing. We look at about half a dozen definitional variants, and evidence the robustness of IND|C by arguing that, under anticipated side conditions, all but one alternative is equivalent to our original formulation. For that final variant, meant to deal with left-or-right style games, we do not know how to prove or disprove equivalence.

2 Indistinguishability up to Correctness

Games. We recall the notion of games from Bellare and Rogaway [3], making some minor adjustments. See Fig. 1.

A game G is an always-halting algorithm given by code. It has entry points Initialize, Oracle, and Finalize. The code can obtain successive coin tosses from a uniformly random string $r \xleftarrow{\$} \{0,1\}^\infty$. One runs G with an adversary A, which can likewise see coins $\rho \xleftarrow{\$} \{0,1\}^\infty$. Both the adversary and game maintain persistent states. A game may depend on an *underlying scheme* Π: $\{0,1\}^* \to \{0,1\}^*$. We may write G_Π to emphasize G's dependence on Π. Normally this dependence is in the form of black-box access to a Π oracle. A game G may also call out to an arbitrary function ψ whose definition need not be in code.

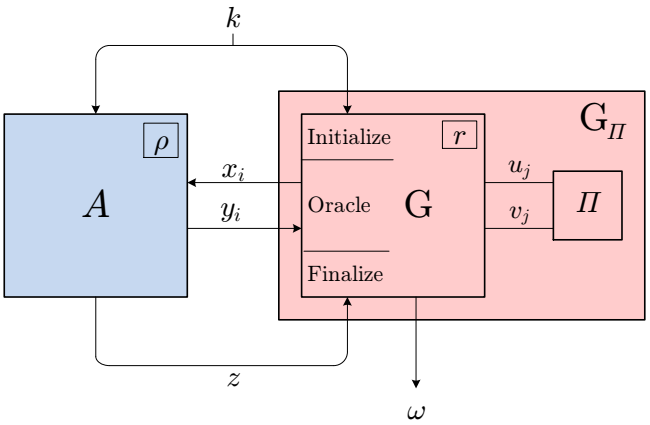


Fig. 1. An adversary interacting with a game. A game G may depend on a cryptographic scheme $\Pi\colon \{0,1\}^* \to \{0,1\}^*$. The game G and adversary A are both provided an initial value k. Adversarial and game randomness are provided by random strings ρ and r. Pairs x_i, y_i and u_j, v_j represent sequences of queries, indexed from 1. The adversary's output is z and the game's outcome is ω.

To execute G with A, the game's Initialize procedure is first run, passing it an initial value k. This is normally assumed to be a number, the security parameter, and presented in unary. Nothing is returned. Next, the adversary A is run, again invoking it on k. The adversary will make a sequence of Oracle calls (oracle queries) $x_1, \ldots, x_q$ obtaining corresponding responses $y_1, \ldots, y_q$. The number of queries q is up to the adversary. When the adversary has asked all the queries it wants to ask, it halts with an output z. The game's Finalize procedure is then called with z. It returns the game outcome ω. Specifying a game entails specifying Initialize, Oracle, and Finalize. If the first is omitted, there is only the default initialization of game variables: 0 for numbers, false for booleans, ε for strings, and the empty vector $\Lambda = ( )$ for vectors. If Finalize is omitted, it is the algorithm that outputs its input, making the game's outcome the adversary's output. The number $\Pr[A^{\mathrm{G}}(k) \to 1]$ is the probability that A outputs 1 after interacting with game G given the initial input k. The Finalize procedure is irrelevant. The number $\Pr[\mathrm{G}^A(k) \to 1]$ is the probability that G (it's Finalize procedure) outputs 1 after an interaction with A on k.

We can regard G_Π as a function, with $y_i = \mathrm{G}_\Pi(k, x_1, \ldots, x_i, r)$ the value returned by the oracle query when the initial value is k, the queries asked are $x_1, \ldots, x_i$, and the coins are r; while $\omega = \mathrm{G}_\Pi(k, x_1, \ldots, x_q, z, r)$ is similarly construed, employing encoding conventions such that the Finalize call is clear. If we omit r from the arguments then G_Π becomes a randomized function. We omit k whenever the Initialize procedure does not depend on it.

As an adversary A interacts with a game G, oracle calls and responses can be recorded in a *transcript*, which is a vector of strings. Query-terminated tran-

scripts $(x_1, y_1, x_2, y_2 \ldots, x_i)$ have an odd number of strings; response-terminated transcripts $(x_1, y_1, x_2, y_2 \ldots, x_i, y_i)$ have an even number of strings.

Discussion. There is no loss of generality in regarding the underlying cryptographic scheme Π as a function from strings to strings; suitable encoding conventions allow any scheme of interest to be so encoded. Similarly, games are routinely described as supporting different *types* of queries, like "Enc" and "Dec" queries. This is handled by regarding each query x as encoding a vector whose first component, $x[1]$, is a label drawn from a specified set.

An oracle query x might be intended only to adjust the game's internal state, not to elicit any response. Such queries are called *declarative*. All other queries are *investigative*. We do not adopt any special syntax to differentiate declarative and investigative queries, but the designer of a game is always free to adopt some convention to serve this purpose.

Correctness. What does it mean to say that a scheme Π is correct? The simplest answer is to say Π belongs to some class of schemes C, which are those deemed correct. That is what we will do; for us a *correctness class* is a set C of functions from strings to strings, and defining correctness means specifying C.

Graded notions of correctness, where a scheme is $(1-\varepsilon)$-correct if some bad event happens with probability at most ε, are outside the scope of our definitions.

Silencing. Given a correctness class C and a game G we define a predicate on response-terminated transcripts

$$\begin{aligned}\mathit{Valid}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j, y_j) =& (\exists \Pi \in \mathrm{C})(\exists k \in \{0,1\}^*)(\exists r \in \{0,1\}^\infty)(\forall i \in [1..j]) \\ & [\,\mathrm{G}_\Pi(k, x_1, x_2, \ldots, x_i, r) = y_i\,]\,.\end{aligned}$$

In English, a response-terminated transcript is valid if there exists a scheme in the specified class that could give rise to it. Since adversaries can ask anything they please, we say that a query-terminated transcript is valid when its longest proper prefix is: $\mathit{Valid}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j, y_j, x) = \mathit{Valid}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j, y_j)$.

Building on the notion of validity, we define a boolean function on query-terminated transcripts

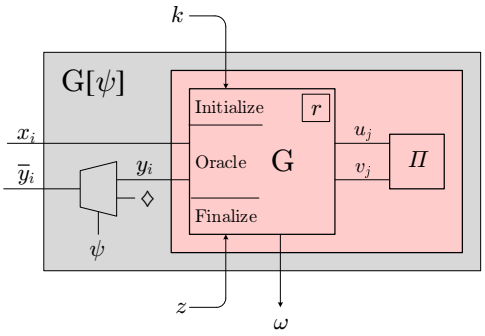
$$\mathit{Fixed}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j, y_j, x) = (\exists!\, y)\ \mathit{Valid}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j, y_j, x, y)\,.$$

Here $(\exists!\, y)P(y)$ means $(\exists y)P(y) \wedge (\forall y_1)(\forall y_2)((P(y_1) \wedge P(y_2)) \Rightarrow y_1 = y_2)$. In English, a query-terminated transcript is fixed if the last indicated query has exactly one valid response. Note that when the transcript $\boldsymbol{t}$ is invalid then $\mathit{Fixed}_{\mathrm{C,G}}(\boldsymbol{t})$ is false, since $(\exists!\, y)P(y) \Rightarrow (\exists y)P(y)$.

Finally, given a correctness class C and game G, we define our preferred *silencing function* for this pair by

$$\mathit{Silence}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j) = \bigvee_{1 \le i \le j} \mathit{Fixed}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_i)\,.$$

That is, we silence an oracle response that terminates a transcript $\boldsymbol{t}$ if that response is now fixed, or was previously. We call this *silence-then-shut-down*.



procedure G[ψ].Initialize(k)
$i \leftarrow 0$; G.Initialize(k)
return

procedure G[ψ].Oracle(x)
$i \leftarrow i + 1$; $x_i \leftarrow x$
$y_i \leftarrow$ G.Oracle(x)
if $\psi(x_1, y_1, x_2, y_2, \ldots, x_i)$ **then**
  return $\Diamond$
return y_i

Fig. 2. Oracle silencing. Left: Given a game G and a function ψ: $\{0,1\}^{**} \to \{0,1\}$ we define the silenced game G[ψ] by silencing the oracle once the boolean value $\psi(x_1, y_1, \ldots, x_i)$ becomes true. **Right:** The formal definition for the game G[ψ]. The game's Finalize procedure is irrelevant.

IND|C SECURITY. Given a game G and a boolean function ψ, which we call a *silencing function*, we define the *silenced game* G[ψ] in Fig. 2. In that game, oracle responses are adjusted according to ψ: when ψ applied to the y_i-terminated transcript is true, we return $\Diamond$ instead of y_i.

Now given games G and H and a silencing function ψ, let $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H},\psi}(A,k) = \Pr[A^{\mathrm{G}[\psi]}(k) \to 1] - \Pr[A^{\mathrm{H}[\psi]}(k) \to 1]$.

Finally, given games G and H and a correctness class C, let $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}(A,k) = \Pr[A^{\mathrm{G}[\psi]}(k) \to 1] - \Pr[A^{\mathrm{H}[\psi]}(k) \to 1]$ where $\psi = \mathit{Silence}_{\mathrm{C,G}}$. We call this notion INDC security, or, perhaps more pretty, IND|C security. (The vertical bar is meant to suggest conditioning.) Note that the silencing that is applied to the ideal game H is determined by the real game G.

For an asymptotic notion of INDC security, we assert that games G and H are *indistinguishable up to* C if $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}(A,k)$ is negligible for any probabilistic polynomial-time (PPT) adversary A. As usual, $\varepsilon(k)$ is negligible if for any polynomial p there exists a number N such that $\varepsilon(k) < 1/p(k)$ for all $k \geq N$.

Remember that games $\mathrm{G} = \mathrm{G}_{\Pi}$ and $\mathrm{H} = \mathrm{H}_{\Pi}$ may depend on some underlying scheme Π. A cryptographer who specifies G, H and C has specified a security measure on protocols $\Pi \in \mathrm{C}$ defined by $\mathbf{Adv}_{\Pi}(A,k) = \mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G}_{\Pi},\mathrm{H}_{\Pi},\mathrm{C}}(A,k)$.

COMPUTABILITY OF FIXEDNESS. There is no *a priori* reason to believe that $\mathit{Fixed}_{\mathrm{C,G}}$ or $\mathit{Silence}_{\mathrm{C,G}}$ will be computable, let alone efficiently. Yet for IND|C security to be meaningful, we need $\mathit{Fixed}_{\mathrm{C,G}}$ to be efficiently computable: if the adversary doesn't *know* that the response to its query is determined by the correctness constraint, then the query is *not* trivial, and making it should *not* be disqualifying. The most straightforward way of capturing the stated expectation is to demand that $\mathit{Fixed}_{\mathrm{C,G}}$ be polynomial-time (PT) computable (if one is in the asymptotic setting). This is overkill, however, insofar as the only transcripts $\boldsymbol{t}$

to which $\mathit{Fixed}_{\mathrm{C,G}}$ will ever be applied are those that are *legitimate*—those that can arise in an interaction between A and G or between A and H.

