Secure Channels based on Authenticated Encryption Schemes: A Simple Characterization

Chanathip Namprempre

Department of Computer Science, Thammasat University
41-42 km. Paholyothin Road, Khong Luang, Rangsit
Pathum Thani, Thailand 12121
E-Mail: `meaw@alum.mit.edu`. URL: `www-cse.ucsd.edu/users/cnamprem`.

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 [9] 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. In addition, we relate the two new notions to existing security notions for authenticated encryption schemes.

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 [1, 15, 11, 23]. 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 [9]. 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 [17] and adapted to symmetric-key settings in [3], non-malleability notions defined in [13, 3] and refined in [8], and integrity notions defined in [19, 5, 20]. 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 [9]? 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 [20]: 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.

Security model of Canetti and Krawczyk. In [9], 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

[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.

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 [9] 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, [9] 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 [17] 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 [9], 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, one needs 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 [9], the same cannot be said regarding privacy. In particular, there exists a channel protocol that clearly does not provide semantic security [17] (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 [9]. This new notion is called *universally composable secure channels* [10]. 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 [9] 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, 9] and restate the security model of [9] 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 $K \xleftarrow{R} \mathcal{K}(k)$. 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 $C \xleftarrow{R} \mathcal{E}_K(M)$. 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. Notice that this imposes no correctness requirement when ciphertexts are decrypted out of order. It is up to an individual scheme to decide how to handle ciphertexts that are decrypted out of order. For example, it can reject all such ciphertexts or accept only the ones that decrypt to certain seen messages. We stress that since this requirement is a part of the *syntax* of encryption schemes, it is liberal by design (messages that arrive out of order can have arbitrary decryptions under this requirement!).[2] The goal here is to ensure that as many encryption schemes as possible can be analyzed under the security notions of interest.

SYNTAX OF MESSAGE-DRIVEN PROTOCOLS. A message-driven protocol $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \textsf{activation list})$ 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 string and returns n strings $(x_1, \ldots, x_n)$. The randomized *bootstrapping* algorithm[3] $\mathcal{B}$ takes as inputs k and an l-bit 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 string. Executing the initialization algorithm 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 [9]. 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 algo-

[2] Recall that *syntax* and *security notion* are two separate concepts. Apparently "insecure" schemes such as one that allows arbitrary decryptions for messages that arrive out-of-order are in fact legitimate encryption schemes, i.e. they follow the syntax defined here. However, they are not secure under integrity notions, for instance.

[3] Also known as an initialization function in [2, 9]. We drop their terminology here to avoid confusion with the initialization algorithm.

rithm 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 [9]. 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. In other words, if in an execution of a protocol an initiating party P has a session (P, P', s) and a responding party P' has a session (P, P', s), then we say that the two sessions are matching. The defining feature of session-based protocols is that individual sessions are maintained separately from one another even when 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. The goal of this activation is to allow KE protocols to provide *perfect forward secrecy* of sessions, a property that past session keys remain secret even after long-term keys are compromised [18, 12].

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 to 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, a session-output reveal, a session-key reveal, or has a matching session that has been exposed.

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 $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r,$ activation list$)$ be a session-based message-driven protocol. We follow the notation of [2, 9] 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 $\mathrm{UNADV}_{\pi,U}(k, \vec{x}, \vec{r})$ 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 $\mathrm{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_i$ 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 the protocol output $\mathrm{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r}) = \mathrm{UNADV}_{\pi,U}(k, \vec{x}, \vec{r}), \mathrm{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_1, \ldots, \mathrm{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})_n$ and let $\mathrm{UNAUTH}_{\pi,U}(k)$ be the random variable describing $\mathrm{UNAUTH}_{\pi,U}(k, \vec{x}, \vec{r})$ when $\vec{r}$ is randomly chosen and $\vec{x}$ is generated via $\mathcal{IG}(k, r')$. We denote by $\mathrm{UNAUTH}_{\pi,U}$ the ensemble $\{\mathrm{UNAUTH}_{\pi,U}(k)\}_{k \in N}$.

2.2 Secure Channels per Canetti and Krawczyk [9]

In [9], 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 1, Construction 2, and Definition 3, respectively.

Definition 1 (Protocol Emulation [2, 9]). 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 $\mathrm{AUTH}_{\pi,A}$ and $\mathrm{UNAUTH}_{\pi',U}$ are computationally indistinguishable. ▮

Construction 2 (SMT Protocol [9]). 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 3 (Network Authentication Protocol Security [9]). A protocol is considered to be a *secure authentication protocol* if it emulates the SMT protocol in the UM. ▮

In defining secure encryption protocols, [9] 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".

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, [9] requires that an adversary never ask for an encryption of a particular message more than once. This requirement can be easily implemented using counters. For example, the encryption algorithm can prepend an internal counter to the input message before encrypting the resulting string to obtain the ciphertext. In fact, the use of this mechanism is common in practical Internet protocols including SSH [23], SSL [15], and TLS [11]. Definition 4 below describes the security of network encryption protocols more precisely.

