Secure Channels based on Authenticated Encryption Schemes: A Simple Characterization

Chanathip Namprempre*

May 28, 2002

Abstract

We consider communication sessions in which a pair of parties begin by running an authenticated key-exchange protocol to obtain a shared session key, and then secure successive data transmissions between them via an authenticated encryption scheme based on the session key. We show that such a communication session meets the notion of a secure channel protocol proposed by Canetti and Krawczyk [8] if and only if the underlying authenticated encryption scheme meets two new, simple definitions of security that we introduce, and the key-exchange protocol is secure. In other words, we reduce the secure channel requirements of Canetti and Krawczyk to easier to use, stand-alone security requirements on the underlying authenticated encryption scheme.

Keywords: Secure channels, authenticated encryption, security notions.

*Dept. of Computer Science & Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: `meaw@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/cnamprem`. Supported in part by Mihir Bellare's 1996 Packard Foundation Fellowship in Science and Engineering and NSF CAREER Award CCR-9624439.

Contents

1 Introduction **3**

2 Definitions **5**
2.1 Preliminaries 5
2.2 Secure Channels per Canetti and Krawczyk [8] 7
2.3 From KE and Authenticated Encryption Schemes to Channel Protocols 9

3 Simple Characterizations of Authenticated Encryption Schemes for Secure Channels **10**

4 SINT-PTXT and IND-CCVA are Necessary and Sufficient **11**

5 Understanding Secure Channels through SINT-PTXT and IND-CCVA **13**
5.1 SINT-PTXT is Reasonably Strong 13
5.2 Integrity is Necessary for IND-CCVA 14
5.3 IND-CCVA is Technically Weak 14
5.4 Strength of the Combination of IND-CCVA and SINT-PTXT is Unclear 14

A Adversary Actions **16**

B Description of the send* and incoming* Activations **16**

C Proofs that SINT-PTXT and IND-CCVA are Necessary and Sufficient **17**
C.1 Proof of Lemma C.1 18
C.2 Proof of Lemma C.2 20
C.3 Proof of Lemma C.3 22
C.4 Proof of Lemma C.4 23

D A Deterministic Encryption Scheme Secure under IND-CCVA **24**

1 Introduction

We consider communication sessions in which a pair of parties begin by running an authenticated *key-exchange* (KE) protocol to obtain a shared *session key*, and then secure successive data transmissions between them via an *authenticated encryption scheme*, a shared-key-based encryption scheme whose goal is to provide *both* privacy *and* authenticity, based on the session key. Many popular Internet protocols follow this structure [17, 12, 11, 1]. One reason is that it minimizes computationally intensive public-key cryptography by using more efficient symmetric-key cryptography for the bulk of the communication.

At Eurocrypt 2001, Canetti and Krawczyk presented security definitions for protocols of this form [8]. They refer to such protocols as *network channel protocols* (or *channel protocols* for short). In their work, they derive a realistic adversarial model from [2] and formulate security definitions using a mixture of both *simulation-based* and *indistinguishability-based* approaches. The former allows them to realistically and naturally capture the security properties of channel protocols and the settings in which the protocols are deployed. The latter allows them to prove security of the protocols with relative ease. The result is the notion of *secure channels*, a notion that captures the desired security properties of the communication channels themselves, rather than those of the components used in constructing them, namely the underlying authenticated encryption schemes.

In contrast, most existing work has traditionally focused on security properties of encryption schemes. Examples include indistinguishability notions for asymmetric encryption schemes pioneered in [13] and adapted to symmetric-key settings in [3], non-malleability notions defined in [10, 3] and refined in [7], and integrity notions defined in [15, 5, 16]. Due to the simplicity and ease of use of these definitions, this approach has proved fruitful and has become the standard way to prove security of encryption schemes.

Our work uses this traditional approach to investigate security properties of the authenticated encryption schemes underlying channel protocols. In particular, our goal is to address the following question. Suppose one takes a "secure" KE protocol and combines it with an authenticated encryption scheme as described above to obtain a channel protocol. What are the necessary and sufficient conditions on the underlying authenticated encryption scheme for the resulting channel protocol to be a secure channel per [8]? The answer to this question will allow us to analyze security of channel protocols in a modular fashion: first consider the underlying KE protocol and the underlying authenticated encryption scheme separately, then determine whether the former is "secure" and whether the latter meets the necessary and sufficient conditions. If both are affirmative, then the channel protocol in question is a secure channel. Not only does this approach simplify protocol analysis, but the necessary and sufficient conditions also help distill exactly the security properties of authenticated encryption schemes that are needed to obtain secure channels. This understanding can help guide cryptographers in designing future schemes for building secure channels.

Krawczyk has already made some progress in this direction in [16]: he provides a necessary condition for a class of authenticated encryption schemes, namely those constructed via the "Authenticate-then-Encrypt" method,[1] to yield a secure channel, assuming that the underlying KE protocol is "secure." Our goal is to provide *both* necessary *and* sufficient conditions that are easy-to-use and can be applied to *any* authenticated encryption schemes, as opposed to schemes of a certain form. To this end, we use the traditional approach of defining security since it yields definitions that are simple and relatively easy to use.

[1]Under this paradigm, a message authentication scheme and an encryption scheme are composed to obtain an authenticated encryption scheme as follows. To encrypt a message M, first compute its MAC via a message authentication scheme and encrypt the concatenation of M and the MAC to obtain the ciphertext to be transmitted. Decryption works in a natural way.

Security model of Canetti and Krawczyk. In [8], Canetti and Krawczyk use the adversarial model of [2]: an adversary is in control of all message delivery and the execution of the protocol. In particular, once the setup phase of the protocol is completed, all parties in the system simply wait for *activations* from the adversary. Possible activations include sending messages, receiving messages, and establishing a session. Messages are delivered solely by the adversary under either of the following models: the Authenticated-links Model (AM) and the Unauthenticated-links Model (UM). Both models allow the adversary to drop messages and to deliver them out of order. In the former, an adversary cannot inject messages and must deliver messages without modifications. In the latter, it can inject fabricated messages and modify messages before delivering them. Section 2.1 describes the security model of [8] in more detail.

Canetti and Krawczyk also present a security definition for KE protocols based on the approach of [6] in this adversarial model. Intuitively, they consider a KE protocol to be secure if, when the two parties involved in the exchange complete the protocol, (1) they arrive at the same session key, and (2) it is hard for an adversary to distinguish the session key from a random value chosen from the distribution of keys generated by the protocol.

Secure channels. Canetti and Krawczyk define a secure channel as a channel protocol that is both a *secure (network) authentication* protocol and a *secure (network) encryption* protocol. The definition of the former uses a simulation-based approach: a protocol secure in this sense must *emulate* ideal message transmissions where the notion of emulation amounts to computational indistinguishability of protocol outputs. To this end, [8] defines a *session-based message transmission* (SMT) protocol, a protocol that does nothing more than its name suggests. For example, to establish a session, a party simply records in its output that a session has been established. To send a message, a party simply puts the message in the message buffer and records in its output that the message has been sent.

The definition of secure encryption protocols applies an indistinguishability-based approach similar to the "find-then-guess" game in [3] (which in turn is an adaptation of *semantic security* of [13] into the symmetric setting) in this adversarial model. Specifically, the protocol is run in the UM against an adversary which, at some point during the run, chooses a session it wishes to break. The rest of the run closely follows the standard find-then-guess game with a few important exceptions. See Section 2.2 for details.

Capturing the essence of secure channels. Following [8], we define a *transform* to specify how the channel protocols considered in this paper are generated: given a KE protocol π and an authenticated encryption scheme $\mathcal{AE}$, we associate with them a channel protocol $\mathsf{NC} = \mathsf{NetAE}(\pi, \mathcal{AE})$ obtained by applying the transform to π and $\mathcal{AE}$. This transform is defined in Section 2.3. We focus on protocols constructed via this transform. Our goal is to find simple necessary and sufficient conditions on the underlying authenticated encryption scheme such that the protocol is a secure channel, assuming that the KE protocol is secure. We define two simple notions: SINT-PTXT and IND-CCVA. The former (resp. the latter) is a necessary and sufficient condition on the underlying authenticated encryption scheme such that the channel protocol is a secure authentication (resp. encryption) protocol. In effect, this reduces the secure channel requirements of Canetti and Krawczyk to easier to use, stand-alone security requirements on the underlying authenticated encryption scheme.

We define the two notions using the traditional approach: we give an adversary access to certain oracles, run it in an experiment, and then measure the probability that it succeeds. Section 3 describes these notions in detail. Precise statements of our main results are presented in Section 4 along with the proof ideas.

Technical issue. The notion of secure authentication protocols captures reasonable authenticity guarantees such as resistance against replay attacks and forgeries. Therefore, to determine if a channel protocol provides authenticity when these attacks are of concern, it suffices to simply determine whether the protocol is a secure authentication protocol. However, due to a technical issue arisen from the notion of secure encryption protocol per [8], the same cannot be said regarding privacy. In particular, there exists a channel protocol that clearly does not provide semantic security [13] (i.e., partial information about transmitted messages may be leaked) and yet *is* provably a secure encryption protocol. Arguably, however, this technical issue does not arise in many practical protocols, including the popular SSH, SSL, and TLS. Consequently, the notion of secure encryption protocol can still be applied to these protocols to obtain meaningful results regarding their privacy guarantees. Section 5 discusses this issue in more detail.