Based on this, we say that *fixedness is efficiently computable for* $(\mathrm{C}, \mathrm{G}, \mathrm{H})$ if there exists a PT-computable function ϕ such that $\phi(\boldsymbol{t}) = \mathit{Fixed}_{\mathrm{C,G}}(\boldsymbol{t})$ for all query-terminated transcripts $\boldsymbol{t}$ satisfying $\mathit{Valid}_{\mathrm{C,G}}(\boldsymbol{t}) \vee \mathit{Valid}_{\mathrm{C,H}}(\boldsymbol{t})$. Taking this a step further, we say that *fixedness is efficiently computable for* $(\mathrm{C}, \mathrm{G}, \mathrm{H}, q)$ if there exists a PT-computable function ϕ such that $\phi(\boldsymbol{t}) = \mathit{Fixed}_{\mathrm{C,G}}(\boldsymbol{t})$ for all query-terminated transcripts $\boldsymbol{t}$ satisfying $\mathit{Valid}_{\mathrm{C,G}}(\boldsymbol{t}) \vee \mathit{Valid}_{\mathrm{C,H}}(\boldsymbol{t})$ and $|\boldsymbol{t}| < 2q$. The last part says that $\boldsymbol{t}$ involves at most q queries (where $|\boldsymbol{t}|$ is the number of components in $\boldsymbol{t}$). For positive results, we must verify that fixedness is efficiently computable for $(\mathrm{C}, \mathrm{G}, \mathrm{H})$, or for $(\mathrm{C}, \mathrm{G}, \mathrm{H}, q)$ with $q(k)$ adequately large.

Further relaxations for efficient computability of fixedness are possible. Since it is safe to silence too little, it is enough to find an efficiently computable function ϕ satisfying $\phi(\boldsymbol{t}) \Rightarrow \mathit{Silence}_{\mathrm{C,G}}(\boldsymbol{t})$ when $\mathit{Valid}_{\mathrm{C,G}}(\boldsymbol{t}) \vee \mathit{Valid}_{\mathrm{C,H}}(\boldsymbol{t})$. Our examples won't need this relaxation.

Discussion. We have spoken about the efficient computability of *Fixed*, but we could as well have spoken of the efficient computability of *Silence*. The former is the more basic object, and simpler to think about. In fact, we not only anticipate that the boolean *Fixed* should be efficiently computable, but also the string-valued function $\mathit{fixed}_{\mathrm{G,C}}$ that specifies the real-oracle's response when it is in fact fixed (or indicates, alternatively, that it is not). See Section 5.

The silencing function ψ used in defining IND|C was not *Fixed* but the logical-or of it applied to all transcript prefixes. Once an oracle is silenced, it stays silenced. An alternative approach, *silence-then-forgive*, is essentially equivalent; see Section 5. It is to simplify the description of silence-then-forgive that, in Fig. 2, when a response y_i is silenced, we let the growing "transcript" retain the original (unsilenced) value. This choice is irrelevant for silence-then-shut-down.

As already explained, if fixedness is not efficiently computable the intuition underlying oracle silencing breaks down, and IND|C becomes meaningless. It could even happen that silenced games are harder to distinguish than the utopian ones. For example, given a one-way permutation F with hardcore bit B, game G is constructed to select random values x_0 and x_1 and, on a first oracle query, provide $F(x_0)$ and $F(x_1)$. A second oracle query selects $b \xleftarrow{\$} \{0,1\}$ and returns $B(x_b)$. Now whether or not this query is silenced provides information that the adversary cannot compute. The idea can be elaborated to create indistinguishable games whose silenced versions are distinguishable.

The usual notion of indistinguishability, $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G,H}}(A,k) = \Pr[A^{\mathrm{G}}(k) \to 1] - \Pr[A^{\mathrm{H}}(k) \to 1]$, coincides with $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H},\psi}(A,k)$ when $\psi(\boldsymbol{t}) = \mathsf{false}$. Of course IND-security is symmetric: $\mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G,H}}(A,k) = \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{H,G}}(A,k)$. This is not true of INDC: it may be that $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}(A,k) \neq \mathbf{Adv}^{\mathrm{indc}}_{\mathrm{H,G,C}}(A,k)$. The asymmetry stems from the fact that we silence based on the *real* game, listed first in the subscripts.

Oracle silencing provides an alternative to penalty-style and exclusion-style definitions [1]. We wrap up our discussion by observing that IND|C security could have been defined using those alternatives, too.

procedure $\mathrm{G}[\![\psi]\!]$.Initialize(k)
$q \leftarrow 0$; G.Initialize(k)
return

procedure $\mathrm{G}[\![\psi]\!]$.Oracle(x)
$q \leftarrow q + 1$; $x_q \leftarrow x$
return $y_q \leftarrow$ G.Oracle(x)

procedure $\mathrm{G}[\![\psi]\!]$.Finalize(z)
if $\psi(x_1, y_1, x_2, y_2, \ldots, x_q)$ **then return** 0
return z

Fig. 3. Penalty-style oracle editing. Oracle queries are answered as usual, but if the final transcript triggers ψ, the game's outcome is set to zero.

Penalty-style alternative. Instead of turning off an adversary's oracle when it asks an offending question, we could answer the query as usual but, at the end of the game, declare it forfeit. This is what Bellare, Hofheinz, and Kiltz call a *penalty-style* definition [1]. We formalize what is needed in Fig. 3, mapping a game G and a function ψ to a corresponding game $\mathrm{G}[\![\psi]\!]$. An alternative version of indistinguishability up to correctness, INDC0, is then defined by saying that $\mathbf{Adv}^{\mathrm{indc0}}_{\mathrm{G,H,C}}(A,k) = \Pr[(\mathrm{G}[\![\psi]\!])^A(k) \to 1] - \Pr[(\mathrm{H}[\![\psi]\!])^A(k) \to 1]$ where $\psi = \mathit{Silence}_{\mathrm{C,G}}$. In effect, the adversary's output z has been replaced by $z \wedge \bigwedge_j \neg \mathit{Fixed}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_j)$. For an asymptotic notion of INDC0 security, we say that games G and H are penalty-style indistinguishable up to C if for any PPT adversary A, the function $\mathbf{Adv}^{\mathrm{indc0}}_{\mathrm{G,H,C}}(A,k)$ is negligible.

What is the relationship between oracle-silencing IND|C and penalty-style INDC0? Assuming fixedness is efficiently computable, the two ways of adjusting games are equivalent. For concision, we give an asymptotic version of the result. The proof, which is easy, is in Appendix A.1.

Theorem 1. *Let* G *and* H *be games and let* C *be a correctness class. Assume fixedness is efficiently computable for* $(\mathrm{G},\mathrm{H},\mathrm{C})$. *Then* G *and* H *are indistinguishable up to* C *iff they are penalty-style indistinguishable up to* C.

The above might be interpreted as saying that oracle silencing is new language for something that doesn't need it. That misses the point, that oracle-silencing grounds the natural explanation how and why one edits the utopian games.

Exclusion-style alternative. And what of exclusion-style definitions [1], where one limits consideration to adversaries that are "well-behaved"? It is possible, although awkward, to describe IND|C in this way. After defining games G_{Π} and H_{Π} and the correctness class C, we restrict attention from all adversaries $\mathcal{U}$ to the subset $\mathcal{A}$ that, when interacting with G_{Π} or H_{Π}, never create a transcript $\boldsymbol{t}$ such that $\mathit{Fixed}_{\mathrm{G,C}}(\boldsymbol{t})$ is true. One attends only to adversaries in $\mathcal{A}$.

The above description might sound problematic because there is no way to inspect an adversary's description and know if it's in $\mathcal{A}$. It doesn't matter. As long as fixedness is efficiently computable for $(\mathrm{G},\mathrm{H},\mathrm{C})$, one can take an adversary

$A \in \mathcal{U}$ and put a "wrapper" around it so that it conforms with $\mathcal{A}$. The wrapped adversary behaves like A unless it is about to ask a query that would make $\mathit{Fixed}_{\mathrm{G,H,C}}(\boldsymbol{t})$ true, in which case it outputs 0 and halts. In this way one names a class of adversaries $\mathcal{A}$ such that the ind-advantage among adversaries in it coincides with the indc-advantage over adversaries in $\mathcal{U}$. So security notions that can be described by oracle silencing can be described exclusion-style. Not that doing so is wise. Exclusion-style definitions compel consideration of adversary classes. They disqualify adversaries that only rarely misbehave. They ignore whether or not an adversary can "know" it has misbehaved. And they promote ambiguity, as the relevant restrictions are not expressed in game code.

Further variants. Beyond penalty-style and exclusion-style formulations of IND|C, more alternatives are possible. See Section 5 for some interesting ones.

3 Public-Key Encryption

Let us consider the well-known IND-CCA security notion for a public-key encryption (PKE) scheme. We first review the syntax. A PKE scheme Π is a tuple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where probabilistic algorithm $\mathcal{K}$ takes in a security parameter k, encoded in unary, and generates a public key pk and a secret key sk; probabilistic algorithm $\mathcal{E}$ takes in a public key pk and a plaintext m, and returns a ciphertext c; and deterministic decryption algorithm $\mathcal{D}$ takes in a secret key sk and a ciphertext c, and returns a message m. For simplicity, we assume a message space of $\{0,1\}^*$. An appropriate encoding of the component algorithms is implicitly assumed whenever we regard Π as a map $\Pi\colon \{0,1\}^* \to \{0,1\}^*$.

To apply our techniques, the first step is to specify the class of correct PKE schemes. This is easily done, letting

$$\mathrm{C1} = \{\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}) \mid (\forall k)(\forall m)\ [(pk, sk) \twoheadleftarrow \mathcal{K}(k); c \twoheadleftarrow \mathcal{E}(pk, m)\colon \mathcal{D}(sk, c) = m]\}$$

denote the schemes we consider *correct*. The condition is absolute: decryption of $c \twoheadleftarrow \mathcal{E}(k, m)$ must *always* return m, which is the customary requirement.

The second step is to write down the utopian real and ideal games. For this, we ask the adversary to distinguish between a game that encrypts a message m of the adversary's choice and a game that encrypts an equal length string of zero-bits. For both games, the adversary can request the public key and has access to a proper decryption oracle. See Fig. 4. Those games only allow the adversary a single Enc query. This restriction is unnecessary, but including it reduces the gap between our new notion and the traditional one for IND-CCA that we use.

The games are indeed utopian: if the adversary queries Enc(1), getting back c, then queries Dec(c), getting back m, it will earn advantage 1 by returning m. Naturally this is where oracle silencing comes into play: if Dec is queried with the response c returned by a previous Enc query then $\mathit{Fixed}_{\mathrm{C1,G1}}$ will almost always be true, resulting in the query being silenced. Why do we say *almost* always, and not always? The answer is closely related to how one can efficiently compute $\mathit{Fixed}_{\mathrm{C1,G1}}$.

procedure Initialize (k) Game G1	**procedure** Initialize (k) Game H1
111 $(pk, sk) \twoheadleftarrow \mathcal{K}(k)$	121 $(pk, sk) \twoheadleftarrow \mathcal{K}(k)$
procedure Key	**procedure** Key
112 **return** pk	122 **return** pk
procedure Enc (m)	**procedure** Enc (m)
113 **if** asked **return** $\perp$	123 **if** asked **return** $\perp$
114 asked $\leftarrow$ true	124 asked $\leftarrow$ true
115 **return** $\mathcal{E}(pk, m)$	125 **return** $\mathcal{E}(pk, 0^{\lvert m\rvert})$
procedure Dec (c)	**procedure** Dec (c)
116 **return** $\mathcal{D}(sk, c)$	126 **return** $\mathcal{D}(sk, c)$

Fig. 4. Utopian games used to define PKE.new. The games are easily distinguished in the ind-sense. The problem is fixed by switching to indc-advantage.

