A Survey of TLS 1.3 Record Protocol

Yuheng (Elle) Wen[1]

***Abstract*— The TLS 1.3 Record Protocol is a crucial component of the TLS protocol suite, aiming for providing a secure communication channel between a browser and a server. In this paper, I explain the construction of the TLS 1.3 Record Protocol based on the most updated RFC document. My main work lies in the security analysis of the TLS 1.3 Record Protocol, where I first explain the basic concepts and proof techniques used in constructive cryptography proposed by U. Maurer [2], then review the security proof of TLS 1.3 Record Protocol given by C. Badertscher et al [1]. Then I discuss the performance of TLS 1.3 Record Protocol against length field attack, replaying attack, cookie cutter attack, Lucky 13, and BEAST. I aim to provide an accessible explanation about the construction and security analysis of TLS 1.3 Record Protocol, which can be useful for anyone interested in the security of TLS.**

I. INTRODUCTION

The TLS (Transport Layer Security) protocol is a widely used cryptographic protocol that provides secure communication over the internet. It is widely used to secure web traffic, email, instant messaging, and other forms of online communication. The TLS protocol consists of several layers, each with a specific responsibility in the secure communication process. The TLS Handshake Protocol is a critical component of the TLS protocol, responsible for establishing the cryptographic parameters of the session, authenticate the server to the client, and optionally authenticate the client to the server.

The focus of this paper is the other critical components of the TLS protocol which is the TLS Record Protocol. The TLS Record Protocol is responsible for fragmenting messages into manageable blocks, adding integrity and confidentiality protection to these blocks, and delivering them to the recipient in a secure and reliable manner.

[1]This is the term paper for CSCI 388 Cryptography at Reed College, Spring 2023. Course Instructor: Prof, Chanathip (Meaw) Namprempre

TLS 1.3, the latest version of the TLS protocol, introduced significant changes to the Record Protocol to improve security, performance, and privacy. In particular, TLS 1.3 Record Protocol eliminated support for many older cryptographic algorithms and replaced them with newer, more secure algorithms. It also simplified the handshake process, reducing the number of round trips required to establish a secure connection, and introduced a zero-RTT mode that allows clients to resume a previous session without a full handshake.

Given the critical role of the TLS Record Protocol in securing online communication, it is essential to understand its inner workings and its security conditions.

II. LITERATURE REVIEW

While there are a great amount of paper that discuss the attacks of the TLS record layer for prior version such as the Lucky Thirteen [5] and the BEAST, there is minimal for 1.3 version.

A long line of work analyzes the security of TLS for versions prior to 1.3. Several recent papers use a game-based definition (ACCE) that models both the handshake and the record layer at the same time, since TLS versions prior to 1.3 could formally not be proved as the composition of the two sub-protocols. Thanks to the adoption of AEAD and the better separation of the two sub-protocols in TLS 1.3, looking at the security of the two subprotocols separately becomes possible and meaningful. Two attempts have been made that took different approaches. A. Delignat-Lavaud et al. [3] gave a proof using game based reductions and typing. C. Badertscher et al. [1] gave a proof using Augmented Secure Channels (ASC) and constructive cryptography. This paper reviews the later proof.

III. Construction

The most updated standard document for TLS 1.3 specifies the implementation of the TLS 1.3 record protocol [4]. In the setting of a browser-server communication as shown in Figure 1, let A be a browser and B be a server. During the hand-

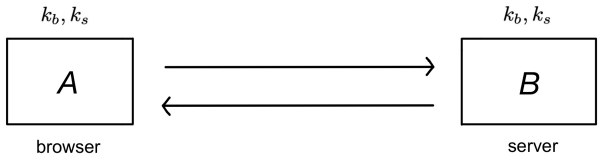

Fig. 1. A browser-server communication setting.

shake protocol, A and B have already established two shared keys, denoted as k_a and k_b. Specifically, k_b is the browser key used later in encrypting or decrypting messages coming from the browser, and k_s is the server key for encrypting or decrypting messages coming from the server. When A wants to deliver a message m to B, it first formats m to be *TLSInnerPlaintext* as shown in Figure 2, where the type bits are encoded to represent the type of the message m, 0^n are padding bits, and m is the actual content A wants to send.