Future work. Canetti and Krawczyk have recently proposed an alternative notion for secure channels that implies their secure channel notion of [8]. This new notion is called *universally composable secure channels* [9]. It provides strong composability guarantees, which means that its security guarantees hold even if the channel protocol is used in combination with other protocols. Thus, a natural research direction is to determine whether we can use the same approach taken here to derive simple necessary and sufficient conditions for an authenticated encryption scheme to yield a universally composable secure channel.

2 Definitions

2.1 Preliminaries

Since the authenticated encryption schemes considered in [8] have stateful decryption algorithms, we modify the standard syntax of symmetric authenticated encryption schemes, which assumes that decryption algorithms are stateless [3], to allow for stateful decryption algorithms. We also explicitly specify the syntax of a message-driven protocol based on [2, 8] and restate the security model of [8] in more detail here.

Syntax of (symmetric) authenticated encryption schemes. A (symmetric) authenticated encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. The randomized *key generation* algorithm $\mathcal{K}$ takes as input a security parameter $k \in \mathsf{N}$ and returns a key K; we write $K \stackrel{R}{\leftarrow} \mathcal{K}(k)$. The *encryption* algorithm $\mathcal{E}$ could be randomized or stateful. It takes the key K and a *plaintext* M to return a *ciphertext* C; we write $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$. The *decryption* algorithm $\mathcal{D}$ could be deterministic, and it could be either stateless or stateful. It takes the key K and a string C to return either the corresponding plaintext M or the symbol $\perp$; we write $x \leftarrow \mathcal{D}_K(C)$ where $x \in \{0,1\}^* \cup \{\perp\}$. Above, a randomized algorithm flips coins anew on each invocation, and a stateful algorithm uses and then updates a state that is maintained across invocations.

Since the decryption algorithm is allowed to be stateful here, the usual correctness condition, which requires that $\mathcal{D}_K(\mathcal{E}_K(M)) = M$ for all M in the message space, is replaced with a less stringent condition requiring only that decryption succeed when the encryption and decryption processes are in synchrony. More precisely, the following must be true for any key K and plaintexts $M_1, M_2, \ldots$. Suppose that both $\mathcal{E}_K$ and $\mathcal{D}_K$ are in their initial states. For $i = 1, 2, \ldots$, let $C_i = \mathcal{E}_K(M_i)$ and let $M_i' = \mathcal{D}_K(C_i)$. It must be that $M_i = M_i'$ for all i.

Syntax of message-driven protocols. A message-driven protocol $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r,$ $\mathsf{activation\ list})$ consists of three algorithms, four positive integer parameters, and a list of *activations* that can be invoked on a party along with instructions on how the party should handle them. Let $k \in \mathsf{N}$ be the security parameter. The parameter n specifies the upper bound of the number of parties in the system. The randomized *input generation* algorithm $\mathcal{IG}$ takes as inputs k and an x-bit random string and returns n strings $(x_1, \ldots, x_n)$. The randomized *bootstrapping* algorithm[2] $\mathcal{B}$ takes as inputs k and an l-bit random string and returns $n+1$ strings $(I_0, \ldots, I_n)$. For each party P_i, the possibly randomized *initialization* algorithm $\mathcal{I}$ takes as inputs I_0, I_i, x_i, and an r-bit random string. This may cause the party to update its *internal state*, to generate outputs to be appended to its *local output*, and/or to produce messages to be sent to other parties.

MESSAGE-DRIVEN PROTOCOL EXECUTION [8]. Let $k \in \mathsf{N}$ be the security parameter. A protocol $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \textsf{activation list})$ is executed against an adversary as follows. First, random coins for $\mathcal{IG}, \mathcal{B}$, and $\mathcal{I}$ are generated, and $\mathcal{IG}$ and $\mathcal{B}$ are executed. Then, each party P_i executes the initialization algorithm $\mathcal{I}$ giving it appropriate inputs as described above. When the initialization algorithm completes, the party waits for incoming activations. Finally, the adversary is run using k, I_0, and as many random coins as it needs. The adversary takes over and activates any parties it wishes to at this point.

Upon receiving an activation, a party executes the corresponding algorithm as specified in activation list. Again, the result of the execution may be internal state updates, local output generation, and/or *outgoing messages*. In the last case, the party appends the message in the *message buffer* $\mathcal{M}$ along with its source, destination, and, in the case of a session-based protocol, the associated session. As an example, upon receiving a "send" activation from the adversary, a party finds the algorithm for handling a send activation in its activation list and executes the algorithm. This typically involves encrypting the message, appending the ciphertext (along with its source, destination, and session ID) to $\mathcal{M}$, and recording the event (e.g., a record to the effect "`sent` M `to` P `within session` s") in the party's local output.

PROTOCOL OUTPUT. The output of a running protocol is the concatenation of the cumulative local outputs of all the parties, together with the output of the adversary. Furthermore, since all actions of the adversary are recorded in the local outputs, they are part of the protocol output.

SESSION-BASED MESSAGE-DRIVEN PROTOCOLS [8]. A *session-based message-driven* protocol defines at least two activations: establish-session and expire-session. They specify how each party can establish a *session* between itself and another. We denote by (P, P', s) a session defined by the initiating party P, the responding party P', and the session ID s. The two parties P and P' are said to play the *roles* of an initiator and a responder, respectively. Two identical sessions (i.e., identical session IDs, participating parties, and their respective roles) from the point of view of the initiator and the responder are called *matching* sessions. The defining feature of session-based protocols is that individual sessions are maintained separately from one another even if they are established between the same pair of parties.

KEY-EXCHANGE PROTOCOLS. A *key-exchange (KE)* protocol is a session-based message-driven protocol that specifies how two parties can establish a shared *session key* to be used during a session. Upon an establish-session activation, a party triggers a sub-protocol to establish a session with another party. This sub-protocol will likely result in further activations such as message sends and receipts. Once the sub-protocol completes, the two parties write on their outputs the resulting session key and mark the entry as "secret." Note that, although potentially confusing, the term "key-exchange protocol" is commonly used in the literature to refer to this sub-protocol rather than the entire protocol. Upon an expire-session activation of a particular session, the party erases the corresponding session key from its output and any internal state it may have (e.g., its memory) and terminate the session. Notice that this means that a session can be unilaterally expired.

[2]Known as an initialization function in [2, 8]. We drop their terminology here to avoid confusion with the initialization algorithm.

Network channel protocols. A *network channel* protocol (or a channel protocol for short) is a session-based message-driven protocol with two additional activations: send and incoming. They specify what a party running the protocol should do to send and receive a message.

Power of an adversary. When interacting with parties executing a session-based message-driven protocol, an adversary is allowed to access the contents of each party's local output except those marked as "secret." It can also perform the following *actions*: party activation, party corruption, session-state reveal, and session-output reveal. In addition to these actions, an adversary against a KE protocol can also perform a session-key reveal action against a party to obtain a session key. A session is considered *exposed* if it belongs to a corrupted party, has been subjected to a session-state reveal or a session-output reveal, or has a matching session that has been exposed. For completeness, we include a detailed description of these actions in Appendix A.

Authenticated and unauthenticated links models. In the Authenticated-links Model (AM), the adversary can perform all of the actions mentioned above. Furthermore, all message delivery is performed by A: to deliver a message in the message buffer $\mathcal{M}$, the adversary A removes it from $\mathcal{M}$ and activates the receiving party with the message as an incoming message. We emphasize that A can deliver messages in any arbitrary order and can drop messages from $\mathcal{M}$ entirely. However, it cannot deliver messages that are not in $\mathcal{M}$, and when it does deliver a message, it must do so without any modifications to the message. On the other hand, in the Unauthenticated-links Model (UM), not only can a UM adversary perform all of the actions permitted to an AM adversary, but it can also deliver messages that are not in $\mathcal{M}$ or modify messages in $\mathcal{M}$ before delivering them.

Notation. We use $|r|$ to denote the length in bits of a string r. Let $k \in \mathsf{N}$ be the security parameter, and let U be an adversary. Let $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$ be a session-based message-driven protocol. We follow the notation of [2, 8] for the protocol output. We describe it here in detail for the UM. The AM is done similarly except that the bootstrapping algorithm is ignored and its outputs are omitted. We denote by $\text{UNADV}_{\pi,U}(k, \vec{x}, \vec{r})$ the output of the UM adversary U running against parties executing the protocol π with security parameter k, inputs $\vec{x} = (x_1, \ldots, x_n)$, and coins $\vec{r} = r', r'', r_0, \ldots, r_n$ where $|r'| = x, |r''| = l$, and $|r_0| = \ldots = |r_n| = r$. We denote by $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_i$ the cumulative output of the party P_i running the protocol π with security parameter k, inputs $\vec{x}$, and coins $\vec{r}$ against the UM adversary U. Then, we let $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r}) = \text{UNADV}_{\pi,U}(k, \vec{x}, \vec{r}), \text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_1, \ldots, \text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_n$ and let $\text{UNAUTH}_{\pi,U}(k)$ be the random variable describing $\text{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})$ when $\vec{r}$ is randomly chosen and $\vec{x}$ is generated via $\mathcal{IG}(k, r')$. We denote by $\text{UNAUTH}_{\pi,U}$ the ensemble $\{\text{UNAUTH}_{\pi,U}(k)\}_{k \in N}$.

