From Stateless to Stateful: Generic Authentication and Authenticated Encryption Constructions with Application to TLS

Colin Boyd[1] Britta Hale[1] Stig Frode Mjølsnes[1] Douglas Stebila[2,*]

[1] *Norwegian University of Science and Technology, NTNU, Trondheim, Norway*
{colin.boyd,britta.hale,stig.mjolsnes}@item.ntnu.no
[2] *Queensland University of Technology, Brisbane, Australia*
stebila@qut.edu.au

September 19, 2016

Abstract

Authentication and authenticated encryption with associated data (AEAD) are applied in cryptographic protocols to provide message integrity. The definitions in the literature and the constructions used in practice all protect against forgeries, but offer varying levels of protection against replays, reordering, and drops. As a result of the lack of a systematic hierarchy of authentication and AEAD security notions, gaps have arisen in the literature, specifically in the provable security analysis of the Transport Layer Security (TLS) protocol. We present a hierarchy of authentication and AEAD security notions, interpolating between the lowest level of protection (against forgeries) and the highest level (against forgeries, replays, reordering, and drops). We show generically how to construct higher level schemes from a basic scheme and appropriate use of sequence numbers, and apply that to close the gap in the analysis of TLS record layer encryption.

Keywords: authentication, authenticated encryption with associated data (AEAD), Transport Layer Security (TLS) protocol, secure channels

1 Introduction

Message integrity is a vital security service demanded of cryptographic protocols, and is usually provided either by a message authentication code (MAC) or by a combined authenticated encryption scheme. The standard security property for a MAC is existential unforgeability under a chosen message attack.

There has been an extensive line of research on security notions and constructions for authenticated encryption schemes, with initial definitions given by Katz and Yung [KY01], Bellare and Namprempre [BN00], and Krawczyk [Kra01]. For message confidentiality, an authenticated encryption scheme could achieve indistinguishability under either an adaptive chosen plaintext (IND-CPA) or an adaptive chosen ciphertext (IND-CCA a.k.a. IND-CCA2) attack. For message integrity, an authenticated encryption scheme could achieve either integrity of plaintexts (INT-PTXT) or of ciphertexts (INT-CTXT). Shrimpton [Shr04] combined the separate INT-CTXT and IND-CCA experiments into a single experiment which he called IND-CCA3.

*Supported by Australian Research Council (ARC) Discovery Project grant DP130104304.

Bellare and Namprempre [BN00] and Krawczyk [Kra01] also investigated how to construct authenticated encryption schemes from MACs and symmetric encryption, evaluating three construction paradigms: encrypt-and-MAC, MAC-then-encrypt, and encrypt-then-MAC.

Rogaway [Rog02] defined the notion of *authenticated encryption with associated data (AEAD)*, to capture the common real-world scenario in which some data (such as packet headers) needs to be sent authentically alongside a ciphertext, but need not be encrypted, and AEAD has taken prominence over plain authenticated encryption in recent years.

Despite the utility of authenticated encryption and AEAD, it is not enough to realize the secure channel property expected of cryptographic protocols for two reasons. First, secure channel protocols are often expected to perform an initial establishment of the encryption key using a key exchange protocol; see for example the original paper on secure channels by Canetti and Krawczyk [CK01] (and the follow-up by Namprempre [Nam02]) as well as recent realizations such as the authenticated and confidential channel establishment (ACCE) model of Jager et al. [JKSS12]. (In this paper, we will not focus on the key exchange establishment phase of secure channels.) Second, and more important for this paper, applications often expect reliable delivery of a sequence of messages: that no attacker can replay messages, deliver them in a different order in which they were sent, or drop some messages without later detection.

To capture the notion of delivery of a sequence of messages, Bellare et al. [BKN02] introduced *stateful authenticated encryption*, with two security properties: stateful integrity of ciphertexts (INT-SFCTXT) and stateful indistinguishability of ciphertexts (IND-SFCCA). Kohno et al. [KPB03] extended the statefulness to AEAD schemes, and gave a hierarchy of 5 integrity notions: type 1) security against forgeries; type 2) type 1 plus security against replays; type 3) type 2 plus security against reordering; type 4) type 3 plus detection of previous drops but still accepting subsequent messages; type 5) type 4 plus but not accepting subsequent messages. The type 5 notion of Kohno et al. [KPB03] is equivalent to the stateful authenticated encryption notion of Bellare et al. [BKN02].

Paterson et al. [PRS11] revisit AEAD definitions in the context of the Transport Layer Security (TLS) protocol. They present a combined AEAD security notion called *length-hiding authenticated encryption (LHAE)*, which provides message integrity and confidentiality similar to the type-5 security of Kohno et al. [KPB03], even for messages of different length (hence "length-hiding"), and in a single combined security property (following Shrimpton [Shr04]). Paterson et al. then go on to show that, under appropriate length conditions on the message authentication tag, a simplified form of the encode-then-MAC-then-encrypt form of encryption in the TLS record layer in ciphersuites that use a block cipher in CBC mode is a secure length-hiding authenticated encryption scheme. The simplification is that the statefulness aspects (sequence numbers) are not considered.

Jager et al. [JKSS12] and Krawczyk et al. [KPW13], in their provable security analyses of the full TLS protocol (covering both the authenticated key exchange in the TLS handshake and the TLS record layer), rely on an extension of the work of Paterson et al. [PRS11], namely a form of *stateful length-hiding authenticated encryption (sLHAE)*. Unfortunately, the work of Paterson et al. did not show that TLS encode-then-MAC-then-encrypt satisfies sLHAE, only LHAE. To our knowledge, this gap remains in the literature until now.

1.1 Our contributions

In this work, we construct a hierarchy of authentication and AEAD security notions, show how to construct schemes with higher levels of security from a scheme with the lowest level of security combined with sequence numbers, and apply these techniques to TLS record layer encryption to bridge the gap between LHAE [PRS11] and sLHAE [JKSS12].

First, we construct a hierarchy of authentication levels:

1. protection against forgeries,

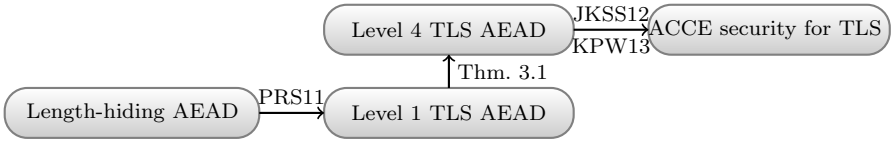

Figure 1: TLS channel analysis.

2. protection against forgeries and replays,

3. protection against forgeries, replays, and reordering of messages, and

4. protection against forgeries, replays, reordering of messages, and dropped messages.

We give a similar hierarchy of definitions for AEAD, with single-experiment AEAD notions that combine integrity and indistinguishability, following Shrimpton [Shr04]. In both cases, these hierarchy levels can be viewed as interpolating between existing *stateless* notions at our level 1 and existing *stateful* notions at our level 4.

Continuing, we show how to construct level 2, 3, and 4 schemes from level 1 schemes. The constructions are not surprising: by appropriate incorporation and checking of sequence numbers, the receiver can ensure it is receiving a valid sequence of sent messages. However, our constructions incorporate a degree of generality: rather than fixing how the sequence numbers are incorporated, we allow an *encoding scheme* to include them either *implicitly* or *explicitly*. For example, in an explicit encoding scheme, the sequence number might be authenticated and then transmitted alongside the ciphertext, in the manner of DTLS. Alternatively, in an implicit encoding scheme, the sequence number might be incorporated into the authentication calculation but not actually transmitted across the wire (since the receiving party ought to know what packet number to expect); this is how TLS works, for example.

We use this generic construction to close the gap in the provable security analysis of TLS record layer encryption. Paterson et al.'s analysis of a simplified form of TLS encode-then-MAC-then-encrypt (Π_{PRS}) shows that it satisfies the LHAE notion, equivalent to our level 1. We can formulate TLS's use of sequence numbers as an encoding scheme in our generic construction, and then see that the full form of TLS encode-then-MAC-then-encrypt (Π_{TLS}) is equivalent to our level-4 generic construction applied to Π_{PRS}, and thus Π_{TLS} achieves level-4 AEAD security, equivalent to sLHAE. Fig. 1 illustrates the connection between our work and that of Paterson et al., Jager et al., and Krawczyk et al., depicting how the construction from level-1 AEAD to level-4 AEAD builds a missing and necessary bridge in the analysis of TLS.

Relation with existing work. The work most closely related to ours is the manuscript of Kohno et al. [KPB03], who gave a hierarchy of AEAD notions. Our AEAD hierarchy maps on to theirs: our levels 1, 2, 3, and 4 correspond to their types 1, 2, 3, and 5, respectively. There are several differences with our work. They give constructions of higher level schemes directly from encryption and MAC schemes in the encrypt-and-MAC, MAC-then-encrypt, and encrypt-then-MAC paradigms, whereas we show how to construct higher levels generically from lower level schemes. Their AEAD hierarchy uses separate integrity and indistinguishability experiments at each level, whereas we use a single combined experiment at each level. We also give a hierarchy of authentication notions, not just AEAD notions, and thereby expand applicability to schemes outside of the AEAD context. Finally, we connect the hierarchy and our generic constructions with TLS record layer encryption.