type	0^n	m

Fig. 2. TLSInnerPlaintext. The type bit is a single bit to represent one of handshake type(data sent during handshake protocol), alert data type(alert data sent to terminate the communication), and application data type(data sent during actual communication of A and B). The number of 0 used for padding are determined by the parameters shared by A and B during handshake protocol

Next, A uses an Authenticated Encryption with Associated Data (AEAD) algorithm that is agreed by both A and B during handshake protocol to encrypt *TLSInnerPlaintext*:

c $\leftarrow$ AEAD(k_b or k_s, nonce, *TLSInnerPlaintext*, additional data)

The key used is either k_b or k_s depending on which party the message comes from. The nonce is created as follows:

$$\text{nounce} = (0^n \parallel \text{sequence number}) \oplus \text{IV}$$

The 64 bit sequence number is maintained separately for encrypting and decrypting *TLSInnerPlaintext*. It is initialized to be 0 every time a new key is used, and it increments by 1 after encrypting or decrypting each *TLSInnerPlaintext*. IV is a static value determined during handshake, and it can be server_IV or browser_IV depending on which party the message comes from. The "additional data" is obtained by concatenating a fixed value 23, a redundant value 0x0303, and the length of c:

$$\text{additional data} = 23 \parallel 0x0303 \parallel |c|$$

Finally, the ciphertext c is formatted as a record that is sent to the internet for transmission (Figure 3). When B received the record, it runs $AEAD^{-1}$

opaque_type	version	length	c

Fig. 3. TLS record. Opaque_type is always set to the value 23 for for outward compatibility with middleboxes accustomed to parsing previous versions of TLS. Version is set to be 0x0303. Length is the sum of the lengths of plus one for the content type, plus any expansion added by the AEAD algorithm.

to decrypt c and gets the actual content m.

IV. Proof of Security

A. Overview

This section will only focus on C. Badertscher et al.'s approach [1] to prove the security of TLS 1.3 record layer. The authors introduced a new abstraction of a secure channel called Augmented Secure Channels (ASC), and then proceed to prove that an insecure channel (IC) can securely construct ASC. Finally, they demonstrated that ASC can securely construct the TLS 1.3 record protocol. Before delving into the details of the proof, it is necessary to explain the rationale behind constructive cryptography.

B. Constructive Cryptography: basic concepts

U. Maurer [2] proposed constructive cryptography as a new paradigm for security definition and proofs in 2012. In constructive cryptography, any abstract object with **interfaces** is considered a **system**. In other words, any cryptographic entity, from low-level tools(ex. block ciphers), higher level primitives(ex. digital signatures), to highest level protocols(ex. key exchange protocols), is viewed as a system. An interface enables a system to interact with the environment or other systems. Analogous to I/O interface defined in the context

of processor where it means any interface that transfers data between the CPU and the rest of the world, interface of a system simply means any input/output interface of any cryptographic entity. Notably, two system can be composed into a single system by connecting their interfaces.

We consider three special types of systems, **resource systems**, **converter systems**, and **distinguisher systems**.

A resource system provides a certain service to parties (usually Alice, Bob, and Eve). Typically (but not always), a resource system contains interfaces that access all parties. Figure 4 shows a simple shared-secrete-key resource that can be constructed by some key exchange protocol. The service of this system is to provide shared secrete key to party A and party B. This resource system initially chooses a key k from some key distribution $\mathcal{K}$. Interface A connects $SK_{\mathcal{K}}$ to party A. When party A wants to get the shared secrete key, it sends the input "getKey" to $\text{SK}_{\mathcal{K}}$, then $SK_{\mathcal{K}}$ output k to party A. An illustration is shown in Figure 5.