2.2 Secure Channels per Canetti and Krawczyk [8]

In [8], Canetti and Krawczyk define a secure channel as a channel protocol that is both a (secure) *authentication protocol* and a (secure) *encryption protocol*. For authentication protocols, their approach is to first define a protocol considered ideal as a message authentication protocol called the *SMT* protocol. A channel protocol is considered a secure authentication protocol if it emulates the SMT protocol in the UM. Below, we present the concept of protocol emulation, the SMT protocol, and the definition of secure authentication protocols in Definition 2.1, Construction 2.2, and Definition 2.3, respectively.

Definition 2.1 **[Protocol Emulation** [8]**]** Let π, π' be message-driven protocols. We say that π' *emulates* π in the UM if, for any UM adversary U, there exists an AM adversary A such that $\text{AUTH}_{\pi,A}$ and $\text{UNAUTH}_{\pi',U}$ are computationally indistinguishable. ▮

Construction 2.2 [**SMT Protocol** [8]] The protocol SMT is a session-based message-driven protocol with the following activations: establish-session, expire-session, send, and incoming. Upon an establish-session activation, a party records the event accordingly in its output. Upon an expire-session activation, a party checks that the session exists, marks the session as expired, and records the event accordingly in its output. When a party receives a send activation involving a message, a partner, and a session ID, it checks that the session is established and is not expired. If so, it sends the given message to its partner via the specified session. Then, it records the event accordingly in its output. Finally, upon an incoming activation, a party checks that the session is established and is not expired. If so, it records the event accordingly in its output. ▌

Definition 2.3 [**Network Authentication Protocol Security** [8]] A protocol is considered to be a *secure authentication protocol* if it emulates the SMT protocol in the UM. ▌

In defining secure encryption protocols, [8] adapts the indistinguishability-based approach to a multi-party computation setting. We present their security definition here. In what follows, the activation $\mathsf{send}^*(P, Q, s, M_b)$ has the same effects as $\mathsf{send}(P, Q, s, M_b)$ except that the party Q merely records the fact that a message is sent but not the actual contents of the message, i.e., P records the entry "`sent a message to` Q `within session` s". Similarly, the activation $\mathsf{incoming}^*(Q, P, s, C, M_b)$ has the same effects as $\mathsf{incoming}(Q, P, s, C)$ except that, if the decrypted message of C is equal to M_b, then Q merely records the fact that a message is received but not the actual contents of the message M_b, i.e., Q records the entry "`received a message from` P `within session` s". For completeness, these two activations are defined in detail in Appendix B.

Let b be a bit. In the experiment below, an adversary U runs in the UM, and its goal is to break one session of its choice by performing an action called test-session against the session and then doing what it can to guess the bit b. Once U picks a session, say (P, Q, s), it outputs a pair of messages, say (M_0, M_1). The sender P is then activated to send M_b. However, if P records in its local output at this point that it sends M_b, then U can easily win the game by simply looking at P's output. Therefore, P is activated with $\mathsf{send}^*(P, Q, s, M_b)$, rather than a regular send activation. The rest of the run continues in the same way as before except that now the receiving party of the tested session uses $\mathsf{incoming}^*(Q, P, s, C, M_b)$ to handle incoming messages. The reason for this is the following: if Q records all decryptions of incoming ciphertexts, U can easily determine the bit b by simply taking the challenge ciphertext corresponding to M_b, handing it to Q as an incoming ciphertext, then seeing what Q writes on its output. The activation $\mathsf{incoming}^*$ prevents this trivial attack.

Unfortunately, the game in its present form allows U to easily win via another trivial attack. Suppose the tested session is (P, Q, s). First, U picks any message M, activates P with a send activation to send M to Q via s, and outputs the challenge message pair (M, M') where $M \neq M'$. As a result of the send activation, P encrypts M to obtain a ciphertext C and appends C to the message buffer. Now, U activates the receiver Q with the ciphertext C as an incoming message from P via session s. If Q does not record the decrypted message, then C corresponds to M, and thus $b = 0$. Otherwise, C corresponds to M', and thus $b = 1$. Therefore, to prevent this trivial attack, [8] requires that an adversary never ask for an encryption of a particular message more than once. Definition 2.4 below describes the security of network encryption protocols more precisely.

Definition 2.4 [**Network Encryption Protocol Security** [8]] Let $k \in \mathsf{N}$. Let $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \text{activation list})$ be a channel protocol. Let U be a UM attacker, and let $r_U\colon \mathsf{N} \to \mathsf{N}$ be the function specifying the upper bound of the running time of U in terms of k. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{ind-ne-}b}_{\mathsf{NC},U}(k)$

$r' \xleftarrow{R} \{0,1\}^x$; $r'' \xleftarrow{R} \{0,1\}^l$; $r_0 \xleftarrow{R} \{0,1\}^{r_U(k)}$; $(x_1,\ldots,x_n) \leftarrow \mathcal{IG}(k,r')$; $(I_0,\ldots I_n) \leftarrow \mathcal{B}(k,r'')$

For $i = 1,\ldots,n$ do $r_i \xleftarrow{R} \{0,1\}^r$; start P_i on (I_0, I_i, x_i, r_i)

Run U on input (k, I_0, r_0), carrying out U's actions as specified in NC

▷ When U submits $\mathsf{test\text{-}session}(P_i, P_j, s_0)$ and outputs (M_0, M_1)

— Activate P_i with $\mathsf{send}^*(P_i, P_j, s_0, M_b)$

▷ Continue carrying out U's actions as specified in NC *except*

— Whenever U activates P_j with $\mathsf{incoming}(P_j, P_i, s_0, C)$,

Activate P_j with $\mathsf{incoming}^*(P_j, P_i, s_0, C, M_b)$ instead

Until U halts and outputs a bit d

Output d

Above, we require that U submit only one test-session query and that it not expose the tested session thereafter. Furthermore, for the tested session, we require that U never invoke send activations involving M_0 or M_1 and also never invoke send activations involving a particular message more than once. We define the advantage of the adversary via

$$\mathbf{Adv}^{\text{ind-ne}}_{\mathsf{NC},U}(k) = \Pr[\,\mathbf{Exp}^{\text{ind-ne-}1}_{\mathsf{NC},U}(k) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-ne-}0}_{\mathsf{NC},U}(k) = 1\,]\ .$$

The channel protocol NC is said to be a *secure encryption protocol* in the UM if the function $\mathbf{Adv}^{\text{ind-ne}}_{\mathsf{NC},U}(\cdot)$ is negligible for any UM adversary U whose time-complexity is polynomial in k. ▌

2.3 From KE and Authenticated Encryption Schemes to Channel Protocols

In [8], Canetti and Krawczyk use a template by which one can describe how a KE protocol and an authenticated encryption scheme can be used as building blocks for a channel protocol. We define a transform based on this template.

Construction 2.5 [**Transform** [8]] Let $\pi = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \mathsf{activation\ list})$ be a KE protocol, and let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme. We associate with π and $\mathcal{AE}$ a channel protocol $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE}) = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \mathsf{alist})$ where alist contains the activations in activation list together with the following activations.

1. $\mathsf{establish\text{-}session}(P_i, P_j, s, \mathit{role})$: This triggers a KE-session under π within P_i with partner P_j, session ID s, and $\mathit{role} \in \{\mathsf{initiator}, \mathsf{responder}\}$. If the KE-session completes, P_i records in its local output the entry "`established session` s `with` P_j" and the generated session key marked as "secret." Otherwise, no action is taken.
2. $\mathsf{expire\text{-}session}(P_i, P_j, s)$: If the session (P_i, P_j, s) exists at P_i, the party P_i marks the session as expired and erases the session key. Then, P_i records in its local output "`expired session` s `with` P_j". Otherwise, no action is taken.
3. $\mathsf{send}(P_i, P_j, s, M)$: The party P_i checks that the session (P_i, P_j, s) has been completed and not expired. If so, it computes $C \xleftarrow{R} \mathcal{E}_K(M)$ using the corresponding session key K, puts (P_i, P_j, s, C) in the message buffer $\mathcal{M}$, and records "`sent` M `to` P_j `within session` s" in the local output. Otherwise, no action is taken.
4. $\mathsf{incoming}(P_j, P_i, s, C)$: The party P_j checks that the session (P_i, P_j, s) has been completed and not expired. If so, it computes $M \leftarrow \mathcal{D}_K(C)$ under the corresponding session key K. If $M \neq \perp$, then P_j records "`received` M `from` P_i `within session` s". Otherwise, no action is taken. ▌

3 Simple Characterizations of Authenticated Encryption Schemes for Secure Channels

We propose two new security notions for authenticated encryption schemes: SINT-PTXT and IND-CCVA. The goal is to capture the necessary and sufficient properties of the authenticated encryption scheme such that, once the transform per Construction 2.5 is applied to the scheme and a KE protocol, the resulting channel protocol is a secure channel, assuming that the KE protocol "securely implements" the key generation algorithm of the authenticated encryption scheme. We postpone a precise definition of the term in quotes to Section 4. In what follows, we use $x \stackrel{R}{\leftarrow} f(y)$ to denote the process of running a possibly randomized algorithm f on an input y and assigning the result to x. If A is a program, $A \Leftarrow x$ means "return x to A." The *time-complexity* referred to in our definitions is the worst case total execution time of the entire experiment, plus the size of the code of the adversary, in some fixed RAM model of computation. Also, oracles corresponding to stateful algorithms maintain their states across invocations.