Computing fixedness. As just indicated, even if a transcript $\boldsymbol{t}$ has a Dec(c) follow an Enc(m) that returns c, it is not *always* the case that $\mathit{Fixed}_{\mathrm{C1,G1}}(\boldsymbol{t}) = \mathsf{true}$. At issue is the fact that there are some peculiar transcripts that can arise in the ideal setting but would never arise in the real setting. Recall that our formalization demands that we do *not* silence a query ending a transcript $\boldsymbol{t}$ that could never arise in the "real" setting. One such counterexample is a Dec(c) query that returns m, followed by an Enc(m') query that returns c, where $m \neq m'$. This can't happen in the "real" game, since it would violate correctness. Since we only silence valid transcripts, once such an invalid event takes place, in a run with H, we never silence any further queries—even for a Dec(c) following some Enc(m) query that returns c.

The code of Fig. 5 attends to such subtleties. There we write out a formula for a candidate function ϕ that efficiently computes fixedness for $(\mathrm{C1}, \mathrm{G1}, \mathrm{H1})$. Function ϕ makes sure the mentioned counterexample does not occur (first line), and it also checks for the "usual" concern: a decryption query that asks to decrypt the challenge ciphertext (second line). But there are still some additional, naïve queries to deal with (the last three lines). These are: a Key query subsequent to the first such query; an Enc query subsequent to the first such query; and a repeating Dec(c) query, for some value c. The responses to any of those queries will be silenced. Our result on the computability of fixedness is as follows.

Theorem 2. *There is a PT algorithm that computes fixedness for* $(\mathrm{C1}, \mathrm{G1}, \mathrm{H1})$. *In fact, the algorithm of Fig. 5 computes it.*

For a proof, see Appendix A.2.

To define the security of a PKE scheme against IND-CCA attack, we let $\mathbf{Adv}^{\mathrm{pke.new}}_{\Pi}(A, k) = \mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G1}[\Pi],\mathrm{H1}[\Pi],\mathrm{C1}}(A, k)$ for the games and correctness class described. We say that a PKE scheme Π is PKE.new-secure if $\mathbf{Adv}^{\mathrm{pke.new}}_{\Pi}(A, k)$

procedure $\phi(x_1, y_1, \ldots, x_t)$
201 **return** $\big((\nexists\, i, j)\ x_i = (\text{Dec}, y_j) \wedge x_j[1] = \text{Enc} \wedge x_j[2] \neq y_i\big) \bigwedge$
202 $\Big((\exists\, j)\ \big(x_j[1] = \text{Enc} \wedge x_t = (\text{Dec}, y_j)\big) \vee$
203 $(\exists\, j)\ \big(x_j[1] = \text{Key} \wedge x_t[1] = \text{Key}\big) \vee$
204 $(\exists\, j)\ \big(x_j[1] = \text{Enc} \wedge x_t[1] = \text{Enc}\big) \vee$
205 $(\exists\, j, c)\ \big(x_j = x_t = (\text{Dec}, c)\big)\Big)$

Fig. 5. Formula for computing fixedness for PKE.new. Line 201 is the validity check, while line 202–205 are the fixedness checks.

Game G0	Game H0
procedure Initialize (k)	**procedure** Initialize (k)
311 $(pk, sk) \twoheadleftarrow \mathcal{K}(k)$	321 $(pk, sk) \twoheadleftarrow \mathcal{K}(k)$
procedure Key	**procedure** Key
312 **return** pk	322 **return** pk
procedure Test (m_1, m_2)	**procedure** Test (m_1, m_2)
313 **if** tested **return** $\perp$	323 **if** tested **return** $\perp$
314 tested $\leftarrow$ true	324 tested $\leftarrow$ true
315 $c^* \twoheadleftarrow \mathcal{E}(pk, m_1)$	325 $c^* \twoheadleftarrow \mathcal{E}(pk, m_2)$
316 **return** c^*	326 **return** c^*
procedure Dec (c)	**procedure** Dec (c)
317 **if** $c = c^*$ **then return** $\perp$	327 **if** $c = c^*$ **then return** $\perp$
318 **return** $\mathcal{D}(sk, c)$	328 **return** $\mathcal{D}(sk, c)$

Fig. 6. The PKE.old notion for IND-CCA secure public-key encryption. The formulation is equivalent to the SE and SP notions from BHK [1].

is negligible for all PPT adversaries A. We have already shown that fixedness is efficiently computable for (C1, G1, H1).

How does our PKE.new notion compare with "standard" IND-CCA security for a public-key encryption scheme? By the latter we mean the (equivalent) IND-CCA-SE and IND-CCA-SP notions of BHK [1]. We define it using the G0 and H0 games of Fig. 6. Let $\mathbf{Adv}^{\text{pke.old}}_{\Pi}(A, k) = \mathbf{Adv}^{\text{ind}}_{\text{G0,H0}}(A, k)$ and define Π as PKE.old-secure if $\mathbf{Adv}^{\text{pke.old}}_{\Pi}(A, k)$ is negligible for any PPT A.

The new and old PKE security notions are equivalent. Equivalence isn't quite obvious, because the silencing criteria not only includes adversaries querying a Dec on the challenge ciphertext—the sole criterion for PKE.old—but, also, adversaries not having triggered any "invalid" events. Less significantly, we're also looking at a real-vs-ideal game, rather than a left-or-right style one. Still, one can show that the notions are equivalent.

Theorem 3. *A PKE scheme is PKE.new-secure iff it is PKE.old-secure.*

The proof is in Appendix A.3.

Theorem 3 supports the idea that the (equivalent) SE and SP notions of BHK are *right*, while the other two notions are not [1]. One of the uses of IND|C security is to justify or call into question an existing definition by, in effect, looking at what the correctness condition itself has to say.

The structure of the proof of Theorem 3 can be generalized. We observe that *Fixed* can always be decomposed into a *validity check* and a *fixedness check*:

$$\begin{aligned} \mathit{Fixed}(x_1, y_1, \ldots, x_q) = &\mathit{Valid}(x_1, y_1, \ldots, x_{q-1}, y_{q-1}) && \text{(validity)} \\ &\wedge \big((\forall y_q, y'_q)\ \mathit{Valid}(x_1, y_1, \ldots, x_q, y_q) \wedge \\ &\quad \mathit{Valid}(x_1, y_1, \ldots, x_q, y'_q) \Rightarrow y_q = y'_q\big) && \text{(fixedness)} \end{aligned}$$

A recapitulation of the proof with the decomposition above allows us to draw the following conclusion: as long as both validity and fixedness checks are efficiently computable, the removal of validity checks will give us an equivalent indistinguishability notion. Related discussions can be found in section 5.

4 Stateful AE

Syntax. A scheme for stateful AE (sAE) is a tuple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where key-generation algorithm $\mathcal{K}$ is a probabilistic algorithm that returns a string, while encryption algorithm $\mathcal{E}\colon \mathcal{K} \times \mathcal{A} \times \mathcal{M} \times \mathcal{S} \to (\mathcal{C} \cup \{\perp\}) \times \mathcal{S}$ and decryption algorithm $\mathcal{D}\colon \mathcal{K} \times \mathcal{A} \times \mathcal{C} \times \mathcal{S} \to (\mathcal{M} \cup \{\perp\}) \times \mathcal{S}$ are deterministic. We call $\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$, $\mathcal{A}$, and $\mathcal{S}$ the key space, message space, ciphertext space, associated-data (AD) space, and state space, respectively. We assume that $\mathcal{K}$ contains the support of $\mathcal{K}$, and that there's a constant τ, the *ciphertext expansion*, such that $(c, s') = \mathcal{E}(k, a, m, s)$ and $c \neq \perp$ implies $|c| = |m| + \tau$. For simplicity, we regard the ciphertext expansion of sAE schemes as a fixed and universal constant (e.g., $\tau = 128$), referring to τ without tying it to any specific scheme.

Level sets. Suppose a party encrypts messages $1, 2, \ldots, 100$, sending them, encrypted and in order, to some receiver. Due to an active adversary or an unreliable transport, that receiver might recover the sequence of messages $(1, 3, 2)$, or maybe $(1, 10)$, or perhaps $(1, 2, 2, 3)$. In each case, should an authentication error be generated? The answer depends on multiple factors: the anticipated properties of the communication channel; your willingness to have the decrypting party maintain state; how much state you think that party should maintain; and the damage you anticipate from omissions, insertions, and reorderings.

How might one specify the targeted level of channel fidelity? It can be done by giving a *level-set*, a set $L \subseteq \mathbb{N}^*$ (where $\mathbb{N} = \{1, 2, 3, \ldots\}$ excludes 0). An element $\mathbf{n} \in L$ is called a *permissible ordering*. The intended semantics of $\mathbf{n} = (n_1, \ldots, n_\beta)$ being in L is that *if* the sender transmits a sequence of messages $1, 2, \ldots$, and the receiver recovers, in order, messages $n_1, \ldots, n_\beta$, then this *is* an acceptable degree of fidelity if and only if $\mathbf{n} \in L$. To make sense, we require of any level-set L that $\mathbf{n} \in L$ implies $\mathbf{n}' \in L$ for any prefix $\mathbf{n}'$ of $\mathbf{n}$.

Examples of significant level-sets are given in Fig. 7. We call the level-sets named there the *basic* level-sets. Due to the superscript ℓ, there are infinitely

Level	Definition and description
L_0	$\mathbb{N}^*$. This level-set deems all orderings permissible, regardless of omissions, replays, or reorderings. A receiver for this level-set can be stateless. This is the level-set that corresponds to conventional (stateless) AE.
L_1^ℓ	$\{\mathbf{n} \in \mathbb{N}^* \colon i \neq j \Rightarrow n_i \neq n_j \text{ and } \lvert n_j - \max_{0 \leq i < j} n_i \rvert \leq \ell + 1 \text{ for all } 1 \leq j \leq \lvert n \rvert\}$. Here we do not permit replays, but do allow omissions and reorderings up to the specified lag. When $\ell = \infty$ there is no limit on the lag and the notion roughly corresponds to level-2 in Kohno et al. [11] and Boyd et al. [4].
L_2^ℓ	$\{\mathbf{n} \in \mathbb{N}^* \colon 1 \leq n_i - n_{i-1} \leq \ell + 1 \text{ for all } 1 \leq i \leq \lvert \mathbf{n} \rvert\}$. This level-set does not permit replays or reorderings, but allows omissions up to ℓ lost packets. When $\ell = \infty$ there is no limit on permissible gaps and the notion roughly corresponds to level-3 in Kohno et al. [11] and Boyd et al. [4]
L_3	$\{\mathbf{n} \in \mathbb{N}^* \colon n_i = i \text{ for all } 1 \leq i \leq \lvert \mathbf{n} \rvert\}$. This is the strictest level-set: the only permissible receipt order is sending order. This matches the notion for sAE put forward by Bellare et al. [2], level-5 in Kohno et al. [11], and level-4 in Boyd et al. [4]. It is what one expects to achieve over a reliable transport.