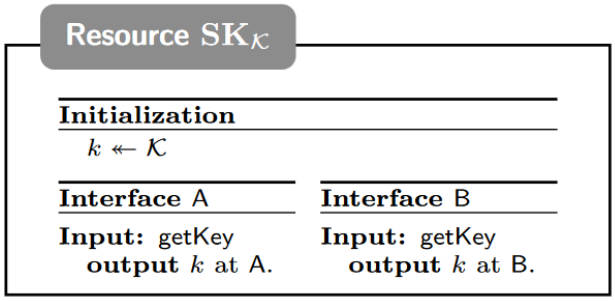

Fig. 4. The shared secrete key resource [1, Fig 4]

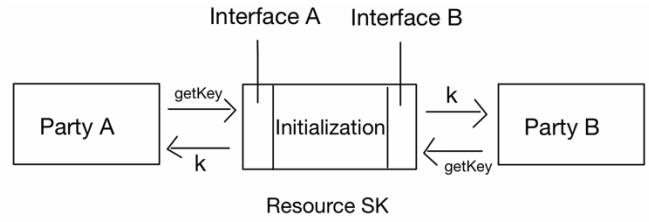

Fig. 5. The shared secrete key resource demonstration

A converter system is a system with an *inside* interface and an *outside* interface. Figure 6 shows that a converter α is connected to a resource R by connecting the inside interface of α to the interface i of R. The outside interface of α now serves as the new interface for the combined system, which is again a resource system denoted by $\alpha^i R$. An easy way to make sense of this notation is reading it as "α connects to R through interface i".

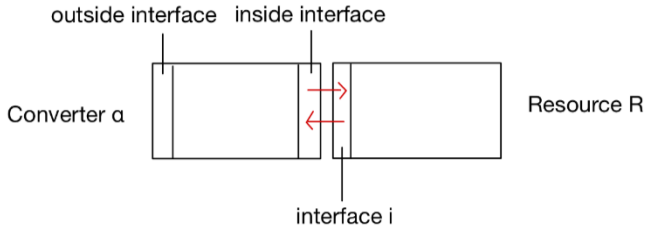

Fig. 6. A new resource system $\alpha^i R$ created by combining α and R through interface i.

A distinguisher system D is designed specifically to capture the notion of two resource systems S and T "behaving essentially identically". D provides inputs to the connected resource system and receives the outputs generated by the resource. For example, a distinguisher D for S and T is combined with one of S or T. Then D repetitively sends inputs to any interface of S or T and receives the corresponding outputs. D stops until it outputs a bit 0 or 1 that indicates its guess to which system it is connected. The distinguishing advantage of D for S and T is defined as

$$\triangle^D(S,T) = Pr[DS \to 1] - Pr[DT \to 1]$$

In words, it means the advantage of D to distinguish S and T equals the probability of D outputs 1 when connecting to S minus the probability of D outputs 1 when connecting to T. Notably, we define the "difference" between system S and T denoted as $d(S,T)$ to be equal to $\triangle^D(S,T)$. Figure 7 illustrates how D works.

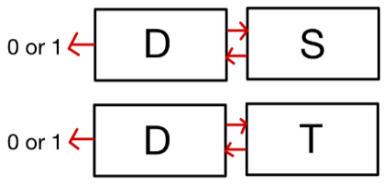

Fig. 7. The distinguisher D for S and T. D is connected to either S or T. D needs to guess which one it has been connected to by adaptively sending input and receiving output from its connected resource. If it guesses S, it outputs 1. If it guesses T, it outputs 0.

C. Constructive Cryptography: proof of security

As the name suggests, constructive cryptography is all about construction. A system is proved to be secure if it can be securely constructed from some insecure resource. A general description of the proof process is as follows. Suppose we want to have a secure resource that provides a specific service. We first spell out what an ideal resource should be able to do. Then we spell out what the adversary should be able to do and should not be able to do in the ideal resource. Based on these, we then create two modes of an ideal resource, S for the adversary is present and $\perp^E S$ for the adversary is not present. $\perp^E$ is a converter that "shields" the interface of the adversary (can be done easily by making the adversary sends inputs when c = 1, and set c = 0). Next, we spell out the resource R of the starting point, or simply "what we can do right now". So now we have what it is called "the real resource" R and "the ideal resource" S. The next step and also the hardest step is to create some converters π_1, π_2 such that the converters "securely construct" S from R.

