TLS Crypto Seminar : *28 February 2019*

Partially Specified Channels

The TLS 1.3 Record Layer without elision

Christopher Patton,
Tom Shrimpton
CCS 2018

Presented by:
Vivek Arte

UC San Diego

Overview

- Motivation
- Partially specified channels
- The TLS 1.3 Record Layer

Motivation

- Protocols are often only partially specified.
- Standard:
  - collection of implementations with a shared core set of behaviors
- Challenge for provable security
  - What is relevant to security?!

Multiplexing in TLS 1.3

- The TLS 1.3 Record Layer handles streams for three distinct sub-protocols
  - handshake
  - alert
  - application-data

- Each sub-protocol has side-effects on the sender and receiver state, and thus could affect security

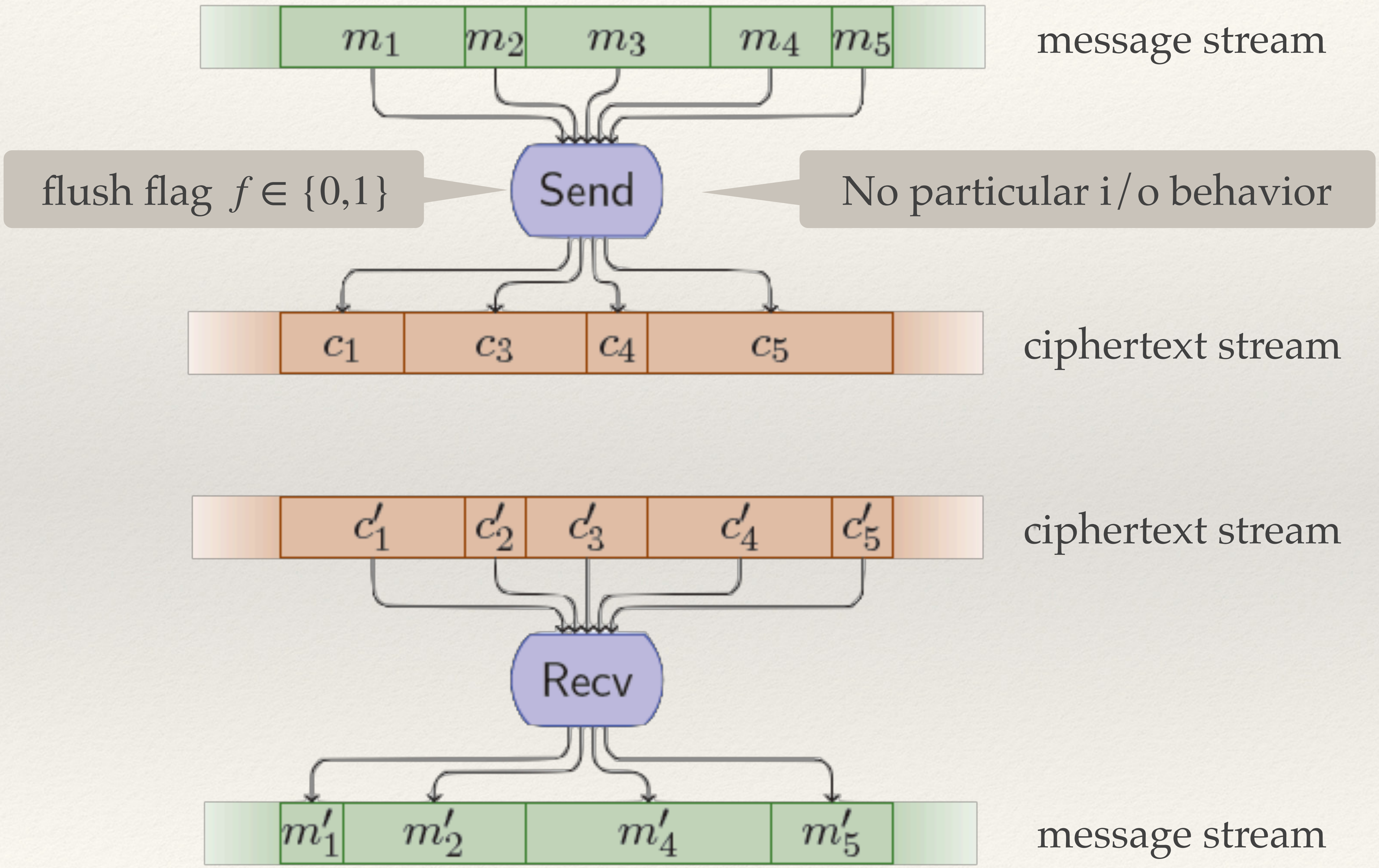
Stateful Authenticated Encryption [BKN02]

- Defined security notions of confidentiality and integrity for stateful symmetric encryption
- Accounts for replay and out-of-order delivery attacks
- Ciphertexts are atomic!

