TLS 1.3 Record **Protocol**

Elle Wen

Approach taken in this course

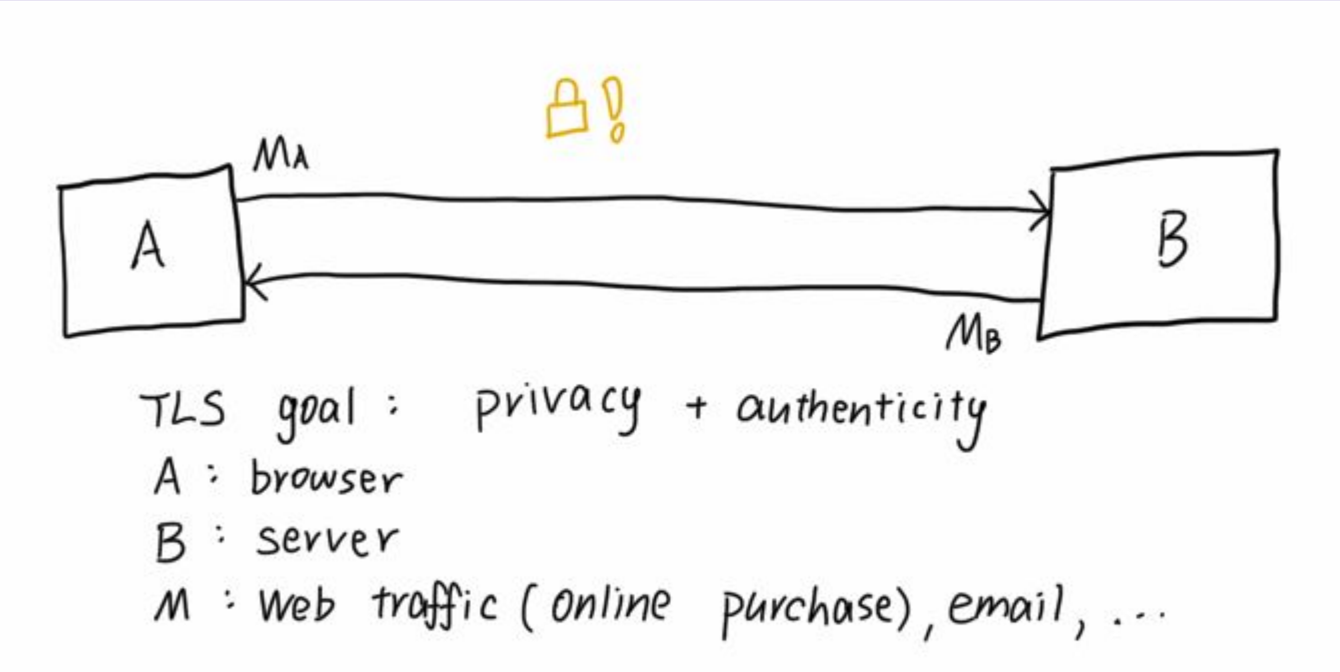
abstraction, abstraction, abstraction, ...

We separate things into layers.

- low-level building blocks
- primitives
- high-level protocols

	Symmetric setting	Asymmetric setting
Low-level tools	block ciphers pseudorandom function families	one-way functions (OWF) trapdoor OWF
Primitives	symmetric encryption AEAD message authentication codes	asymmetric encryption digital signatures
High-level protocols	key exchange TLS1.3	key exchange electronic voting payment

Big Picture



TLS 1.3 = Handshake Protocol -> set up shared parameters, key exchange, authenticate parties
+ Record Protocol -> protect traffic