Definition (secure construction). For resource R and S we say that converters (π_1, π_2) securely construct S from R within arbitrarily small value ϵ, denoted

$$R \xrightarrow{(\pi_1,\pi_2,\epsilon)} S,$$

if the following two conditions are satisfied:

$$d(\pi_1^A \pi_2^B \perp^E R, \perp^E S) \leq \epsilon \quad (1)$$

$$d(\pi_1^A \pi_2^B R, \sigma^E S) \leq \epsilon \quad (2)$$

where σ is a converter that simulates the behaviors of the adversary present in R, and A, B are interfaces.

Finally, once we've found such converters π_1, π_2 and have proved the two conditions are met, we've found a resource $\pi_1^A \pi_2^B R$ that is indistinguishable from the ideal resource S. Hence $\pi_1^A \pi_2^B R$ must be the secure resource that we wanted.

D. TLS 1.3 Record Protocol: proof of security

Follows from the approach outlined in the previous section, we proceed to construct a secure TLS 1.3 Record Protocol. The first step is to spell out an ideal resource based on what we think a good TLS record protocol should provide. In many relevant communication channels, transmitted data packets have two parts. The first part is a header that contains the information needed for transmission and requires authenticity but not confidentiality, and the second part is a payload that contains the encrypted message and hence needs both authenticity and confidentiality. We further observes that a header conceptually can be split into two parts: an implicit part which is the context that is known by the receiver, and an explicit part that is unknown to the receiver. The ideal resource that is designed to formalize these service is called a Augmented Secure Channel(ASC) which is presented in Figure 8.

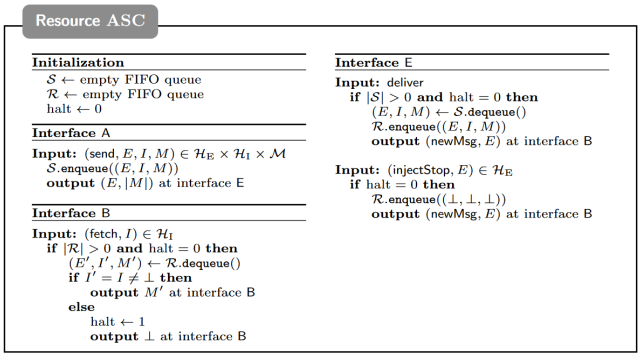

Fig. 8. The augmented secure channel resource [1, Fig 3].

Interface A is the sender. It sends an explicit part of the header E, an implicit part of the header I, and the message M to the channel. In addition, we assume that only the information about the length of the message can be leaked so only $|M|$ is sent. Interface E is the channel between the sender A and the receiver B. E and $|M|$ are delivered at E. What E is allowed to do is either deliver all the inputs to the receiver B, or inject special element $\perp$ that will terminate the channel at the receiver's side. When interface B received new messages, it first fetches the context information it has and verifies if the received header is legitimate or not. If yes, it outputs the message M. If not, it terminates. This can be implemented using a FIFO queue at the sender's side and the receiver's side respectively.

The next step is to formalize the assumed resource. We start our construction having only an insecure channel IC and and shared-key resource

SK built during handshake protocol. We denote this by $[IC, SK_{\mathcal{K}}]$. See Figure 9 and Figure 4. In IC, the message M is sent in clear, and an adversary can inject any message of its choice to receiver B.

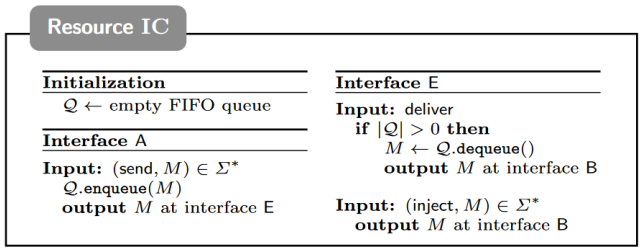

Fig. 9. The insecure channel resource [1, Fig 2]. Interface B simply receives any message that is sent by E and is omitted here.