Stream-based Channels [FGMP15]

- TLS - provides a streaming interface for applications
  - Never promised to treat messages atomically!
- Fragmentation at the sender and receiver ends could differ
- [FGMP15] gives specifications and security notions for stream-based channels

Stream-based Channels [FGMP15]



Correctness [FGMP15]

$$(\mathsf{st}_{S,0}, \mathsf{st}_{R,0}) \leftarrow_\$ \mathsf{Init}(1^\lambda)$$

$$(\mathsf{st}'_S, c) \leftarrow_\$ \mathsf{Send}(\mathsf{st}_S, m, f)$$

$$(\mathsf{st}'_R, m) \leftarrow_\$ \mathsf{Recv}(\mathsf{st}_R, c)$$

- ## Correctness
  - No matter how ciphertexts are fragmented at the sender side, and re-fragmented at the receiver side, the returned message stream is a prefix of the initial message stream

$$\|\mathbf{c}[1,\ldots,i] \preccurlyeq \|\mathbf{c}' \preccurlyeq \|\mathbf{c} \implies \|\mathbf{m}[1,\ldots,i] \preccurlyeq \|\mathbf{m}' \preccurlyeq \|\mathbf{m}.$$

$$i \in \{0\} \cup \{j : f_j = 1\}$$

everything up to last flush

Multiplexing [PS18]

- Streams are of the form $(M_1, sc_1), (M_2, sc_2), \ldots$

Partially Specified Channels

things that are mandated and explicitly described

[RS09]

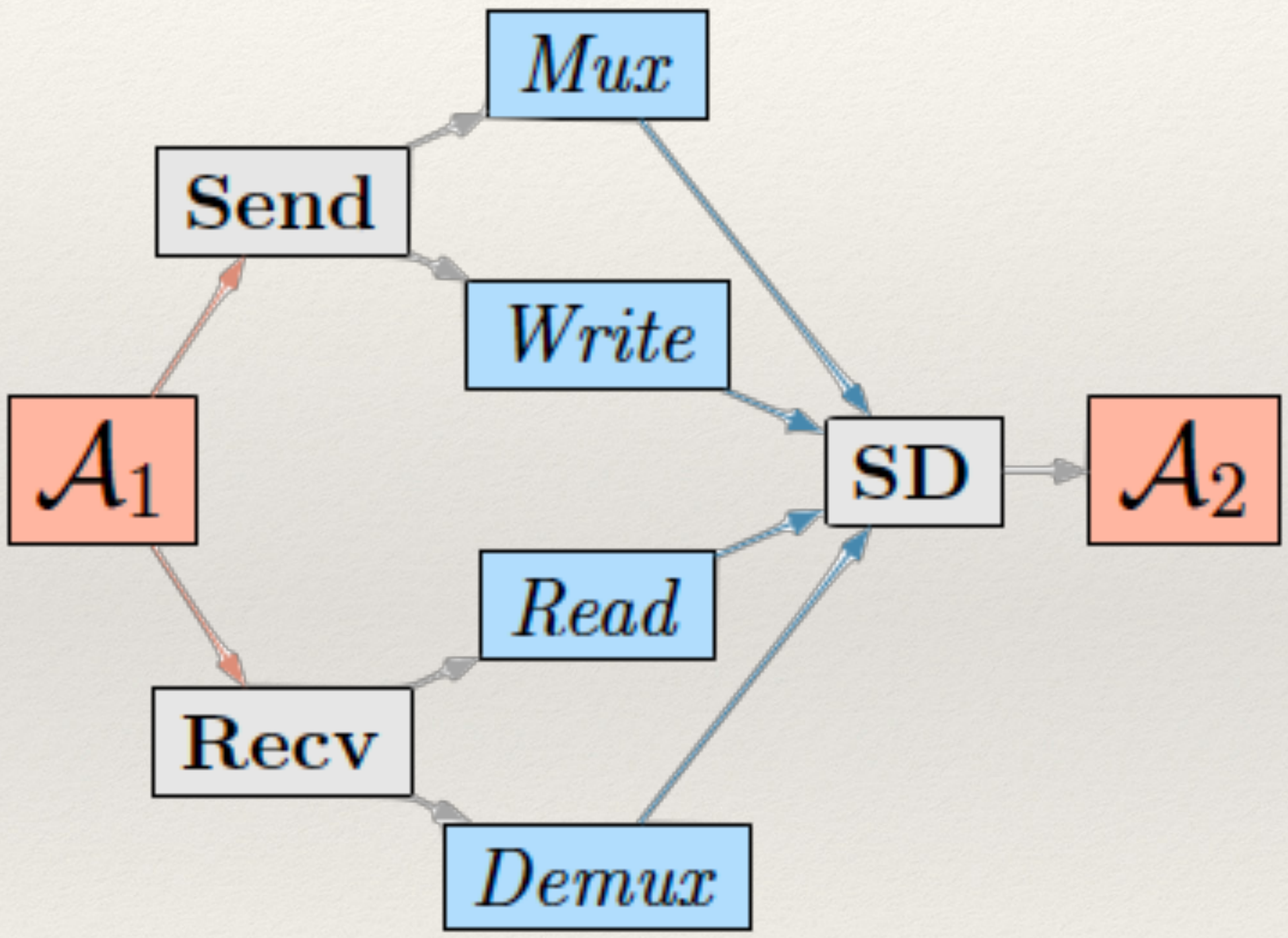
- Standard = partial specification + additional details