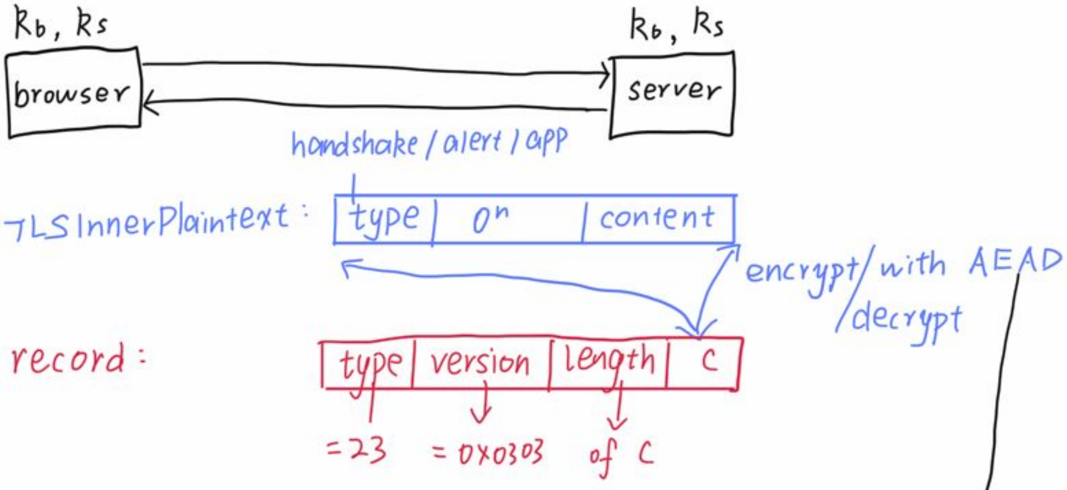
NOT SEQUENTIAL!

TLS 1.3 Record Protocol

1. Construction
2. Security

1. Construction

Let’s look at the most updated specification document for TLS 1.3: RFC 8446



AEAD (key, nonce, plaintext, additional data) ⟶ c (could pad)

key → k_b / k_s

nonce → 0^n || sequence number ⊕ IV

plaintext → TLSInnerPlaintext

additional data → type || version || length

$AEAD^{-1}$ (same key, ...) ⟶ plaintext

nonce: every time we use a new key, sequence number == 0, then ++ by record

2. Security

Wait! Do we know how to prove a PROTOCOL is secure?!

Is it possible to decide whether a cryptographic protocol is secure or not ?

Hubert Comon and Vitaly Shmatikov

Abstract —

We consider the so called "cryptographic protocols" whose aim is to ensure some security properties when communication channels are not reliable. Such protocols usually rely on cryptographic primitives. Even if it is assumed that the cryptographic primitives are perfect, the security goals may not be achieved: the protocol itself may have weaknesses which can be exploited by an attacker. We survey recent work on decision techniques for the cryptographic protocol analysis.

1. Introduction

Security questions are not new. They become increasingly important, however, with the development of the In-

2. Abstract protocol modeling

In the presence of insecure communication channels, an attacker may be able to observe network traffic and/or intercept messages, modify them in transit, and construct fake messages. In this context, securing communication relies on a set of basic functions that we will refer to as *cryptographic primitives*. For example, an encryption primitive can be used to encode messages prior to tranmission on an insecure channel in such a way that the original message content (*cleartext*) can only be retrieved by recipients who possess the "right" decryption key. A number of cryptographic primitives have been designed to achieve information security goals such as secrecy, integrity, authentication, etc.