Connection with secure channel definitions. One motivation of our work was to understand the difference between the original CK01 secure channel definition of Canetti and Krawczyk [CK01] and the ACCE model of Jager et al. [JKSS12]. The confidentiality and integrity notions in CK01 and their NetAut protocol correspond with level 1 of our AEAD hierarchy –

stateless authenticated encryption. A comment in their paper does require that the receiver "check for uniqueness of the incoming message", which would upgrade to level 2 in our hierarchy, and this is the notion that was used in a subsequent work by Namprempre [Nam02]. In contrast, Jager et al.'s ACCE notion maps to level 4 of our AEAD hierarchy – sLHAE.

Application to real-world protocols. Each level of our AEAD hierarchy maps to the requirements expected in some real-world protocols:

- Level 1: DTLS [RM06, RM12]: Datagram TLS provides basic authentication, allows packets to be dropped, and will receive packets out of order, queuing them for future processing.
- Level 2: IPsec Authentication Header (AH) [Ken05]: IPsec Authentication Header protocol provides similar replay detection using a window of recently received packets combined with dropping packets that are "too old".
- Level 2: DTLS with optional replay detection: Datagram TLS does allow optional replay detection [RM06, RM12, §3.3] using a similar technique to IPsec AH.
- Level 3: 802.11 [IEE12] is designed to preventing reordering and to detect replays but allows for packet dropping.
- Level 4: TLS [DR08] is designed to receive a message sequence strictly as a sent, and will be discussed at greater length in Section 4.

A recent analysis [LJBN15] of the QUIC protocol [LC15] employed an AEAD level comparable to our level 1 AEAD; however, the replay-detection abilities of QUIC suggest that a higher authentication level should be achievable.

1.2 Additional related work

There are several additional lines of work on authenticated encryption.

One line of research views data "as a stream", rather than a discrete sequence of messages; practical implementations receive data byte-by-byte rather than as atomic messages in security definitions. Albrecht et al. [APW09] showed how to carry out a plaintext recovery attack against the Secure Shell (SSH) protocol as a result of byte-by-byte processing. This motivated the need for non-atomic authenticated encryption definitions [BDPS12, FGMP15]. The work of Fischlin et al. [FGMP15] in particular is motivated by protocols such as TLS, SSH, and QUIC, and describes checks that can again be correlated with our level-4 AEAD notion. It would be interesting to expand stream-based analysis in the direction of our hierarchical levels for protocols that allow packet dropping. For example, the QUIC protocol [LC15] runs over UDP and tolerates a degree of packet loss, making analysis under a level-4 stream-based notion inappropriate.

Another line of research focuses on the use of nonces in authenticated encryption [Rog02, RBBK01], and more recently for the specific purposes of protecting implementations that misuse counters or nonces [RS06, FFL12, HRRV15]. Meanwhile, Hoang et al. [HKR14] define a notion of robust authenticated encryption which incorporates padding properties similar to the stateless form of LHAE of Paterson et al. [PRS11]. Finally, additional recent work focuses on defining authenticated encryption results in the constructive cryptography framework [MT10, BMM+15].

2 Authentication Hierarchy

In this section, we formalize our 4-tier hierarchy of authentication notions, each level building on the previous, and show how to achieve higher level notions from level-1 combined with appropriate checks on sequence numbers.

$\underline{\mathsf{Exp}^{\mathsf{auth}_i}_{\Pi,\mathcal{A}}()}$:

1: $k \xleftarrow{\$} \mathrm{Kgn}()$
2: $st_{\mathrm{E}} \leftarrow \perp, st_{\mathrm{D}} \leftarrow \perp$
3: $u \leftarrow 0, v \leftarrow 0$
4: $\mathtt{r} \leftarrow 0$
5: $\mathcal{A}^{\mathsf{Send}(\cdot),\mathsf{Recv}(\cdot)}()$
6: **return** $\mathtt{r}$

Oracle $\mathsf{Send}(m)$:

1: $u \leftarrow u + 1$
2: $(sent_u, st_{\mathrm{E}}) \leftarrow \mathrm{Snd}(k, m, st_{\mathrm{E}})$
3: **return** $sent_u$ to $\mathcal{A}$

Oracle $\mathsf{Recv}(c)$:

1: $v \leftarrow v + 1$
2: $rcvd_v \leftarrow c$
3: $(m, \alpha, st_{\mathrm{D}}) \leftarrow \mathrm{Rcv}(k, c, st_{\mathrm{D}})$
4: **if** $(i = 4) \wedge (\alpha = 1) \wedge [\mathsf{cond}_4 \vee (\alpha_{v-1} = 0)]$ **then**
5: $\quad \mathtt{r} \leftarrow 1$
6: **else if** $(\alpha = 1) \wedge \mathsf{cond}_i$ **then**
7: $\quad \mathtt{r} \leftarrow 1$
8: $\quad$ **return** $\mathtt{r}$ from experiment
9: **return** $\perp$ to $\mathcal{A}$

1. **Basic authentication:**
   $\mathsf{cond}_1 = (\nexists w : c = sent_w)$
2. **Basic authentication, no replays:**
   $\mathsf{cond}_2 = (\nexists w : c = sent_w) \vee (\exists w < v : c = rcvd_w)$
3. **Basic authentication, no replays, strictly increasing:**
   $\mathsf{cond}_3 = (\nexists w : c = sent_w) \vee (\exists w, x, y : (w < v) \wedge (sent_x = rcvd_w) \wedge (sent_y = rcvd_v) \wedge (x \geq y))$
4. **Basic authentication, no replays, strictly increasing, no drops:**
   $\mathsf{cond}_4 = (u < v) \vee (c \neq sent_v)$

Figure 2: Stateful authentication experiment auth_i with authentication condition cond_i for stateful authentication scheme $\Pi = (\mathrm{Kgn}, \mathrm{Snd}, \mathrm{Rcv})$ and adversary $\mathcal{A}$.

2.1 Definitions

Definition 2.1. *A* stateful authentication scheme Π *for a message space $\mathcal{M}$, a key space $\mathcal{K}$, and an output space $\mathcal{C}$ is a tuple of algorithms:*

- $\mathrm{Kgn}() \xrightarrow{\$} k$*: A probabilistic key generation algorithm that outputs a key k.*
- $\mathrm{Snd}(k, m, st_{\mathrm{E}}) \xrightarrow{\$} (c, st_{\mathrm{E}})$*: A probabilistic authentication algorithm that takes as input a key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$, and an authentication state st_{E}, and outputs a tagged message $c \in \mathcal{C}$ and updated state st_{E}.*
- $\mathrm{Rcv}(k, c, st_{\mathrm{D}}) \rightarrow (m, \alpha, st_{\mathrm{D}})$*: A deterministic verification algorithm that takes as input a key $k \in \mathcal{K}$, a tagged message $c \in \mathcal{C}$, and a verification state st_{D}, and outputs either a message $m \in \mathcal{M}$ or an error symbol $\perp$, a bit $\alpha \in \{0, 1\}$, and an updated state st_{D}.*

On first use, st_{E} and st_{D} are initialized to $\perp$.

Correctness is defined in the natural way: for all $m \in \mathcal{M}$, all $k \xleftarrow{\$} \mathrm{Kgn}()$, all st_{E} and st_{D} defined in any sequence of sends and receives respectively, and all c such that $(c, st'_{\mathrm{E}}) \leftarrow \mathrm{Snd}(k, m, st_{\mathrm{E}})$, we have that $\mathrm{Rcv}(k, c, st_{\mathrm{D}}) = (m, 1, st'_{\mathrm{D}})$.

Note that in the case of a Rcv (message authentication check) failure, the receive algorithm outputs a failure symbol $\perp$, $\alpha = 0$ to denote a failed receipt, and an updated state st_{D}: $(\perp, 0, st_{\mathrm{D}}) \leftarrow \mathrm{Rcv}(k, c, st_{\mathrm{D}})$. Otherwise, the algorithm outputs the correctly received message m, $\alpha = 1$ to denote successful receipt, and an updated state st_{D}: $(m, 1, st_{\mathrm{D}}) \leftarrow \mathrm{Rcv}(k, c, st_{\mathrm{D}})$.

Formally we define a stateful authentication security experiment that can be *parameterized* with different authentication conditions to capture various levels of authentication. Four graded levels of authentication are defined for the experiment, correlated to different conditions, cond_i, under which an adversary $\mathcal{A}$ wins, as shown in Fig. 2. Note that cond_4 is strongly linked to authentication demands in analyses of TLS [PRS11, JKSS12], a protocol with strict authentication requirements.