First, we capture the notion of a secure authentication protocol with SINT-PTXT. Recall that a protocol is considered a secure authentication protocol if it emulates the SMT protocol in the UM where SMT is an ideal session-based message transmission protocol. Under the SMT protocol in the AM, when a party sends a message M to another party, the message M is simply put on the buffer. Since the adversary is operating in the AM, it can drop messages but cannot modify or inject messages. Therefore, a secure authentication protocol must ensure that each sent message is received *at most once* (i.e., replay attacks are unsuccessful), and that its contents are left intact.

We define the SINT-PTXT notion in Definition 3.1. An adversary is given access to an encryption oracle and a decryption oracle. This captures its ability to obtain encryption and decryption of messages and ciphertexts of its choice. We use a multiset, denoted T below, to keep track of messages that have been sent but not yet received. Whenever a message is received, it is removed from the multiset. If an adversary is able to submit a query to the decryption oracle that results in a message that is not in the multiset T, i.e., the message is not one of those waiting to be received, then it wins.

Definition 3.1 [**SINT-PTXT**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme. Let $k \in \mathsf{N}$. Let A be an adversary with access to two oracles. Consider the following experiment.

Experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},A}(k)$
    $K \stackrel{R}{\leftarrow} \mathcal{K}(k)$; $T \leftarrow \emptyset$ // T is a multiset
    Run $A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}(k)$
        Reply to $\mathcal{E}_K(M)$ as follows:
            $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$; $T \leftarrow T \cup \{M\}$; $A \Leftarrow C$
        Reply to $\mathcal{D}_K(C)$ as follows:
            $M \leftarrow \mathcal{D}_K(C)$
            `If` $M = \perp$ `Then` $A \Leftarrow M$
            `Else` `If` $M \in T$ `Then` $T \leftarrow T - \{M\}$; $A \Leftarrow M$
                `Else` return 1
    Until A halts
    Return 0

We define the *advantage* of the adversary via

$$\mathbf{Adv}^{\text{sint-ptxt}}_{\mathcal{AE},A}(k) = \Pr[\,\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},A}(k) = 1\,] \,.$$

The scheme $\mathcal{AE}$ is said to be *SINT-PTXT secure* if the function $\mathbf{Adv}^{\text{sint-ptxt}}_{\mathcal{AE},A}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in k. ▮

Now, we capture the notion of a secure encryption protocol. To capture an adversary's ability to obtain encryption and decryption of messages and ciphertexts of its choice, we give it access to an encryption oracle $\mathcal{E}_K(\cdot)$ and a decryption oracle $\mathcal{D}_K(\cdot)$. The definition follows that of [8] closely and straightforwardly. Let $b \in \{0,1\}$. Recall that, in the definition of secure encryption protocol per [8], once the adversary outputs a challenge message pair (M_0, M_1), the receiver of the tested session does not record the decrypted message if it is equal to the secret message M_b. Therefore, we capture this through an oracle denoted by $\mathcal{D}_K(\cdot, M_b)$. This oracle is the same as the standard decryption oracle $\mathcal{D}_K(\cdot)$ except the following. If a given ciphertext decrypts to M_b, then the oracle $\mathcal{D}_K(\cdot, M_b)$ returns a special symbol $\pm$. Otherwise, it returns the decrypted message. Additionally, since an adversary in the definition per [8] cannot obtain encryptions of a particular message more than once, we also impose the same restriction on the adversary in our experiment.

Definition 3.2 [**IND-CCVA**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme. Let $b \in \{0,1\}$ and $k \in \mathsf{N}$. Let A be an adversary that has access to three oracles. Consider the following experiment.

Experiment $\mathbf{Exp}^{\text{ind-ccva-}b}_{\mathcal{AE},A}(k)$
$K \stackrel{R}{\leftarrow} \mathcal{K}(k)$
$(M_0, M_1, st) \leftarrow A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}(k, \mathsf{find})$; $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M_b)$; $d \leftarrow A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot,M_b)}(k, \mathsf{guess}, C, st)$
Return d

The computation $\mathcal{E}_K(M_b)$ above is a call to the encryption oracle. Also, the oracle $\mathcal{D}_K(\cdot, M_b)$ shares states with (i.e., is initialized with the current states of) $\mathcal{D}_K(\cdot)$ if any. Furthermore, we require that A never query $\mathcal{E}_K(\cdot)$ on M_0 or M_1 and also never query $\mathcal{E}_K(\cdot)$ on a particular message more than once. We define the *advantage* of the adversary via

$$\mathbf{Adv}^{\text{ind-ccva}}_{\mathcal{AE},A}(k) = \Pr[\,\mathbf{Exp}^{\text{ind-ccva-1}}_{\mathcal{AE},A}(k) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-ccva-0}}_{\mathcal{AE},A}(k) = 1\,] \,.$$

The scheme $\mathcal{AE}$ is said to be *IND-CCVA secure* if the function $\mathbf{Adv}^{\text{ind-ccva}}_{\mathcal{AE},A}(\cdot)$ is negligible for any adversary A whose time-complexity is polynomial in k. ▮

4 SINT-PTXT and IND-CCVA are Necessary and Sufficient

Our results uses Definition 4.1 below. It describes how a key generation algorithm of an authenticated encryption scheme should relate to a KE protocol of a channel protocol based on the authenticated encryption scheme. In particular, the KE protocol should "implement" the key generation algorithm, meaning that two parties that have completed the KE protocol with each other should end up with the same key which in turn should be drawn from the distribution generated by the key generation algorithm. The definition, which is adapted from [8], captures this property more precisely via the following game. Let $k \in \mathsf{N}$ be the security parameter. Let Π be a session-based message-driven protocol that includes a KE protocol π as a sub-protocol, and let U be a UM adversary running against Π. The adversary U can carry out actions specified in Π plus one additional activation, namely a test-session-key query, against at most one unexpired and unexposed session s whose KE portion is completed. Once U perform a test-session-key query, a bit b is chosen at random. If $b = 0$, then U receives the session key for s. Otherwise, it receives a value $r \stackrel{R}{\leftarrow} \mathcal{K}(k)$. The adversary wins if it correctly guesses the bit b.

Definition 4.1 [**Securely Implementing a Key Generation Algorithm via a Key Exchange Protocol.**] Let $k \in \mathsf{N}$ be the security parameter. A KE protocol π is said to *securely implement* a key generation algorithm $\mathcal{K}$ in the UM during the run of a protocol if, for any adversary U in the UM,

— When an uncorrupted party completes π with another uncorrupted party, they both arrive at the same session key, AND

— U wins the game above with probability no more than 1/2 plus a negligible function of k. ▌

We present our results here. They state that, respectively, SINT-PTXT and IND-CCVA are necessary and sufficient for the notions of network authentication and network encryption of Canetti and Krawczyk [8]. We present the theorems and their proof ideas below. The full proofs in detail are in Appendix C. For brevity, we write $X \overset{s}{\approx} Y$ when the ensembles X and Y are *statistically indistinguishable.*

Theorem 4.2 [**Given a secure KE, SINT-PTXT $\Leftrightarrow$ Secure Authentication Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. Then, $\mathcal{AE}$ is SINT-PTXT secure *if and only if* NAE is a secure authentication protocol. ▌

We sketch the proof for each direction of the "if and only if," assuming throughout that π securely implements $\mathcal{K}$. For the "if" direction, we show that if $\mathcal{AE}$ is SINT-PTXT, then given any UM adversary U against NAE, we can construct an AM adversary A against SMT such that $\mathrm{AUTH}_{\mathsf{SMT},A} \overset{s}{\approx} \mathrm{UNAUTH}_{\mathsf{NAE},U}$. The crux of this proof is essentially the same as that of Theorem 12 of [8], and thus, we do not discuss it further.

For the "only if" direction, we show that, given any sint-ptxt adversary F against $\mathcal{AE}$, we can construct a UM adversary U against NAE such that, for any AM adversary A against SMT, $\mathrm{AUTH}_{\mathsf{SMT},A} \overset{s}{\not\approx} \mathrm{UNAUTH}_{\mathsf{NAE},U}$ as follows. The adversary U starts two parties P_1 and P_2. Then, it activates P_1 with $\mathsf{establish\text{-}session}(P_1, P_2, s)$ and runs F. Whenever F submits an encryption query $\mathcal{E}_K(M)$, the adversary U activates the party P_1 with $\mathsf{send}(P_1, P_2, M, s)$. Similarly, whenever F submits a decryption query $\mathcal{D}_K(C)$, the adversary U activates the party P_2 with $\mathsf{incoming}(P_2, P_1, C, s)$. Recall that a successful adversary F can essentially replay a message or forge a ciphertext the decrypts to a previously-unseen message. Since such actions are not allowed in the AM, there can be no AM adversaries that can generate the global output that is statistically indistinguishable from that generated by U.