The analysis techniques discussed in this survey assume *per-*

Traditional Way - game based reduction

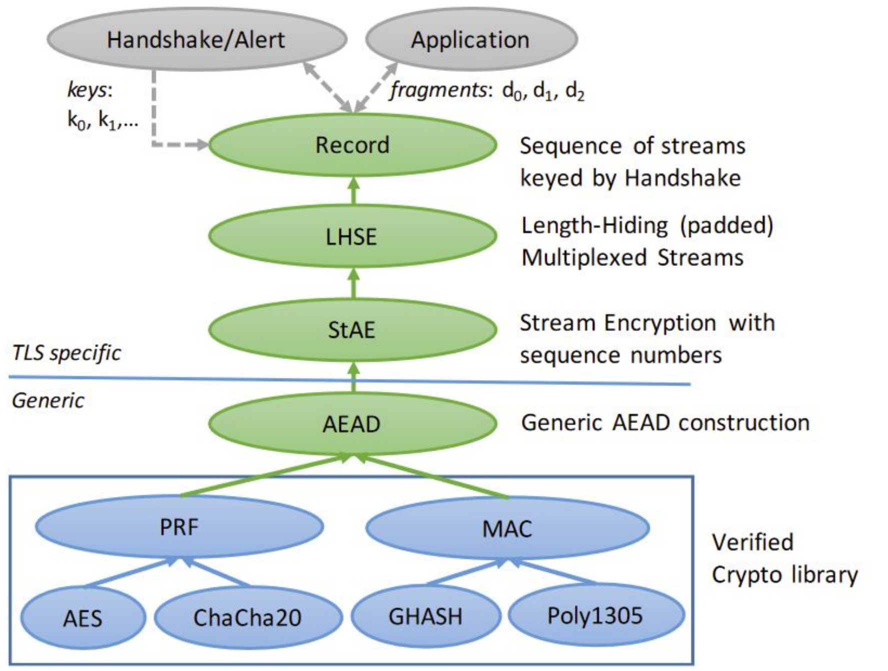


Figure 2. Modular structure of our proof. Green arrows denote security reductions proved by typing.

Proved by typing
Verification tool F*

Constructive Cryptography

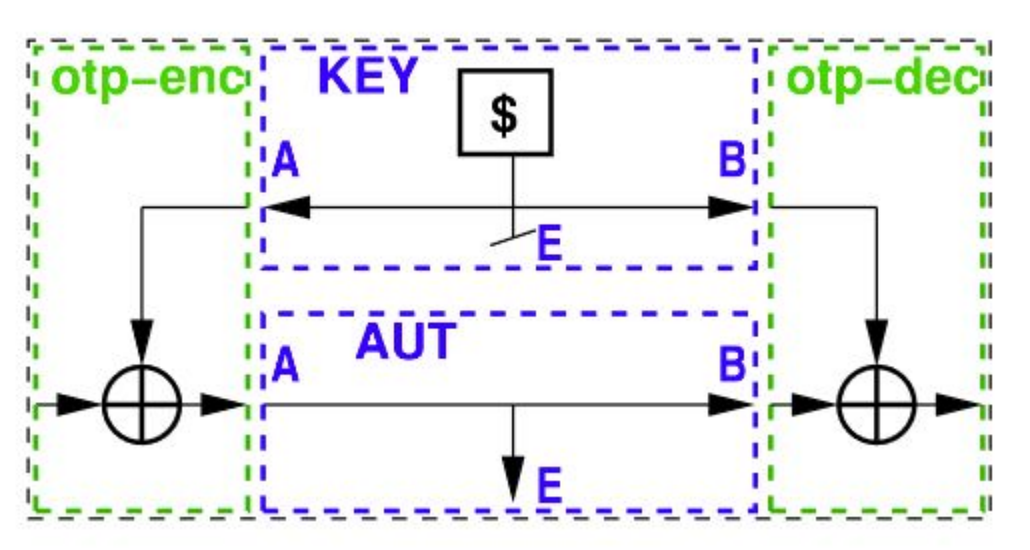
Basic concepts:

system: an abstract object with **interface**

interface: interact with environment/other system(input/output)

- Two system can be composed into a single system by connecting their interfaces
- There are many types of system defined in lower abstraction level, in this context, we only consider 3 types of system:

1. I-resource system: guarantees and expectation that provide a specified service to parties, R, S, interface label set I = {A, B, E}
2. Converter system: α, π, otp-enc(a system with two interfaces: in-out)
3. Distinguisher system



$\textsf{otp-dec}^{\mathrm{B}}\ \textsf{otp-enc}^{\mathrm{A}}\ (\textsf{KEY}||\textsf{AUT}).$

Def 1.