Definition 2.2. *Let* Π *be a stateful authentication scheme and let* $\mathcal{A}$ *be an adversary algorithm. Let* $i \in \{1, \ldots, 4\}$*. The stateful authentication experiment for* Π *with authentication condition* cond_i *is given by* $\mathsf{Exp}^{\mathsf{auth}_i}_{\Pi,\mathcal{A}}$ *in Fig. 2. We define*

$$\mathbf{Adv}^{\mathsf{auth}_i}_{\Pi}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\mathsf{auth}_i}_{\Pi}(\mathcal{A}) = 1\right] \ .$$

We now give a brief explanation of the "winning" conditions in Figure 2. As we can see in lines 4–8 of the Recv oracle, the receiver is processing its vth ciphertext c; if the ciphertext is accepted by the receiving algorithm Rcv (in which case $\alpha = 1$), then the adversary is deemed to have "won" if cond_i is true, for $i = 1, 2, 3$. For $i = 4$, we do not restrict a win to a correct receipt ($\alpha = 1$); since level-4 does not allow drops, an adversary may win under a failed receipt if it can succeed in having further messages correctly received.

- cond_1: To capture basic authentication of ciphertexts, we say that the adversary has won if the received ciphertext c was not one of the ciphertexts output by the Send oracle.
- cond_2: To capture basic authentication of ciphertexts with no replays, we say that the adversary has won if the received ciphertext c was not one of the ciphertexts output by the Send oracle, or if it was output by the Send oracle and previously accepted by the Recv oracle.
- cond_3: To capture basic authentication of ciphertexts with no replays and strictly increasing receipt, we say that the adversary has won if the received ciphertext c was not one of the ciphertexts output by the Send oracle, or if there were two messages received out of order: if $rcvd_w$ was received before $rcvd_v$, but $rcvd_w$ was sent after $rcvd_v$.
- cond_4: To capture basic authentication of ciphertexts with no replays, strictly increasing receipt, and no drops, we say that the adversary has won if the v-th ciphertext received (c) is not the v-th ciphertext sent, or if fewer than v ciphertexts have been sent.

Remark. If the authenticated message c takes the form of a ciphertext, then level-1 authentication is equivalent to INT-CTXT. If c is such that $c = (m, \mathrm{MAC}(m))$, where MAC is a message authentication code, then level-1 authentication is equivalent to SUF-CMA. In order to maximize the application potential of our results, we provide the generality for either application.

2.2 Relations among authentication notions

Each of the authentication notions sequentially implies the security of the levels below it. In the following theorem, the security implications between levels are formalized, with security at Level 2 implying security at Level 1, etc.

Theorem 2.1 (Level-(i+1) authentication implies level-i authentication)**.** *Let* $\Pi = (\mathrm{Kgn}, \mathrm{Snd}, \mathrm{Rcv})$ *be an authentication scheme and let* $i \in \{1, 2, 3\}$*. For any adversary* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathsf{auth}_i}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{auth}_{i+1}}_{\Pi}(\mathcal{A}) \ .$$

The proof of Theorem 2.1 can be found in Appendix A.

2.3 Constructing higher level authentication schemes

In this section, we generically show how to build higher level authentication schemes based on lower level authentication schemes and the inclusion of sequence numbers with appropriate checks. Since currently implemented protocols use both implicit and explicit sequence numbers, we generalize our model for an arbitrary *encoding scheme* which captures both implicit and explicit sequence numbers.

Definition 2.3 (Authentication encoding scheme)**.** *An* (authentication) encoding scheme $\texttt{Coding}$ *for a sequence number space* S *and message space* $\mathcal{M}$ *is a pair of algorithms:*

- $\texttt{Ecd}(\texttt{sqn}, m) \to m_{\texttt{ecd}}$*: A deterministic encoding algorithm that takes as input a sequence number* $\texttt{sqn} \in \mathsf{S}$ *and a message* $m \in \mathcal{M}$*, and outputs an encoded message* $m_{\texttt{ecd}} \in \mathcal{M}_{\texttt{ecd}}$*, where* $\mathcal{M}_{\texttt{ecd}}$ *is the encoded version of* $\mathcal{M}$*.*
- $\texttt{Dcd}(\mathsf{sqnlist}, m_{\texttt{ecd}}) \to (\texttt{sqn}, m, \alpha)$*: A deterministic decoding algorithm that takes as input a sequence number list* $\mathsf{sqnlist} \subset \mathsf{S}$ *and an encoded message* $m_{\texttt{ecd}} \in \mathcal{M}_{\texttt{ecd}}$*, and outputs a sequence number* $\texttt{sqn} \in \mathsf{S}$*, a message* $m \in \mathcal{M}$ *or an error symbol* $\perp$*, and a status variable* $\alpha = 1$ *if decoding was successful or* $\alpha = 0$ *otherwise.*

In our construction of higher level authentications, we will require that $\texttt{Ecd}$ is collision-resistant.

We can construct schemes that use either implicit or explicit sequence numbers using Definition 2.3. For example, the scheme with $\texttt{Ecd}(\texttt{sqn}, m) := \texttt{sqn} \| m$ has an explicit sequence number, and may be very applicable in practice since $\texttt{sqn}$ is sent explicitly with the message. An alternative scheme with implicit sequence numbers would be $\texttt{Ecd}(\texttt{sqn}, m) := m \| \mathrm{MAC}(\texttt{sqn})$. Thus elements of the space $\mathcal{M}_{\texttt{ecd}}$ may take various forms, contingent on the properties desirable for $\texttt{Coding}$. We will see in Section 4.2 that the TLS record layer protocol uses an encoding scheme based on the second example above. We formally distinguish explicit and implicit sequence numbers as follows:

Definition 2.4. *We say that authentication encoding scheme* $\texttt{Coding}$ *uses* explicit sequence numbers *if* $\texttt{Dcd}(\emptyset, \texttt{Ecd}(\texttt{sqn}, m)) = (\texttt{sqn}, m, 1)$ *for all* $\texttt{sqn}$ *and all* m*, and that* $\texttt{Coding}$ *uses* implicit sequence numbers *otherwise.*

We now present our generic constructions of level-i authentication schemes from a level-1 authentication scheme. The heart of our construction is a sequence number check $\texttt{TEST}i$ that will correspond to the authentication condition cond_i. Our constructions can accommodate any collision-resistant encoding scheme $\texttt{Coding}$, with either implicit or explicit sequence numbers; this requirement is specifically important in implicit authentication where the sequence number is not physically present on receipt. For conciseness, the notation Π'_i for $P(\Pi, \texttt{Ecd}, \texttt{TEST}i)$ will be generally employed.

Definition 2.5 (P construction)**.** *Let* Π *be a (level-1) authentication scheme,* $\texttt{Coding}$ *be an encoding scheme, and let* $\texttt{TEST}i$ *be one of the conditions specified in Fig. 3. Define* $\Pi'_i := P(\Pi, \texttt{Coding}, \texttt{TEST}i)$ *as the authentication scheme resulting from apply construction* P *in Fig. 3.*

In this construction, the check $\texttt{TEST2}$ corresponds to the condition for level-2 authentication. Basic level-1 authentication is assumed, so $\texttt{TEST2}$'s protection against replays implies replay protection for condition cond_2. Namely, if $\exists w < v : c = rcvd_w$ then $\exists j : \texttt{sqn} = st_{\mathrm{D}}.\mathsf{sqnlist}_j$, since identical authenticated messages must contain identical sequence numbers. Similar connections exist between $\texttt{TEST3}$ and cond_3 and $\texttt{TEST4}$ and cond_4. Note that to check $\texttt{TEST2}$ it is necessary to maintain a record of all previously received $\texttt{sqn}$; thus $st_{\mathrm{D}}.\mathsf{sqnlist}$ must be a complete record. However, for $\texttt{TEST3}$ and $\texttt{TEST4}$, it is strictly only necessary for $st_{\mathrm{D}}.\mathsf{sqnlist}$ to contain the last received $\texttt{sqn}$.

The following theorem shows that the P construction with $\texttt{TEST}i$ achieves level-i authentication. Notably Theorem 2.2 depends on the collision-resistance of $\texttt{Ecd}$. For many encoding schemes, this follows immediately. For example, the simple concatenation scheme $\texttt{Ecd}(\mathsf{ctr}, m) = \mathsf{ctr} \| m$ is clearly collision-resistant when assuming unambiguous concatenation. When such a scheme is used, the advantage of $\mathcal{A}$ is then directly reducible to the advantage of $\mathcal{F}$. For the proof of Theorem 2.2, see appendix A.

