On security of TLS 1.2 Record layer with Russian ciphersuites

Liliya R. Akhmetzyanova,
Lead Cryptography Analyst,
CryptoPro LLC

Evgeny K. Alekseev, Grigory K. Sedov,
Stanislav V. Smyshlyaev

Motivation

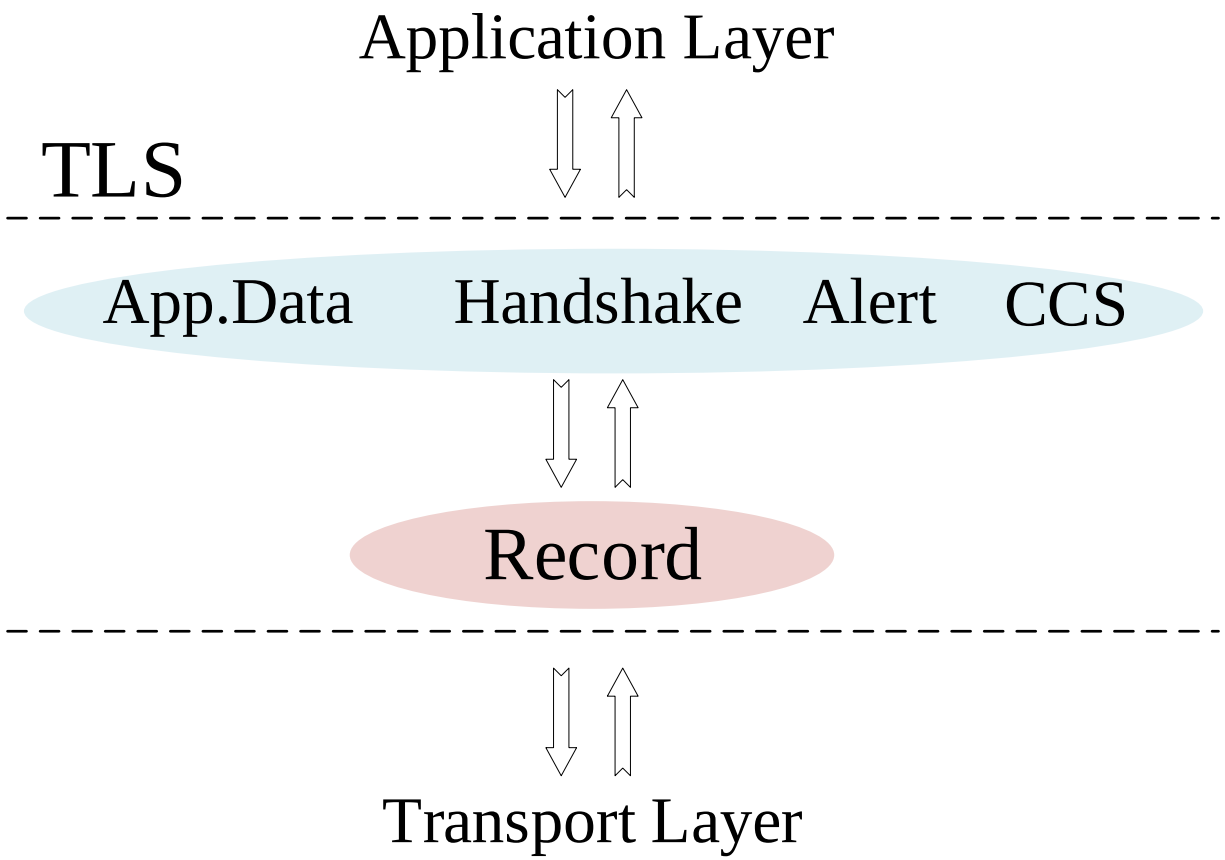
In 2019 Rosstandart has approved recommendations for standardization which define the use of TLS 1.2 ciphersuites based on the actual Russian cryptographic algortihms. In this document the following ciphersuites are defined:

- TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC;
- TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC.

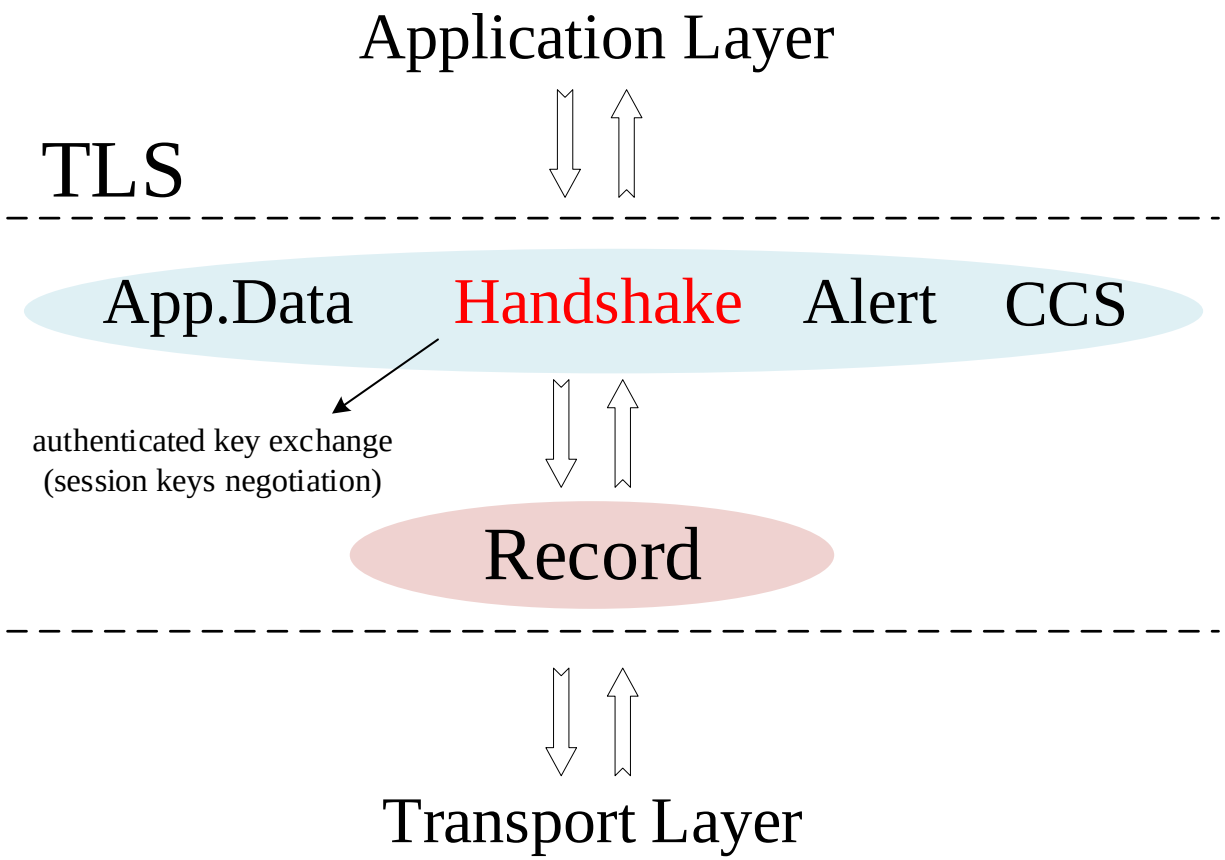
In 2019 IANA has added these ciphersuites to the "TLS Cipher Suite" registry.