A cryptographic algebra (Φ, Σ) for an interface set I consists of

Φ - a set of resource, $\|$ - parallel composition operation,

Σ - a set of converter. a mapping $\Sigma \times \Phi \times I \longmapsto \Phi$

that defines the resource obtained when converter α

is attached to interface i of resource R: $\alpha^i R$

s.t.

i) $\alpha^i \beta^j R = \beta^j \alpha^i R \quad \forall\, i \neq j,\ R \in \Phi,\ \alpha, \beta \in \Sigma$

ii) neutral converter $1 \in \Sigma$ (attaching no converter) s.t.

$1^i R = R \quad \forall\, i \in I,\ R \in \Phi$

can naturally define : $(\alpha\beta)^i R = \alpha^i \beta^i R$

$(\alpha \| \beta)^i (R \| S) = \alpha^i R \,\|\, \beta^i S$

We use pseudo-metric to measure the similarity or dissimilarity of resources -> measure security

A pseudometric space (X, d) is a set X together with a non-negative real-valued function $d : X \times X \longrightarrow \mathbb{R}_{\geq 0}$, called a **pseudometric**, such that for every $x, y, z \in X$,

1. $d(x, x) = 0$.
2. *Symmetry*: $d(x, y) = d(y, x)$
3. *Subadditivity*/*Triangle inequality*: $d(x, z) \leq d(x, y) + d(y, z)$

Unlike a metric space, points in a pseudometric space need not be distinguishable; that is, one may have $d(x, y) = 0$ for distinct values $x \neq y$.

To make sure "non-expanding"

Definition 2. A pseudo-metric d on Φ is *compatible* with the cryptographic algebra $\langle \Phi, \Sigma \rangle$ if

$$d(R \| R', S \| S') \leq d(R, S) + d(R', S') \tag{3}$$

for all $R, R', S, S' \in \Phi$, and

$$d(\alpha^i R, \alpha^i S) \ \leq \ d(R, S) \tag{4}$$

for all $i \in \mathcal{I}$, $R, S \in \Phi$ and $\alpha \in \Sigma$.

Define the distance function d for a class of distinguishers D to distinguish R from S:

$$d(R,S) = \Delta^{\mathcal{D}}(R,S) := \sup_{D\in\mathcal{D}} \Delta^{D}(R,S),$$

ΔD(R, S) is the advantage of D in distinguishing R and S

Distinguisher system: a system with n+1 interfaces

A distinguisher D emulating (internally) a converter $\alpha \in \Sigma$ at interface i induces a new distinguisher, denoted $D\alpha^i$, defined by

$$\Delta^{D\alpha^i}(R,S) = \Delta^{D}(\alpha^i R, \alpha^i S).$$

Similarly, a distinguisher D emulating a resource $T \in \Phi$ in parallel induces a new distinguisher, denoted $D[\cdot\|T]$, defined by

$$\Delta^{D[\cdot\|T]}(R,S) = \Delta^{D}(R\|T, S\|T).$$

Ppty: Class D is closed under emulation of resource and interface

Lemma 1. *For a distinguisher class $\mathcal{D}$ for resources in Φ, the pseudo-metric $\Delta^{\mathcal{D}}$ is compatible with the cryptographic algebra $\langle \Phi, \Sigma \rangle$ if*

$$\mathcal{D}\Sigma^i \subseteq \mathcal{D}, \qquad \mathcal{D}[\cdot \| \Phi] \subseteq \mathcal{D}, \qquad \textit{and} \qquad \mathcal{D}[\Phi \| \cdot] \subseteq \mathcal{D}.$$