everything else

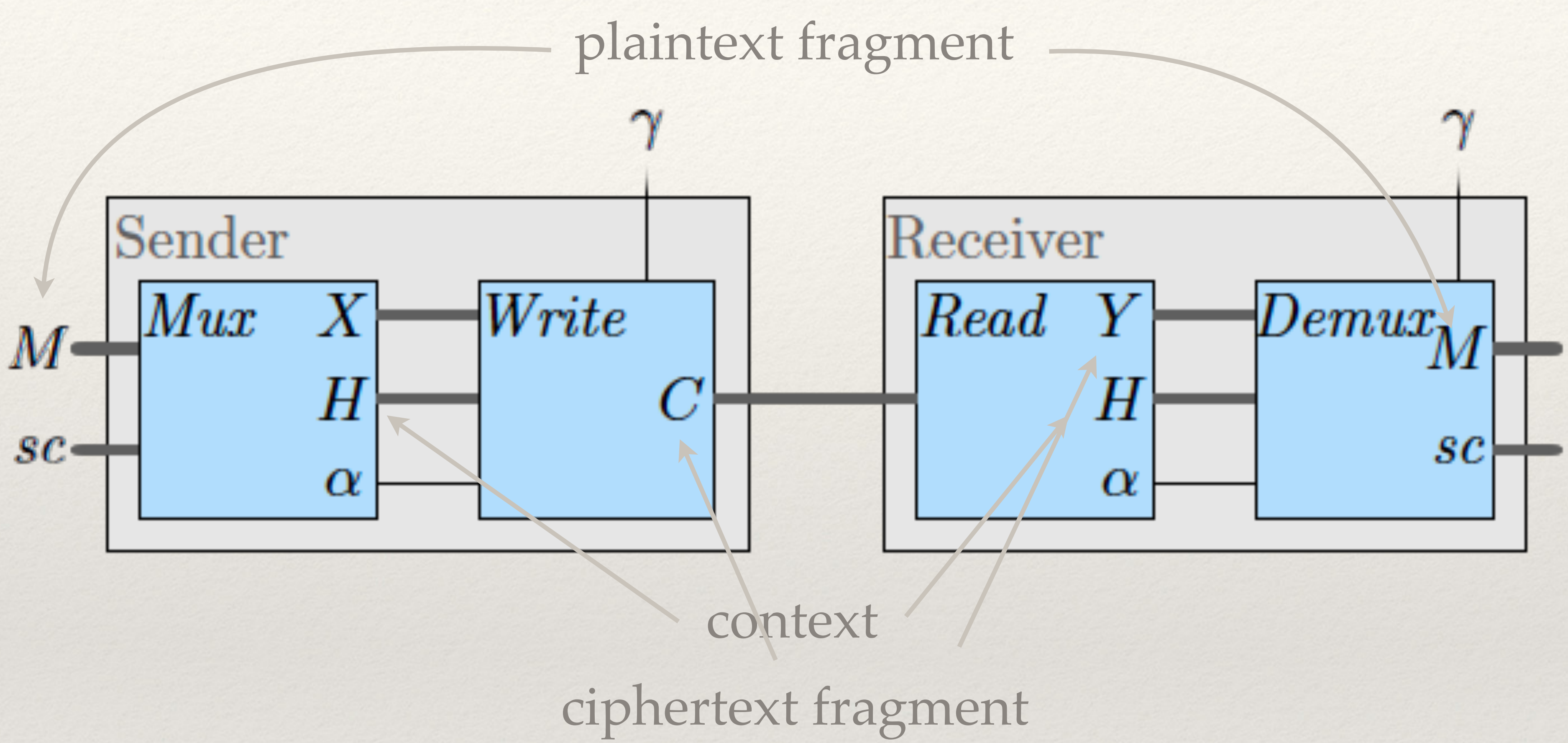
- *Mux, Write, Read, Demux* : fully specified
- The rest of the details are formalized as an oracle given to each algorithm