Fig. 7. Basic level-sets for sAE. The value $\ell \geq 0$, the maximal *lag*, is a number or the value ∞. The named sets impose increasingly stringent requirements for rejecting replays, omissions, and out-of-order delivery. Throughout, $\mathbf{n} = (n_1, \ldots, n_\beta)$ and $n_0 = 0$.

many basic level-sets. The goals associated to levels $L_0, L_1^\infty, L_2^\infty$ and $L_3 = L_1^0 = L_2^0$ are described in prior work [4,11], while the L_1^ℓ and L_2^ℓ goals, for $\ell \in \mathbb{N}$, have not been formalized, although they would seem to be targeted by secure messaging apps like Signal [13].

To apply oracle silencing we need to specify a class of correct sAE schemes. That class will depend on the level-set L. Intuitively, a correct sAE scheme for level-L should satisfy the following condition. Suppose you encrypt a sequence of plaintexts to create ciphertexts we number $1, 2, 3, \ldots$, and then you decrypt, in order, the ciphertexts numbered $n_1, n_2, \ldots, n_\beta$. If $(n_1, \ldots, n_\beta) \in L$ then you must get back the correct sequence of plaintexts. Correctness places no demands on what happens for sequences outside of L. Nor does it levy demands once $\mathcal{E}$ declines to encrypt a string. The correctness class $\mathrm{C2}(L)$ associated to level-set L is formalized at the top of Fig. 8.

Utopian setting. We specify the utopian games for sAE in the bottom of Fig. 8, which defines games G2 and H2. The only thing peculiar in the code is the boolean flag σ provided to Dec queries. When set, only a random bit is returned by the game. This is a way for the adversary to mark a declarative query (p. 6), meaning an oracle call in which the adversary is not seeking information, but only trying to side-effect the game's internal state. Returning a random bit is just an idiom to exempt a declarative query from getting silenced (as *Fixed* will never be true). Without supporting such an ability, our adversary would effectively be unable to ask a decryption query that it knows the answer to, even if asking such a query would help set the oracle to a state in which the adversary could subsequently cause damage. We call σ the *declarative flag*.

C2(L) is the set of all sAE schemes $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that satisfy:

$\big(\forall k \in \mathcal{K}\big)\big(\forall (a_1, m_1), (a_2, m_2), \ldots \in \mathcal{A} \times \mathcal{M}\big)\big(\forall (n_1, \ldots, n_\beta) \in L\big)$
$\big[\ s_0 \leftarrow \varepsilon;\ r_0 \leftarrow \varepsilon;\ \alpha \leftarrow \max(n_1, \ldots n_\beta);$
for $i \leftarrow 1$ **to** α **do** $(c_i, s_i) \leftarrow \mathcal{E}(k, a_i, m_i, s_{i-1});$
for $i \leftarrow 1$ **to** β **do** $(m'_i, r_i) \leftarrow \mathcal{D}(k, a_{n_i}, c_{n_i}, r_{i-1})$:
$((\forall i \in [1..\alpha])\ (c_i \neq \perp)) \Rightarrow ((\forall i \in [1..\beta])\ (m'_i = m_{n_i}))\big]$

procedure Initialize — Game G2	**procedure** Initialize — Game H2
511 $k \twoheadleftarrow \mathcal{K}$	521 $k \twoheadleftarrow \mathcal{K}$
procedure Enc(a, m)	**procedure** Enc(a, m)
512 $(c, s) \twoheadleftarrow \mathcal{E}(k, a, m, s)$	522 $(c, s) \twoheadleftarrow \mathcal{E}(k, a, 0^{\lvert m\rvert}, s)$
513 **return** c	523 **return** c
procedure Dec(a, c, σ)	**procedure** Dec(a, c, σ)
514 $(m, r) \leftarrow \mathcal{D}(k, a, c, r)$	524 **if** σ **then return** $b \twoheadleftarrow \{0, 1\}$
515 **if** σ **then return** $b \twoheadleftarrow \{0, 1\}$	525 **return** $\perp$
516 **return** m	

Fig. 8. Top: Correctness classes for sAE. The function maps a level-set L to a correctness class C2(L). **Bottom: The utopian real and ideal games for sAE.** The games depend on an underlying sAE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$.

Given an sAE protocol Π and a level-set L, we define $\mathbf{Adv}^{\mathrm{sae}[L]}_{\Pi}(A)$ as $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G2}[\Pi],\mathrm{H2}[\Pi],\mathrm{C2}(L)}(A)$. Informally, scheme Π is sAE[L]-secure if $\mathbf{Adv}^{\mathrm{sae}[L]}_{\Pi}(A)$ is small for any reasonable adversary A. Following prevailing traditions in symmetric cryptography, our notion is concrete, not asymptotic, although one could always provide $\mathcal{K}$ with a security parameter and support an asymptotic notion.

Computing fixedness. It would be nice to give an efficiently computable formula for $\mathit{Fixed}_{\mathrm{C2}(L),\mathrm{G2}}$, hereinafter abbreviated as Fixed_L, for an arbitrary level-set L. But this is not possible—there is no such algorithm—because, in our treatment, level-sets can be arbitrarily bizarre. So we content ourselves with showing efficient computability of fixedness for the basic level-sets. We believe that any "natural" level-set L will have the same property, but stating a sufficient condition on L seems to get rather technical.

Theorem 4. *For any basic level-set L the fixedness function is efficiently computable for* $(\mathrm{C2}(L), \mathrm{G2}, \mathrm{H2}, 2^\tau - 3)$.

See Appendix A.4 for the proof.

N2S construction. We now give a simple construction for making an sAE scheme out of a classical nonce-based AE scheme (an nAE scheme) [14]. First we review the syntax and security notions for nonce-based AE.

An nAE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of a probabilistic key-generation algorithm that draws a key from the key space $\mathcal{K}$; a deterministic encryption algorithm $\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C}$ that takes in a key $k \in \mathcal{K}$, a nonce $n \in \mathcal{N}$, an AD $a \in \mathcal{A}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$; and a deterministic decryption algorithm $\mathcal{D}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ that takes in a key $k \in \mathcal{K}$, a nonce $n \in \mathcal{N}$, an AD $a \in \mathcal{A}$ and a ciphertext $c \in \mathcal{C}$ and either outputs a decrypted message $m \in \mathcal{M}$ or a failure symbol $\perp$. Correctness is defined in the natural way: for all $(k, n, a, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$ and $c \leftarrow \mathcal{E}(k, n, a, m)$ it holds that $\mathcal{D}(k, n, a, c) = m$. We also assume that for all $(k, n, a, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$, the *expansion* $\tau = |\mathcal{E}(k, n, a, m)| - |m|$ is a constant.

For the nAE security definition, let $\$(\cdot, \cdot, \cdot)$ be an oracle that takes in $n \in \mathcal{N}$ and $a \in \mathcal{A}$ and $m \in \mathcal{M}$ and returns a fresh random string of $|m| + \tau$ bits; and let $\perp(\cdot, \cdot, \cdot)$ be an oracle that takes in $n \in \mathcal{N}$ and $a \in \mathcal{A}$ and $c \in \mathcal{C}$ and always returns $\perp$. The advantage of an adversary A against an nAE scheme Π is then defined as

$$\mathbf{Adv}^{\mathrm{nae}}_{\Pi}(A) = \Pr\left[k \in \mathcal{K}\colon A^{\mathcal{E}(k,\cdot,\cdot,\cdot),\mathcal{D}(k,\cdot,\cdot,\cdot)} \to 1\right] - \Pr\left[A^{\$(\cdot,\cdot,\cdot),\perp(\cdot,\cdot,\cdot)} \to 1\right].$$

We require that A never asks (n, a, c) of its right oracle if some previous left oracle query (n, a, m) returned c; and that A does not repeat nonces when asking its left oracle. (The first condition could itself be recovered via IND|C.) Informally, an nAE scheme Π is secure if for all such adversaries with reasonable resources, the advantage $\mathbf{Adv}^{\mathsf{nae}}_{\Pi}(A)$ is small.

Construction N2S turns an nAE scheme Π with key space $\mathcal{K} \subseteq \{0,1\}^*$, nonce space $\mathcal{N} = \{0,1\}^\eta$, AD space $\mathcal{A} \subseteq \{0,1\}^*$ and message space $\mathcal{M} \subseteq \{0,1\}^*$ and ciphertext expansion τ into an sAE scheme $\overline{\Pi} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ with the same key space, AD space, and message space. Given an nAE scheme Π and a level-set L, the sAE scheme $\overline{\Pi} = \mathrm{N2S}(\Pi, L)$ is defined and illustrated in Fig. 9.

The construction is quite simple. For encryption, the state is maintained as a counter n that gets incremented with each message sent. When n is used as a string, it is encoded into η bits. The ciphertext is formed by concatenating n and the ciphertext returned by the nAE scheme. For decryption, the state is the vector $\mathbf{n}$ of nonces received so far. The decryption algorithm outputs failure if either the underlying nAE scheme says so *or* the received nonce, when appended to the list of prior ones, does not comprise a permissible ordering in L. We have the following result for the security of N2S:

Theorem 5. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an nAE scheme with nonce length η and ciphertext expansion τ. Let L be a level-set and let A an adversary that asks $q \le \min\{2^\eta, 2^\tau - 3\}$ queries. Then there exists an adversary B, generically described in the proof of this theorem, such that*

$$\mathbf{Adv}^{\mathrm{nae}}_{\Pi}(B) \ge \mathbf{Adv}^{\mathrm{sae}[L]}_{\mathrm{N2S}[\Pi,L]}(A)\,.$$

Adversary B is efficient if A is efficient and L is a basic level-set.

The efficiency referred to in the theorem statement is made more concrete by the theorem's proof, which is in Appendix A.5.

procedure $\overline{\mathcal{K}}$
611 **return** $k \twoheadleftarrow \mathcal{K}$

procedure $\overline{\mathcal{E}}(k, a, m, n)$
621 **if** $n = 2^\eta - 1$ **then return** $(\bot, n)$
622 $n \leftarrow n + 1$
623 $c \leftarrow \mathcal{E}(k, n, a, m)$
624 **return** $(n \,||\, c, n)$

procedure $\overline{\mathcal{D}}(k, a, nc, \mathbf{n})$
631 **if** $\mathbf{n} = \bot$ **then return** $(\bot, \bot)$
632 $n \,||\, c \leftarrow nc$; $\mathbf{n} \leftarrow \mathbf{n} \,||\, n$
633 **if** $\mathbf{n} \notin L$ **then return** $(\bot, \bot)$
634 $m \leftarrow \mathcal{D}(k, n, a, c)$
635 **if** $m = \bot$ **then** $\mathbf{n} \leftarrow \bot$
636 **return** $(m, \mathbf{n})$

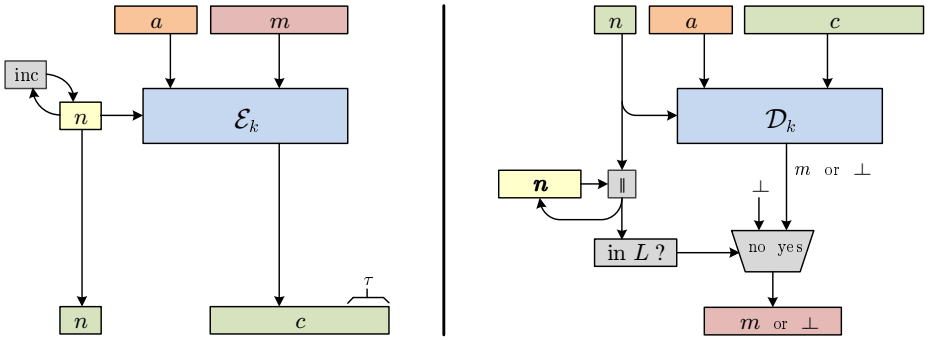


Fig. 9. Top: Definition of the N2S construction. For $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ an nAE scheme with η-bit nonces and L a level-set, we construct the sAE scheme $\mathrm{N2S}(\Pi, L) = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$. **Bottom: Illustration of the N2S construction.** Messages can be rejected because $\mathcal{D}$ calls for this or because the provided nonce, once concatenated to the prior ones received, is not in L. Various optimizations are possible, depending on L.