Theorem 4.3 [**Given a secure KE, IND-CCVA $\Leftrightarrow$ Secure Encryption Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. Then, $\mathcal{AE}$ is IND-CCVA secure *if and only if* NAE is a secure encryption protocol. ▌

We sketch the proof for each direction of the "if and only if," assuming throughout that π securely implements $\mathcal{K}$. For the "if" direction, we show that, given any ind-ne adversary U against NAE, we can construct an ind-ccva adversary A against $\mathcal{AE}$ such that A's success probability is no less than that of U divided by the total number of sessions established by U over its run. The adversary A simply simulates U as in the experiment $\mathbf{Exp}^{\text{ind-ne-}b}_{\mathsf{NAE},U}(k)$ (where b is a bit) with one exception: during the find phase, A chooses a session at random and uses its oracles to encrypt and decrypt messages in this session. If U submits a test-session query on the chosen session and outputs a pair of test messages, A does too. (Otherwise, A aborts.) Then, A enters its guess phase and continues the simulation exactly as before. It halts and outputs what U outputs. Since π securely implements $\mathcal{K}$, the adversary A correctly simulates U. Thus, it succeeds if U does.

For the "only if" direction, we show that, given any ind-ccva adversary A against $\mathcal{AE}$, we can construct an ind-ne adversary U against NAE such that U's success probability is no less than that of U using a similar technique as before: U establishes a session between two parties, then runs A, answering its encryption and decryption queries by making send and incoming activations respectively for the session. Finally, U halts and outputs what A outputs. Since π securely implements $\mathcal{K}$, the adversary U correctly simulates A. Thus, it succeeds if A does.

5 Understanding Secure Channels through SINT-PTXT and IND-CCVA

We explore the strengths and weaknesses of the new notions and the combination of both of them. Since the two notions are necessary and sufficient for secure channels, the strengths and weaknesses of their combination bear direct implications to those of secure channels. Here, we say that a notion A implies a notion B if any scheme secure under A is secure under B. The proofs of the results in this section are simple and are omitted.

5.1 SINT-PTXT is Reasonably Strong

We compare SINT-PTXT to existing integrity notions, namely integrity of plaintexts INT-PTXT [5] and integrity of ciphertexts INT-CTXT [15, 5]. We briefly describe the two notions here. An adversary attacking a scheme under these notions is given access to two oracles: a standard encryption oracle and a verification oracle— an oracle that returns a bit indicating whether the given ciphertext is valid, i.e., whether it decrypts to $\perp$. An adversary succeeds in breaking a scheme under the INT-PTXT notion if it can forge a ciphertext that decrypts to a "new" message, i.e., a message that has not been submitted to the encryption oracle before. Similarly, it succeeds in breaking a scheme under the INT-CTXT notion if it can forge a "new" and valid ciphertext, i.e., a valid ciphertext that has not been returned by the encryption oracle. It is easy to see that INT-CTXT implies INT-PTXT but not vice versa [5].

First, we argue that SINT-PTXT implies INT-PTXT. The reason is simple. If an adversary can forge a ciphertext for a message that has not been previously encrypted, i.e., it defeats INT-PTXT, it can also defeat SINT-PTXT with the same attack. However, INT-PTXT does not imply SINT-PTXT. Consider stateless schemes constructed via the encrypt-then-MAC composition[3] as defined and shown in [5] to be INT-PTXT secure if the underlying MAC and encryption schemes are secure. Being stateless, these schemes are susceptible to replay attacks and therefore are not secure under SINT-PTXT. Note that this does not contradict the result in [8] since the encrypt-then-MAC composition defined there is stateful.

Now, we argue that neither of SINT-PTXT or INT-CTXT imply the other. First, we argue that INT-CTXT does not imply SINT-PTXT. The reason is similar to before. Consider stateless schemes constructed via the encrypt-then-MAC composition as defined and shown in [5] to be INT-CTXT secure if the underlying MAC and encryption schemes are secure (the security assumption on the MAC here is stronger than in the case of INT-PTXT security above). Being stateless, however,

[3]Under this paradigm, to encrypt a message M, first encrypt M then MAC the result to obtain the ciphertext to be transmitted. Decryption works in a natural way.

they are not secure under SINT-PTXT. Now, consider a scheme secure under SINT-PTXT. It is easy to see that adding a redundant bit to every ciphertext generated via this scheme yields a scheme that is insecure under INT-CTXT (ciphertexts can now be easily forged) but is still secure under SINT-PTXT (the underlying messages are unaffected and so will still be hard to forge).

5.2 Integrity is Necessary for IND-CCVA

The definition of IND-CCVA closely resembles standard privacy notions such as the "find-then-guess" definitions originally proposed in [13] for the asymmetric setting and later adapted to the symmetric setting in [3]. However, upon closer inspection, it is not hard to see that, for a scheme to be secure under the IND-CCVA notion, it must also provide some form of integrity. Suppose toward a contradiction that there exists an authenticated encryption scheme $\mathcal{AE}$ secure under IND-CCVA but does not provide plaintext integrity. In particular, there exists an adversary A that can easily forge a ciphertext C of a message M that has not been previously encrypted, i.e., it can generate C on its own without ever submitting M to the encryption oracle. (This is the INT-PTXT notion.) Then, A can easily break $\mathcal{AE}$ under the IND-CCVA notion as follows. It forges the ciphertext C corresponding to a message M in the find stage, outputs (M, M') where $M \neq M'$ as the challenge message pair, then submit C to the decryption oracle in the guess stage. If it receives the special symbol $\pm$ as a response, then it returns 0. Otherwise, it returns 1. Thus, we have a contradiction.

5.3 IND-CCVA is Technically Weak

Recall that, in the definition of secure encryption protocols, an ind-ne adversary U is not allowed to submit send activations involving a particular message more than once for the tested session. This translates into a similar restriction for ind-ccva adversaries since IND-CCVA is necessary and sufficient for the notion of secure encryption protocols. Unfortunately, under this restriction, one can show that there exists a deterministic encryption scheme secure under IND-CCVA. An example of such a scheme is presented in Appendix D. Since it is well-known that deterministic encryption schemes are not secure under existing standard privacy notions (e.g., IND-CPA, IND-CCA, NM-CPA and NM-CCA [4, 10, 14]), this means that IND-CCVA does not imply any of the standard privacy notions. Thus, schemes proven secure under IND-CCVA are not guaranteed to be secure under the standard notions and thus are not guaranteed to provide semantic security.

It is easy to see that channel protocols constructed from deterministic encryption schemes secure under IND-CCVA do not provide semantic security either (i.e., partial information about transmitted messages may be leaked). The unfortunate implication here is that channel protocols proven secure as an encryption protocol may *not* provide semantic security. On the other hand, this is arguably a technical issue. As pointed out in [8], one can ensure uniqueness of messages by a simple use of unique message IDs. In fact, many Internet protocols in use today (e.g., SSH, SSL, and TLS) already do so: they include in every packet a sequence number maintained internally by the communicating parties [17, 12, 11].

5.4 Strength of the Combination of IND-CCVA and SINT-PTXT is Unclear

Secure channels are meant to capture both privacy and integrity. Therefore, it is natural to ask whether the combination of the two necessary and sufficient conditions implies standard privacy and integrity notions. First, we ask whether IND-CCVA $\wedge$ SINT-PTXT implies a weak integrity notion, namely INT-PTXT. Since both notions independently imply INT-PTXT as discussed above, the answer is affirmative. Now, we ask whether IND-CCVA $\wedge$ SINT-PTXT implies a weak privacy notion, namely IND-CPA. Surprisingly, the answer to this question is unclear. To prove that it does not, one may be tempted to use a stateless deterministic encryption scheme as we did previously. However, since no stateless schemes are SINT-PTXT, such a scheme cannot be used for this purpose. On the other hand, it is not at all clear how to prove that the combination of the two notions implies IND-CPA. Roughly speaking, the problem is due to the restriction that adversaries attacking a scheme under IND-CCVA cannot ask for repeated encryption of the same message. Since this restriction does not apply to IND-CPA (or any other standard privacy notions), it is hard to see how one can prove the implication using the standard black-box reduction approach.

References

[1] R. Atkinson. Security architecture for the Internet protocol. RFC 1825, 1995.

[2] M. Bellare, R. Canetti, and H. Krawczyk. Modular approach to the design and analysis of key exchange protocols. In *30th ACM STOC*, pages 419–428, New York, NY, May 23–26 1998. ACM Press.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In IEEE, editor, *38th FOCS*, pages 394–403. IEEE Computer Society Press, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, August 1998.

[5] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer-Verlag, December 2000.

[6] M. Bellare and P. Rogaway. Entity authentication and key distribution. In Y. Desmedt, editor, *CRYPTO '94*, volume 839 of *LNCS*, pages 232–249. Springer-Verlag, 1994.

[7] M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In E. Brickell, editor, *CRYPTO '99*, volume 740 of *LNCS*, pages 519–536. Springer-Verlag, August 1999.

[8] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 451–472. Springer-Verlag, 2001.

[9] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer-Verlag, 2002.

[10] C. Dwork D. Dolev and M. Naor. Non-malleable cryptography. In ACM, editor, *23rd ACM STOC*, pages 542–552, New Orleans, Louisiana, May 6–8 1991. ACM Press.

[11] T. Dierks and C. Allen. The TLS protocol: Version 1.0. RFC 2246, 1999.

[12] A. Freier, P. Karlton, and P. Kocher. The SSL protocol: Version 3.0, 1996.

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

[14] J. Katz and M. Yung. Complete characterization of security notions for probabilistic private-key encryption. In ACM, editor, *32nd ACM STOC*, pages 245–254. ACM Press, 2000.