Proof. Since $\mathcal{D}\alpha^i \subseteq \mathcal{D}\Sigma^i \subseteq \mathcal{D}$ we have

$$\Delta^{\mathcal{D}}(\alpha^i R, \alpha^i S) = \Delta^{\mathcal{D}\alpha^i}(R, S) \leq \Delta^{\mathcal{D}}(R, S),$$

which is (4). Similarly, since $\mathcal{D}[\cdot \| T] \subseteq \mathcal{D}[\cdot \| \Phi] \subseteq \mathcal{D}$ we have

$$\Delta^{\mathcal{D}}(R \| T, S \| T) = \Delta^{\mathcal{D}[\cdot \| T]}(R, S) \leq \Delta^{\mathcal{D}}(R, S).$$

As mentioned, this inequality together with the dual inequality $\Delta^{\mathcal{D}}(T \| R, T \| S) \leq \Delta^{\mathcal{D}}(R, S)$ implies (3). $\square$

Main Definition - secure construction

Definition 3. Consider a cryptographic algebra $\langle \Phi, \Sigma \rangle$ for interface set $\mathcal{I} = \{A, B, E\}$ and a pseudo-metric d on Φ. For resources R and S we say that protocol (π_1, π_2) for $\pi_1, \pi_2 \in \Sigma$ *(securely) constructs* S from R, within ε, denoted

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

if the following two conditions (availability and security) are satisfied:

1. $d(\pi_1^A \pi_2^B \bot^E R, \bot^E S) \leq \varepsilon$
2. $\exists \sigma \in \Sigma : \quad d(\pi_1^A \pi_2^B R,\ \sigma^E S) \leq \varepsilon.$

OK. Back to proving the security of TLS record protocol…

Proof idea:
[SK, IC] -> ASC ->TLS 1.3 record protocol

Proof Part 1: [SK, IC] securely constructs ASC

Resource IC

Initialization

$\mathcal{Q} \leftarrow$ empty FIFO queue

Interface A

Input: $(\mathsf{send}, M) \in \Sigma^*$
  $\mathcal{Q}.\mathsf{enqueue}(M)$
  output M at interface E

Interface E

Input: deliver
  if $|\mathcal{Q}| > 0$ **then**
    $M \leftarrow \mathcal{Q}.\mathsf{dequeue}()$
    output M at interface B

Input: $(\mathsf{inject}, M) \in \Sigma^*$
  output M at interface B

Resource $\mathrm{SK}_{\mathcal{K}}$

Initialization

$k \twoheadleftarrow \mathcal{K}$

Interface A

Input: getKey
  output k at A.

Interface B

Input: getKey
  output k at B.

Augmented Secure Channel(ASC)

Secure Chanel(SC) ->

Resource ASC

Initialization

$\mathcal{S} \leftarrow$ empty FIFO queue
$\mathcal{R} \leftarrow$ empty FIFO queue
halt $\leftarrow 0$

Interface A

Input: $(\mathsf{send}, E, I, M) \in \mathcal{H}_{\mathrm{E}} \times \mathcal{H}_{\mathrm{I}} \times \mathcal{M}$
$\mathcal{S}.\mathsf{enqueue}((E, I, M))$
output $(E, |M|)$ at interface E

Interface B

Input: $(\mathsf{fetch}, I) \in \mathcal{H}_{\mathrm{I}}$
if $|\mathcal{R}| > 0$ **and** halt $= 0$ **then**
$(E', I', M') \leftarrow \mathcal{R}.\mathsf{dequeue}()$
if $I' = I \neq \perp$ **then**
output M' at interface B
else
halt $\leftarrow 1$
output $\perp$ at interface B

Interface E

Input: deliver
if $|\mathcal{S}| > 0$ **and** halt $= 0$ **then**
$(E, I, M) \leftarrow \mathcal{S}.\mathsf{dequeue}()$
$\mathcal{R}.\mathsf{enqueue}((E, I, M))$
output (newMsg, E) at interface B