In security games, queries made to the oracle are serviced by the adversary

Execution Model



Syntax



$$Init() \to (Mu, Wr, Re, De)$$

$$Mux^{\mathcal{O}}(M, sc, Mu) \to (X, H, \alpha)$$

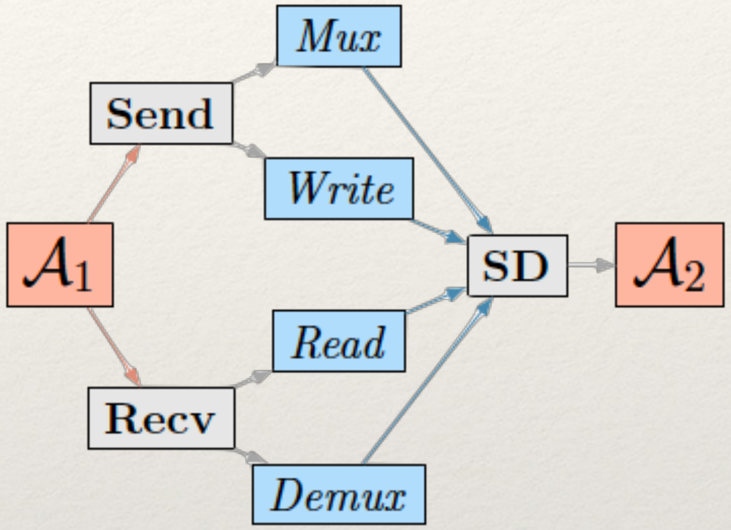
$$Write^{\mathcal{O}}(X, H, \alpha, Wr) \to (C, \gamma)$$

$$Read^{\mathcal{O}}(C, Re) \to (Y, H, \alpha)$$

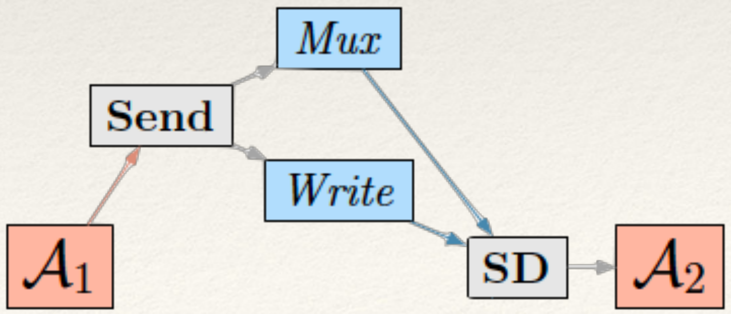
$$Demux^{\mathcal{O}}(Y, H, \alpha, De) \to (M, sc, \gamma)$$

Privacy Notions

- PRIV-SR
  - **Send** : allows adversary to provide the sender with arbitrary message fragments and stream contexts
  - **Recv** : allows adversary to deliver arbitrary ciphertext fragments to the receiver