Value	Description	DTLS-OK	Recommended	Reference
0xC1,0x00	TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC	N	N	[draft-smyshlyaev-tls12-gost-suites]
0xC1,0x01	TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC	N	N	[draft-smyshlyaev-tls12-gost-suites]
0xC1,0x02	TLS_GOSTR341112_256_WITH_28147_CNT_IMIT	N	N	[draft-smyshlyaev-tls12-gost-suites]

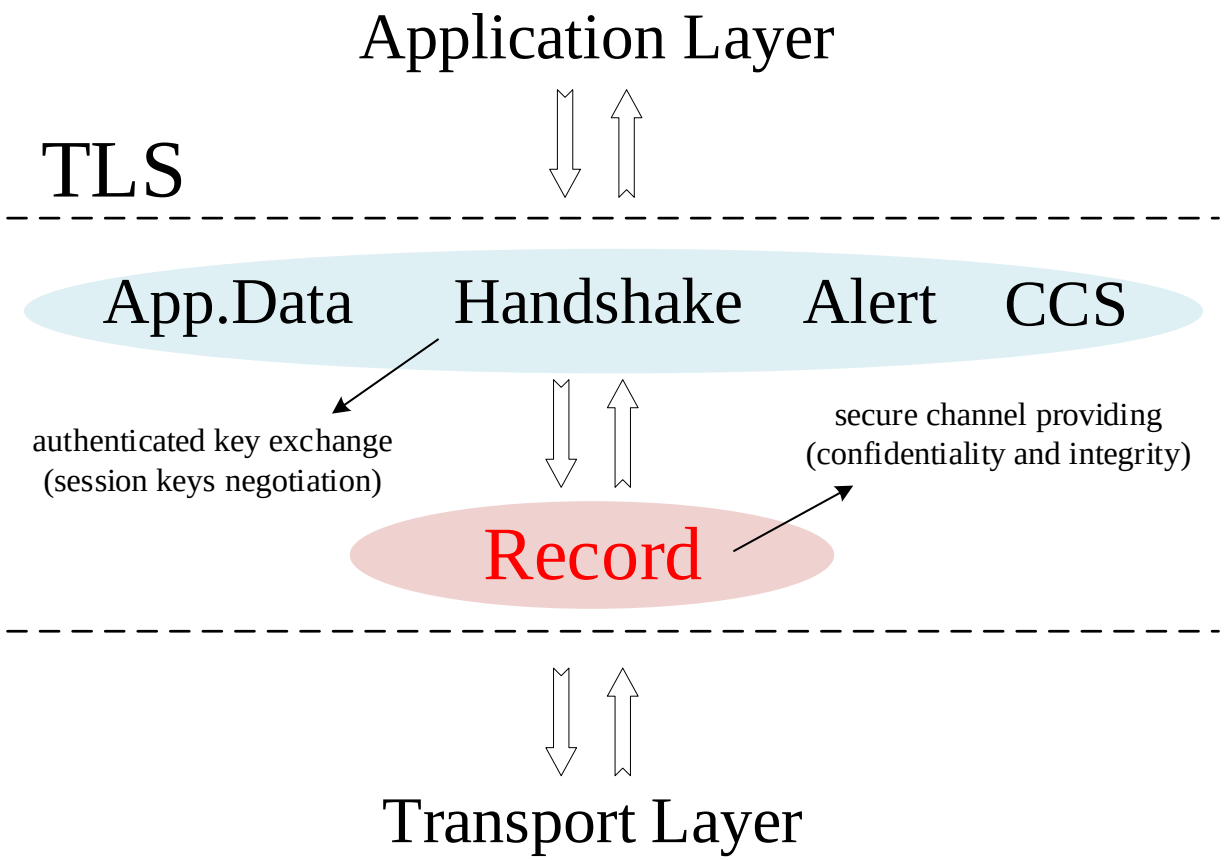
TLS structure



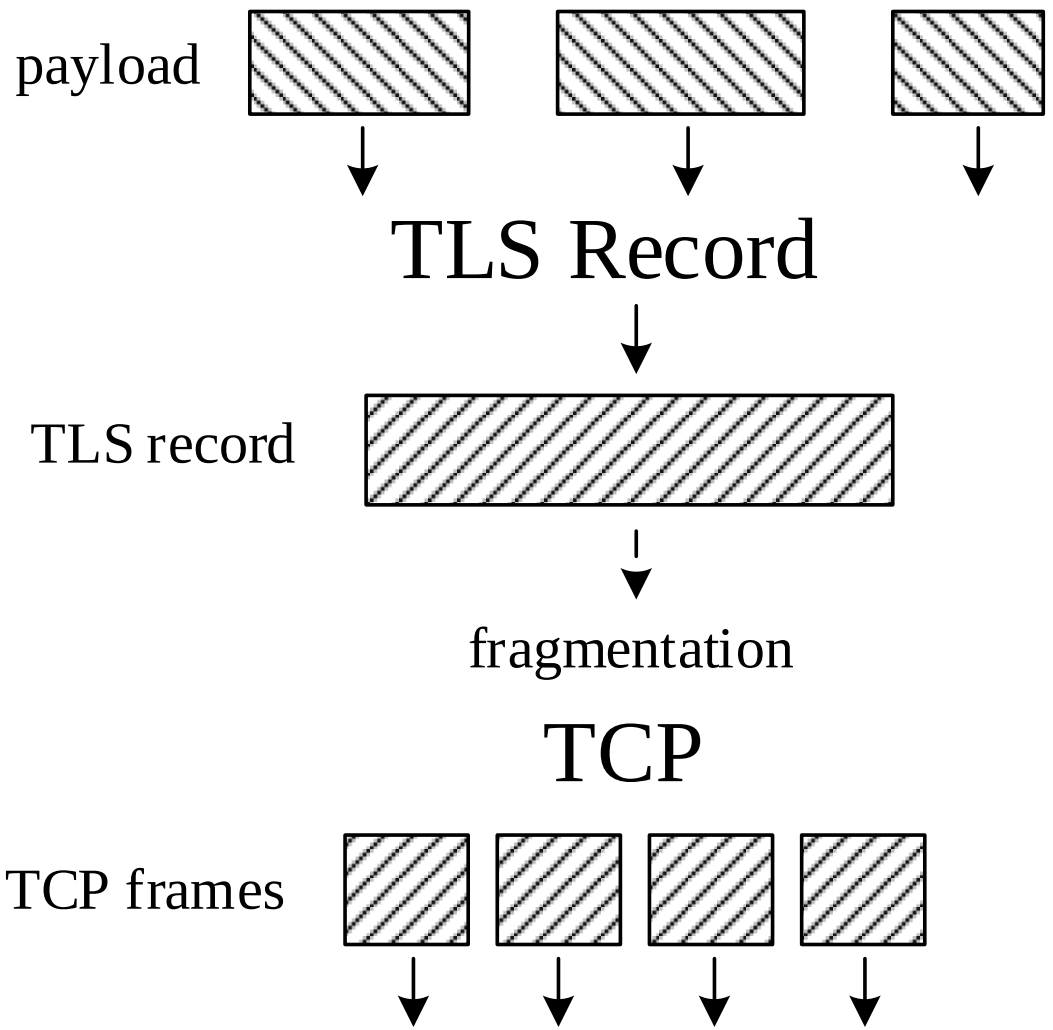
TLS structure



TLS structure



Record Payload Protection



Assumption

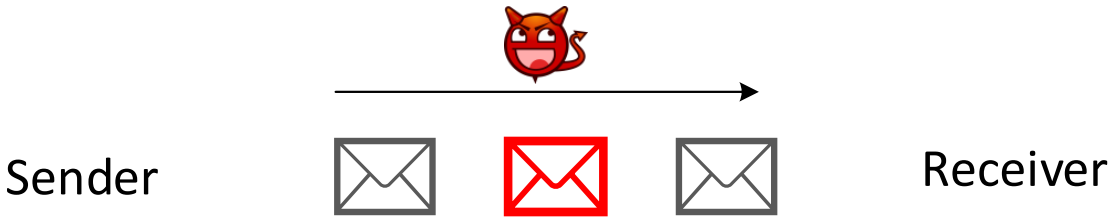