$\underline{\Pi'_i.\mathrm{Kgn}()}$:
1: **return** $\Pi.\mathrm{Kgn}()$

$\underline{\Pi'_i.\mathrm{Snd}(k, m, st'_{\mathrm{E}})}$:
1: $(c, st'_{\mathrm{E}}.\mathsf{subst}) \leftarrow \Pi.\mathrm{Snd}(k, \mathtt{Ecd}(st'_{\mathrm{E}}.\mathsf{ctr}, m), st'_{\mathrm{E}}.\mathsf{subst})$
2: $st'_{\mathrm{E}}.\mathsf{ctr} \leftarrow st'_{\mathrm{E}}.\mathsf{ctr} + 1$
3: **return** (c, st'_{E})

$\underline{\Pi'_i.\mathrm{Rcv}(k, c, st'_{\mathrm{D}})}$:
1: **if** $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$ **then**
2: **return** $(\perp, 0, st_{\mathrm{D}})$
3: $(m_\Pi, \alpha, st'_{\mathrm{D}}.\mathsf{subst}) \leftarrow \Pi.\mathrm{Rcv}(k, c, st'_{\mathrm{D}}.\mathsf{subst})$
4: **if** $\alpha = 1$ **then**
5: $(\mathtt{sqn}, m, \alpha) \leftarrow \mathtt{Dcd}(st'_{\mathrm{D}}.\mathsf{sqnlist}, m_\Pi)$
6: **if** $(\alpha = 0) \vee \mathtt{TEST}i$ **then**
7: $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$
8: **return** $(\perp, 0, st'_{\mathrm{D}})$
9: $st'_{\mathrm{D}}.\mathsf{sqnlist} = st'_{\mathrm{D}}.\mathsf{sqnlist}||\mathtt{sqn}$
10: **return** $(m, \alpha, st'_{\mathrm{D}})$

Sequence number tests for building Π', correlated to authentication levels:

- **Basic authentication, no replays:**
  $\mathtt{TEST2} = (\exists j : \mathtt{sqn} = st'_{\mathrm{D}}.\mathsf{sqnlist}_j)$
- **Basic authentication, no replays, strictly increasing:**
  $\mathtt{TEST3} = (\exists j : \mathtt{sqn} \not> st'_{\mathrm{D}}.\mathsf{sqnlist}_j)$
- **Basic authentication, no replays, strictly increasing, no drops:**
  $\mathtt{TEST4} = (\exists j : \mathtt{sqn} \not> st'_{\mathrm{D}}.\mathsf{sqnlist}_j) \vee (\mathtt{sqn} \neq \max\{st'_{\mathrm{D}}.\mathsf{sqnlist}_j\} + 1)$

Description of states st'_{E} and st'_{D}:

- $st'_{\mathrm{E}}.\mathsf{subst} := st_{\mathrm{E}}$, where st_{E} is the state in Π
- $st'_{\mathrm{E}}.\mathsf{ctr}$. When $\Pi'.\mathrm{Snd}$ is initialized, $st'_{\mathrm{E}}.\mathsf{ctr} \leftarrow 0$.
- $st'_{\mathrm{D}}.\mathsf{subst} := st_{\mathrm{D}}$, where st_{D} is the state in Π
- $st'_{\mathrm{D}}.\mathsf{status}$. Once $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$ it is not reset and all subsequently received messages are also immediately aborted.
- $st'_{\mathrm{D}}.\mathsf{sqnlist}$, an ordered list of sequence numbers previously received. It is required that $|st'_{\mathrm{D}}.\mathsf{sqnlist}| \geq 1$ after the first received $\mathtt{sqn}$; i.e. the size of the ordered set is maintained at 1 or greater. When $\Pi'.\mathrm{Snd}$ is initialized, $st'_{\mathrm{D}}.\mathsf{sqnlist} \leftarrow \perp$.

Figure 3: Construction P of a level-i authentication scheme Π'_i from a level-1 authentication scheme Π and encoding scheme $\mathtt{Coding} = (\mathtt{Ecd}, \mathtt{Dcd})$.

Theorem 2.2. *Let Π be a secure level-1 authentication scheme and $\mathtt{Coding}$ be an authentication encoding scheme with collision-resistant encoding. Let $i \in \{2, 3, 4\}$. Then $\Pi'_i = P(\Pi, \mathtt{Coding}, \mathtt{TEST}i)$, constructed as in Fig. 3, is a secure level-i authentication scheme. Specifically, let $\mathcal{A}$ be an adversary algorithm that runs in time t and asks q_s Send queries and q_r Recv queries, and let $q = q_s + q_r$. Then there exists an adversary $\mathcal{B}$ that runs in time $t_{\mathcal{B}} \approx t$ and asks no more than $q_{\mathcal{B}} = \frac{1}{2}q_s(q_s - 1)$ queries, and an adversary $\mathcal{F}$ that runs in time $t_{\mathcal{F}} \approx t$ and asks $q_{\mathcal{F}} = q$ queries, such that*

$$\mathbf{Adv}^{\mathsf{auth}_i}_{P(\Pi,\mathtt{Coding},\mathtt{TEST}i)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{auth}_1}_{\Pi}(\mathcal{F}) + \mathbf{Adv}^{\mathrm{collision}}_{\mathtt{Ecd}}(\mathcal{B}) \ .$$

The time-cost for checking using implicit sequence numbers could be considerable when using a Level 2 or Level 3 authentication notion due to the need to check against all previously received messages (see appendix A for details). However, to our knowledge, there are no real-world implementations using implicit sequence numbers at these levels. Implicit sequence numbers have been used in instances where Level 4 authentication is desired, but explicit sequence numbers are usually employed at the lower levels. Logically, this also corresponds to desirable real-world instantiation formats; if a protocol allows packets to be dropped then it would be inconvenient to base authentication upon information that is not explicitly sent in each packet. Alternatively, if no drops are allowed, authentication can be checked against explicit or implicit information.

3 Authenticated Encryption Hierarchy

In this section, we build equivalent notions for *authenticated encryption with associated data (AEAD) schemes*. AEAD security is typically defined by extending the authentication notion with a type of *left-or-right* encryption game.

3.1 Definitions

Definition 3.1. *A* stateful AEAD scheme Π *for a message space* $\mathcal{M}$*, an associated data space* $\mathcal{AD}$*, a key space* $\mathcal{K}$*, and a ciphertext space* $\mathcal{C}$*, is a tuple of algorithms:*

- $\mathrm{Kgn}() \xrightarrow{\$} k$*: A probabilistic key generation algorithm that outputs a key* k
- $\mathrm{E}(k, \ell, \mathsf{ad}, m, st_{\mathrm{E}}) \xrightarrow{\$} (c, st'_{\mathrm{E}})$*: A probabilistic encryption algorithm that takes as input a key* $k \in \mathcal{K}$*, a length* $\ell \in \mathbb{Z}$*, associated data* $\mathsf{ad} \in \mathcal{AD}$*, a message* $m \in \mathcal{M}$*, and an encryption state* st_{E}*, and outputs a ciphertext* $c \in \mathcal{C}$ *and updated state* st'_{E}*.*
- $\mathrm{D}(k, \mathsf{ad}, c, st_{\mathrm{D}}) \to (\mathsf{ad}, m, \alpha, st'_{\mathrm{D}})$*: A deterministic decryption algorithm that takes as input a key* $k \in \mathcal{K}$*, associated data* $\mathsf{ad} \in \mathcal{AD}$*, a ciphertext* c*, and a decryption state* st_{D}*, and outputs either associated data* ad *or an error symbol* $\perp$*, a message* $m \in \mathcal{M}$ *or an error symbol* $\perp$*, a bit* $\alpha \in \{0, 1\}$*, and an updated state* st'_{D}*.*

Compared with stateful authentication schemes in Definition 2.1, AEAD schemes utilize two further fields: ad, which is for associated data (such as authenticated but unencrypted header data), and an optional length field ℓ.

Correctness is defined in an analogous manner to that of stateful authentication schemes. Correspondingly we define 4 levels of stateful AEAD security.

Definition 3.2. *Let* Π *be a stateful AEAD scheme and let* $\mathcal{A}$ *be an PPT adversarial algorithm. Let* $i \in \{1, \ldots, 4\}$ *and let* $b \in \{0, 1\}$*. The stateful AEAD experiment for* Π *with condition* cond_i *and bit* b *is given by* $\mathsf{Exp}_{\Pi}^{\mathsf{aead}_i\text{-}b}(\mathcal{A})$ *in Fig. 4. We define*

$$\mathbf{Adv}_{\Pi}^{\mathsf{aead}_i}(\mathcal{A}) = \left| \Pr\left[\mathsf{Exp}_{\Pi}^{\mathsf{aead}_i\text{-}1}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}_{\Pi}^{\mathsf{aead}_i\text{-}0}(\mathcal{A}) = 1\right] \right| \ .$$

The $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ oracles in Fig. 4 work together to provide both an authentication experiment and ciphertext indistinguishability experiment. When $b = 0$, the adversary always gets m_0 encrypted and never receives any decryption information. When $b = 1$, the adversary always gets m_1 encrypted and potentially receives decryption information. If the adversary makes an attempt to forge ciphertexts or violate the sequencing condition, then a secure stateful AEAD scheme should return $\perp$ in all subsequent decryption queries. If the adversary has caused the encryptor and decryptor to get out of sync (by forging a ciphertext or violating the sequencing condition) and ever receives non-$\perp$ from $\mathsf{Decrypt}$, the adversary learns $b = 1$.