[15] J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In B. Schneier, editor, *FSE 2000*, volume 1978 of *LNCS*, pages 284–299. Springer-Verlag, April 2000.

[16] H. Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*. Springer-Verlag, August 2001.

[17] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH transport layer protocol, 2002. Draft, available at `http://www.ietf.org/internet-drafts/draft-ietf-secsh-transport-12.txt`.

A Adversary Actions

An adversary A running against a session-based message-driven protocol can perform the first four actions below. An adversary A against a key exchange protocol can perform all of the actions below.

1. Party activation. The activation list specifies what can be activated on a party. Examples include asking a party to *send* a message to, *receive* a message from, or *establish a session* with another party. An adversary can also ask a party to *expire* an existing session. This causes the party to permanently erases all state information relevant to the session.
2. Party corruption. A obtains from a party all of its state, including its long-term secrets. The party appends to its output the entry "`corrupted`" and terminates. It generates no further output.
3. Session-state reveal. A obtains from a party the portion of its state that is "local" to the specified session. The protocol specifies what information is considered "local" to a session. This query is valid only for sessions that have *not* completed. The party appends to its output the entry "`revealed state of` (P, P', s)" where (P, P', s) is the session being revealed.
4. Session-output reveal. A obtains from a party all of its transcripts that have been created for the specified session (P, P', s) and are marked "secret." The party appends to its output the entry "`revealed output of` (P, P', s)".
5. Session-key reveal. A obtains from a party the session key for the specified session which must be completed but has not expired. The party appends to its output the entry "`revealed session key for` (P, P', s)" where (P, P', s) is the session in question.

B Description of the send* and incoming* Activations

Let NC be a network channel protocol. Let $k \in \mathsf{N}$ be the security parameter, let $b \in \{0, 1\}$, and let U be a UM attacker. The activations send^* and $\mathsf{incoming}^*$ used in the experiment $\mathbf{Exp}^{\text{ind-ne-}b}_{\mathsf{NC},U}(k)$ are defined as follows.

Activation $\mathsf{send}^*(P_i, P_j, s, M)$ at P_i

If the session (P_i, P_j, s) is expired or exposed, then return

If the key exchange protocol for the session (P_i, P_j, s) is not completed, then return
$C \leftarrow \mathcal{E}_K(M)$ where K is the session key for the session (P_i, P_j, s)
Record "sent a message to P_j within session s" on P_i's output
Put (P_i, P_j, s, C) in the message buffer $\mathcal{M}$

Activation $\mathsf{incoming}^*(P_j, P_i, s, C, M_b)$ at P_j
If the session (P_i, P_j, s) is expired, then return
If the key exchange protocol for the session (P_i, P_j, s) is not completed, then return
$M \leftarrow \mathcal{D}_K(C)$ where K is the session key for the session (P_i, P_j, s)
If $M = M_b$ then record "received a message from P_i within session s" on P_j's output
else if $M \neq \perp$ then record "received M from P_i within session s" on P_j's output

C Proofs that SINT-PTXT and IND-CCVA are Necessary and Sufficient

We state the lemmas from which Theorem 4.2 and Theorem 4.3 directly follow. Lemma C.1 and Lemma C.2 prove the former. Lemma C.3 and Lemma C.4 prove the latter. Then, we present their proofs in detail.

Lemma C.1 [**Given a secure KE, SINT-PTXT $\Rightarrow$ Secure Authentication Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. If $\mathcal{AE}$ is SINT-PTXT secure, then given any UM adversary U against NAE, we can construct an AM adversary A against SMT such that

$$\mathrm{AUTH}_{\mathsf{SMT},A} \overset{s}{\approx} \mathrm{UNAUTH}_{\mathsf{NAE},U} \ . \ \blacksquare$$

Lemma C.2 [**Given a secure KE, SINT-PTXT $\Leftarrow$ Secure Authentication Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. Then, given any sint-ptxt adversary F against $\mathcal{AE}$, we can construct a UM adversary U against NAE such that, for any AM adversary A against SMT,

$$\mathrm{AUTH}_{\mathsf{SMT},A} \overset{s}{\not\approx} \mathrm{UNAUTH}_{\mathsf{NAE},U} \ . \ \blacksquare$$

Lemma C.3 [**Given a secure KE, IND-CCVA $\Rightarrow$ Secure Encryption Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. Then, given any ind-ne adversary U against NAE, we can construct an ind-ccva adversary A against $\mathcal{AE}$ such that

$$\mathbf{Adv}^{\text{ind-ne}}_{\mathsf{NAE},U}(k) \leq S \cdot \mathbf{Adv}^{\text{ind-ccva}}_{\mathcal{AE},A}(k)$$

where U establishes at most S sessions and A's time-complexity is polynomially-related to that of U. $\blacksquare$

Lemma C.4 [**Given a secure KE, IND-CCVA $\Leftarrow$ Secure Encryption Protocol**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} =$

$\mathsf{NetAE}(\pi, \mathcal{AE})$ be the associated channel protocol as per Construction 2.5. Suppose that π securely implements $\mathcal{K}$ in the UM during the run of NAE. Then, given any ind-ccva adversary A against $\mathcal{AE}$, we can construct an ind-ne adversary U against NAE such that

$$\mathbf{Adv}^{\text{ind-ccva}}_{\mathcal{AE},A}(k) \leq \mathbf{Adv}^{\text{ind-ne}}_{\mathsf{NAE},U}(k) .$$

Furthermore, U's time-complexity is polynomially-related to that of A. ▌

C.1 Proof of Lemma C.1

Proof Idea. The crux of this proof is the same as that of Theorem 12 of [8]. Let $k \in \mathsf{N}$ and $i \in \{1, \ldots, n\}$. Given a UM adversary U, we construct an AM adversary A. We denote the parties interacting with A and U by P_i and P_i', respectively. To run U, the AM adversary A simulates the parties P_i' by carrying out all requests and activations from U by itself on P_i''s behalf and only makes requests and activations to a party P_i for events that have been recorded and events that involve corruption or exposure of a party.

Then, we show that for any security parameter $k \in \mathsf{N}$, any UM adversary U, and the AM adversary A defined above, if $\mathcal{AE}$ is SINT-PTXT secure, then the random variables $\text{AUTH}_{\mathsf{SMT},A}(k)$ and $\text{UNAUTH}_{\mathsf{NAE},U}(k)$ are statistically indistinguishable.

We do so by first arguing that, for any $k \in \mathsf{N}$, if $\text{AUTH}_{\mathsf{SMT},A}(k)$ and $\text{UNAUTH}_{\mathsf{NAE},U}(k)$ are statistically distinguishable, then a *forgery event* has occurred. Before defining a forgery event, we first describe the concept of *matching entries*. An entry in the local output of a party P_i that reads "`sent` M `to` P_j `within session` s" is said to be a *match* of an entry in the local output of a party P_j that reads "`received` M `from` P_i `within session` s". We mandate that once two entries are matched, they cannot be matched with any other entries, in which case we say that they become *unavailable*. An entry that has not been matched (i.e., is not unavailable) is considered *available*. A forgery event is an event in which the local output of a party P_j contains an entry of the form "`received` M `from` P_i `within session` s" while the local output of P_i does not contain an available matching entry. In other words, a forgery event occurs if, for some M, P_i, P_j, and s, the output of P_j contains a receipt record of M from P_i within session s and the record is available.

Then, we construct an adversary F so that, if a forgery event occurs, then F wins as follows. First, F chooses a session s at random from all sessions and uses its oracles, rather than the actual session key, to compute the messages transmitted via s. Then, F runs the UM adversary U until it halts. We argue that, if a forgery event occurs, then F wins as follows. Since the multiset T in the experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},F}(k)$ keeps track of sent messages that are yet to be received, an occurrence of a forgery event means that there exists a message M that has been received but $M \notin T$. Therefore, the adversary F will succeed in breaking the SINT-PTXT security of $\mathcal{AE}$. Since we assume that the KE protocol securely implements the key generation algorithm, this concludes the proof.

Proof Details. Given a UM adversary U, we construct A as follows. Here, $r_U(\cdot)$ specifies the upper bound on the running time of U.

Adversary $A(k, r_A)$
  $r' \xleftarrow{R} \{0,1\}^x$; $r'' \xleftarrow{R} \{0,1\}^l$; $r_0 \xleftarrow{R} \{0,1\}^{r_U(k)}$; $(x_1, \ldots, x_n) \leftarrow \mathcal{IG}(k, r')$; $(I_0, \ldots I_n) \leftarrow \mathcal{B}(k, r'')$
  For $i = 1, \ldots, n$ do $r_i \xleftarrow{R} \{0,1\}^r$; start P_i' on (I_0, I_i, x_i, r_i)
  Run U on (k, I_0, r_0), carrying out U's actions as follows:
  ▷ When U activates P_i' with $\mathsf{establish\text{-}session}(P_i', P_j', s, \mathit{role})$,
    $\mathsf{expire\text{-}session}(P_i', P_j', s)$, $\mathsf{send}(P_i', P_j', s, M)$, $\mathsf{incoming}(P_i', P_j', s, C)$,
    or any activations as part of the run of the KE protocol
    — Invoke the same activation against P_i'

— Put any resulting messages to be delivered on U's message buffer $\mathcal{M}$

▷ When U corrupts P'_i

— Corrupt P'_i and give P'_i's internal states to U

— Corrupt P_i

▷ When U exposes a session (P'_i, P'_j, s) at P'_i

— Expose the same session at P'_i and give resulting data to U

— Expose the session (P_i, P_j, s) at P_i

▷ When P'_i records "`established session` s `with` P'_j", "`expired session` s `with` P'_j", or "`sent` M `to` P'_j `within session` s"

— Activate P_i with establish-session(P_i, P_j, s), expire-session(P_i, P_j, s), or send(P_i, P_j, s, M)

▷ When P'_i records "`received` M `from` P'_j `within session` s"

— Find an available match of this entry in the local output of P'_j

— If an available match is found

Then match the two entries and activate P_i with incoming(P_i, P_j, s, M)

Else If P_j is corrupted or exposed

Then activate P_j with send(P_j, P_i, s, M) and P_i with incoming(P_i, P_j, s, M)

Else abort

Until U halts

Output what U outputs

We define the following event, make a few observations, then state and prove Claim C.6. Lemma C.1 follows directly.