Input: $(\mathsf{injectStop}, E) \in \mathcal{H}_{\mathrm{E}}$
if halt $= 0$ **then**
$\mathcal{R}.\mathsf{enqueue}((\perp, \perp, \perp))$
output (newMsg, E) at interface B

Capital Epsilon means AEAD

Converter enc_{Π}

Initialization

$N \leftarrow 0$
output getKey to $\mathbf{SK}_{\mathcal{K}}$
let K be returned value from $\mathbf{SK}_{\mathcal{K}}$

Interface out

Input: $(\mathsf{send}, E, I, M) \in \mathcal{H}_{\mathrm{E}} \times \mathcal{H}_{\mathrm{I}} \times \mathcal{M}$
$A \leftarrow (E, I)$
$C \leftarrow \mathcal{E}(K, N, A, M)$
$N \leftarrow N + 1$
output $(\mathsf{send}, (E, C))$ to **IC**

Converter dec_{Π}

Initialization

$\mathcal{Q} \leftarrow$ empty FIFO queue
$N \leftarrow 0$
halt $\leftarrow 0$
output getKey to $\mathbf{SK}_{\mathcal{K}}$
let K be returned value from $\mathbf{SK}_{\mathcal{K}}$

Interface in

Input: $(E, C) \in \mathcal{H}_{\mathrm{E}} \times \mathcal{C}$ from **IC**
if halt $= 0$ **then**
$\mathcal{Q}.\mathsf{enqueue}((E, C))$
output (newMsg, E) at out

Interface out

Input: $(\mathsf{fetch}, I) \in \mathcal{H}_{\mathrm{I}}$
if $|\mathcal{Q}| > 0$ **and** halt $= 0$ **then**
$(E, C) \leftarrow \mathcal{Q}.\mathsf{dequeue}()$
$A \leftarrow (E, I)$
$M \leftarrow \mathcal{D}(K, N, A, C)$
$N \leftarrow N + 1$
if $M = \perp$ **then** halt $\leftarrow 1$
else output M at out

Define $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \;=\; \Pr[\mathbf{DR} = 1] - \Pr[\mathbf{DS} = 1].$

Lemma 1. *There is an (efficient) transformation ρ described in the proof that maps distinguishers $\mathbf{D}$ for two resources to valid adversaries $\mathcal{A} = \rho(\mathbf{D})$ for the AEAD-security game such that*

$$\Delta^{\mathbf{D}}\left(\mathsf{enc}_{\Pi}{}^{\mathsf{A}}\mathsf{dec}_{\Pi}{}^{\mathsf{B}}\mathsf{dlv}^{\mathsf{E}}\,[\mathbf{SK}_{\mathcal{K}}, \mathbf{IC}], \mathsf{dlv}^{\mathsf{E}}\,\mathbf{ASC}\right) \leq \mathbf{Adv}^{\mathrm{ae}}_{\Pi}(\rho(\mathbf{D})).$$

Lemma 2. *For the simulator $\mathsf{sim}_{\mathrm{ASC}}$ defined in Fig. 6, there is an (efficient) transformation ρ' described in the proof that maps distinguishers $\mathbf{D}$ for two resources to valid adversaries $\mathcal{A} = \rho'(\mathbf{D})$ for the AEAD-security game such that*

$$\Delta^{\mathbf{D}}\left(\mathsf{enc}_{\Pi}{}^{\mathsf{A}}\mathsf{dec}_{\Pi}{}^{\mathsf{B}}\,[\mathbf{SK}_{\mathcal{K}}, \mathbf{IC}], \mathsf{sim}^{\mathsf{E}}_{\mathrm{ASC}}\,\mathbf{ASC}\right) \leq \mathbf{Adv}^{\mathrm{ae}}_{\Pi}(\rho'(\mathbf{D})).$$

reduction is needed

Proof part 2: ASC -> secure TLS 1.3 protocol

Resource $\mathrm{SEC}_{\mathrm{TLS}}$

Initialization

$\mathcal{S} \leftarrow$ empty FIFO queue
halt $\leftarrow 0$

Interface A

Input: $(\mathsf{send}, T, M) \in \mathcal{T} \times \mathcal{M}$
$\mathcal{S}.\mathsf{enqueue}((T, M))$
output $(T, |M|)$ at interface E