Then we propose two converters enc_π and dec_π (Figure 10) and prove that they securely construct ASC from IC using the definition of secure construction. The enc_π converter first gets a key

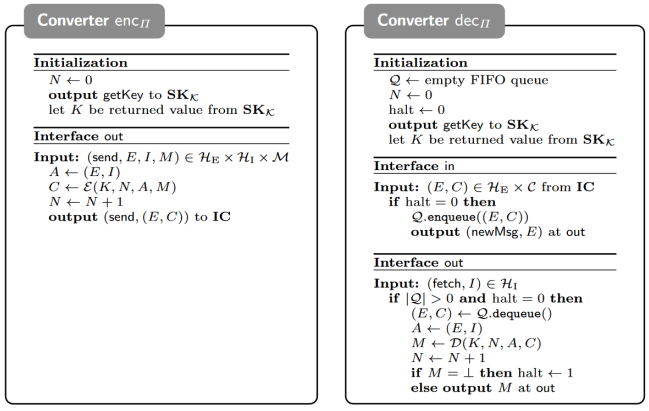

Fig. 10. [1,Fig 5]

K by asking $SK_{\mathcal{K}}$. Then for every packet that the sender wants to send, it creates an "additional data" A using E and I, and then uses and AEAD encryption algorithm $\mathcal{E}$ to encrypt the message M with a nonce N created by incrementing 1 for every encryption to get C. It then sends (E, C) to IC where IC performs simple delivery. The dec_π converter first gets the key from $SK_{\mathcal{K}}$ and upon receiving (E, C) from IC at interface *in*, it delivers the inputs to interface *out*. At interface *out*, it decrypts C using decryption algorithm of AEAD to get back M and sends it to the receiver. The combined system of the two converts and $[IC, SK_{\mathcal{K}}]$ is denoted by $enc_\pi^A dec_\pi^B[IC, SK_{\mathcal{K}}]$ as illustrated in Figure 12.

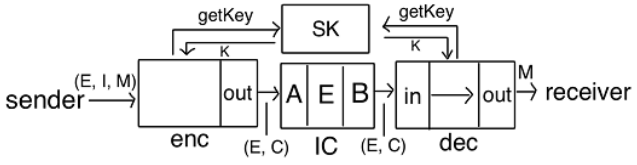

Fig. 11. The combined system $enc_\pi^A dec_\pi^B[IC, SK_{\mathcal{K}}]$

Next, we want to prove the two conditions (1) and (2) are met.

Condition 1. For the first condition when the adversary is absent, we want to prove that for a distinguisher D, the value of

$$d(enc_\pi^A dec_\pi^B \perp^E [IC, SK_{\mathcal{K}}], \perp^E ASC)$$
$$= \triangle^D(enc_\pi^A dec_\pi^B \perp^E [IC, SK_{\mathcal{K}}], \perp^E ASC)$$
$$= Pr[D(enc_\pi^A dec_\pi^B \perp^E [IC, SK_{\mathcal{K}}]) \rightarrow 1] - Pr[D(\perp^E ASC) \rightarrow 1]$$

is bounded. Roughly speaking, we found that the security of AEAD implies the security of ASC. The security game for AEAD is defined using the common real or ideal game as shown in Figure 12. An adversary A needs to output a bit 1 or 0 indicating which oracle it thinks it gets. The advantage of A is defined as

$$Adv_\pi^{ae}(A) = Pr[A^{Real_\pi} \rightarrow 1] - Pr[A^{Ideal_\pi} \rightarrow 1]$$

Now we show that the difference between the real

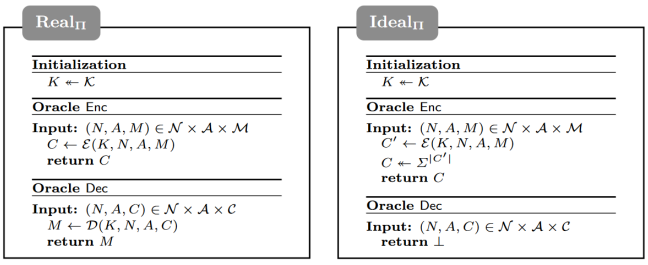

Fig. 12. Real and ideal security game for AEAD-schemes[1, Fig 1].

and ideal resource is bounded by the advantage of an adversary A playing AEAD game. More formally, we prove the following lemma.