When ℓ is not used, the level-1 notion aead_1 corresponds to IND-CCA and INT-CTXT security of a stateless AEAD scheme.

When ℓ is used for length, the level-4 notion aead_4 corresponds to the stateful length-hiding authenticated encryption security notion of Krawczyk et al. [KPW13] which is a slight modification of that of Jager et al. [JKSS12].

Analogously to Section 2.2, level-$(i + 1)$ AEAD security implies level-i AEAD security.

3.2 Constructing higher level AEAD schemes

Similarly to Section 3, we can construct higher level AEAD schemes based on a level-1 AEAD scheme with the inclusion of sequence numbers with appropriate checks. We again generalize the approach using an encoding scheme that captures both implicit and explicit sequence numbers.

$\underline{\mathsf{Exp}^{\mathsf{aead}_i\text{-}b}_{\Pi,\mathcal{A}}()}$:

$k \xleftarrow{\$} \mathrm{Kgn}()$
$st_{\mathrm{E}} \leftarrow \bot, st_{\mathrm{D}} \leftarrow \bot$
$u \leftarrow 0, v \leftarrow 0$
$\mathtt{out\text{-}of\text{-}sync} \leftarrow 0$
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Encrypt}(\cdot),\mathsf{Decrypt}(\cdot)}()$
return b'

Oracle $\mathsf{Encrypt}(\ell, \mathtt{ad}, m_0, m_1)$:

$u \leftarrow u + 1$
$(sent.c^{(0)}, st_{\mathrm{E}}^{(0)}) \leftarrow \mathrm{E}(k, \ell, \mathtt{ad}, m_0, st_{\mathrm{E}})$
$(sent.c^{(1)}, st_{\mathrm{E}}^{(1)}) \leftarrow \mathrm{E}(k, \ell, \mathtt{ad}, m_1, st_{\mathrm{E}})$
if $sent.c^{(0)} = \bot$ or $sent.c^{(1)} = \bot$ **then**
  return $\bot$
$(sent.ad_u, sent.c_u, st_{\mathrm{E}}) := (\mathtt{ad}, sent.c^{(b)}, st_{\mathrm{E}}^{(b)})$
return $sent.c_u$

Oracle $\mathsf{Decrypt}(\mathtt{ad}, c)$:

if $b = 0$ **then**
  return $\bot$
$v \leftarrow v + 1$
$rcvd.c_v \leftarrow c$
$(\mathtt{ad}, m, \alpha, st_{\mathrm{D}}) \leftarrow \mathrm{D}(k, \mathtt{ad}, c, st_{\mathrm{D}})$
if $(i = 4) \wedge \mathsf{cond}_4$ **then**
  $\mathtt{out\text{-}of\text{-}sync} \leftarrow 1$
else if $(\alpha = 1) \wedge \mathsf{cond}_i$ **then**
  $\mathtt{out\text{-}of\text{-}sync} \leftarrow 1$
if $\mathtt{out\text{-}of\text{-}sync} = 1$ **then**
  return m
return $\bot$

1. **Basic authenticated encryption:**
   $\mathsf{cond}_1 = (\nexists w : (c = sent.c_w) \wedge (\mathtt{ad} = sent.ad_w))$
2. **Basic authenticated encryption, no replays:**
   $\mathsf{cond}_2 = (\nexists w : (c = sent.c_w) \wedge (\mathtt{ad} = sent.ad_w)) \vee (\exists w < v : c = rcvd.c_w)$
3. **Basic authenticated encryption, no replays, strictly increasing:**
   $\mathsf{cond}_3 = (\nexists w : (c = sent.c_w) \wedge (\mathtt{ad} = sent.ad_w)) \vee (\exists w, x, y : (w < v) \wedge (sent.c_x = rcvd.c_w) \wedge (sent.c_y = rcvd.c_v) \wedge (x \geq y))$
4. **Basic authenticated encryption, no replays, strictly increasing, no drops:**
   $\mathsf{cond}_4 = (u < v) \vee (c \neq sent.c_v) \vee (\mathtt{ad} \neq sent.ad_v)$

Figure 4: Stateful AEAD experiment aead_i with authentication condition cond_i for stateful AEAD scheme $\Pi = (\mathrm{Kgn}, \mathrm{E}, \mathrm{D})$ and adversary $\mathcal{A}$.

Definition 3.3 (AEAD encoding scheme)**.** *An* AEAD encoding scheme $\mathtt{Coding}$ *for a sequence number space* S*, a message space* $\mathcal{M}$*, and an associated data space* $\mathcal{AD}$ *is a pair of algorithms:*

- $\mathtt{Ecd}(\mathtt{sqn}, \mathtt{ad}, m) \to (\mathtt{ad}_{\mathtt{ecd}}, m_{\mathtt{ecd}})$*: A deterministic encoding algorithm that takes as input a sequence number* $\mathtt{sqn} \in \mathsf{S}$*, associated data* $\mathtt{ad} \in \mathcal{AD}$*, and a message* $m \in \mathcal{M}$*, and outputs an encoded associated data value* $\mathtt{ad}_{\mathtt{ecd}} \in \mathcal{AD}_{\mathtt{ecd}}$ *and message* $m_{\mathtt{ecd}} \in \mathcal{M}_{\mathtt{ecd}}$*, where* $\mathcal{AD}_{\mathtt{ecd}}$ *and* $\mathcal{M}_{\mathtt{ecd}}$ *are the encoded versions of associated data space* $\mathcal{AD}$ *and message space* $\mathcal{M}$*, respectively.*
- $\mathtt{Dcd}(\mathsf{sqnlist}, \mathtt{ad}_{\mathtt{ecd}}, m_{\mathtt{ecd}}) \to (\mathtt{sqn}, \mathtt{ad}, m, \alpha)$*: A deterministic decoding algorithm that takes as input a sequence number list* $\mathsf{sqnlist} \subset \mathsf{S}$*, an encoded associated data value* $\mathtt{ad}_{\mathtt{ecd}}$*, and an encoded message* $m_{\mathtt{ecd}} \in \mathcal{M}_{\mathtt{ecd}}$*, and outputs a sequence number* $\mathtt{sqn} \in \mathsf{S}$*, associated data* $\mathtt{ad} \in \mathcal{AD}$ *or an error symbol* $\bot$*, a message* $m \in \mathcal{M}$ *or an error symbol* $\bot$*, and a status variable* $\alpha = 1$ *if decoding was successful or* $\alpha = 0$ *otherwise.*

We similarly distinguish between schemes with explicit and implicit sequence numbers:

Definition 3.4. *We say that AEAD encoding scheme* $\mathtt{Coding}$ *uses* explicit sequence numbers *if, for all* $\mathtt{sqn}$*,* $\mathtt{ad}$*, and* m*, when* $\mathtt{Ecd}(\mathtt{sqn}, \mathtt{ad}, m) = (\mathtt{ad}_{\mathtt{ecd}}, m_{\mathtt{ecd}})$*, we have that* $\mathtt{Dcd}(\bot, \mathtt{ad}_{\mathtt{ecd}}, m_{\mathtt{ecd}}) = (\mathtt{sqn}, \mathtt{ad}, m, 1)$*. Otherwise, we say that* $\mathtt{Coding}$ *uses* implicit sequence numbers*.*

Our construction of level-i AEAD schemes from a level-1 AEAD scheme is as follows:

Definition 3.5 (P_{AEAD} construction)**.** *Let* Π *be a (level-1) AEAD scheme,* $\mathtt{Coding}$ *be an AEAD encoding scheme, and let* $\mathtt{TEST}i$ *be a condition specified in Fig. 3. Define* $\Pi'_i := P_{\mathrm{AEAD}}(\Pi, \mathtt{Ecd}, \mathtt{TEST}i)$ *as the AEAD scheme resulting from applying construction* P_{AEAD} *in Fig. 5.*

$\underline{\Pi'_i.\mathrm{Kgn}():}$
1: **return** $\Pi.\mathrm{Kgn}()$

$\underline{\Pi'_i.\mathrm{E}(k, \ell, \mathsf{ad}, m, st'_{\mathrm{E}}):}$
1: $(\mathsf{ad}_\Pi, m_\Pi) \leftarrow \mathtt{Ecd}(st'_{\mathrm{E}}.\mathsf{ctr}, \mathsf{ad}, m)$
2: $(c, st'_{\mathrm{E}}.\mathsf{subst}) \leftarrow \Pi.\mathrm{E}(k, m_\Pi, \mathsf{ad}_\Pi, l, st'_{\mathrm{E}}.\mathsf{subst})$
3: $st'_{\mathrm{E}}.\mathsf{ctr} \leftarrow st'_{\mathrm{E}}.\mathsf{ctr} + 1$
4: **return** (c, st'_{E})