Forgery Event: There exists two parties P'_i and P'_j such that at some point during the protocol execution, the local output of P'_j contains an entry "`received` M `from` P'_i `within session` s" where M is a message and s is a session ID, and this entry cannot be matched with any available entry in the local output of P'_i.

Remark C.5

1. The adversary A above simulates U exactly as in any run of U against parties running NAE.
2. The adversary A aborts if a forgery event occurs.
3. Suppose that A does not abort. Then, for each entry recorded in the local outputs of the simulated parties running NAE against U, there is an entry recorded in the local outputs of the parties running SMT against A.

Claim C.6 Let $k \in \mathsf{N}$ be the security parameter, let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an authenticated encryption scheme, and let π be a KE protocol. Let $\mathsf{NAE} = \mathsf{NetAE}(\pi, \mathcal{AE})$. Let U be a UM adversary, and let A be the AM adversary defined above. Suppose that the KE protocol π securely implements $\mathcal{K}$. If $\mathcal{AE}$ is SINT-PTXT secure, then $\mathrm{AUTH}_{\mathsf{SMT},A}(k)$ and $\mathrm{UNAUTH}_{\mathsf{NAE},U}(k)$ are statistically indistinguishable. ▌

We prove Claim C.6 by contradiction. Suppose that $\mathrm{AUTH}_{\mathsf{SMT},A}(k)$ and $\mathrm{UNAUTH}_{\mathsf{NAE},U}(k)$ are statistically distinguishable. From Remark C.5, this means that a forgery event occurs. We construct an adversary F that breaks SINT-PTXT security of $\mathcal{AE}$ with non-negligible probability. The adversary F works as follows. Here, $r_U(\cdot)$ specifies the upper bound on the running time of U.

Adversary $F^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)}$

$r' \stackrel{R}{\leftarrow} \{0,1\}^x$; $r'' \stackrel{R}{\leftarrow} \{0,1\}^l$; $r_0 \stackrel{R}{\leftarrow} \{0,1\}^{r_U(k)}$; $(x_1, \ldots, x_n) \leftarrow \mathcal{IG}(k, r')$; $(I_0, \ldots I_n) \leftarrow \mathcal{B}(k, r'')$

For $i = 1, \ldots, n$ do $r_i \xleftarrow{R} \{0,1\}^r$; start P'_i on (I_0, I_i, x_i, r_i)
Pick a session (P'_i, P'_j, s_0) at random from all sessions
Run U on (k, I_0, r_0) carrying out U's actions as specified in NAE *except*
— When U activates P'_i with $\mathsf{send}(P'_i, P'_j, s_0, M)$,
Take P'_i's code for handling a send activation
Replace execution of the encryption algorithm in the code with call to the oracle $\mathcal{E}_K(\cdot)$
Execute the resulting code
— When U activates P'_j with $\mathsf{incoming}(P'_j, P'_i, s_0, C)$,
Take P'_j's code for handling an incoming activation
Replace execution of the decryption algorithm in the code with call to the oracle $\mathcal{D}_K(\cdot)$
Execute the resulting code
Until U halts
Output what U outputs

Notice that, when F picks a session at random, it does not yet know the total number of sessions to be established. We address this by putting an upper bound on the total number of sessions using the running time of U and letting F choose a session at random. Also, recall that the KE protocol π is assumed to securely implement the key generation algorithm $\mathcal{K}$. This means that the session keys and the keys used by the oracles are drawn from the same distribution. Therefore, the probability that a forgery event occurs in a regular run of U and the probability that it occurs in F's run of U above are the same.

We argue here that, if a forgery event occurs, then the experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},F}(k)$ returns 1. First, we observe that the code of F above ensures that each send activation results in the corresponding encryption query and that each incoming activation results in the corresponding decryption query. Now, recall that in the experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},F}(k)$, whenever F submits an encryption query $\mathcal{E}_K(M)$ (or, equivalently here, whenever U activates P'_i with a send activation involving M), the message M is added to the multiset T. Furthermore, whenever F submits a decryption query $\mathcal{D}_K(C)$ (or, equivalently here, whenever U activates P'_j with an incoming activation involving C), if C decrypts to some message $M \neq \perp$, then M is removed from T. In short, whenever a message is sent, it is added to T, and whenever a message is received, it is removed from T.

If a forgery event occurs in the session (P'_i, P'_j, s_0), then we know that (1) there is a receipt record involving M, P'_i, and s in the output of P'_j but (2) it cannot be matched with any available matching send entry in the output of P'_i. The first condition implies that F will submit a query that decrypts to $M \neq \perp$ to the decryption oracle. The second condition implies that this query results in $M \notin T$. Therefore, the experiment returns 1, and F succeeds. Finally, since F chooses the session (P'_i, P'_j, s_0) from the total number of sessions which is polynomial in k, the probability that F succeeds remains non-negligible. Moreover, F runs in time polynomial in k since U does. Hence, Claim C.6 follows.

C.2 Proof of Lemma C.2

Proof Idea. Given an sint-ptxt adversary F against $\mathcal{AE}$, we construct a UM adversary U as follows. The adversary U starts two parties P_1 and P_2, activates P_1 with $\mathsf{establish\text{-}session}(P_1, P_2, s)$, then runs F. Whenever F submits an encryption query $\mathcal{E}_K(M)$, the adversary U activates the party P_1 with $\mathsf{send}(P_1, P_2, M, s)$. Similarly, whenever F submits a decryption query $\mathcal{D}_K(C)$, the adversary U activates the party P_2 with $\mathsf{incoming}(P_2, P_1, C, s)$.

Similar to the proof of Lemma C.1, our analysis involves a forgery event in the simulation. The forgery event is defined exactly as in the proof of Lemma C.1, so we do not repeat it here. Now, suppose that F wins its game, meaning that it has submitted a decryption query that results in a message $M \neq \perp$ so that $M \notin T$. Since the multiset T in the experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},F}(k)$ keeps track of sent messages that are yet to be received, this means that there exists a receipt record of a message with no available matching send record. In other words, U has caused a forgery event to occur. Now, since no two plaintext messages can encrypt to the same ciphertext, the fact that the received message has not been sent implies that no ciphertext whose decryption is the received message has been inserted into the message buffer $\mathcal{M}$ before U delivers the ciphertext to the recipient. Therefore, U has indeed activated a party with an incoming string that is not in the message buffer $\mathcal{M}$. Since such an action is not permitted in the AM and since its effect is actually recorded by a party, there can be no AM adversaries that can generate the global output that is statistically indistinguishable from that generated by U. Thus, Lemma C.2 follows.

Proof Details. Given an sint-ptxt adversary F, we construct a UM adversary U as follows.

Adversary $U(k, I_0, r_0)$
  Activate P_1 with establish-session$(P_1, P_2, s, \text{initiator})$
  Wait until the KE protocol for the session (P_1, P_2, s) is completed
  Run $F^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}(k)$
  ▷ Reply to $\mathcal{E}_K(M)$ queries as follows:
    — Activate P_1 with send(P_1, P_2, s, M)
    — Wait until an entry (P_1, P_2, s, C) is appended to the message buffer
    — Return C to F
  ▷ Reply to $\mathcal{D}_K(C)$ queries as follows:
    — Activate P_2 with incoming(P_2, P_1, s, C)
    — If P_2 records "`received` M `from` P_1 `within session` s"
      Then return M to A ; Else return $\perp$ to A
  Until F halts

Recall that the KE protocol π is assumed to securely implement the key generation algorithm $\mathcal{K}$. This means that the session key and the key used by the oracles are drawn from the same distribution. Therefore, the probability that F successfully breaks SINT-PTXT security of $\mathcal{AE}$ remains unaffected.

Now we argue that, if F succeeds, then a forgery event has occurred. We use a similar line of reasoning as in the proof of Lemma C.1. First, we observe that the code of U above ensures that each encryption query results in the corresponding send activation and that each decryption query results in the corresponding incoming activation. Now, recall that in the experiment $\mathbf{Exp}^{\text{sint-ptxt}}_{\mathcal{AE},F}(k)$, whenever F submits an encryption query $\mathcal{E}_K(M)$ or, equivalently here, whenever U activates P'_i with a send activation involving M, the message M is added to the multiset T. Furthermore, whenever F submits a decryption query $\mathcal{D}_K(C)$ or, equivalently here, whenever U activates P'_j with an incoming activation involving C, if C decrypts to some message $M \neq \perp$, then M is removed from T. In short, whenever a message is sent, it is added to T, and whenever a message is received, it is removed from T.