- PRIV-S
  - PRIV-SR without access to the Recv oracle

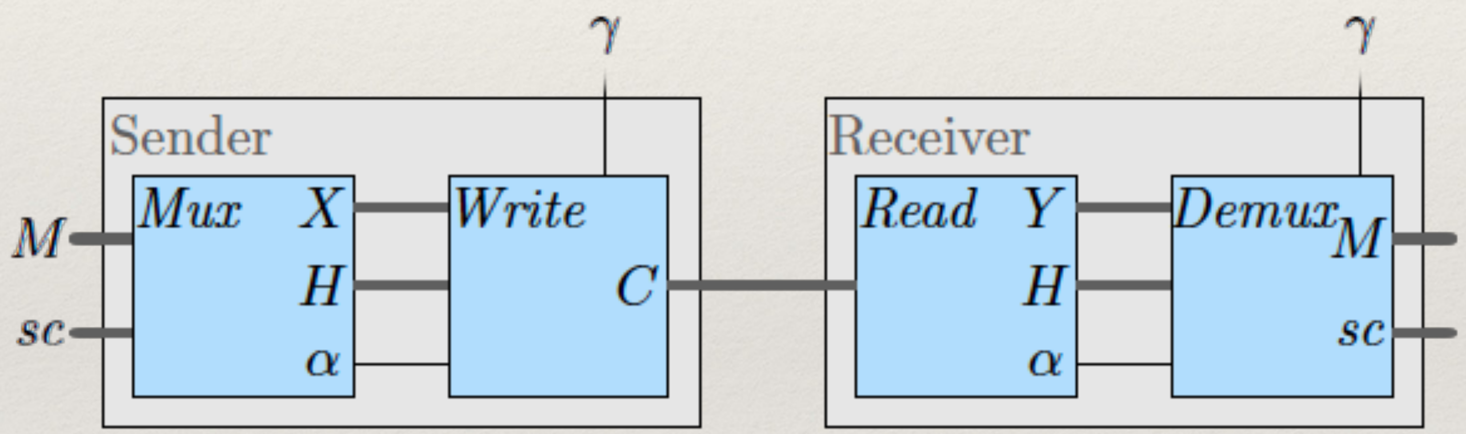


Privacy Notions

- Privacy is in terms of left-or-right indistinguishability of ciphertexts
  - PRIV-SR : must suppress the output of **Recv** in situations that will give trivial distinguishing attacks
  - These "situations" are when the channel is in-sync

Channel Synchronization

- *Read* : models the receiver side buffering and defragmentation



- Channel is in-sync as long as the ciphertext fragments Y output by Read remain a prefix of the ciphertext stream C transmitted by the sender

PRIV-SR Security Notion

$\mathbf{Exp}^{\text{priv-sr}}_{\mathcal{CH},\ell,b}(\mathcal{A})$

$\textbf{declare str } S, \mathit{Env}, \textbf{ bool } \mathit{sync}$
$(\mathit{Mu}, \mathit{Wr}, \mathit{Re}, \mathit{De}) \twoheadleftarrow \mathit{Init}(\,)$
$\mathit{sync} \leftarrow 1$
$b' \twoheadleftarrow \mathcal{A}_1{}^{\textbf{Send},\textbf{Recv}}(\textbf{var } \mathit{Env})$
$\textbf{return } b'$

$\textbf{Send}(M_0, sc_0, M_1, sc_1)$

$L_0 \leftarrow \mathit{leak}(\ell, M_0, sc_0)$
$L_1 \leftarrow \mathit{leak}(\ell, M_1, sc_1)$
$\textbf{if } L_0 \neq L_1 \textbf{ then return } (\bot, \bot)$
$(X, H, \alpha) \twoheadleftarrow \mathit{Mux}^{\textbf{SD}}(M_b, sc_b, \textbf{var } \mathit{Mu})$
$(C, \gamma) \twoheadleftarrow \mathit{Write}^{\textbf{SD}}(X, H, \alpha, \textbf{var } \mathit{Wr})$
$S \leftarrow S \,\|\, C$
$\textbf{return } (C, \gamma)$

$\textbf{Recv}(C)$

$(Y, H, \alpha) \twoheadleftarrow \mathit{Read}^{\textbf{SD}}(C, \textbf{var } \mathit{Re})$
$(M, sc, \gamma) \twoheadleftarrow \mathit{Demux}^{\textbf{SD}}(Y, H, \alpha, \textbf{var } \mathit{De})$
$\textbf{if } \mathit{sync} \textbf{ and } Y \preceq S \textbf{ then}$
$\quad S \leftarrow S \,\%\, Y;\ M, sc \leftarrow \bot$
$\textbf{else } \mathit{sync} \leftarrow 0$
$\textbf{return } (M, sc, \gamma)$

$\textbf{SD}(I)$

$O \twoheadleftarrow \mathcal{A}_2(I, \textbf{var } \mathit{Env});\ \textbf{return } O$

$\boxed{\begin{array}{l} \mathit{leak}(\ell, M, sc) \\ \textbf{switch } (\ell) \\ \quad \textbf{case } \mathsf{lensc}\textbf{: return } \langle |M|, sc \rangle \\ \quad \textbf{case } \mathsf{len}\textbf{: return } \langle |M|, |sc| \rangle \\ \quad \textbf{case } \mathsf{none}\textbf{: return } \varepsilon \end{array}}$

$$\mathbf{Adv}^{\mathrm{priv-sr}}_{\mathcal{CH},l}(\mathcal{A}) = 2\Pr_b[\mathbf{Exp}^{\mathrm{priv-sr}}_{\mathcal{CH},l,b}(\mathcal{A}) = b] - 1$$

Integrity Notions

- INT-CS
  - Requires that the channel (i.e. the ciphertext stream) should remain in-sync
  - The adversary wins if it can make the out-of-sync **Recv** oracle output a valid message fragment and context
- INT-PS
  - Requires that the plaintext streams carried by the channel should remain in-sync
  - The adversary wins if at any point in the game, the output plaintext stream is not a prefix of the input plaintext stream