Discussion. While the decrypting party must, in general, maintain an unboundedly long state vector $\mathbf{n}$, for many level-sets this is unnecessary: the decryption algorithm will be able to make the decision it needs to make, at line 633, by retaining a finite amount of state. In particular, level-set L_0 needs no retained state; level-set L_1^ℓ needs the last $\ell + 1$ nonces; and level-sets L_2^ℓ and L_3 need the last nonce received.

Our N2S construction includes in the ciphertext the nonce used for the underlying nAE encryption. This is the usual way to use an nAE scheme, and the choice keeps our construction simple. But it has downsides, both for security and efficiency. The presence of the nonce reveals information that one might wish to hide. It might identify which user a message was sent by (as when one user has sent many messages, and another user has sent few). The presence of the nonces excludes the possibility of achieving IND$-security, meaning indistinguishability from random bits; it is, in fact, the reason we defined sAE security using the weaker notion of indistinguishability from the encryption of zero-bits (line 522 of Fig. 8). As for efficiency, N2S increases the ciphertext expansion from the nAE scheme's τ bits to $\eta + \tau$ bits, which may be unnecessary.

Addressing the efficiency complaint first, we note that if one is targeting sAE level L_3 (a reliable channel), the nonce n need not be included with the ciphertext, for the receiver will know what it must be if the ciphertext is to be valid. For levels L_1^ℓ and L_2^ℓ, with $\ell \in \mathbb{N}$, we can also reduce the ciphertext length. Instead of including the entire nonce n in the ciphertext, it is sufficient to include n mod $(2\ell+2)$ for L_1^ℓ or n mod $(\ell+1)$ for L_2^ℓ. From this the receiver can reconstruct the only possible value of n for a valid message. In practical settings, one would expect this information to fit in a single byte. Thus L_1^ℓ and L_2^ℓ are nicer than L_1^∞ and L_2^∞ not only for capping the state of the decrypting party but, also, for reducing ciphertext expansion.

As for security, what change to N2S is needed to achieve the stronger IND\$ definition? (For that, change line 522 to replace c by $|m| + \tau$ many uniformly random bits.) Perhaps the most obvious approach is to include in the ciphertext the enciphered nonce, rather than the nonce itself. One would use a blockcipher and a separate key. If η is small, like 32 or 96, one would need a blockcipher with an unusual block length. And the IND\$ security would now degrade, unpleasantly, with $q^2/2^\eta$. So a better construction, perhaps, is to append the nonce n to the plaintext and encrypt using a zero-nonce MRAE scheme [15], rather than a conventional nAE scheme like we used for N2S. This avoids the quantitative security loss and works for any level-set L. For L_1^ℓ and L_2^ℓ one can use the trick from the last paragraph and include only n mod ℓ within the scope of what is MRAE-encrypted. It is tempting to try to eliminate this too, using the nonce as the AD value and have the decrypting party employ trial decryptions. But this scheme is problematic because it does not achieve perfect correctness, which is required in our treatment of IND|C.

While it is beyond the scope of this paper to formalize and prove all of the claims made in the last couple of paragraphs, it is our contention that all of them are straightforward to establish within the framework of IND|C.

5 Variants

Formalizations of IND|C are quite robust with respect to definitional adjustments. In this section we describe three IND|C variants and explain in what sense each is equivalent. The three variants are: (1) whether or not to silence oracle responses from the "ideal" game that are invalid in the "real" game; (2) whether to silence-then-shut-down or silence-then-forgive, the latter meaning that oracle responses after silencing will still be returned to the adversary; and (3) whether to silence ideal-side responses, as we have done throughout, or to replace them with the real-side values.

At the end of Section 2 we described further alternatives, (0) a penalty-style version of IND|C, and (0′) an exclusion-style variant. One concludes from these examples that many of the definitional choices we have made are not significant.

We go on to look at a more distant alternative to INDC, which we call *symmetric* INDC. Meant to deal with left-or-right games instead of real-or-ideal ones, this variant silences oracle responses whenever the correctness condition

dictates fixed but distinct responses from the two sides. We suspect that this approach is, once again, as expressive as our other treatments of IND|C.

Before we describe our IND|C variants, let us clarify what it means to say that one way of defining advantage is equivalent to another. Suppose first that one has defined security measures $\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(A)$ and $\mathbf{Adv}^{\mathrm{yyy}}_{\Pi}(A)$. Then we may regard them as *equivalent* if any adversary A can be generically converted into an almost-as-efficient adversary B for which $\mathbf{Adv}^{\mathrm{yyy}}_{\Pi}(B)$ is nearly as high as $\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(A)$; and the other way around.

Now, for our more abstract setting, suppose we have two ways of associating an advantage measure to a primitive Π, a class C containing it, and a pair of Π-dependent games (G, H). Call these $\mathbf{Adv}^{\mathrm{xxx}}_{\mathrm{G}_\Pi,\mathrm{H}_\Pi,\mathrm{C}}$ and $\mathbf{Adv}^{\mathrm{yyy}}_{\mathrm{G}_\Pi,\mathrm{H}_\Pi,\mathrm{C}}$. Then these approaches for defining security are *equivalently expressive*, or just *equivalent*, if there's a generic method to construct from (G, H) a pair $(\mathrm{G}', \mathrm{H}')$ such that $\mathbf{Adv}^{\mathrm{xxx}}_{\mathrm{G}_\Pi,\mathrm{H}_\Pi,\mathrm{C}}$ and $\mathbf{Adv}^{\mathrm{yyy}}_{\mathrm{G}'_\Pi,\mathrm{H}'_\Pi,\mathrm{C}}$ are equivalent (in the sense of the last paragraph); and, also, the other way around.

(1) Silencing invalid transcripts. Recall that the formula we've been using for $\mathit{Fixed}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i)$ is $(\exists!\ y_i)\ \mathit{Valid}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i, y_i)$ where the symbol $\exists!$ means *there exists one and only one.* This choice implies that an adversary, when interacting with the ideal game H, will receive responses y_i (in not yet silenced games) for which the y_i-ending transcript could *not* occur with the real game—that is, when $\mathit{Valid}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i, y_i) = \mathsf{false}$. The rationale behind this choice is that the adversary should be given a chance to win the distinguishing game by observing that a response is invalid—that it could not occur with the "real" oracle—but that determination should still fall on the adversary.

Yet a natural variant is to silence invalid replies, effectively marking transcripts where the ideal oracle has failed to provide a plausible response. The new silencing condition would define $\mathit{Fixed}^{1}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i)$ as

$$(\forall y, y')\,(\mathit{Valid}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i, y) \wedge \mathit{Valid}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i, y') \Rightarrow y = y')$$

and would silence by

$$\mathit{Silence}^{1}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_j) = \bigvee_{1 \le i \le j} \mathit{Fixed}^{1}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i).$$

In words, we silence whenever there is *at most one* valid response, rather than demanding that there be *exactly one* valid response. We denote the advantage of adversary A under this new silencing condition by $\mathbf{Adv}^{\mathsf{indc1}}_{\mathrm{G,H,C}}(A) = \mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G}[\psi],\mathrm{H}[\psi],\mathrm{C}}$ where $\psi = \mathit{Silence}^{1}_{\mathrm{G,C}}$. We call the notion INDC1 security.

We argue that when $\mathit{Valid}_{\mathrm{G,C}}$ is efficiently computable, this alteration is irrelevant. Given an INDC adversary A, one can construct an INDC1 adversary A' that behaves as A does except when it sees a response y_i for which $\mathit{Valid}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i, y_i)$ is false. When this happens, adversary A' halts with a return value of 0. The constructed adversary is about as efficient as the original one (if $\mathit{Valid}_{\mathrm{G,C}}$ is easily computed) and has advantage no smaller than A's.

Conversely, the exact same reduction turns an INDC1 adversary A' to an INDC adversary A of comparable efficiency and undiminished advantage.

(2) SILENCE-THEN-FORGIVE. Our INDC formalization effectively punishes the adversary for triggering silencing: once silencing happens, the oracle shuts down and becomes useless. One might argue that this is overly punitive—that there is no reason to do anything other than silence just the offending query. We call this alternative *silence-then-forgive*. We explain that, when the silencing function is efficiently computable, the difference is inconsequential.

The silence-then-forgive notion is easy to formalize. We use the same *Valid* and *Fixed* predicates as defined in Section 2, but for the silencing function, instead of using the logical-or of *Fixed* applied to transcript prefixes, we use *Fixed* directly. That is, we let $\mathit{Silence}^2 = \mathit{Fixed}$ and define

$$\mathbf{Adv}^{\mathsf{ind2}}_{\mathrm{G,H,C}}(A) = \Pr[A^{\mathrm{G}[\psi]} \Rightarrow 1] - \Pr[A^{\mathrm{H}[\psi]} \Rightarrow 1]$$

where $\psi = \mathit{Silence}^2_{\mathrm{C,G}}$. We call this the INDC2 advantage of adversary A.

Given an INDC adversary A differentiating G and H, an INDC2 adversary A' can simply execute A in a black-box manner and whenever A asks a query that will be silenced according to ψ, adversary A' would stop its own interaction and continue simulating the $\Diamond$ response to A. Conversely, given an INDC2 adversary A', an INDC adversary A can simply execute A' and whenever it asks a query that will be silenced according to ψ, adversary A would ask the same query, but setting the declarative flag. It then returns a $\Diamond$ response to A'. Since setting the declarative flag guarantees the response would not be silenced, adversary A would never trigger silencing. The simulation is perfect. The argument implies that the two silencing notions are equally expressive.

(3) IDEAL-SIDE EDITING. So far, all of our INDC variants silence both the real and ideal sides. Consider the following alternative to oracle silencing: the real game G is never changed, while the ideal game H, instead of being silenced when a response is fixed according to G, returns that fixed response.