$\underline{\Pi'_i.\mathrm{D}(k, \mathsf{ad}, c, st'_{\mathrm{D}}):}$
1: **if** $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$ **then**
2: **return** $(\perp, 0, st_{\mathrm{D}})$
3: $(\mathsf{ad}_\Pi, m_\Pi, \alpha, st'_{\mathrm{D}}.\mathsf{subst}) \leftarrow \Pi.\mathrm{D}(k, \mathsf{ad}, c, st'_{\mathrm{D}}.\mathsf{subst})$
4: **if** $\alpha = 1$ **then**
5: $(\mathtt{sqn}, \mathsf{ad}, m, \alpha) \leftarrow \mathtt{Dcd}(st'_{\mathrm{D}}.\mathsf{sqnlist}, \mathsf{ad}_\Pi, m_\Pi)$
6: **if** $(\alpha = 0) \vee \mathtt{TEST}i$ **then**
7: $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$
8: **return** $(\perp, 0, st'_{\mathrm{D}})$
9: $st'_{\mathrm{D}}.\mathsf{sqnlist} = st'_{\mathrm{D}}.\mathsf{sqnlist} \| \mathtt{sqn}$
10: **return** $(m, \alpha, st'_{\mathrm{D}})$

Description of states st'_{E} and st'_{D}:

- $st'_{\mathrm{E}}.\mathsf{subst} := st_{\mathrm{E}}$, where st_{E} is the state in Π
- $st'_{\mathrm{E}}.\mathsf{ctr}$. When $\Pi'.\mathrm{E}$ is initialized, $st'_{\mathrm{E}}.\mathsf{ctr} \leftarrow 0$.
- $st'_{\mathrm{D}}.\mathsf{subst} := st_{\mathrm{D}}$, where st_{D} is the state in Π
- $st'_{\mathrm{D}}.\mathsf{status}$. Once $st'_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$ it is not reset and all subsequently received messages are also immediately aborted.
- $st'_{\mathrm{D}}.\mathsf{sqnlist}$, an ordered list of sequence numbers previously received. It is required that $|st'_{\mathrm{D}}.\mathsf{sqnlist}| \geq 1$ after the first received $\mathtt{sqn}$; i.e. the size of the ordered set is maintained at 1 or greater. When $\Pi'.\mathrm{E}$ is initialized, $st'_{\mathrm{D}}.\mathsf{sqnlist} \leftarrow \perp$.

Figure 5: Construction P_{AEAD} of a level-i AEAD scheme Π'_i from a level-1 AEAD scheme Π and AEAD encoding scheme $\mathtt{Coding} = (\mathtt{Ecd}, \mathtt{Dcd})$, with $\mathtt{TEST}i$ as shown in Fig. 3.

Security of the P_{AEAD} construction follows analogously:

Theorem 3.1. *Let Π be a secure level-1 AEAD scheme and $\mathtt{Coding}$ be an AEAD encoding scheme with collision-resistant encoding. Let $\mathtt{TEST}i$ be defined as in Fig. 3 and $i \in \{2, 3, 4\}$. Then $\Pi'_i = P_{\mathrm{AEAD}}(\Pi, \mathtt{Coding}, \mathtt{TEST}i)$, constructed as in Fig. 5, is a secure level-i AEAD scheme. Specifically, let $\mathcal{A}$ be an adversary algorithm that runs in time t and asks q_e $\mathsf{Encrypt}$ queries and q_d $\mathsf{Decrypt}$ queries, and let $q = q_e + q_d$. Then there exists an adversary $\mathcal{B}$ that runs in time $t_{\mathcal{B}} \approx t$ and asks no more than $q_{\mathcal{B}} = \frac{1}{2} q_e(q_e - 1)$ queries, and an adversary $\mathcal{F}$ that runs in time $t_{\mathcal{F}} \approx t$ and asks $q_{\mathcal{F}} = q$ queries, such that*

$$\mathbf{Adv}^{\mathsf{aead}_i}_{P_{\mathrm{AEAD}}(\Pi, \mathtt{Coding}, \mathtt{TEST}i)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{aead}_1}_{\Pi}(\mathcal{F}) + \mathbf{Adv}^{\mathrm{collision}}_{\mathtt{Ecd}}(\mathcal{B}) \ .$$

The proof of Theorem 3.1 is similar to that of Theorem 2.2 (Appendix A).

4 Authenticated Encryption in TLS

The work of Paterson et al. [KPW13] showed that the MAC-then-encode-then-encrypt mode of CBC encryption in TLS 1.2 (with sufficiently long MAC tags) is a secure length-hiding authenticated encryption (LHAE) scheme, assuming the encryption function is a strong pseudorandom permutation and the MAC is a pseudorandom function. Their definition corresponds to level 1 of our AEAD hierarchy. Several subsequent work on the provable security of TLS, such as that of Jager et al. [JKSS12] and Krawczyk et al. [KPW13], assume that the TLS record layer is a secure *stateful* length-hiding authenticated encryption (sLHAE) scheme, corresponding to level 4 of our AEAD hierarchy. To our knowledge, there has as of yet been no formal connection between the LHAE result of Paterson et al. and the sLHAE requirement of subsequent works; we address that gap in this section by bringing sequence numbers into the modeling using the framework in the previous sections.

4.1 TLS sequence numbers and authentication level

The TLS record layer utilizes sequence numbers to ensure detection of deleted or reordered records [DR08, p. 94]. Being 64-bits long, sequence number exhaustion for any given connection is unlikely and the specification demands renegotiation should it occur. Sequence numbers are sent implicitly by inclusion under the MAC (or AEAD). When instantiated, "the first record transmitted under a particular connection state MUST use sequence number 0" [DR08, §6.1] and each subsequent record increments the sequence number. Sequence numbers are continuous across record types (application and alert).

When the ciphersuite uses MAC-then-encode-then-encrypt, the MAC tag is computed as follows, where k is the MAC key (either `MAC_write_key` or `MAC_read_key`, depending on the direction), `sqn` is the 64-bit sequence number, and `m` is the (possibly compressed) TLS plaintext object (called `TLSCompressed`) [DR08]:

$$\mathrm{MAC}(k, \texttt{sqn} \parallel \texttt{m.type} \parallel \texttt{m.version} \parallel \texttt{m.length} \parallel \texttt{m.fragment}) \ .$$

Since the sequence number is implicit, a receiver will check the MAC verification using the expected sequence number. If the check fails, a `bad_record_mac` alert (type 20) will be generated – an alert that is always fatal [DR08, §7.2.2].

When the ciphersuite is uses a combined AEAD scheme, the sequence number, as well as several other values, are included in the additional data field [DR08]:

$$\texttt{ad} = \texttt{sqn} \parallel \texttt{m.type} \parallel \texttt{m.version} \parallel \texttt{m.length} \ .$$

The ciphertext is then

$$c \leftarrow \mathrm{Encrypt}(k, \texttt{m.length}, \texttt{ad}, \texttt{m.fragment}, st_{\mathrm{E}}) \ .$$

The sequence number is not transmitted in the ciphertext. AEAD decryption is applied using the expected sequence number. Decryption failure must also result in a `bad_record_mac` fatal alert [DR08, §6.2.3.3].

4.2 From TLS Level-1 AEAD to Level-4 AEAD

Paterson et al. [PRS11] show that a simplified version of TLS MAC-then-encode-then-encrypt, which we call Π_{PRS} and describe in the top half of Fig. 6, satisfies level-1 AEAD security. By design, Π_{PRS} includes the sequence number field in the `ad`, but never initializes it as Π_{PRS} is not stateful. However, the TLS record layer protocol as actually used is stateful and, as such, ought to achieve a higher level of AEAD; namely, it should satisfy level-4 AEAD. The bottom half of Fig. 6 shows the TLS MAC-then-encode-then-encrypt record layer with the use of sequence numbers as specified in the standard.

Our framework allows us to immediately show that Π_{TLS} satisfies level-4 AEAD security: we incorporate the sequence numbers in an implicit AEAD encoding scheme $\texttt{Coding}_{TLS}$, and then view Π_{TLS} as the result of applying the P_{AEAD} construction to Π_{PRS} and $\texttt{Coding}_{TLS}$.

Define AEAD encoding scheme $\texttt{Coding}_{TLS} = (\texttt{Ecd}_{TLS}, \texttt{Dcd}_{TLS})$ as follows:

- $\texttt{Ecd}_{TLS}(\texttt{sqn}, \texttt{ad}, m) = (\texttt{sqn} \| \texttt{ad}, m)$
- $\texttt{Dcd}_{TLS}(\texttt{sqnlist}, \texttt{sqn} \| \texttt{ad}, m) = (\texttt{sqn}, \texttt{ad}, m, \alpha)$

where $\alpha = 1$ if and only if `sqn` and `sqnlist` satisfy `TEST4` in Fig. 3, $\texttt{ad} \neq \perp$, and $m \neq \perp$.