INT-CS and INT-PS Notions

$\mathbf{Exp}^{\text{int-cs}}_{\mathcal{CH}}(\mathcal{A})$

declare str $\mathit{Env}, S,$ **bool** $\mathit{sync}, \mathit{win}$
$(\mathit{Mu}, \mathit{Wr}, \mathit{Re}, \mathit{De}) \twoheadleftarrow \mathit{Init}()$
$\mathit{sync} \leftarrow 1;\ \mathcal{A}_1^{\mathbf{Send},\mathbf{Recv}}(\mathbf{var}\ \mathit{Env})$
return win

$\mathbf{Send}(M, sc)$

$(X, H, \alpha) \twoheadleftarrow \mathit{Mux}^{\mathbf{SD}}(M, sc, \mathbf{var}\ \mathit{Mu})$
$(C, \gamma) \twoheadleftarrow \mathit{Write}^{\mathbf{SD}}(X, H, \alpha, \mathbf{var}\ \mathit{Wr})$
$S \leftarrow S \,\|\, C$
return (C, γ)

$\mathbf{Recv}(C)$

$(Y, H, \alpha) \twoheadleftarrow \mathit{Read}^{\mathbf{SD}}(C, \mathbf{var}\ \mathit{Re})$
$(M, sc, \gamma) \twoheadleftarrow \mathit{Demux}^{\mathbf{SD}}(Y, H, \alpha, \mathbf{var}\ \mathit{De})$
if sync **and** $Y \preceq S$ **then** $S \leftarrow S \,\%\, Y$
else $\mathit{sync} \leftarrow 0$
    $\mathit{win} \leftarrow \mathit{win} \vee (M \neq \bot \wedge sc \neq \bot)$
return (M, sc, γ)

$\mathbf{SD}(I)$

$O \twoheadleftarrow \mathcal{A}_2(I, \mathbf{var}\ \mathit{Env})$; **return** O

$\mathbf{Exp}^{\text{int-ps}}_{\mathcal{CH}}(\mathcal{A})$

declare str $\mathit{Env}, S[\,],$ **str** $R[\,],$ **bool** win
$(\mathit{Mu}, \mathit{Wr}, \mathit{Re}, \mathit{De}) \twoheadleftarrow \mathit{Init}()$
$\mathcal{A}_1^{\mathbf{Send},\mathbf{Recv}}(\mathbf{var}\ \mathit{Env})$
return win

$\mathbf{Send}(M, sc)$

$(X, H, \alpha) \twoheadleftarrow \mathit{Mux}^{\mathbf{SD}}(M, sc, \mathbf{var}\ \mathit{Mu})$
$(C, \gamma) \twoheadleftarrow \mathit{Write}^{\mathbf{SD}}(X, H, \alpha, \mathbf{var}\ \mathit{Wr})$
$S_{sc} \leftarrow S_{sc} \,\|\, M$
return (C, γ)