Handshake provides a «good» key material.

Secure channel

The records streams on the sender and receiver sides are equal.

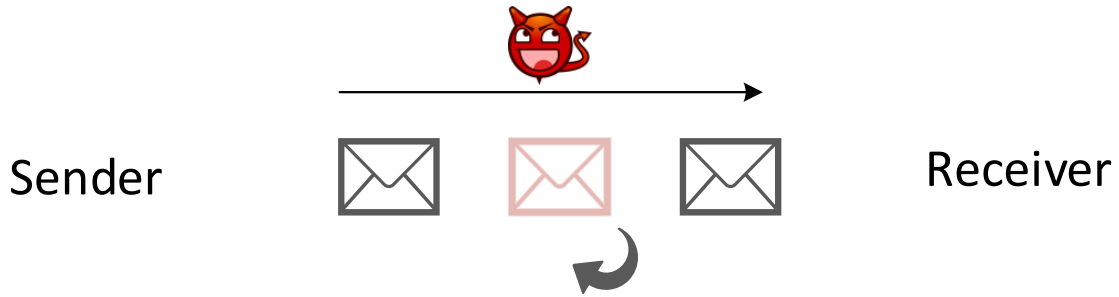
Standard security properties

Should protect against undetectable records modification:

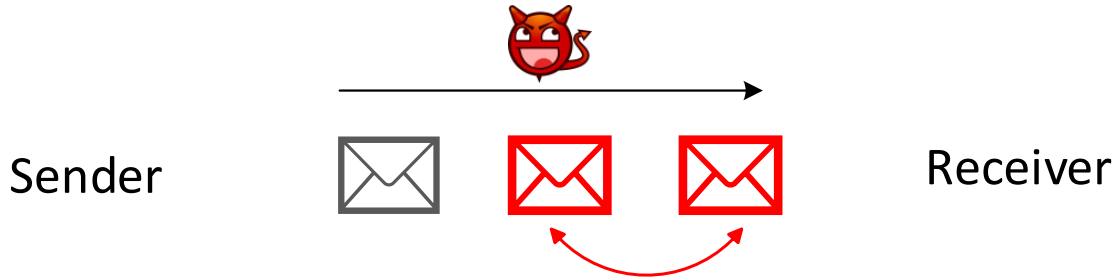


Security properties on the stream level

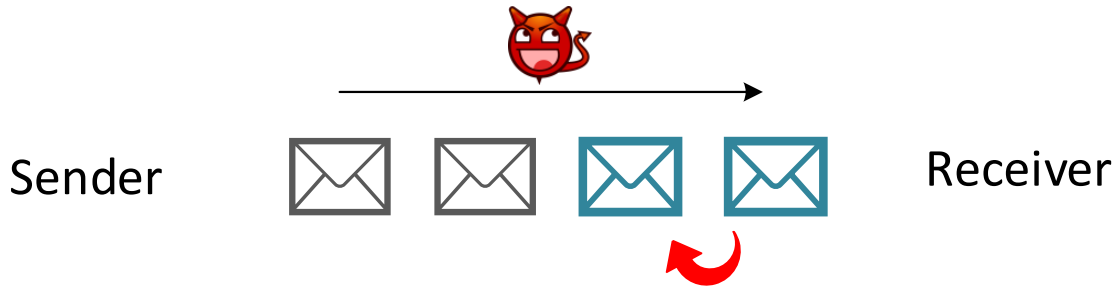
Should protect against undetectable records dropping:



Should protect against undetectable records reordering:



Should protect against undetectable records replaying:



How to formilize these security properties?