Lemma 1. If there is a distinguisher D for the two resources, then there is an adversary A

playing the AEAD-security game such that

$$\triangle^D(enc_\pi^A dec_\pi^B \perp^E [IC, SK_\mathcal{K}], \perp^E ASC) \leq Adv_\pi^{ae}(A) \tag{3}$$

Proof: First note that in this condition, $\perp$ blocks the adversary from attacking. This essentially means IC is now a reliable channel. So for the real resource, any tuple (E, I, M) that is sent to enc_π gets output M to the receiver. For the ideal resource ASC, any tuple (E, I, M) that is sent to interface A gets output M at interface B only if the I being sent to A is equal to the I that is fetched by interface B. This means that only if in ASC the ith input at interface B is (fetch, I_i') for $I_i' \neq I_i$ where I_i is the implicit part of the ith input at interface A, then the behavior of the two systems can differ: while ASC always returns $\perp$ in this case, the real combined resource might return $M \neq \perp$. Since this is the only difference between the two systems, we can upper bound the distinguishing advantage by the probability that the distinguisher D can provoke such an output at interface B when interacting with the real resource. We denote this event by $\mathcal{F}$.

Our next job is to bound the probability of the event $\mathcal{F}$. Observe that $\mathcal{F}$ occurs exactly if the decryption algorithm of the AEAD-scheme returns a message $M \neq \perp$ on input a different additional data than used for encryption. This is because in the cases when the implicit part I that is being sent is different from the implicit part I that is being fetch, the I in enc must be different from the I in dec. And the additional data A in both converters is derived by their own (E, I) pair. Based on this observation, we construct an adversary A that plays AEAD security game using the distinguisher D for the real and ideal resource (Figure 13). The algorithm is shown in Algorithm 1. We conclude the proof by noting that $Adv_\pi^{ae}(A) = Pr[\mathcal{F}\, occurs]$ ■

Condition 2. For the second condition where the adversary is present, we want to construct a converter sim for the ideal resource ASC that simulates the adversary's action in the real resource. The simulator is constructed as shown in Figure 14. Similarly as in condition 1, we want to prove the following lemma.

Lemma 2. If there is a distinguisher D for

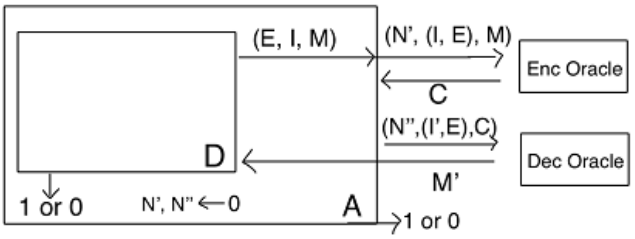

Fig. 13. The adversary A that plays AEAD security game using the distinguisher D that distinguish real and ideal resource

the two resources, then there is an adversary A playing the AEAD-security game such that

$$\triangle^D(enc_\pi^A dec_\pi^B [IC, SK_\mathcal{K}], sim_{ASC}^E ASC) \leq Adv_\pi^{ae}(A) \tag{4}$$

For the proof of that, see Lemma 2 of [1]. The idea is similar with the proof of lemma 1. We construct an adversary using D to play AEAD game. There are four possible inputs D can make, and we need to specify how A should do for each of them.

Lemma 1 and Lemma 2 together imply that the two converters enc, dec allow us to securely construct ASC from insecure channel with shared-key resource. The next step is to securely construct

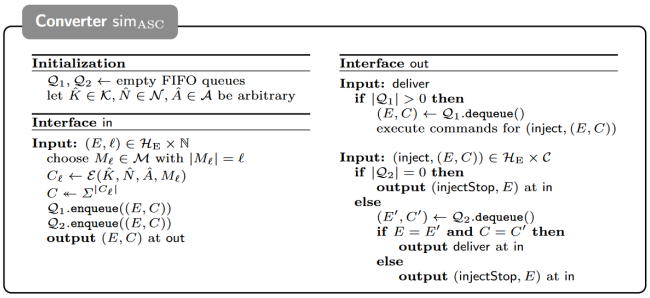

Fig. 14. The simulator for ASC [1, Fig 6].

TLS Record Protocol from ASC. We state that for a secure TLS, there is no need to have an implicit part of the header. The construction explained in the construction section of this paper shows that this is true, since the TLS record contains only fix value and a encrypted ciphertext (See Figure 3).We present the TLS Record, the two converters $tlsSnd$ and $tlsRcv$, and the simulator for tls in Figure 16, Figure 17, and Figure 15. Given these, it is easy to see actually there is "no difference" between the real resource ASC and the ideal resource TLS.