Interface E

Input: deliver
if $|\mathcal{S}| > 0$ **and** halt $= 0$ **then**
$(T, M) \leftarrow \mathcal{S}.\mathsf{dequeue}()$
output (T, M) at interface B

Input: terminate
if halt $= 0$ **then**
halt $\leftarrow 1$
output $\perp$ at interface B

Converter tlsSnd

Initialization

$V \leftarrow \{3, 4\}$

Interface out

Input: $(\mathsf{send}, T, M) \in \mathcal{T} \times \mathcal{M}$
  output (send, T, V, M) to **ASC**

Converter tlsRcv

Initialization

$V \leftarrow \{3, 4\}$

Interface in

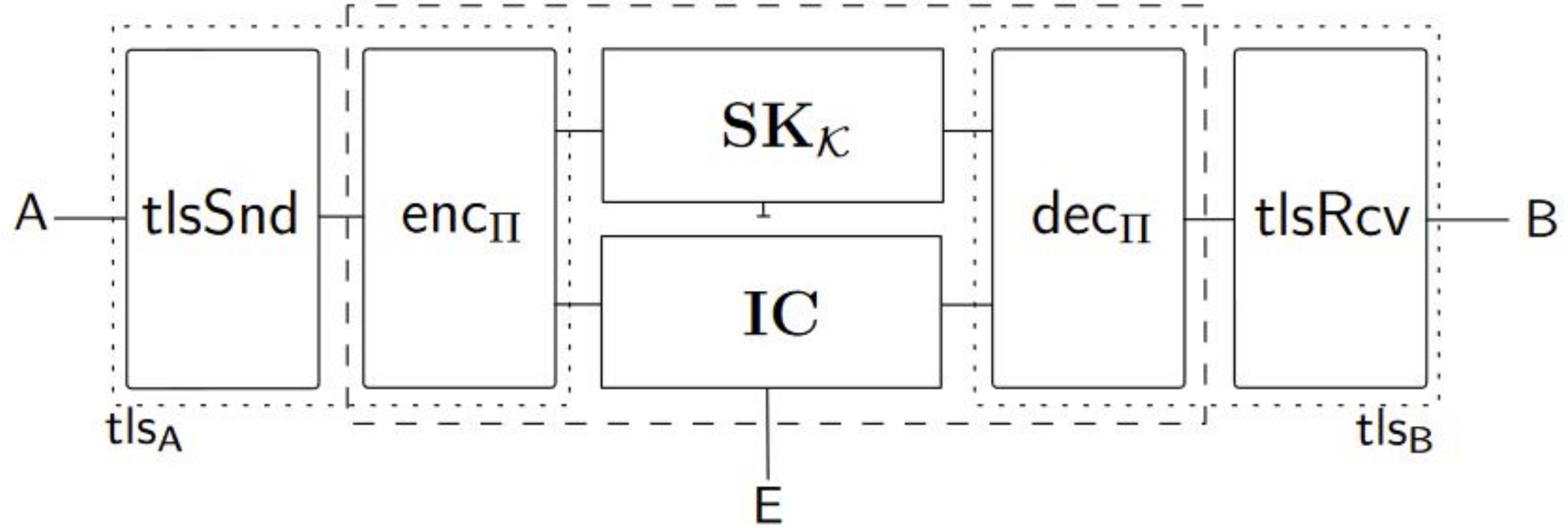
Input: $(\mathsf{newMsg}, T) \in \mathcal{T}$
  if halt $= 0$ **then**
    output (fetch, V) to **ASC**
    let M be returned value from **ASC**
    if $M \neq \perp$ **then**
      output (T, M) at out
    else
      halt $\leftarrow 1$
      output $\perp$ at out