Definition (Stateful authenticated encryption with associated data scheme)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of keys, $\mathcal{M} \subseteq \{0,1\}^*$ be a set of messages, $\mathcal{AD} \subseteq \{0,1\}^*$ be a set of associated data, $\mathcal{C} \subseteq \{0,1\}^*$ be a set of ciphertexts, and $\mathcal{S}$ be a set of states. A *stateful AEAD scheme* sfAEAD is a set of algorithms

- Key generation: $\mathsf{sfAEAD.K}() \xrightarrow{\$} K \in \mathcal{K}$.
- State initialization: $\mathsf{sfAEAD.Init}(st) \to (st_E, st_D) \in \mathcal{S} \times \mathcal{S}$.
- State updating: $\mathsf{sfAEAD.Upd}(st) \to st' \in \mathcal{S}$.
- Authenticated encryption: $\mathsf{sfAEAD.E}(K, ad, m, st_E) \xrightarrow{\$} c \in \mathcal{C}$, where $ad \in \mathcal{AD}$, $m \in \mathcal{M}$.
- Authenticated decryption $\mathsf{sfAEAD.D}(K, ad, c, st_D) \to m \in M$.

IND-sfCCSA notion

Experiment is a «game» between a challenger and an adversary.

We need two challengers.

$$\underline{\mathbf{Exp}^{\text{IND-sfCCSA}-b}_{\text{sfAEAD}}(A), b \in \{0,1\}}$$

$K \xleftarrow{\$} \text{sfAEAD.K}()$
$u \leftarrow 0,\ v \leftarrow 0$
$sent \leftarrow \emptyset$
$st \leftarrow A$
$(st_E,\ st_D) \leftarrow \text{sfAEAD.Init}(st)$
$b' \leftarrow A^{\text{Encrypt}^b, \text{Decrypt}^b}$
return b'

IND-sfCCSA notion

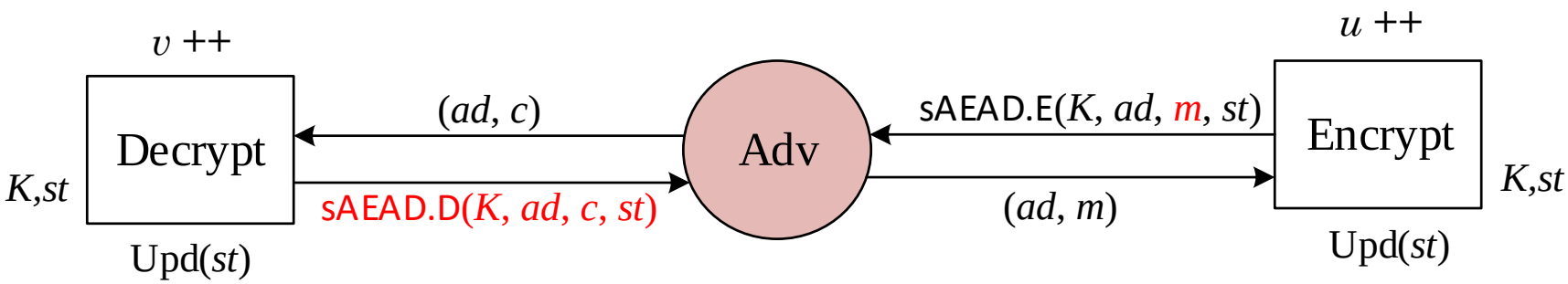
Experiment is a «game» between a challenger and an adversary.

We need two challengers.

$$\underline{\mathbf{Exp}^{\text{IND-sfCCSA}-b}_{\text{sfAEAD}}(A), b \in \{0,1\}}$$

$K \xleftarrow{\$} \text{sfAEAD.K}()$
$u \leftarrow 0,\ v \leftarrow 0$
$sent \leftarrow \emptyset$
$st \leftarrow A$
$(st_E,\ st_D) \leftarrow \text{sfAEAD.Init}(st)$
$b' \leftarrow A^{\text{Encrypt}^b, \text{Decrypt}^b}$
return b'

Experiment $\mathbf{Exp}^{\text{IND-sfCCSA}-1}_{\text{sfAEAD}}(A)$: «real world»



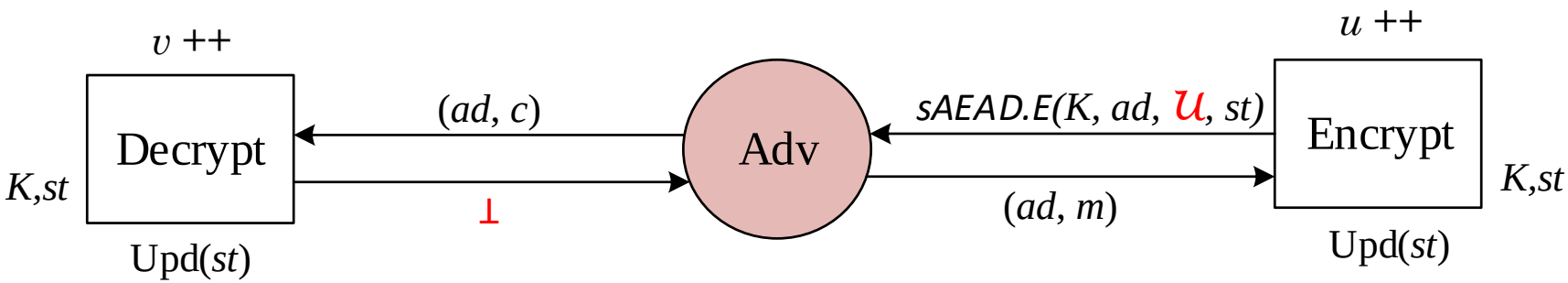
Oracle $\text{Encrypt}^1(ad, m)$

$c \leftarrow \text{sfAEAD.E}(K, ad, m, st_E)$
$sent \leftarrow sent \cup (ad, c, u)$
$st_E \leftarrow \text{sfAEAD.Upd}(st_E)$
$u \leftarrow u + 1$
return c