Algorithm 1: Adversary A that plays AEAD game

$N', N'' \leftarrow 0$;
Initialize FIFO empty queue Q;
Run D as follows:
if *D inputs (E, I, M) at interface A* **then**
  $C \leftarrow Enc((N', (I, E), M))$;
  $N' \leftarrow N' + 1$;
  $Q.enqueue((E, I, M, C))$;
  if *Q is empty* **then**
    nothing
  else
    $(E, I, M, C) \leftarrow Q.dequeue$;
    if $I' = I$ **then**
      $M' \leftarrow M$
    end
    if $I' \neq I$ **then**
      $M' \leftarrow Dec(N, (I', E), C)$
    end
  end
  $N'' \leftarrow N'' + 1$;
  if $M' \neq \perp$ *and* $I \neq I$ **then**
    A stops and return 1
  end
  if *D output a bit* **then**
    A return 0
  end
  output M' at interface B of D;
end

Actually, we can easily prove that

$$\triangle^D(tlsSnd^A tlsRcv^B \perp^E ASC, \perp^E SEC_{tls}) = 0$$
$$\triangle^D(tlsSnd^A tlsRcv^B ASC, sim^E_{tls} SEC_{tls}) = 0$$

by first observing that when there is no adversary, for any input on the two systems, they output the same, and second noting that when there is an adversary, the two systems both terminate the session if an empty message is injected into the channel and that all inputs are delivered in order until termination.

Hence, we can conclude that the composition of all systems construct a secure TLS Record Protocol. An illustration is shown in Figure 18

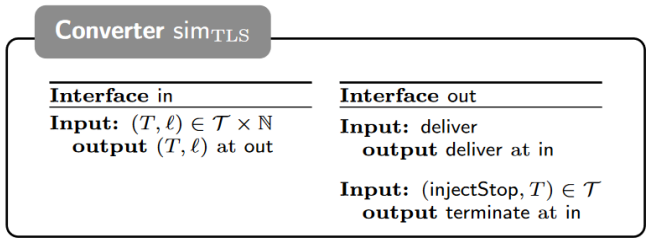

Fig. 15. The simulator for TLS [1, Fig 9].

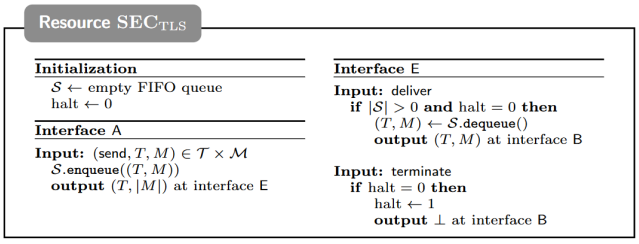

Fig. 16. The ideal TLS [1, Fig 7].

V. Performance Against Common Attacks

A. The length field attack

For network communication protocols, it is common to send the length of the ciphertext in clear. The length of the ciphertext can be exploited to deduce information about the content of the message. For example, if an encrypted message contains one of two images of different size, then

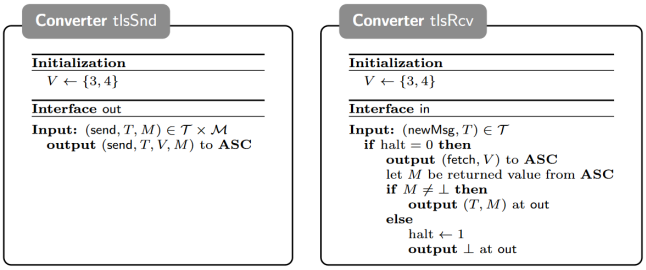

Fig. 17. The converters for ideal TLS[1, Fig 8].