Definition 4 (Network Encryption Protocol Security [9]). Let $k \in \mathsf{N}$. Let $\mathsf{NC} = (\mathcal{IG}, \mathcal{B}, \mathcal{I}, x, l, n, r, \mathsf{activation\ list})$ be a channel protocol. Let U be a UM attacker, and let $r_U\colon\ \mathsf{N} \rightarrow \mathsf{N}$ 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, it is required 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 [9], 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 5 (Transform [9]). 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 $\mathsf{activation\ list}$ 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 $C \stackrel{R}{\leftarrow} \mathcal{E}_K(M)$ 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 (for strong integrity of plaintexts) and IND-CCVA (for indistinguishability against chosen-ciphertext attacks with verification). The goal is to capture the necessary and sufficient properties of the authenticated encryption scheme such that, once the transform per Construction 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 6. 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 6 (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 [9] closely and straightforwardly. Let $b \in \{0,1\}$. Recall that, in the definition of secure encryption protocol per [9], 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 [9] cannot obtain encryptions of a particular message more than once, we also impose the same restriction on the adversary in our experiment.

Definition 7 (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 $\mathbf{Exp}^{\text{ind-ccva-}b}_{\mathcal{AE},A}(k)$
  $K \xleftarrow{R} \mathcal{K}(k)$
  $(M_0, M_1, st) \leftarrow A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}(k, \mathsf{find})$
  $C \xleftarrow{R} \mathcal{E}_K(M_b)$
  $d \leftarrow A^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot,M_b)}(k, \mathsf{guess}, C, st)$
  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 use Definition 8 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 [9], 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. From this point on, U is not allowed to expose the tested session. 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 8 (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 main 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 [9]. We present the theorems and their proof ideas below. The full proofs in detail are in the full version of this paper [21]. For brevity, we write $X \stackrel{s}{\approx} Y$ when the ensembles X and Y are *statistically indistinguishable*. Note that statistical indistinguishability implies computational indistinguishability.

Theorem 9 (Given a secure KE, SINT-PTXT ⇔ 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 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 $\mathrm{AUTH}_{\mathsf{SMT},A} \stackrel{s}{\approx} \mathrm{UNAUTH}_{\mathsf{NAE},U}$. The crux of this proof is essentially the same as that of Theorem 12 of [9], 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 $\mathsf{establish\text{-}session}(P_1, P_2, s, \mathsf{initiator})$ 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 sint-ptxt 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 10 (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 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 $\mathsf{test\text{-}session}$ 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 $\mathsf{incoming}$ 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 new notions by taking the standard approach of relating them to familiar notions. Since the two notions are necessary and sufficient for secure channels, the knowledge we gain from this exercise is applicable to secure channels as well. In our comparisons, we use the following terminology. Suppose X and Y are security notions. We say that X *implies* Y if any scheme secure under X is secure under Y. We say that X does *not* imply Y if there exists an encryption scheme that is secure under X but is insecure under Y. We say that A is *equivalent* to B if A implies B and vice versa. We say that X is *strictly stronger* than Y if X implies Y but Y does not imply X. Finally, we say that X and Y are *incomparable* if X does not imply Y and if Y does not imply X.

In this section, we discuss relations among notions of symmetric encryption as summarized in Figure 1. Our strategy for showing that X implies Y is the standard reduction approach: given an adversary that successfully breaks the scheme under the notion Y, construct an adversary that successfully breaks the scheme under the notion X. To show that X does not imply Y, we start with a scheme secure under X, then modify it to obtain a scheme that remains secure under X but is insecure under Y.

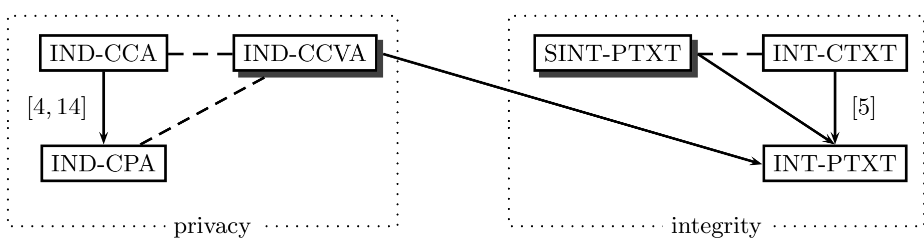

Fig. 1. Relations among notions of symmetric encryption: An arrow from a notion X to a notion Y denotes that X is *strictly stronger* than Y. A dashed line between a notion X and a notion Y denotes that the two notions are *incomparable*. The relations established in other papers are annotated with the corresponding citations. For simplicity, only interesting relations are shown here. We emphasize that the existing notions in this figure (those in unshadowed frames) are variants of the standard notions in the literature. In particular, the oracles here maintain states across invocations.