Oracle $\text{Decrypt}^1(ad, c)$

$m \leftarrow \text{sfAEAD.D}(K, ad, c, st_D)$
if $(m \neq \perp)$ **then**
  if $(ad, c, v) \in sent$ **then**
    $m \leftarrow \perp$ // *trivial query*
  $st_D \leftarrow \text{sfAEAD.Upd}(st_D)$
  $v \leftarrow v + 1$
return m

Experiment $\mathbf{Exp}^{\text{IND-sfCCSA}-0}_{\text{sfAEAD}}(A)$: «ideal world»



Oracle $\text{Encrypt}^0(ad, m)$

$m \xleftarrow{u} \{0,1\}^{|m|}$
$c \leftarrow \text{sfAEAD.E}(K, ad, m, st_E)$
$st_E \leftarrow \text{sfAEAD.Upd}(st_E)$
$u \leftarrow u + 1$
return c

Oracle $\text{Decrypt}^0(ad, c)$

return $\perp$

Definition (IND-sfCCSA - advantage)

The advantage of an adversary A in the model IND-sfCCSA for the stateful AEAD scheme sfAEAD is defined as:

$$\mathsf{Adv}^{\text{IND-sfCCSA}}_{\mathsf{sfAEAD}}(A) = \Pr\left[\mathbf{Exp}^{\text{IND-sfCCSA}-1}_{\mathsf{sfAEAD}}(A) \to 1\right] - \Pr\left[\mathbf{Exp}^{\text{IND-sfCCSA}-0}_{\mathsf{sfAEAD}}(A) \to 1\right].$$

Trivial queries

Oracle $\mathsf{Encrypt}^1(ad, m)$

$c \leftarrow \mathsf{sfAEAD.E}(K, ad, m, st_E)$
$sent \leftarrow sent \cup (ad, c, u)$
$st_E \leftarrow \mathsf{sfAEAD.Upd}(st_E)$
$u \leftarrow u + 1$
return c

Oracle $\mathsf{Decrypt}^1(ad, c)$

$m \leftarrow \mathsf{sfAEAD.D}(K, ad, c, st_D)$
if $(m \neq \perp)$ **then**
  if $(ad, c, v) \in sent$ **then**
    $m \leftarrow \perp$ *// trivial query*
  $st_D \leftarrow \mathsf{sfAEAD.Upd}(st_D)$
  $v \leftarrow v + 1$
return m

Definition (IND-sfCCSA - advantage)

The advantage of an adversary A in the model IND-sfCCSA for the stateful AEAD scheme sfAEAD is defined as:

$$\mathsf{Adv}^{\text{IND-sfCCSA}}_{\mathsf{sfAEAD}}(A) = \Pr\left[\mathbf{Exp}^{\text{IND-sfCCSA}-1}_{\mathsf{sfAEAD}}(A) \to 1\right] - \Pr\left[\mathbf{Exp}^{\text{IND-sfCCSA}-0}_{\mathsf{sfAEAD}}(A) \to 1\right].$$

Trivial queries

Oracle $\mathsf{Encrypt}^1(ad, m)$

$c \leftarrow \mathsf{sfAEAD.E}(K, ad, m, st_E)$
$sent \leftarrow sent \cup (ad, c, u)$
$st_E \leftarrow \mathsf{sfAEAD.Upd}(st_E)$
$u \leftarrow u + 1$
return c

Oracle $\mathsf{Decrypt}^1(ad, c)$

$m \leftarrow \mathsf{sfAEAD.D}(K, ad, c, st_D)$
if $(m \neq \perp)$ **then**
  if $(ad, c, v) \in sent$ **then**
    $m \leftarrow \perp$ // *trivial query*
  $st_D \leftarrow \mathsf{sfAEAD.Upd}(st_D)$
  $v \leftarrow v + 1$
return m

	Real	Model	Comment
header	strict format	only length restrictions	the model is more general
errors	*unexpected_message* *decode_error* *record_overflow* *bad_record_mac*	*bad_record_mac* ($\perp$)	other errors occur in headers only (no additional info for an adversary, SAE)
records	fragmentation	no fragmentation	the sender/receiver sends info to the channel only after finishing with the entire record (CFA, BCPA)
time	record time processing depends on its length	record time processing is constant	length is not confidential info, since it is written to the header (LH)

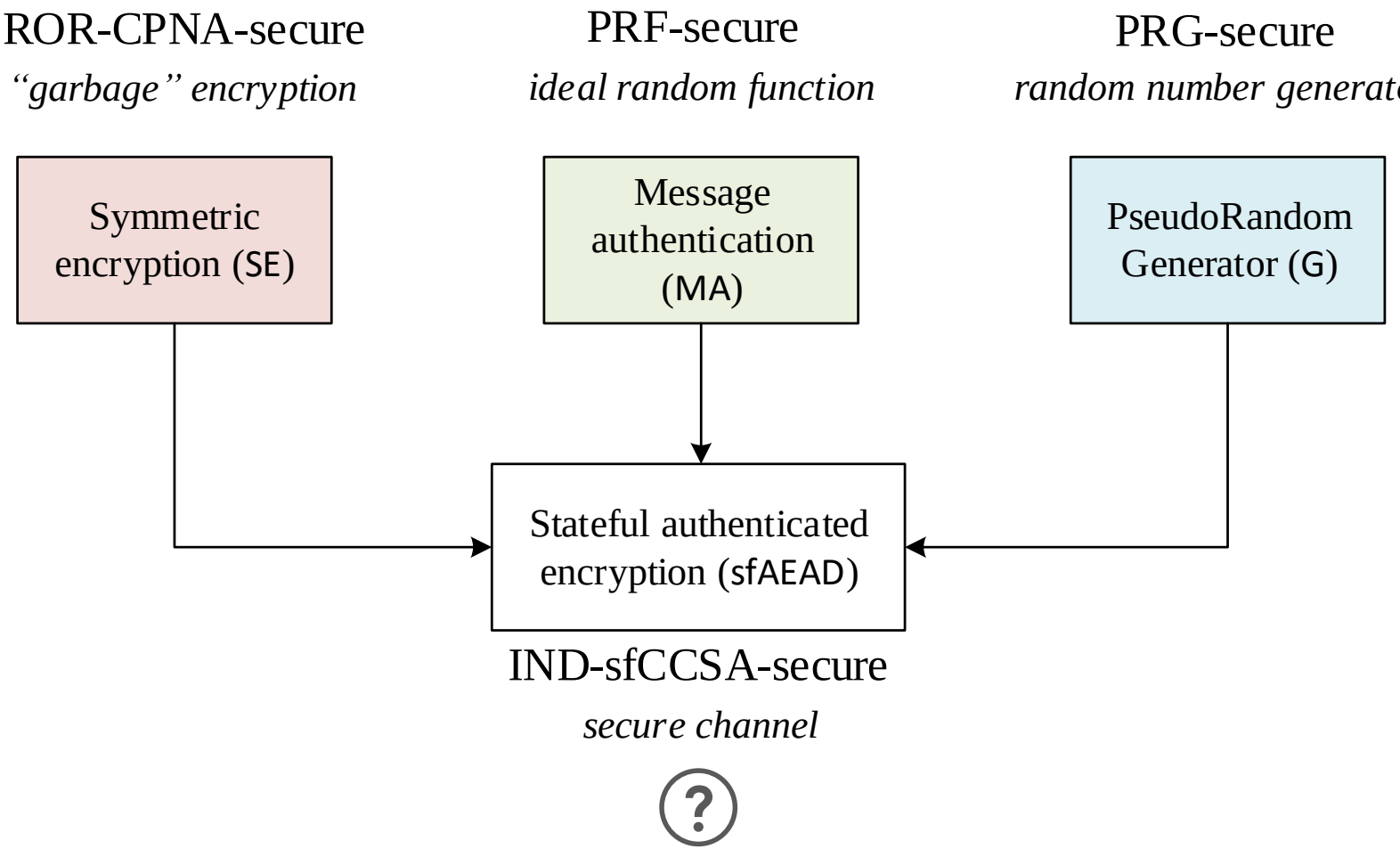
	Real	Model	Comment
message	the Record protocol does not transfer an incoming message to the application level until it is verified for integrity (AE-RUP).		
padding	the Record protocol does not use the padding procedure (no Lucky13).		