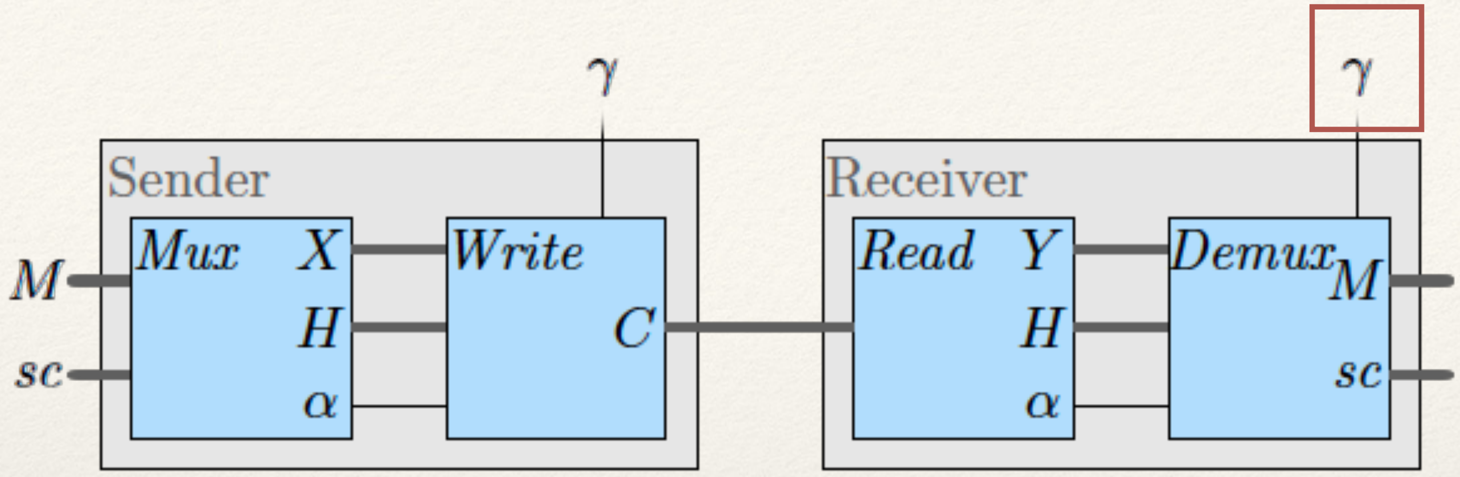
$\mathbf{Recv}(C)$

$(Y, H, \alpha) \twoheadleftarrow \mathit{Read}^{\mathbf{SD}}(C, \mathbf{var}\ \mathit{Re})$
$(M, sc, \gamma) \twoheadleftarrow \mathit{Demux}^{\mathbf{SD}}(Y, H, \alpha, \mathbf{var}\ \mathit{De})$
if $M \neq \bot$ **and** $sc \neq \bot$ **then**
    $R_{sc} \leftarrow R_{sc} \,\|\, M$
if $R_{sc} \not\preceq S_{sc}$ **then** $\mathit{win} \leftarrow 1$
return (M, sc, γ)

$\mathbf{SD}(I)$

$O \twoheadleftarrow \mathcal{A}_2(I, \mathbf{var}\ \mathit{Env})$; **return** O

$$\mathbf{Adv}^{\text{int-cs}}_{\mathcal{CH}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{int-cs}}_{\mathcal{CH}}(\mathcal{A}) = 1] \qquad \mathbf{Adv}^{\text{int-ps}}_{\mathcal{CH}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\text{int-ps}}_{\mathcal{CH}}(\mathcal{A}) = 1]$$



- PRIV-S ^ INT-CS ?⟹? PRIV-SR

Receiver-status Simulatability

- SIM-STAT
  - This notion captures what the adversary learns from the receiver's state by observing the status messages output
  - Simulation-based game : for every efficient adversary, efficient simulator such that real status messages are indistinguishable from fake ones

$\mathbf{Exp}^{\text{sim-stat}}_{\mathcal{CH},\mathcal{S},b}(\mathcal{A})$

$\quad$ **declare str** Env, S

$\quad (Mu, Wr, Re, De) \twoheadleftarrow Init()$

$\quad b' \twoheadleftarrow \mathcal{A}_1^{\mathbf{Send},\mathbf{Recv}}(\mathbf{var}\ Env)$

$\quad$ **return** b'

$\mathbf{Send}(M, sc)$

$\quad (X, H, \alpha) \twoheadleftarrow Mux^{\mathbf{SD}}(M, sc, \mathbf{var}\ Mu)$

$\quad (C, \gamma) \twoheadleftarrow Write^{\mathbf{SD}}(X, H, \alpha, \mathbf{var}\ Wr)$

$\quad S \leftarrow S \,\|\, C$

$\quad$ **return** (C, γ)

$\mathbf{Recv}(C)$

$\quad$ **if** $b = 1$ **then**

$\qquad (Y, H, \alpha) \twoheadleftarrow Read^{\mathbf{SD}}(C, \mathbf{var}\ Re)$

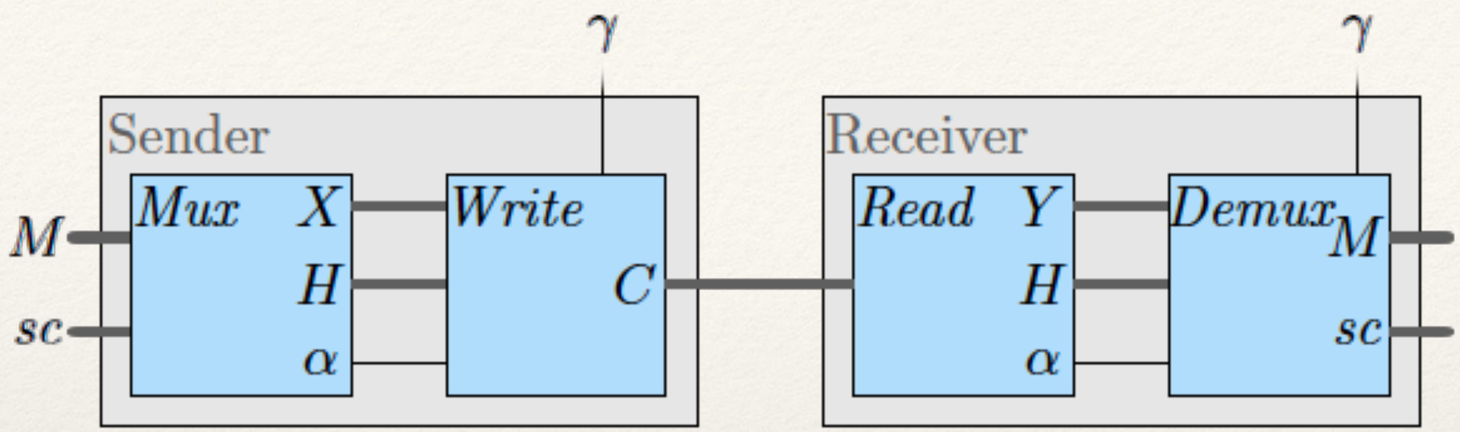
$\qquad (*, *, \gamma) \twoheadleftarrow Demux^{\mathbf{SD}}(Y, H, \alpha, \mathbf{var}\ De)$