Theorem 2. *The protocol* $(\mathsf{tlsSnd}, \mathsf{tlsRcv})$ *constructs* $\mathbf{SEC}_{\mathrm{TLS}}$ *from* $\mathbf{ASC}$. *More specifically, we have for the simulator* $\mathsf{sim}_{\mathrm{TLS}}$ *defined in Fig. 9 and for all distinguishers* $\mathbf{D}$

$$\Delta^{\mathbf{D}}\left(\mathsf{tlsSnd}^{\mathsf{A}}\mathsf{tlsRcv}^{\mathsf{B}}\mathsf{dlv}^{\mathsf{E}}\mathbf{ASC}, \mathsf{dlv}^{\mathsf{E}}\mathbf{SEC}_{\mathrm{TLS}}\right) = 0 \tag{1}$$

$$\textit{and} \quad \Delta^{\mathbf{D}}\left(\mathsf{tlsSnd}^{\mathsf{A}}\mathsf{tlsRcv}^{\mathsf{B}}\mathbf{ASC}, \mathsf{sim}^{\mathsf{E}}_{\mathrm{TLS}}\mathbf{SEC}_{\mathrm{TLS}}\right) = 0. \tag{2}$$

Proof. The availability condition (1) is easy to verify: On input (send, T, M) at interface A, the system $\mathsf{dlv}^{\mathsf{E}}\mathbf{SEC}_{\mathrm{TLS}}$ directly outputs (T, M) at interface B. The same holds for system $\mathsf{tlsSnd}^{\mathsf{A}}\mathsf{tlsRcv}^{\mathsf{B}}\mathsf{dlv}^{\mathsf{E}}\mathbf{ASC}$: On input (send, T, M), the converter tlsSnd inputs (send, T, V, M) to $\mathbf{ASC}$. The converter tlsRcv then obtains the notification (newMsg, T) and queries (fetch, V) to $\mathbf{ASC}$, which results in the output M from $\mathbf{ASC}$, which in turn triggers tlsRcv to output (T, M). Since the two systems behave identically, every distinguisher has advantage 0 in distinguishing them, i.e., (1) follows.


A
tlsSnd
enc$_\Pi$
$\mathbf{SK}_{\mathcal{K}}$
$\mathbf{IC}$
dec$_\Pi$
tlsRcv
B
tls$_A$
tls$_B$
E

Does it change anything?

1. The nonce of the AEAD scheme can be set to the counter value left-padded with zeros to be of the appropriate length.
2. The sequence number can be removed from the additional data part.
3. After the handshake, the version number does not need to be transmitted explicitly as part of the TLS record. However, it should still be part of the additional data.

Reference:

1.Rescorla, Eric. “RFC Ft-Ietf-Tls-Tls13: The Transport Layer Security (TLS) Protocol Version 1.3.” IETF Datatracker, August 10, 2018. https://datatracker.ietf.org/doc/html/rfc8446.

2.Delignat-Lavaud, Antoine, Cedric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella-Beguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. “Implementing and Proving the TLS 1.3 Record Layer.” In *2017 IEEE Symposium on Security and Privacy (SP)*, 463–82. San Jose, CA, USA: IEEE, 2017. https://doi.org/10.1109/SP.2017.58.

3.Maurer, Ueli. “Constructive Cryptography – A New Paradigm for Security Definitions and Proofs.” In *Theory of Security and Applications*, edited by Sebastian Mödersheim and Catuscia Palamidessi, 6993:33–56. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-27375-9_3.

4.adertscher, Christian, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. “Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer.” In *Provable Security*, edited by Man-Ho Au and Atsuko Miyaji, 9451:85–104. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-26059-4_5.