Does not capture side-channel attacks!

	Real	Model	Comment
message	the Record protocol does not transfer an incoming message to the application level until it is verified for integrity (AE-RUP).		
padding	the Record protocol does not use the padding procedure (no Lucky13).		

Does not capture side-channel attacks!

ROR-CPNA-secure	PRF-secure	PRG-secure
"garbage" encryption	*ideal random function*	*random number generator*
Symmetric encryption (SE)	Message authentication (MA)	PseudoRandom Generator (G)


ROR-CPNA-secure
"garbage" encryption
PRF-secure
ideal random function
PRG-secure
random number generator
Symmetric encryption (SE)
Message authentication (MA)
PseudoRandom Generator (G)
Stateful authenticated encryption (sfAEAD)
IND-sfCCSA-secure
secure channel
?

Definition (Encryption scheme)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of keys, $\mathcal{M} \subseteq \{0,1\}^*$ be a set of plaintexts, $\mathcal{C} \subseteq \{0,1\}^*$ be a set of ciphertexts, and $\mathcal{IV} \subseteq \{0,1\}^*$ be a set of initialization vectors. An *IV-based symmetric encryption scheme* SE is a set of the following algorithms

- Key generation: $\mathsf{SE.K} \xrightarrow{\$} K \in \mathcal{K}$;
- Encryption $\mathsf{SE.E}(K, IV, m) \to c \in \mathcal{C}$, where $IV \in \mathcal{IV}$, $m \in \mathcal{M}$;
- Decryption: $\mathsf{SE.D}(K, IV, c) \to m$.

Definition (Message authentication scheme)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of keys, $\mathcal{M} \subseteq \{0,1\}^*$ be a set of messages, $\mathcal{T} \subseteq \{0,1\}^*$ be a set of tags. A *deterministic message authentication scheme* MA is a set of the following algorithms

- Key generation: $\mathsf{MA.K} \xrightarrow{\$} K \in \mathcal{K}$;
- Tag generation: $\mathsf{MA.TAG}(K, m) \to t \in \mathcal{T}$, where $m \in \mathcal{M}$;
- Tag verification: $\mathsf{MA.VF}(K, m, t) \to r \in \{\mathsf{true}, \mathsf{false}\}$.

Definition (Encryption scheme)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of keys, $\mathcal{M} \subseteq \{0,1\}^*$ be a set of plaintexts, $\mathcal{C} \subseteq \{0,1\}^*$ be a set of ciphertexts, and $\mathcal{IV} \subseteq \{0,1\}^*$ be a set of initialization vectors. An *IV-based symmetric encryption scheme* SE is a set of the following algorithms

- Key generation: $\mathsf{SE.K} \xrightarrow{\$} K \in \mathcal{K}$;
- Encryption $\mathsf{SE.E}(K, IV, m) \to c \in \mathcal{C}$, where $IV \in \mathcal{IV}$, $m \in \mathcal{M}$;
- Decryption: $\mathsf{SE.D}(K, IV, c) \to m$.

Definition (Message authentication scheme)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of keys, $\mathcal{M} \subseteq \{0,1\}^*$ be a set of messages, $\mathcal{T} \subseteq \{0,1\}^*$ be a set of tags. A *deterministic message authentication scheme* MA is a set of the following algorithms

- Key generation: $\mathsf{MA.K} \xrightarrow{\$} K \in \mathcal{K}$;
- Tag generation: $\mathsf{MA.TAG}(K, m) \to t \in \mathcal{T}$, where $m \in \mathcal{M}$;
- Tag verification: $\mathsf{MA.VF}(K, m, t) \to r \in \{\mathsf{true}, \mathsf{false}\}$.

Definition (Generator)

Let $\mathcal{K} \subseteq \{0,1\}^*$ be a set of states and $\mathcal{B} \subseteq \{0,1\}^*$ be a set of blocks. A *generator* G is a pair of the following algorithms:

- Initial state generation algorithm: $\mathsf{G.K}() \xrightarrow{\$} St \in \mathcal{K}$;
- Next state calculation algorithm: $\mathsf{G.N}(St) \to (Out, St')$, $Out \in \mathcal{B}$, $St' \in \mathcal{K}$

$\mathsf{G.N}(St, i)$ denote the i-th block of the output sequence of the generator with the initial state St.