Theorem 4.1. $\Pi_{TLS} = P_{AEAD}(\Pi_{PRS}, \texttt{Coding}_{TLS}, \texttt{TEST4})$.

$\Pi_{PRS}.\mathrm{E}(k, \ell, \mathsf{ad}, m, \perp)$:
1: $(k_m, k_e) \leftarrow k$
2: $t \leftarrow \mathrm{MAC}(k_m, \mathsf{ad}, m)$
3: $c \leftarrow \mathrm{E}(k_e, \ell, m, t)$
4: **return** $(c, \perp)$

$\Pi_{PRS}.\mathrm{D}(k, \mathsf{ad}, c, \perp)$:
1: $(k_m, k_e) \leftarrow k$
2: $(m, t, \alpha) \leftarrow \mathrm{D}(k_e, c)$
3: **if** $\mathrm{MAC}(k_m, \mathsf{ad}, m) \neq t$ **then**
4: **return** $(\perp, 0, \perp)$
5: **return** $(m, \alpha, \perp)$

$\Pi_{TLS}.\mathrm{E}(k, \ell, \mathsf{ad}, m, st_{\mathrm{E}})$:
1: $(k_m, k_e) \leftarrow k$
2: $t \leftarrow \mathrm{MAC}(k_m, st_{\mathrm{E}}.\mathsf{ctr} \| \mathsf{ad}, m)$
3: $c \leftarrow \mathrm{E}(k_e, \ell, m, t)$
4: $st_{\mathrm{E}}.\mathsf{ctr} \leftarrow st_{\mathrm{E}}.\mathsf{ctr} + 1$
5: **return** (c, st_{E})

$\Pi_{TLS}.\mathrm{D}(k, \mathsf{ad}, c, st_{\mathrm{D}})$:
1: $(k_m, k_e) \leftarrow k$
2: **if** $st_{\mathrm{D}}.\mathsf{status} = \mathsf{failed}$ **then**
3: **return** $(\perp, 0, st_{\mathrm{D}})$
4: $(m, t, \alpha) \leftarrow \mathrm{D}(k_e, c)$
5: **if** $\mathrm{MAC}(k_m, st_{\mathrm{D}}.\mathsf{ctr} \| \mathsf{ad}, m) \neq t$ **then**
6: $\alpha \leftarrow 0$
7: **if** $\alpha = 0$ **then**
8: $st_{\mathrm{D}}.\mathsf{status} \leftarrow \mathsf{failed}$
9: **return** $(\perp, 0, st_{\mathrm{D}})$
10: $st_{\mathrm{D}}.\mathsf{ctr} \leftarrow st_{\mathrm{D}}.\mathsf{ctr} + 1$
11: **return** $(m, \alpha, st_{\mathrm{D}})$

Figure 6: Construction of AEAD schemes Π_{PRS} (Paterson et al. [PRS11] variant of TLS MAC-then-encode-then-encrypt) and Π_{TLS} (TLS MAC-then-encode-then-encrypt) from encode-then-encrypt scheme (E, D).

Theorem 4.1 follows semantically comparing Π_{TLS} and the scheme resulting from the construction $P_{AEAD}(\Pi_{PRS}, \mathtt{Coding}_{TLS}, \mathtt{TEST4})$.

Clearly, $\mathtt{Ecd}_{TLS}$ is collision-resistant due to the unambiguous parsing of `sqn` as a fixed-length 64-bit value. We can thus apply Theorem 3.1 to obtain Corollary 4.1.1.

Corollary 4.1.1. *The TLS record layer with MAC-then-encode-then-encrypt in CBC mode satisfies level-4 AEAD security. Specifically, let $\mathcal{A}$ be an adversary algorithm that runs in time t against Π_{TLS}. Then there exists an adversary $\mathcal{F}$ that runs in time $t_{\mathcal{F}} \approx t$ such that*

$$\mathbf{Adv}^{\mathsf{aead}_4}_{\Pi_{TLS}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{aead}_1}_{\Pi_{PRS}}(\mathcal{F}) \ .$$

From Paterson et al. [PRS11] we know that the TLS record layer encryption in MAC-then-encode-then-encrypt CBC mode satisfies AEAD level-1 security when a secure cipher and message authentication code is used. Combined with Corollary 4.1.1, this means that the sLHAE security definition used by Jager et al. [JKSS12] and Krawczyk et al. [KPW13] in their analyses of full TLS ciphersuites is achieved, and thus TLS is ACCE secure in this scenario.

References

[APW09] Martin R. Albrecht, Kenneth G. Paterson, and Gaven J. Watson. Plaintext recovery attacks against SSH. In *2009 IEEE Symposium on Security and Privacy*, pages 16–26, Oakland, California, USA, May 17–20, 2009. IEEE Computer Society Press.

[BDPS12] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. Security of symmetric encryption in the presence of ciphertext fragmentation. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 682–699, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[BKN02] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In Vijayalakshmi Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 1–11, Washington D.C., USA, November 18–22, 2002. ACM Press.

[BMM+15] Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. Augmented secure channels and the goal of the TLS 1.3 record layer. Cryptology ePrint Archive, Report 2015/394, 2015. `http://eprint.iacr.org/2015/394`.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[DR08] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*, 2008. RFC 5426, `https://tools.ietf.org/html/rfc5426`.

[FFL12] Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A family of almost foolproof on-line authenticated encryption schemes. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 196–215, Washington, DC, USA, March 19–21, 2012. Springer, Heidelberg, Germany.

[FGMP15] Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. Data is a stream: Security of stream-based channels. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 545–564, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[HKR14] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption: AEZ and the problem that it solves. Cryptology ePrint Archive, Report 2014/793, 2014. `http://eprint.iacr.org/2014/793`.

[HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online authenticated-encryption and its nonce-reuse misuse-resistance. Cryptology ePrint Archive, Report 2015/189, 2015. `http://eprint.iacr.org/2015/189`.

[IEE12] IEEE 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2012. `http://dx.doi.org/10.1109/IEEESTD.2012.6178212`.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[Ken05] S. Kent. *IP Authentication Header*, 2005. RFC 4302, `https://tools.ietf.org/html/rfc4302`.

[KPB03] Tadayoshi Kohno, Adriana Palacio, and John Black. Building secure cryptographic transforms, or how to encrypt and MAC. Cryptology ePrint Archive, Report 2003/177, 2003. `http://eprint.iacr.org/2003/177`.

[KPW13] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[KY01] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299, New York, NY, USA, April 10–12, 2001. Springer, Heidelberg, Germany.

[LC15] A. Langley and W.-T. Chang. *QUIC Wire Layout Specification*, 2015. `https://docs.google.com/document/d/1WJvyZflAO2pq77yOLbp9NsGjC1CHetAXV8I0fQe-B_U`.

[LJBN15] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. How secure and quick is QUIC? Provable security and performance analyses. In *2015 IEEE Symposium on Security and Privacy*, pages 214–231, San Jose, California, USA, May 17–21, 2015. IEEE Computer Society Press.

[MT10] Ueli Maurer and Björn Tackmann. On the soundness of authenticate-then-encrypt: formalizing the malleability of symmetric encryption. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 505–515, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.

[Nam02] Chanathip Namprempre. Secure channels based on authenticated encryption schemes: A simple characterization. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 515–532, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

[PRS11] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 372–389, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 196–205, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.

[RM06] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security*, 2006. RFC 4347, `https://tools.ietf.org/html/rfc4347`.

[RM12] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*, 2012. RFC 6347, `https://tools.ietf.org/html/rfc6347`.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 02: 9th Conference on Computer and Communications Security*, pages 98–107, Washington D.C., USA, November 18–22, 2002. ACM Press.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[Shr04] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272, 2004. `http://eprint.iacr.org/2004/272`.

A Appendix

A.1 Proof of Theorem 2.1

Proof. $\mathcal{A}$ starts Π.
For $i = 1$. $\mathcal{A}$ wins as a level-1 adversary with a message c_{win}, where $\nexists w : c_{\text{win}} = sent_w$. Then $\mathcal{A}$ will also win against Π as a level-2 adversary.
For $i = 2$.
Case 1: $\mathcal{A}$ wins by forging c_{win}. This follows as for $i = 1$, above.
Case 2: $\mathcal{A}$ wins against Π as a level-2 adversary with $c_{\text{win}} = rcvd_v$, where there exists some $w < v$ such that $c_{\text{win}} = rcvd_w$ for some $sent_w \in Sent$.