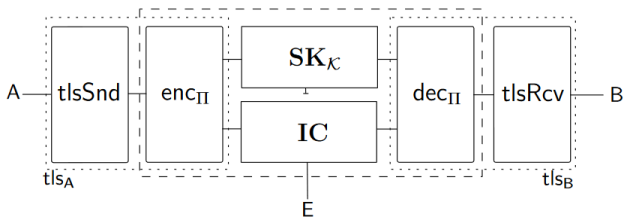

Fig. 18. The final conclusion[1, Fig 10].

the length of the ciphertext might reveal which image was encrypted. This attack is not a concern for the explained TLS 1.3 record protocol because the length of the ciphertext is fixed for any message, which is made possible by adding padding bits during encryption and formatting.

B. The replaying attack

An attacker may want to resend a previous record to cause the wrong action of the receiver. For example, an attacker can replay a purchase order twice to get more money from the buyer. This attack is not a concern for TLS 1.3 record protocol since it uses sequence number which increments by one per record in AEAD.

C. The cookie cutter attack

TLS provides a streaming interface. Records are sent as soon as they are ready. An attacker may want to drop the last record to cause the wrong action of the user. TLS does not provide defense against this. This is because TLS assumes the application layer is responsible for defending this so this is not part of TLS's design goal.

D. Lucky 13

Lucky 13 is a timing attack. An attacker can analyze the traffic response time to guess the padding conditions given a message. However, this attacks works only for a malformed CBC padding. Since TLS 1.3 protocol adopts AEAD instead of CBC, TLS 1.3 protocol is not threated by lucky 13. However, this does not rule out the possibilities of other forms of timing attack. By far, there is not known efficient timing attack.

E. BEAST

This is a man-in-the-middle attack that exploits a vulnerability of the initialization vector in CBC mode. Since TLS 1.3 protocol adopts AEAD instead of CBC, TLS 1.3 protocol is not threated by BEAST.

VI. Conclusion and Discussion

In this paper, I examine TLS 1.3 Record Protocol from its current construction, its security analysis, and its performance against some common attacks. The main part of this paper is spent explaining constructive cryptography and the proof of security by phrasing the logic in my own words and drawing my own illustrative pictures. The two main proofs for the security analysis of TLS 1.3 Record protocol all rely on a reduction from the distinguishing game to AEAD security game where the adversary needs to guess real or ideal world. For more consistency of a "constructive cryptography's mindset", it might be better to frame the AEAD security game to also be a distinguishing game for a real resource and an ideal resource rather than a security game that is defined the traditional game-based cryptography, even though the two games are essentially the same by nature. In addition, in the proof for the first lemma. The author states that "we can upper bound the distinguishing advantage by the probability that the distinguisher D can provoke such an output at interface B when interacting with the real resource." It might be better to first spell out the adversary and then do a full advantage analysis, or just spell out either one, instead of spelling out neither of them which creates confusions to readers.

TLS 1.3 Record Protocol is able to defend almost all of the attacks thanks to the adoption of AEAD algorithm. However, even though it is proven secure, there are still potential system-level vulnerabilities that need to be taken cared of by practitioners.

References

[1] C. Badertscher, C. Matt, U. Maurer, P. Rogaway, and B. Tackmann, "Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer," in Provable Security, M.-H. Au and A. Miyaji, Eds., in Lecture Notes in Computer Science, vol. 9451. Cham: Springer International Publishing, 2015, pp. 85–104. doi: $10.1007/978-3-319-26059-4_5$.

[2] U. Maurer, "Constructive Cryptography – A New Paradigm for Security Definitions and Proofs," in Theory of Security and Applications, S. Mödersheim and C. Palamidessi, Eds., in Lecture Notes in Computer Science, vol. 6993. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 33–56. doi: $10.1007/978-3-642-27375-9_3$.

[3] A. Delignat-Lavaud et al., "Implementing and Proving the TLS 1.3 Record Layer," in 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA: IEEE, May 2017, pp. 463–482. doi: 10.1109/SP.2017.58.

[4] E. Rescorla, "RFC ft-ietf-tls-tls13: The Transport Layer Security (TLS) Protocol Version 1.3," IETF Datatracker, Aug. 10, 2018. https://datatracker.ietf.org/doc/html/rfc8446 (accessed Feb. 05, 2023).

[5] N. J. Al Fardan and K. G. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols," in 2013 IEEE

Symposium on Security and Privacy, Berkeley, CA: IEEE, May 2013, pp. 526–540. doi: 10.1109/SP.2013.42.