Stateful MtE-AD

We have:

- MA for sets $\mathcal{K}_{\mathsf{MA}}, \mathcal{M}_{\mathsf{MA}}, \mathcal{T}$.
- SE for sets $\mathcal{K}_{\mathsf{SE}}, \mathcal{M}_{\mathsf{SE}}, \mathcal{C}_{\mathsf{SE}}, \mathcal{IV}$.

Let combine them to obtain sfAEAD for sets $\mathcal{K}_{\mathsf{MA}} \times \mathcal{K}_{\mathsf{SE}}, \mathcal{AD}, \mathcal{M}, \mathcal{C}_{\mathsf{SE}}, \mathcal{S}$.

We need to define the following deterministic functions:

- $\mathsf{Next}: \mathcal{S} \to \mathcal{S}$;
- $\mathsf{encode}_{\mathsf{SE}}: \mathcal{M} \times \mathcal{T} \to \mathcal{M}_{\mathsf{SE}}$; $\mathsf{decode}_{\mathsf{SE}}: \mathcal{M}_{\mathsf{SE}} \to \mathcal{M} \times \mathcal{T}$;
- $\mathsf{encode}_{\mathsf{MA}}: \mathcal{AD} \times \mathcal{M} \times \mathcal{S} \to \mathcal{M}_{\mathsf{MA}}$;
- $\mathsf{StateToIV}: \mathcal{S} \to \mathcal{IV}$.

Stateful MtE-AD

We have:

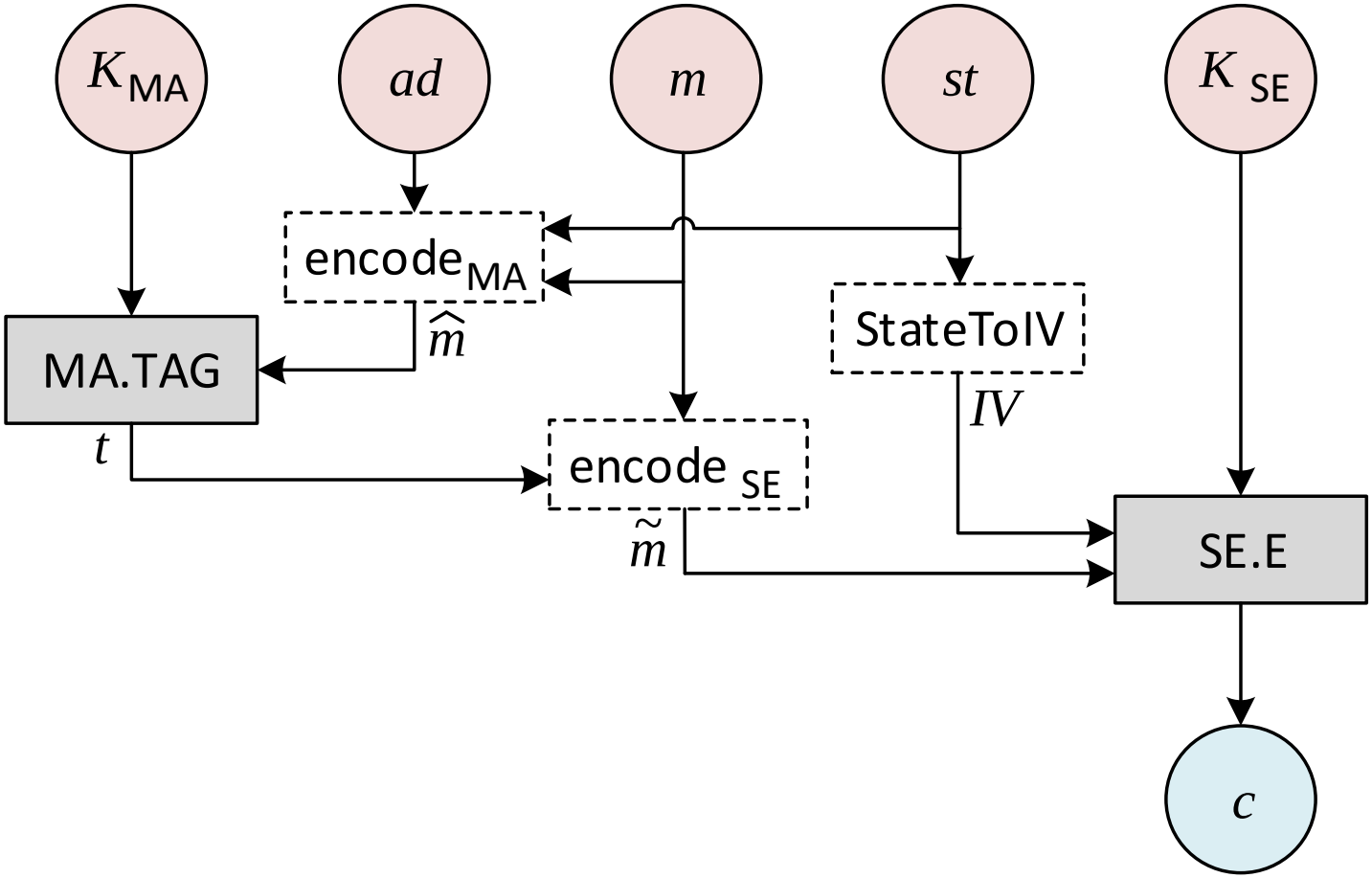
- MA for sets $\mathcal{K}_{\mathsf{MA}}, \mathcal{M}_{\mathsf{MA}}, \mathcal{T}$.
- SE for sets $\mathcal{K}_{\mathsf{SE}}, \mathcal{M}_{\mathsf{SE}}, \mathcal{C}_{\mathsf{SE}}, \mathcal{IV}$.

Let combine them to obtain sfAEAD for sets $\mathcal{K}_{\mathsf{MA}} \times \mathcal{K}_{\mathsf{SE}}, \mathcal{AD}, \mathcal{M}, \mathcal{C}_{\mathsf{SE}}, \mathcal{S}$.

We need to define the following deterministic functions:

- $\mathsf{Next}\colon \mathcal{S} \to \mathcal{S}$;
- $\mathsf{encode}_{\mathsf{SE}}\colon \mathcal{M} \times \mathcal{T} \to \mathcal{M}_{\mathsf{SE}}$; $\mathsf{decode}_{\mathsf{SE}}\colon \mathcal{M}_{\mathsf{SE}} \to \mathcal{M} \times \mathcal{T}$;
- $\mathsf{encode}_{\mathsf{MA}}\colon \mathcal{AD} \times \mathcal{M} \times \mathcal{S} \to \mathcal{M}_{\mathsf{MA}}$;
- $\mathsf{StateToIV}\colon \mathcal{S} \to \mathcal{IV}$.