To formalize this, we change the boolean predicate *Fixed* into a function *fixed* that returns the unique string-valued response that is determined when the original predicate returns true, and returns $*$ (for "not-fixed") otherwise:

$$\mathit{fixed}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_i) = \begin{cases} y & \text{when } (\exists!\, y)\ \mathit{Valid}_{\mathrm{C,G}}(x_1, y_1, \ldots, x_i, y) \\ * & \text{otherwise} \end{cases}$$

We then extend the notion $\mathrm{G}[\psi]$ to include the case where ψ is a string-or-$*$-valued function. Specifically, the Oracle procedure of $\mathrm{G}[\psi]$ behaves as below.

procedure $\mathrm{G}[\psi]$.Oracle(x)
$i \leftarrow i + 1$; $x_i \leftarrow x$; $y_i \leftarrow \mathrm{G}$.Oracle($x$)
if $\psi(x_1, y_1, \ldots, x_i) \neq *$ **then** $y_i \leftarrow \psi(x_1, y_1, \ldots, x_i)$
return y_i

Finally, we define INDC3 advantage by $\mathbf{Adv}^{\mathrm{indc3}}_{\mathrm{G,H,C}}(A) = \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G,H}[\psi]}(A)$ where $\psi = \mathit{fixed}_{\mathrm{C,G}}$. We call this INDC variant *ideal-side editing*.

We argue that INDC3 is equivalent to INDC assuming $\mathit{fixed}_{\mathrm{G,C}}$ is efficiently computable. Let A be an INDC adversary differentiating a real game G and an ideal game H, where C is the underlying class. One can construct an INDC3 adversary A' executing A in a black-box manner. Whenever A asks a query x_i such that the history so far, when applied to $\mathit{fixed}_{\mathrm{G,C}}$, results in a string response y_i, then A' stops its own interaction and provides the silencing mark $\Diamond$ to A. Conversely, with A' an INDC3 adversary we can construct an INDC adversary A executing A' in a black-box manner. Whenever A' first asks a query x_i such that $\mathit{fixed}_{\mathrm{G,C}}(x_1, y_1, \ldots, x_i) = y_i \neq *$, adversary A would forward x_i to its own game, but would set a declarative flag so that silencing is not triggered. It then returns y_i to A'. Both reductions are perfect in simulating the game interaction.

(4) Symmetric silencing. Our last form of game-editing is meant to deal with left-or-right style games instead of real-or-ideal style games. A typical example was given in Section 3, the treatment of CCA-secure PKE in which an oracle accepts two equal-length plaintexts and encrypts either the left or the right one of them. Can one directly use our INDC definition in such a setting?

One can, but doing so doesn't make sense. A real game is different from an ideal one, and its privileged position makes it reasonable that $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}$ is not $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{H,G,C}}$. But a left game and a right game ought not be treated differently: it should be the case that the order of naming them doesn't matter.

Although the rationale just stated is a philosophical one, we have found that trying to apply IND|C to the LR-style games of Section 3 just doesn't work.

Here is a way to realize *symmetric* silencing: silence when the responses of the games are distinct fixed strings. Namely, let

$$\mathit{Fixed}^4_{\mathrm{G,H,C}}(\boldsymbol{t}) = \left(\mathit{fixed}_{\mathrm{C,G}}(\boldsymbol{t}) \neq \mathit{fixed}_{\mathrm{C,H}}(\boldsymbol{t}) \wedge \mathit{fixed}_{\mathrm{C,G}}(\boldsymbol{t}) \neq * \wedge \mathit{fixed}_{\mathrm{C,H}}(\boldsymbol{t}) \neq *\right).$$

Note that the predicate is symmetric: $\mathit{Fixed}^4_{\mathrm{G,H,C}} = \mathit{Fixed}^4_{\mathrm{H,G,C}}$. Define the silencing function $\mathit{Silence}^4_{\mathrm{G,H,C}}(\boldsymbol{t})$ as the logical-or of $\mathit{Fixed}^4_{\mathrm{G,H,C}}(\boldsymbol{t}')$ applied to all prefixes $\boldsymbol{t}'$ of $\boldsymbol{t}$. The INDC-SYM advantage of an adversary A is then defined as $\mathbf{Adv}^{\text{indc-sym}}_{\mathrm{G,H,C}}(A) = \mathbf{Adv}^{\mathrm{ind}}_{\mathrm{G}[\psi],\mathrm{H}[\psi]}(A)$ where $\psi = \mathit{Silence}^4_{\mathrm{G,H,C}}$.

We use the games in section 3 to give an example. Let G and H be the left and right games in Fig. 6 with line 317 and line 327 removed, and let C be the class of correct PKE schemes. We remove the two lines so that the decryption does not exclude challenged ciphertext and thus the games become "utopian." The $\mathit{Fixed}^4_{\mathrm{G,H,C}}$ predicate, in this case, evaluates to true if (1) no two encryptions of the same-side but distinct plaintexts return identical ciphertexts (validity condition); and (2) a decryption of c is asked while there was a previous $\mathrm{ENC}(m_1, m_2)$ oracle returning c and $m_1 \neq m_2$ (fixedness condition). Therefore, apart from the explicit checking of an additional validity condition, the INDC-SYM notion again coincides with the conventional IND-CCA one.

6 Conclusions

Definitions in cryptography often vary in subtle ways, and deciding among them can seem rather subjective. The IND|C framework may help lessen this subjectivity. It embodies a thesis that a definition is "right" when it attends to the limits imposed by correctness, but goes no further than that in restricting adversarial behavior.

We suspect there are many cryptographers who have written definitions with an implicit view that what they aim to do is to disallow all and only the adversarial behaviors that some correctness condition dictates. The challenge of this work has been in figuring out how to make this vague conception real.

The IND|C approach is rather abstract. Definitions one gets out of it may require significant investigation to concretely characterize or understand. For this reason, one might claim that IND|C doesn't banish complexity so much as hide it. At least with a complicated game, the argument might go, you can see the complexity before your eyes.

We regard the critique as mostly off-base. Most fundamentally, it is unrealistic to think that complex cryptographic goals admit simple formulations when described in low-level terms. A more realistic aim is to find abstraction boundaries that help modularize definitions and enhance intuition.

The situation is reminiscent of UC [6], where an ideal functionality can be simply specified, a definition inherited from it, but it may be quite unclear what that notion means. Yet the hidden complexity behind IND|C isn't remotely at the level of UC. Nor, in our simpler setting, is there much difficulty with rigor. Perhaps IND|C may come to serve as an alternative to UC, for some cryptographic problems, the utopian game H corresponding to the specification of the ideal functionality.

Acknowledgments

Many thanks to anonymous reviewers of this paper, whose questions motivated the addition of Section 5. Thanks to the NSF, which provided funding for this work under grants CNS 1314885 and CNS 1717542.

References

1. Bellare, M., Hofheinz, D., Kiltz, E.: Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? Journal of Cryptology 28(1), 29–48 (Jan 2015) 3, 8, 9, 12, 13
2. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the encode-then-encrypt-and-MAC paradigm. ACM Trans. Inf. Syst. Secur. 7(2), 206–241 (2004), `https://doi.acm.org/10.1145/996943.996945` 3, 14
3. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006) 4

4. Boyd, C., Hale, B., Mjølsnes, S.F., Stebila, D.: From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. In: Sako, K. (ed.) Topics in Cryptology – CT-RSA 2016. Lecture Notes in Computer Science, vol. 9610, pp. 55–71. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 29 – Mar 4, 2016) 3, 14
5. Boyd, C., Hale, B., Mjølsnes, S.F., Stebila, D.: From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. Cryptology ePrint Archive, Report 2015/1150, revision 20160919:152253 (2016), `https://eprint.iacr.org/2015/1150` 3
6. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000), `http://eprint.iacr.org/2000/067` 4, 22
7. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 453–474. Springer, Heidelberg, Germany, Innsbruck, Austria (May 6–10, 2001) 4
8. Fischlin, M., Günther, F., Marson, G.A., Paterson, K.G.: Data is a stream: Security of stream-based channels. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 545–564. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015) 4
9. Fischlin, M., Gnther, F., Marson, G.A., Paterson, K.G.: Data is a stream: Security of stream-based channels. Cryptology ePrint Archive, Report 2017/1191 (2017), `https://eprint.iacr.org/2017/1191` 4
10. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 273–293. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012) 4
11. Kohno, T., Palacio, A., Black, J.: Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint Archive, Report 2003/177 (2003), `http://eprint.iacr.org/2003/177` 3, 14
12. Namprempre, C.: Secure channels based on authenticated encryption schemes: A simple characterization. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 515–532. Springer, Heidelberg, Germany, Queenstown, New Zealand (Dec 1–5, 2002) 4
13. Perrin, T., Marlinspike, M.: The double ratchet algorithm. Open Whisper Systems (2016), `https://signal.org/docs/specifications/doubleratchet/` 14
14. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 02: 9th Conference on Computer and Communications Security. pp. 98–107. ACM Press, Washington D.C., USA (Nov 18–22, 2002) 15
15. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006) 18

A Proofs

A.1 Proof of Theorem 1

It suffices to give mutual reductions between INDC and INDC0 adversaries. Since fixedness is efficiently computable for $(\mathrm{G}, \mathrm{H}, \mathrm{C})$ we know there exists a PT

algorithm ϕ that computes $\mathit{Fixed}_{\mathrm{C,G}}$ for all valid transcripts. In the following we give the two reductions.

Let A be an INDC adversary. We construct an INDC0 adversary B that does the following: it runs A as a black-box and forwards every query made by A. Before forwarding a query x_t, however, it appends x_t to the recorded transcript and computes ϕ on it. If ϕ returns true then B stops forwarding and from then on keeps returning $\Diamond$ to A. Clearly B is PPT when A is. In addition, adversary B never triggers the penalty in Finalize, and it perfectly simulates the INDC game for A.

Conversely, let B be an INDC0 adversary. We construct A that does the following: it runs B as a black-box and forwards every query made by B. But A gives up and returns 0 whenever it sees $\Diamond$ returned by the game. When B is PPT then so is A. For the advantage, let $\mathsf{bad}_{\mathrm{G}}$ and $\mathsf{bad}_{\mathrm{H}}$ denote the events that A sees a $\Diamond$ response when interacting with G and H, then we have $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G,H,C}}(A,k) = \Pr[A^{\mathrm{G}[\psi]} \to 1] - \Pr[A^{\mathrm{H}[\psi]} \to 1] = \Pr[A^{\mathrm{G}[\psi]} \to 1 \cap \neg\mathsf{bad}_{\mathrm{G}}] - \Pr[A^{\mathrm{H}[\psi]} \to 1 \cap \neg\mathsf{bad}_{\mathrm{H}}] = \Pr[\mathrm{G}[\![\psi]\!]^B \to 1] - \Pr[\mathrm{H}[\![\psi]\!]^B \to 1] = \mathbf{Adv}^{\mathrm{indc0}}_{\mathrm{G,H,C}}(B,k)$, and the reduction is complete.

A.2 Proof of Theorem 2

First note that the formula ϕ in Fig. 5 is PT-omputable. We must show that $\mathit{Valid}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t) \vee \mathit{Valid}_{\mathrm{C1,H1}}(x_1, y_1, \ldots, x_t)$ implies $\phi(x_1, y_1, \ldots, x_t) = \mathit{Fixed}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t)$.