The standard privacy notions we consider here are indistinguishability under chosen-plaintext and adaptive chosen-ciphertext attacks (IND-CPA and IND-CCA). The original definitions of these notions were in the asymmetric setting [17, 16, 13, 22] but can be "lifted" to the symmetric setting using the encryption oracle based template of [3]. We use the "find-then-guess" definitions per [3] throughout our discussions here. In particular, for both notions, an adversary A plays a game in which it is to "find" a pair of challenge messages (M_0, M_1), obtain the ciphertext corresponding to the encryption of one of the challenge messages, and then "guess" a bit indicating to which challenge message the ciphertext corresponds. For IND-CPA, A is given access to an encryption oracle throughout the game. For IND-CCA, A is given access to both an encryption oracle and a decryption oracle throughout the game. (This notion is also known as IND-CCA2 [4].)

The integrity notions considered here are integrity of plaintexts [5] and integrity of ciphertexts [7, 19, 5]. 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.

Strictly speaking, the original definitions of the existing security notions considered here, namely IND-CPA, IND-CCA, INT-PTXT and INT-CTXT, do not explicitly deal with encryption schemes with stateful decryption algorithms. Therefore, to compare them to our proposed notions, namely IND-CCVA and SINT-PTXT, we make one small modification to existing definitions. Specifically, we allow each oracle used in the definitions to maintain states across invocations. It is easy to see that, this modification notwithstanding, the relations among existing notions shown in [4] and [5] remain the same. It is also easy to see that any schemes secure under the original definitions are secure under the definitions with this modification. Henceforth, we use the original names to refer to the modified definitions.

We provide the justifications of the relations in the full version of this paper [21]. We briefly discuss the relations shown in Figure 1 here. First, we comment that, as Figure 1 shows, SINT-PTXT is reasonably strong: it implies INT-PTXT but not the stronger notion of INT-CTXT. Also, an integrity notion, specifically INT-PTXT, turns out to be necessary for IND-CCVA, a privacy notion.

Being a necessary and sufficient characterization of secure encryption protocol of [9], IND-CCVA is not meant to constitute a complete security measure on its own. Rather, it guarantees secrecy only in conjunction with additional mechanisms that guarantee uniqueness of messages. Consequently, it may be surprising at first glance that IND-CCVA emerges as a notion that is incomparable to both IND-CPA and IND-CCA. In particular, IND-CCVA does not imply even a weak notion of privacy such as IND-CPA. Moreover, the proof of this relation can be easily extended to show that a channel protocol does not provide the stateful variant of semantic security either. (See the full version of this paper [21] for details.) The unfortunate implication here is that channel protocols proven secure as an encryption protocol may in fact leak information. This is a rather unexpected result since one would naturally assume that a secure encryption protocol should protect privacy of transmitted information. On the other hand, it is also arguably simply a technical issue that does not arise in many cases in practice. As pointed out in [9], if one can ensure that all messages are unique, then one can obtain security. One way to ensure uniqueness of messages is to simply prepend unique message IDs to all messages and to verify them when ciphertexts are received. 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 [15, 11, 23].

Acknowledgments

I thank Mihir Bellare for his guidance and advice throughout the research and writing process for this paper. I also thank Ran Canetti and Hugo Krawczyk for their insights and comments especially regarding the notion of secure channels. Finally, I thank Tadayoshi Kohno, Bogdan Warinschi, and Alexandra Boldyreva for their helpful comments on earlier versions of this draft. The author is supported in part by a 1996 Packard Foundation Fellowship in Science and Engineering and NSF CAREER Award CCR-9624439.

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 *Proc. of the 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 *Proc. of the 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 P. Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer-Verlag, December 2000.

8. 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.
9. 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.
10. 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.
11. T. Dierks and C. Allen. The TLS protocol: Version 1.0. RFC 2246, 1999.
12. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, June 1992.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proc. of the 23rd ACM STOC*, pages 542–552, New Orleans, Louisiana, May 6–8 1991. ACM Press.
14. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.
15. A. Freier, P. Karlton, and P. Kocher. The SSL protocol: Version 3.0, 1996.
16. O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *J. Cryptology*, 6(1):21–53, 1993.
17. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.
18. C. Günther. An identity-based key-exchange protocol. In J-J. Quisquater and J. Vandewille, editors, *EUROCRYPT '89*, volume 434 of *LNCS*, pages 29–37. Springer-Verlag, April 1990.
19. 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.
20. 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*, pages 310–331. Springer-Verlag, August 2001.
21. C. Namprempre. Secure channels based on authenticated encryption schemes: A simple characterization. Cryptology ePrint Archive, Report 2002/065, May 2002. `http://eprint.iacr.org/`.
22. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO '91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, August 1991.
23. 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-14.txt`.