In this case $\mathcal{A}$ wins by replay with $c_{\text{win}} = rcvd_v = sent_y$, for some y, since c_{win} is not a forgery. Let $rcvd_w = sent_x$, for some x. Then it follows that $sent_y = rcvd_v = c_{\text{win}} = rcvd_w = sent_x$ and, since $\mathcal{A}$ wins by replay, $x = y$. Finally, combining the above facts, there exist w, x, y such that $(w < v) \wedge (sent_x = rcvd_w) \wedge (sent_y = rcvd_v) \wedge (x = y)$. Ergo, $\mathcal{A}$ wins as a level-3 adversary .
For $i = 3$.
Case 1: This follows as for $i = 1$, above.
Case 2: $\mathcal{A}$ wins as a level-3 adversary against Π with $c_{\text{win}} = rcvd_v$, where there exist w, x, y such that $(w < v) \wedge (sent_x = rcvd_w) \wedge (sent_y = rcvd_v) \wedge (x \geq y)$. By assumption, it follows that $c_{\text{win}} \neq sent_v$; therefore $\mathcal{A}$ also wins as a level-4 adversary. □

A.2 Proof of Theorem 2.2

Due to the similarities in the proofs for the various levels in Theorem 2.2, we underscore the key differences in the proofs using a substitutable *inhibitor step* for conciseness, which indicates the section that changes appropriately for the different conditions being checked.

Proof. Let $\mathcal{A}$ be an adversary against the authentication scheme Π' with $\texttt{TEST}i$, winning according to the Level i authentication experiment. Let $\mathbf{Adv}_0$ and $\mathbf{Adv}_1$ denote $\mathcal{A}$'s advantage in games 0 and 1, respectively.

Game 0. Real experiment: $\mathbf{Adv}_0 = \mathbf{Adv}^{\mathsf{auth}_i}_{P(\Pi,\texttt{Coding},\texttt{TEST}i)}(\mathcal{A})$.

Game 1. Proceed as in Game 0. Should the adversary select messages m and m' such that $(\texttt{sqn}_m, m)$ and $(\texttt{sqn}_{m'}, m')$ are distinct, but $\texttt{Ecd}(\texttt{sqn}_m, m) = \texttt{Ecd}(\texttt{sqn}_{m'}, m')$, then the challenger will abort.

If $\mathcal{A}$ does output messages m and m' as described above, an adversary $\mathcal{B}$ can be constructed against the collision resistance of $\texttt{Ecd}$. Let $\mathcal{B}$ have an $\texttt{Ecd}$ oracle. For every message m output by $\mathcal{A}$, $\mathcal{B}$ incorporates the appropriate sequence number to make the pair $(\texttt{sqn}_m, m)$ and checks if $\texttt{Ecd}(\texttt{sqn}_m, m) = \texttt{Ecd}(\texttt{sqn}_{m^*}, m^*)$ for all m^* previously queried by $\mathcal{A}$. When $\mathcal{A}$ outputs m and m', $\mathcal{B}$ will detect that $\texttt{Ecd}(\texttt{sqn}_m, m) = \texttt{Ecd}(\texttt{sqn}_{m'}, m')$, and will use $(\texttt{sqn}_m, m)$ and $(\texttt{sqn}_{m'}, m')$ to win against the collision resistance of $\texttt{Ecd}$. Hence,

$$\mathbf{Adv}_0 \leq \mathbf{Adv}_1 + \mathbf{Adv}^{\text{collision}}_{\texttt{Ecd}}(\mathcal{B}) \ .$$

For $\mathbf{Adv}_1$, an upper bound can be obtained as follows:
Adversary $\mathcal{F}$ starts schemes Π and Π'. Per the scheme, all the following variables are set: $st_{\text{E}} \leftarrow \perp, st_{\text{D}} \leftarrow \perp, u \leftarrow 0$, and $v \leftarrow 0$. $\mathcal{A}$ selects $\texttt{sqn}_0$. $\mathcal{F}$ answers all of $\mathcal{A}$'s Π'.Snd and Π'.Rcv queries with his Π.Snd and Π.Rcv oracles, applying new sequence numbers for each new Snd query (i.e. incrementation of u) with his $\texttt{Ecd}$ and $\texttt{Dcd}$ algorithms.

When $\mathcal{A}$ asks a Π'.Snd query with a message $m_{\Pi'}$, $\mathcal{F}$ encodes $st'_{\text{E}}.\texttt{ctr}$ with $m_{\Pi'}$, creating the new message $m_\Pi = \texttt{Ecd}(st'_{\text{E}}.\texttt{ctr}, m_{\Pi'})$. In turn, $\mathcal{F}$ forwards this message to his Π.Snd oracle and passes the result, c back to $\mathcal{A}$. When $\mathcal{A}$ asks a Π'.Rcv query with an authenticated message c, $\mathcal{F}$ passes c to his Π.Rcv oracle which outputs either $\perp$ or a successfully received message $m_\Pi = \texttt{Ecd}(st'_{\text{E}}.\texttt{ctr}, m_{\Pi'})$.

If $\mathtt{sqn}$ is sent explicitly, Π.Rcv uses the $\mathtt{Dcd}$ algorithm to check $m_{\Pi'}$ and $\mathtt{sqn}$. If the decoding fails, $\mathcal{F}$ outputs $\perp$. If $\mathtt{sqn}$ is sent implicitly, $\mathcal{F}$ checks the decoding against $\{st'_{\mathrm{E}}.\mathsf{ctr}\}$; a list of all counters previously encoded with messages by $\mathcal{F}$ before being input into Π.Snd, in response to Π'.Snd queries. If $\nexists j$ such that $st'_{\mathrm{E}}.\mathsf{ctr}_j$ yields a valid decoding, $\mathcal{F}$ passes $\perp$ back to $\mathcal{A}$; else he decodes the message into $\mathtt{sqn}$ and $m_{\Pi'}$.

Inhibitor Step $i = 2$: Should $\mathcal{A}$ attempt to win by replay and issue a Π'.Rcv query on a previously queried authenticated message c, $\mathcal{F}$'s Π.Rcv oracle and subsequent $\mathtt{Dcd}$ algorithm will output $\mathtt{sqn}$ and $m_{\Pi'}$. $\mathcal{F}$ checks $\mathtt{sqn}$ against $st_{\mathrm{D}}.\mathsf{sqnlist}$ and will discover that $\mathtt{sqn} = st_{\mathrm{D}}.\mathsf{sqnlist}_j$ for some j, passing $\perp$ back to $\mathcal{A}$.

Inhibitor Step $i = 3$: Should $\mathcal{A}$ attempt to win by replay, $\mathcal{F}$ will act as in the *Inhibitor Step* for $i = 2$, passing $\perp$ to $\mathcal{A}$. If the sequence number corresponding to c is not strictly greater than the one previously output by Π.Rcv, $\mathcal{F}$'s call to its Π.Rcv oracle on c will output an encoded message on which $\mathcal{F}$ will subsequently use his $\mathtt{Dcd}$ algorithm to obtain $\mathtt{sqn}$. $\mathcal{F}$ then checks that $\mathtt{sqn} > st_{\mathrm{D}}.\mathsf{sqnlist}_j$ for all j, passing $\perp$ back to $\mathcal{A}$ if the check fails.

Inhibitor Step $i = 4$: Should $\mathcal{A}$ attempt to win by replay with a authenticated message c or if the sequence number corresponding to c is not strictly greater than the previous one received, $\mathcal{F}$ behaves as described in the *Inhibitor Step* for $i = 2$ and $i = 3$, respectively, passing $\perp$ to $\mathcal{A}$.

Should the sequence number corresponding to c not be the next expected sequence number, $\mathcal{F}$'s call to its Π.Rcv oracle on c will output an encoded message from which $\mathcal{F}$ subsequently obtains $\mathtt{sqn}$ via his $\mathtt{Dcd}$ algorithm. F then checks $\mathtt{sqn}$ against the ordered list of all previously received sequence numbers; if $\mathtt{sqn} \neq \max\{st'_{\mathrm{D}}.\mathsf{sqnlist}_j\} + 1$, i.e. if $\mathtt{sqn}$ is not precisely the next expected sequence number, then $\mathcal{F}$ passes $\perp$ back to $\mathcal{A}$.

Since $\mathcal{A}$ wins the level-i game against Π' with non-negligible advantage $\mathbf{Adv}^{\mathsf{auth}_i}_{P(\Pi,\mathtt{Coding},\mathtt{TEST}i)}(\mathcal{A})$, and $\mathcal{A}$ is not able to win by any other means than forgery, by the above behavior of $\mathcal{F}$, then at some time $\mathcal{A}$ must make a Π'.Rcv query on a message c_{win} such that $\nexists w : c = sent_w$. By the success of $\mathcal{A}$, c_{win} will be successfully received. Moreover, due to the collision resistance of $\mathtt{Coding}$, c_{win} must break the authentication attributes of Π – consequently $\mathcal{F}$ will also win the level-1 game against Π with c_{win}. Hence,

$$\mathbf{Adv}_1 \leq \mathbf{Adv}^{\mathsf{auth}_1}_{\Pi}(\mathcal{F}) \ .$$

□