Fix such a transcript. We claim that $\mathit{Valid}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_{t-1}, y_{t-1})$ if and only if line 201 in Fig. 5 is true. The only-if direction is straightforward: the negation of line 201 violates correctness required by the scheme class C1. For the if direction, consider the artificial scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, whose definition depends on the transcript, with the following behavior:

- $\mathcal{K}(k)$: regardless of k, if there is any $(x_i, y_i) = (\mathrm{Key}, pk)$ then output pk. Otherwise output an arbitrary string.
- $\mathcal{E}(pk, m)$: output $c \xleftarrow{\$} T[m]$ where $T[m] \subseteq \{0,1\}^*$ are sets of strings, indexed by $m \in \{0,1\}^*$, which satisfy:
  - if there is an $(x_i, y_i) = ((\mathrm{Enc}, m), c)$ then $c \in T[m]$.
  - $(m \neq m') \Rightarrow (T[m] \cap T[m'] = \varnothing)$.
  - if there is an $(x_i, y_i) = ((\mathrm{Dec}, c), m)$: when m is not the challenge plaintext then $c \notin T[m']$ for all $m' \in \{0,1\}^*$; otherwise $c \in T[m]$. (By the *challenge plaintext* we mean the input to the first Enc query in the transcript, if it exists.)
- $\mathcal{D}(sk, c)$: if $(\exists m)\ c \in T[m]$, then output m; else if there exists some $(x_i, y_i) = ((\mathrm{Dec}, c), m)$ then output m; else output an arbitrary string.

It is straightforward to verify that Π as constructed above is correct and can generate the given transcript. It remains to show well-definedness, namely, the existence of indexed sets $T[m]$ for $m \in \{0,1\}^*$. Note the only possible contradiction in the construction of T is between the first bullet and the third bullet in the

description of $\mathcal{E}$. However, such a contradiction can only take place when there is an $(x_i, y_i) = ((\mathrm{Enc}, m), c)$ and an $(x_j, y_j) = ((\mathrm{Dec}, c), m')$ such that $m \neq m'$, exactly the case excluded by line 201 in Fig. 5. The if direction is thus proved.

If $\phi(x_1, y_1, \ldots, x_t)$ is true then one of the lines 202–205 is true. From the code of G1 and the definition of C1, it is straightforward to verify that whichever line in 202–205 is true, the values recorded in the transcript determine the value of $\mathrm{G1}_{\Pi}(k, x_1, \ldots, x_t, r)$. Additionally, since the above claim says whenever line 201 is true then the transcript is valid, we conclude that the value of $\mathit{Fixed}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t)$ is true.

Conversely, if $\phi(x_1, y_1, \ldots, x_t)$ is false then either line 201 or the disjunction of line 202–205 is false. In the former case, our claim implies the falseness of $\mathit{Valid}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t)$, so $\mathit{Fixed}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t)$ is also false. In the latter case, consider the artificial scheme we just constructed. Since such a scheme can always generate the given history as long as the indexed set T satisfies the required properties, it suffices to give two instantiations of Π which generate distinct responses for x_t. A routine check of the code of G1 concludes that: for all transcripts not falling in the four cases of line 202–205, such instantiations can indeed be given. We conclude that $\mathit{Fixed}_{\mathrm{C1,G1}}(x_1, y_1, \ldots, x_t)$ in this case, is also false.

A.3 Proof of Theorem 3

We give reductions for both directions. First, let A be a PPT IND-CCA adversary attacking Π. We construct an INDC-adversary B that does the following. For all Dec queries and Key queries asked by A, forward them to its own game if ψ evaluates to false for the current transcript; otherwise simulate the answers by itself without forwarding (which could be done by an inspection of the code of the four games $(\mathrm{G1}, \mathrm{H1}, \mathrm{G0}, \mathrm{H0})$). For the first Test query (m_1, m_2) queried by A, we let B draw a random coin $b \xleftarrow{\$} \{0,1\}$ and query $\mathrm{Enc}(m_b)$. Now $\mathbf{Adv}^{\mathrm{indc}}_{\mathrm{G1,H1,C1}}(B) = \mathbf{Adv}^{\mathrm{ind}}_{\Pi}(A)/2$, and B is also PPT.

Next, let B be a PPT IND|C adversary, we construct an IND-CCA adversary A that does the following: forward all Dec queries and Key queries; for the Enc query m, let A query $\mathrm{Test}(m, 0^{|m|})$. In addition A will also silence queries made by B by computing ψ. This reduction simulates perfectly except for one problematic case: when B triggered an invalid event (the negation of line 201 in Fig. 5) and asks for a decryption of the challenged ciphertext, by the formula of ψ he should see an unsilenced response, but the IND-CCA adversary A cannot simulate such a response for him. However, since an invalid event necessarily implies that A is in the ideal world, we could simply let A return 0. Therefore, the advantage of B is preserved.

A.4 Proof of Theorem 4

We introduce some notation first. Given $\mathbf{n} \in \mathbb{N}^*$ and a vector X, we define $X[\mathbf{n}]$ recursively by $X[\emptyset] = \emptyset$ and $X[\mathbf{n} \,||\, i] = X[\mathbf{n}] \,||\, X_i$ if $1 \leq i \leq |X|$, while $X[\mathbf{n}]$ otherwise. For $\mathbf{n} = [i, i+1, \ldots, j]$ we may write $X[i..j]$ instead of $X[\mathbf{n}]$. We use $\boldsymbol{t} =$

procedure $\phi_L(\boldsymbol{t})$
711 $(x_1, y_1, \ldots, x_q) \leftarrow \boldsymbol{t}$
712 **for** $i \leftarrow 1$ **to** $q-1$ **do**
713 **if** $x_i[1] = \text{Enc}$ **then** $e \leftarrow e+1$; $(a_e, m_e, c_e) \leftarrow (x_i[2], x_i[3], y_i)$
714 **for** $i \leftarrow 1$ **to** q **then**
715 **if** $x_i[1] = \text{Dec}$ **then** $d \leftarrow d+1$; $(a'_d, m'_d, c'_d) \leftarrow (x_i[2], y_i, x_i[3])$
716 **if** $(\exists i)(\exists \mathbf{n} \in L)\ (a'[1..i], c'[1..i]) = (a[\mathbf{n}], c[\mathbf{n}]) \wedge m'_i = \perp$ **then return** false
717 **if** $(\exists \mathbf{n}, \mathbf{n}' \in L)\ \big(\mathbf{n} \neq \mathbf{n}' \wedge (m[\mathbf{n}] \neq m[\mathbf{n}']) \wedge (a[\mathbf{n}], c[\mathbf{n}]) = (a[\mathbf{n}'], c[\mathbf{n}'])\big)$
718 **then return** false
719 **return** $x_q[1] = \text{Dec} \wedge x_q[4] = \mathsf{false} \wedge (\exists \mathbf{n} \in L)\ (a', c') = (a[\mathbf{n}], c[\mathbf{n}])$

Fig. 10. Computing fixedness for sAE.

procedure $\mathcal{K}$
811 **return** 0

procedure $\mathcal{E}(k, a, m, s)$
821 **if** $s = \varepsilon$ **then** $s \leftarrow 0$
822 $s \leftarrow s+1$
823 **return** $(F_{s,a}(m), s)$

procedure $\mathcal{D}(k, a, c, r)$
824 **if** $\mathsf{isNum}(r)$ **then**
825 $r \leftarrow r+1$; **return** $(\text{G}(r, a, c), r)$
826 **if** $r = \varepsilon$ **then** $r \leftarrow \{\Lambda\}$
827 $r' \leftarrow \varnothing$
828 **for** $\mathbf{n} \in r$ **do**
829 **for** $n \in \{n\colon \mathbf{n} \,||\, n \in L\}$ **do**
830 **if** $F^{-1}_{n,a}(c) \neq \perp$ **then**
831 $m \leftarrow F^{-1}_{n,a}(c)$
832 $r' \leftarrow r' \cup \{\mathbf{n} \,||\, n\}$
833 **if** $r' = \varnothing$ **then return** $(\text{G}(0, a, c), 0)$
834 **return** (m, r')

Conditions on F for $\Pi \in \text{C2}(L)$ and Π being able to generate the history:
841 $(\forall q)(\forall a_1, \ldots, a_q \in \mathcal{A})(\forall c_1, \ldots, c_q \in \mathcal{C})(\forall \mathbf{n}, \mathbf{n}' \in L)$
842 $\Big[$**For** $i \leftarrow 1$ **to** q **do** $(m_i, m'_i) \leftarrow (F^{-1}_{\mathbf{n}_i, a_i}(c_i), F^{-1}_{\mathbf{n}'_i, a_i}(c_i))$:
843 $((\forall i)\ m_i \neq \perp \wedge m'_i \neq \perp) \Rightarrow ((\forall i)\ m_i = m'_i)\Big] \bigwedge$
844 $(\forall i \in \{1, \ldots, e\})\ F_{i,a_i}(m_i) = c_i \bigwedge (\forall i \in \mathbb{N})\ c'_{h+1} \notin \mathsf{Range}(F_{i,a'_{h+1}})$

Fig. 11. Code of the artificial sAE scheme.

$(x_1, y_1, \ldots, x_q)$ to denote a query-terminated transcript that is $\mathit{Valid}_{\text{C2}(L),\text{G2}}(\boldsymbol{t}) \vee \mathit{Valid}_{\text{C2}(L),\text{H2}}(\boldsymbol{t})$ and satisfies $q \leq 2^\tau - 3$.

Our proof strategy is as follows. We first give the pseudocode of a function ϕ_L for a general level-set L, and then prove its efficient computability when L is a basic level-set. See Fig. 10 for the code of ϕ_L.

We claim ϕ_L indeed computes fixedness. Let $\boldsymbol{t}$ be such a transcript that satisfies the stated condition in the above, we use $(a_1, c_1), \ldots, (a_e, c_e)$ and $(a'_1, c'_1), \ldots, (a'_d, c'_d)$ to denote its *encryption history* and *decryption history*, defined as in Fig. 10. Given a decryption query (a'_i, c'_i), we say it is $\mathbf{n}$-*honest* if $\mathbf{n} \in L \wedge (a'[1..i], c'[1..i]) = (a[\mathbf{n}], c[\mathbf{n}])$; and it is *honest* if it is $\mathbf{n}$-honest for some $\mathbf{n} \in L$. We use h to denote the largest index of honest decryption queries in the decryption history, namely $h = \max\{i\colon (a'_i, c'_i) \text{ is honest}\}$.

We try to define an artificial scheme $\Pi \in \mathrm{C2}(L)$, out of two functions $F\colon \mathbb{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \cup \{\perp\}$ and $G\colon \mathbb{N} \times \mathcal{A} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$. We require for all $(n, a) \in \mathbb{N} \times \mathcal{A}$, the projected $F_{n,a}(\cdot)$, apart from its possible mapping to $\perp$, is an injection with ciphertext expansion τ: We accordingly write $F_{n,a}^{-1}(c)$ to denote the unique $m \in \mathcal{M}$ such that $F_{n,a}(m) = c$ or $\perp$ if such m does not exist. Basically, the behavior of F has some restrictions that depend on the given transcript $\boldsymbol{t}$, and such restriction serves its best to make sure Π can generate $\boldsymbol{t}$. Ultimately, we expect Π to have the following properties:

> If the validity check is passed (both the if conditions in line 716 and 717 are false) then Π is well-defined, correct, and can generate $\boldsymbol{t}$. On top of that, if the fixedness check does not pass (line 719 returns false), then there are multiple instantiations of Π that generate distinct responses for the last query in $\boldsymbol{t}$.

We call these two properties the *existence property* and the *multiplicity property*.