If F succeeds, then it has submitted a decryption query $\mathcal{D}_K(C)$ such that (1) the response $M = \mathcal{D}_K(C)$ is not equal to $\perp$ and (2) $M \notin T$. The former implies that, at some point during the protocol execution, U activates P_2 with incoming(P_2, P_1, s, C) and P_2 actually records the receipt of M. The latter implies that, at that moment, there is no matching send entry at P_1 for the receipt entry of M recorded at P_2.

Now, since no two plaintext messages can encrypt to the same ciphertext, the fact that the received message has not been sent implies that no ciphertext whose decryption is the received message has been inserted into the message buffer $\mathcal{M}$ before U delivers the ciphertext to the recipient. Therefore, U has activated P_2 with an incoming string that is not present in the buffer $\mathcal{M}$ at the time. Since such an action is not permitted in the AM and since the effect of this activation is actually recorded by P_2, there exists no AM adversaries that can generate the global output that is statistically indistinguishable from that generated by U. Thus, Lemma C.2 follows.

C.3 Proof of Lemma C.3

Given an ind-ne adversary U against NAE, we construct an ind-ccva adversary A against $\mathcal{AE}$ below. The activation $\mathsf{incoming}^{\mathcal{D}_K(\cdot,M_b)}$ is defined in a similar manner as in Appendix B except that here we use the oracle $\mathcal{D}_K(\cdot, M_b)$ to determine whether to write the decrypted message in the local output.

Activation $\mathsf{incoming}^{\mathcal{D}_K(\cdot,M_b)}(P_j, P_i, s, C)$ at P_j
    If the session (P_i, P_j, s) is expired, then return
    If the KE protocol for the session (P_i, P_j, s) is not completed, then return
    $M \leftarrow \mathcal{D}_K(C, M_b)$
    If $M = \pm$ then record "`received a message from` P_i `within session` s" on P_j's output
        else if $M \neq \perp$ then record "`received` M `from` P_i `within session` s" on P_j's output

Now, we define the adversary A as follows. Here, $r_U(\cdot)$ specifies the upper bound on the running time of U.

Adversary $A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}(k, \mathsf{find})$
    $r' \stackrel{R}{\leftarrow} \{0,1\}^x$; $r'' \stackrel{R}{\leftarrow} \{0,1\}^l$; $r_0 \stackrel{R}{\leftarrow} \{0,1\}^{r_U(k)}$; $(x_1, \ldots, x_n) \leftarrow \mathcal{IG}(k, r')$; $(I_0, \ldots I_n) \leftarrow \mathcal{B}(k, r'')$
    For $i = 1, \ldots, n$ do $r_i \stackrel{R}{\leftarrow} \{0,1\}^r$; start P_i on (I_0, I_i, x_i, r_i)
    Pick a session (P_i, P_j, s_0) at random from all sessions
    Run U on (k, I_0, r_0)
    ▷ Carry out U's actions as specified in NAE *except*
        — Whenever U activates P_i with $\mathsf{send}(P_i, P_j, s_0, M)$,
            Take P_i's code for handling a send activation
            Replace execution of the encryption algorithm in the code with call to the oracle $\mathcal{E}_K(\cdot)$
            Execute the resulting code at P_i
        — Whenever U activates P_j with $\mathsf{incoming}(P_j, P_i, s_0, M)$,
            Take P_j's code for handling an $\mathsf{incoming}$ activation
            Replace execution of the decryption algorithm in the code with call to the oracle $\mathcal{D}_K(\cdot)$
            Execute the resulting code at P_j
    Until U submits $\mathsf{test\text{-}session}(P, Q, s)$ and outputs (M_0, M_1)
    If $P \neq P_i$ or $Q \neq P_j$ or $s \neq s_0$ then abort
    $st \leftarrow (P_i, P_j, s_0) \| M_0 \| M_1 \|$ internal states of all parties $\|$ state of U
    Output (M_0, M_1, st)

Adversary $A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot,M_b)}(k, \mathsf{guess}, C, st)$
    Parse st as $(P_i, P_j, s_0) \| M_0 \| M_1 \|$ internal states of all parties $\|$ state of U
    Restart all parties and U to where they were
    ▷ If the session (P_i, P_j, s_0) is expired or exposed, then abort
    ▷ If the KE protocol for the session (P_i, P_j, s_0) is not completed, then abort.
    ▷ Record "`sent a message to` P_j `within session` s_0" on P_i's local output
    ▷ Put (P_i, P_j, s_0, C) in the message buffer

▷ Carry out U's actions as specified in NAE *except*
— Whenever U activates P_j with incoming(P_j, P_i, s_0, C),
Execute incoming$^{\mathcal{D}_K(\cdot, M_b)}(P_j, P_i, s, C)$ at P_j
Until U halts and outputs a bit d
Output d

Notice that A does not yet know the total number of sessions to be established when it picks a session at random. We address this by putting an upper bound on the total number of sessions using U's running time. It is easy to see that A simulates U exactly as in the experiment $\mathbf{Exp}^{\text{ind-ne-}b}_{\mathsf{NAE},U}(k)$ where $b \in \{0, 1\}$. We stress that this is true even though the session key for the tested session (P_i, P_j, s_0) is substituted with the key used by the oracles, the reason being that the KE protocol π securely implements the key generation algorithm $\mathcal{K}$. Therefore, if U can guess the bit b correctly, then so can A. Since there are a total of at most S sessions in the run of U, the probability that A guesses the tested session (P_i, P_j, s_0) correctly is $1/S$. Thus,

$$\frac{1}{S} \cdot \mathbf{Adv}^{\text{ind-ne}}_{\mathsf{NAE},U}(k) = \mathbf{Adv}^{\text{ind-ccva}}_{\mathcal{AE},A}(k) \ .$$

Furthermore, recall that the time-complexity of an adversary pertains to the entire experiment in which it runs. Therefore, the time-complexity of A is polynomially-related to that of U. Thus, Lemma C.3 follows.

C.4 Proof of Lemma C.4

Given an ind-ccva adversary A against $\mathcal{AE}$, we construct an ind-ne adversary U against NAE below.

Adversary $U(k, I_0, r_0)$
Activate P_1 with establish-session$(P_1, P_2, s, \text{initiator})$
Wait until the KE protocol for the session (P_1, P_2, s) is completed
Run $A^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)}(k, \text{find})$
▷ Reply to $\mathcal{E}_K(M)$ queries as follows:
— Activate P_1 with send(P_1, P_2, s, M)
— Wait until an entry (P_1, P_2, s, C) is appended to the message buffer $\mathcal{M}$
— Return C to A
▷ Reply to $\mathcal{D}_K(C)$ queries as follows:
— Activate P_2 with incoming(P_2, P_1, s, C)
— If P_2 records "`received` M `from` P_1 `within session` s"
Then return M to A ; Else return $\perp$ to A
Until A outputs (M_0, M_1, st)
Submit the query test-session(P_1, P_2, s) and output (M_0, M_1)
Wait until an entry (P_1, P_2, s, c) is appended to the message buffer $\mathcal{M}$
Run $A^{\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot, M_b)}(k, \text{guess}, c, st)$
▷ Reply to $\mathcal{E}_K(M)$ queries exactly as before
▷ Reply to $\mathcal{D}_K(C, M_b)$ queries as follows:
— Activate P_2 with incoming(P_2, P_1, s, C)
— If P_2 records "`received` M `from` P_1 `within session` s"
Then return M to A
Else If P_2 records "`received a message from` P_1 `within session` s"
Then return $\pm$ to A ; Else return $\perp$ to A
Until A stops and outputs a bit d
Output d

Since the KE protocol π securely implements the key generation algorithm $\mathcal{K}$, it is easy to see that U runs A in the same environment as the experiment $\mathbf{Exp}^{\text{ind-ccva-}b}_{\mathcal{AE},A}(k)$ where b is a bit. Therefore, if A can guess the bit b correctly, then so can U. Furthermore, time-complexity of U is polynomially-related to that of A. Thus, Lemma C.4 follows.

D A Deterministic Encryption Scheme Secure under IND-CCVA

Let l be a positive integer, and let F be an l-bit block cipher. We denote by $F_K(M)$ and $F_K^{-1}(C)$ an application of the block cipher on M with key K and an application of the inverse cipher on C with key K, respectively. Consider an encryption scheme $\mathcal{SE}$ with message space $\{0,1\}^l$ that works as follows: to encrypt a message M using a key K, compute and return $F_K(M)$; to decrypt a ciphertext C using K, compute and return $F_K^{-1}(M)$. Being deterministic, $\mathcal{SE}$ is clearly not secure under IND-CPA. However, it is easy to see that, if F is a pseudorandom permutation, then $\mathcal{SE}$ is secure under IND-CCVA. To see this, recall that an adversary against $\mathcal{SE}$ under this notion is not allowed to ask for encryptions of its challenge message pair. Furthermore, if it asks for a decryption of the challenge ciphertext C, it will get back only the symbol $\pm$. Therefore, there is not much the adversary can do here to win its game other than breaking the block cipher itself. We omit details.