$\quad$ **else** $\gamma \twoheadleftarrow \mathcal{S}^{\mathbf{SD}}(C, S)$

$\quad$ **return** γ

$\mathbf{SD}(I)$

$\quad O \twoheadleftarrow \mathcal{A}_2(I, \mathbf{var}\ Env)$; **return** O



- PRIV-S ^ INT-CS ^ SIM-STAT $\Longrightarrow$ PRIV-SR

The TLS 1.3 Record Layer

- Three client-server protocols executing concurrently
  - handshake : (re-)initialization of the channel
  - record : exchange application data
  - alert : close the channel

- Each flow is authenticated and encrypted as soon as client and server exchange key material

TLS 1.3 Records

- Plaintext records encode:
  - content type
  - stream fragment
  - length of fragment ($< 2^{14}$ bytes)
  - *legacy_record_version* (for backward compatibility)
- Streams of data are transformed into a sequence of records
- Record boundaries are subject to certain rules

Record Boundary Rules

- Handshake : no interleaving
- Handshake : no spanning a key change
- Handshake and Alert : no zero length messages
- One alert per record

The Core Components

- Which fully specified components can be altered without affecting security?
- Which unspecified or partially specified components are critical to security?

Observations

- Record boundaries may leak the content type!
  - Hiding the content and the type unachievable in general due to the record boundary rules
- Associated data is unauthenticated

Record Header Authentication

- Header : *opaque_type, legacy_record_version, length*
- What if the header is different than specified?
  - *length* changed : invalid with high probability
  - If the others are changed, it should be alright since it doesn't affect decryption - it is left optional in the spec
    - But this is an INT-CS attack!
    - We must authenticate the header
- To formalize that the value should not affect security, we allow the specification details to choose the bits

Is the model too strong?

- One point of view is that this does not constitute a “real attack” on privacy or integrity, since inputs to decryption were not affected
- This is correct only if down-stream handling of the plaintext is independent of these values

The Core Components

- Stream Multiplexer
  - Transforms data streams into records
  - Captures the non-cryptographic functionality
  - Consider it to be partially specified
- AEAD scheme
- Nonce generator
  - Both these are core cryptographic functionalities
  - Required to be fully specified

Partially-specified Multiplexers

- mPRIV-S
  - Captures the adversary's ability to discern information about the inputs to *Mux* given its outputs
  - Like the PRIV-S game earlier, except:
    - No Write oracle.
    - Rather than (X, γ), it returns γ and the length of X
- SIM-mSTAT
  - Captures simulatability of the status message output by *Demux*.

AEAD Scheme

- Encryption and Decryption are both deterministic
- Standard security notions are as follows:
  - PRIV
    - Indistinguishability under Chosen Plaintext Attack
  - INT
    - Integrity of Ciphertexts

Nonce Generator

- It consists of a pair of algorithms:
  - $Init() \rightarrow ng$ (randomized, initializes the state)
  - $Next(ng) \rightarrow N$ (computes the next nonce and updates state)
- $Coll$: outputs 1 if there is a nonce-reuse

Partially Specified Record Layer

- $(\mathrm{PRIV}_{\mathrm{AEAD}}) \wedge (\mathrm{mPRIV\text{-}S}_{\mathrm{Mux}}) \Longrightarrow (\mathrm{PRIV\text{-}S}_{CH})$
- $\mathrm{INT}_{\mathrm{AEAD}} \Longrightarrow \mathrm{INT\text{-}CS}_{CH}$
- The SIM-STAT security of the channel reduces to the SIM-mSTAT security of the multiplexer and the integrity of the AEAD scheme
- Can combine all these with the earlier result regarding PRIV-SR security

Conclusion

- Partial specification of protocols is simple and flexible
- Allows us to think formally about what the protocol must get right, and what it may get wrong
- Helps point out which matters are security-critical

Thank You!