The code of Π is given in Fig. 11. We first make two observations about it:

- When F satisfies line 842–843, the variable m assigned in line 831 is identical across iterations for each decryption in a *correctness experiment* as shown in Fig. 8, and $\Pi \in \mathrm{C2}(L)$. This can be proved by an induction on the number of decryption queries in a correctness experiment. The inductive argument is: after the first i decryptions $(a_1, c_1), (a_2, c_2), \ldots, (a_i, c_i)$ in a correctness experiment, all vectors $\mathbf{n}$ in the receiver state are reorderings in L, and for each $\mathbf{n}$ let $(m_1, \ldots, m_t) \leftarrow (F_{n_1,a_1}^{-1}(c_1), \ldots F_{n_t,a_t}^{-1}(c_t))$ then $(\nexists i)\ m_i = \perp$ and the vector m is identical for all $\mathbf{n}$ in the receiver state.
- When F satisfies all lines 842–844, the scheme Π can generate the encryption history. What's more, as long as the if condition in line 716 in Fig. 10 evaluates to false, then Π, with some instantiation of G, can generate the decryption history as well. The generation of the encryption history is obvious by line 844. For the generation of the decryption history, note that $\mathit{Valid}_{\mathrm{C2}(L),\mathrm{G2}}(\boldsymbol{t}) \vee \mathit{Valid}_{\mathrm{C2}(L),\mathrm{H2}}(\boldsymbol{t})$ implies that an $\mathbf{n}$-honest decryption query (a_i', c_i') must have a response either equal to m_{n_i} (in the real world) or $\perp$ (in the ideal world). Since the falseness of the if condition in line 716 of Fig. 10 excludes the latter case, the correctness of Π thus ensures those honest decryption queries' responses can be generated. For those *post-honest* decryption queries, since line 844 implies that the first of those queries c_{h+1}' is not in the range of $F_{i,a_{h+1}'}(\cdot)$ for any i, the updated receiver state r' will be set to $\varnothing$ and from this point the decryption will depend only on G. With the help of the additive state, by simply assigning $\mathrm{G}(i - h - 1, a_i', c_i') \leftarrow m_i'$ for all $i > h$, we can make Π generate the given decryption history as well.

Based on the above two observations, we first prove the existence property. For a number $i \in \mathbb{N}$, let $\mathsf{num2str}_j(i)$ be the binary representation of i with j bits (leading 0 padded when $i \ll 2^j$). Suppose $\boldsymbol{t}$ is such that both the if conditions

in line 716 and 717 evaluate to false then consider the following F:

$$F_{i,a}(m) = \begin{cases} c_i & \text{if } a = a_i \wedge m = m_i; \\ H(i,m) & \text{else if } i \le e; \\ \bot & \text{otherwise,} \end{cases}$$

where $H\colon \{1,2,\ldots,q\} \times \mathcal{M} \to \mathcal{C}$ satisfies

1. $H(i,\cdot)$ is an injection for all i with ciphertext expansion τ;
2. $i \neq j \Rightarrow \mathsf{Range}(H(i,\cdot)) \cap \mathsf{Range}(H(j,\cdot)) = \varnothing$;
3. $\boldsymbol{t}.c_i \notin \mathsf{Range}(H(i,\cdot))$ for all i;
4. $\boldsymbol{t}.c'_{h+1} \notin \mathsf{Range}(H(i,\cdot))$ for all i.

It's easy to see such an H really exists by the condition $q \le 2^\tau - 3$. We claim that this instantiation satisfies the three conditions in Fig. 11, which would imply the existence property. Indeed, line 844 is obvious. For line 841–843, let $(a_1, a_2, \ldots, a_q)$, $(c_1, c_2, \ldots, c_q)$, $\mathbf{n}$ and $\mathbf{n}'$ be as quantified, then $((\forall i)\ m_i \neq \bot \wedge m'_i \neq \bot)$ implies that for all i, either $c_i \in \mathsf{Range}(H(\mathbf{n}_i,\cdot)) \cap \mathsf{Range}(H(\mathbf{n}'_i,\cdot))$, or c_i is equal to $\boldsymbol{t}.c_j$ for some j (We use the notation $\boldsymbol{t}.c_i$ to differentiate the c_i being quantified in the statement of line 841–843 and the c_i recorded in the transcript $\boldsymbol{t}$). In the former case, by the second property of H above we have $n_i = n'_i$, hence $m_i = m'_i$. In the latter case, suppose for contradiction that $m_i \neq m'_i$ and let i be the minimal such index. Since by the instantiation of F, for $j \le i$ either $n_i = n'_i$ (the case we just analyzed) or $(a_i, c_i) = (\boldsymbol{t}.a[n_i], \boldsymbol{t}.c[n_i]) = (\boldsymbol{t}.a[n'_i], \boldsymbol{t}.c[n'_i])$, we conclude $(\boldsymbol{t}.a[\mathbf{n}], \boldsymbol{t}.c[\mathbf{n}]) = (\boldsymbol{t}.a[\mathbf{n}'], \boldsymbol{t}.c[\mathbf{n}'])$. The assumption $m_i \neq m'_i$ therefore contradicts with the condition that line 717 in Fig. 10 returns false.

We next show the multiplicity property. There are three cases of $\boldsymbol{t}.x_q$ to consider. They are: 1) Dec query with the declarative flag set to true; 2) Dec query that is not honest; 3) Enc query. The first case is trivial. For the second case, since our construction depends on an arbitrary function G after a dishonest decryption, there are always multiple ways of specifying different G so as x_q will have distinct outputs. For the third case, it suffices to extend the instantiation of F by H in the above with e replaced by $e+1$. By the condition $q \le 2^\tau - 3$, the four conditions can still be satisfied by a proper choice of H, and all successive logic thus follows. The different way of instantiating $H(e+1,\cdot)$ thus guarantees multiplicity property.

To complete the proof, we need to write pseudocode of efficient algorithms for the procedure ϕ_L where L is a basic level set. See Fig. 12 for the concrete code of these algorithms that instantiate ϕ_L.

A.5 Proof of Theorem 5

We describe the code of B in terms of A. See Fig. 13. We claim that this reduction achieves perfect simulation. To see why, note that the only bad event which semantically differs from an otherwise perfect simulation is line 1032, which

```
procedure φ_j^ℓ(t)
911 (x_1, y_1, ..., x_q) ← t
912 if x_q[1] ≠ Dec ∨ x_q[4] ≠ false then return false
913 for i ← 1 to q − 1 do
914   if x_i[1] = Enc then
915     e ← e + 1; (a_e, m_e, c_e) ← (x_i[2], x_i[3], y_i);
916     ac2i[a_e, c_e] ← ac2i[a_e, c_e] ∪ {e}
917 if j = 0 then
918   for i ← 1 to e do
919     if ac2m[a_e, c_e] ≠ ◊ and ac2m[a_e, c_e] ≠ m_e then return false
920     ac2m[a_e, c_e] ← m_e
921 else
922   queue ← ((Λ, Λ))
923   while (n_1, n_2) ← pop(queue) do
924     for (n_1, n_2) ∈ Next(L_j^ℓ, n_1, e) × Next(L_j^ℓ, n_2, e) do
925       if (a[n_1], c[n_1]) = (a[n_2], c[n_2]) then
926         if m[n_1] ≠ m[n_2] then return false
927         push(queue, (n_1 || n_1, n_2 || n_2))
928 for i ← 1 to q do
929   if x_i[1] = Dec then
930     d ← d + 1; (a'_d, m'_d, c'_d) ← (x_i[2], y_i, x_i[3])
931 N ← {Λ}
932 for i ← 1 to d do
933   for n ∈ N do
934     for n ∈ ac2i[a'_i, c'_i] do
935       if n || n ∈ L_j^ℓ then
936         N' ← N' ∪ {n || n}
937         if m'_i = ⊥ then return false
938   N ← N'; N' ← ∅
939 return N ≠ ∅
```

Fig. 12. Pseudocode of algorithms computing fixedness for sAE. The algorithm ϕ_j^ℓ computes fixedness for $(\mathrm{C2}(L_j^\ell), \mathrm{G2}, \mathrm{H2}, 2^\tau - 3)$ where τ is the ciphertext expansion for sAE schemes. The only dependences on level-sets are in line 924 and line 935, where $\mathsf{Next}(L, \mathbf{n}, e) = \{n \in \{1, \ldots, e\} \colon \mathbf{n} \,||\, n \in L\}$. The value $\mathsf{Next}(L, \cdot, \cdot)$ is efficiently computable for all $L \in \{L_0, L_1^\ell, L_2^\ell, L_3\}$.

When A queries $\mathrm{Enc}(a, m)$
1011 **if** silenced **return** $\Diamond$
1012 $\boldsymbol{t} \leftarrow \boldsymbol{t} \,||\, (\mathrm{Enc}, a, m)$
1013 **if** $\psi_L(\boldsymbol{t})$ **then**
1014 silenced $\leftarrow$ true; **return** $\Diamond$
1015 $n \leftarrow n + 1$
1016 $c \leftarrow \mathrm{Enc}(n, a, m)$
1017 $\boldsymbol{t} \leftarrow \boldsymbol{t} \,||\, c$
1018 **return** $n \,||\, c$

When A queries $\mathrm{Dec}(a, nc)$
1021 **if** silenced **return** $\Diamond$
1022 $\boldsymbol{t} \leftarrow \boldsymbol{t} \,||\, (\mathrm{Dec}, a, c)$
1023 **if** $\psi_L(\boldsymbol{t})$ **then**
1024 silenced $\leftarrow$ true; **return** $\Diamond$

1025 **if** $\mathbf{n} = \bot$ **then** $m \leftarrow \bot$
1026 **else**
1027 $n \,||\, c \leftarrow nc$; $\mathbf{n} \leftarrow \mathbf{n} \,||\, n$
1028 **if** $\mathbf{n} \notin L$ **then** $m \leftarrow \bot$
1029 **else if**
1030 $(\nexists m')\ ((\mathrm{Enc}, n, m'), c) = \boldsymbol{t}.x_n$
1031 **then** $m \leftarrow \mathrm{Dec}(n, a, c)$
1032 **else** $m \leftarrow m'$
1033 **if** $m = \bot$ **then** $\mathbf{n} \leftarrow \bot$
1034 $\boldsymbol{t} \leftarrow \boldsymbol{t} \,||\, m$
1035 **return** m

When A outputs b
1036 Output b

Fig. 13. Construction of an nAE adversary out of an sAE adversary. The reduction simulates perfectly since the only bad event in line 1032 never takes place in the ideal setting.

causes digression from the semantics of H2 but not that of G2, hence it suffices to show in an execution between B and the ideal side, line 1032 is never reached. Suppose for contradiction that it is reached, then by the code semantics, oracle silencing has not taken place, and the nonces in the nc input for Dec queries so far form a reorder $\mathbf{n} \in L$. Consider the first Dec query of which $nc = (n_1, c)$ for some c. Due to the monotonicity of silencing and the event $\mathbf{n} = \bot$, at the point of the query, line 1029 is reached and the if conditions there can be either true or false. If it is true, then in the ideal world H2 we must have m assigned as $\bot$, and accordingly $\mathbf{n}$ gets assigned to $\bot$, contradicting to the assumption that line 1032 is reached later. If it is false, then at this query the vector $\mathbf{n}_1$ already forms a valid reorder of the encryption history at the time, so this query should already have been silenced at line 1024, a contradiction again.

We conclude that B simulates for A a perfect execution of the silenced $(\mathrm{G2}_{\mathrm{N2S}[\Pi,L]}, \mathrm{H2}_{\mathrm{N2S}[\Pi,L]})$ games. Adversary B is efficient when A is and the proof is complete.