Algorithm sfAEAD.E(K, ad, m, st_E)



Algorithms sfAEAD.E, sfAEAD.D

A *stateful AEAD-scheme of type MtE-AD* sfAEAD is a set of algorithms

sfAEAD.K :

$K_{\mathsf{SE}} \xleftarrow{\$} \mathsf{SE.K}()$
$K_{\mathsf{MA}} \xleftarrow{\$} \mathsf{MA.K}()$
return K

sfAEAD.Init(st) :

$st_E \leftarrow st$
$st_D \leftarrow st$
return (st_E, st_D)

sfAEAD.Upd(st) :

$st' \leftarrow \mathsf{Next}(st)$
return st'

sfAEAD.E(K, ad, m, st_E)

$\widehat{m} \leftarrow \mathsf{encode}_{\mathsf{MA}}(ad, m, st_E)$
$t \leftarrow \mathsf{MA.TAG}(K_{\mathsf{MA}}, \widehat{m})$
$IV \leftarrow \mathsf{StateToIV}(st_E)$
$\widetilde{m} \leftarrow \mathsf{encode}_{\mathsf{SE}}(m, t)$
$c \xleftarrow{\$} \mathsf{SE.E}(K_{\mathsf{SE}}, IV, \widetilde{m})$
$st_E \leftarrow \mathsf{sfAEAD.Upd}(st_E)$
return c

sfAEAD.D(K, ad, c, st_D)

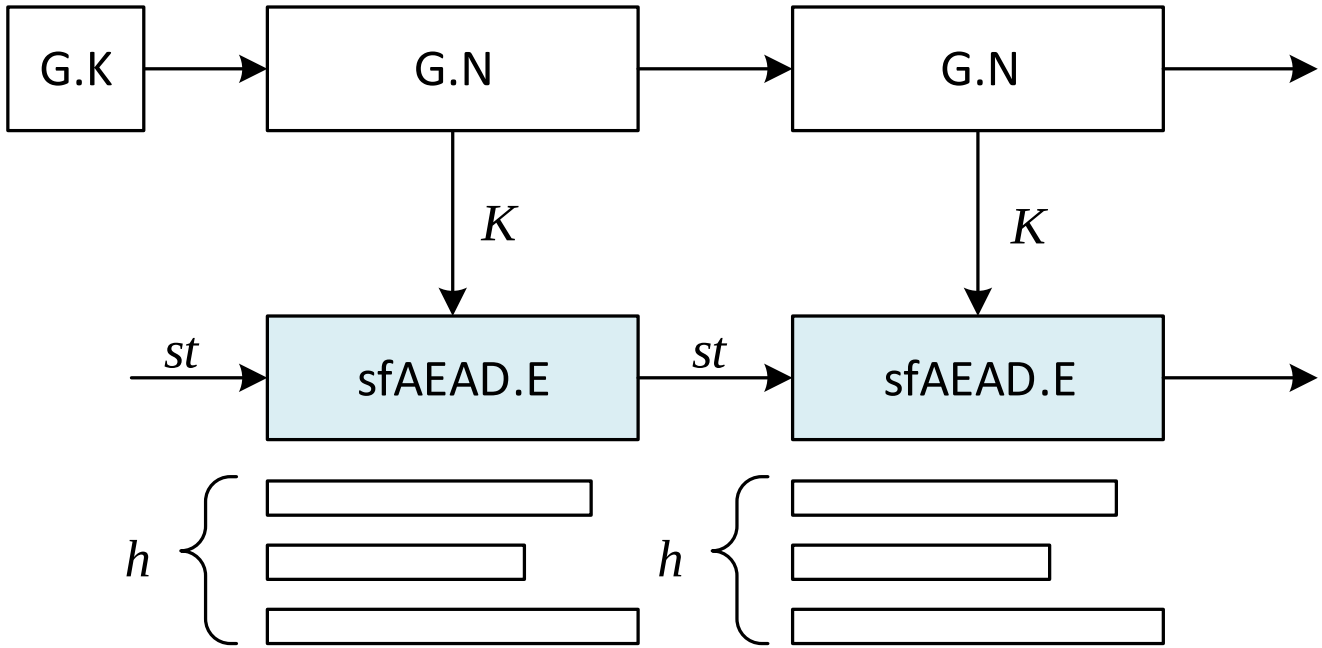
$IV \leftarrow \mathsf{StateToIV}(st_D)$
$\widetilde{m} \leftarrow \mathsf{SE.D}(K_{\mathsf{SE}}, IV, c)$
$(m, t) \leftarrow \mathsf{decode}_{\mathsf{SE}}(\widetilde{m})$
$\widehat{m} \leftarrow \mathsf{encode}_{\mathsf{MA}}(ad, m, st_D)$
if $\mathsf{MA.VF}(K_{\mathsf{MA}}, \widehat{m}, t) \neq \mathsf{true}$ **then**
    return $\perp$
$st_D \leftarrow \mathsf{sfAEAD.Upd}(st_D)$
return m

Stateful AEAD with generator

We have:

- sfAEAD for sets $\mathcal{B}$ (set of keys), $\mathcal{AD}$, $\mathcal{M}$, $\mathcal{C}$, $\mathcal{S}$.
- G for sets $\mathcal{K}$, $\mathcal{B}$.

Let combine them to obtain $(\mathsf{sfAEAD}, \mathsf{G})_h$ scheme with key diversification for sets $\mathcal{K}$, $\mathcal{AD}$, $\mathcal{M}$, $\mathcal{C}$, $\mathcal{S} \times \mathbb{N}_0$.



$(\mathsf{sfAEAD}, \mathsf{G})_h$ scheme

$\underline{(\mathsf{sfAEAD}, \mathsf{G})_h.\mathsf{K}:}$

$K \xleftarrow{\$} \mathsf{G.K}()$
return K

$\underline{(\mathsf{sfAEAD}, \mathsf{G})_h.\mathsf{Init}(st):}$

$(st_E, st_D) \leftarrow \mathsf{sfAEAD.Init}(st)$
return $(st_E, 0), (st_D, 0)$

$\underline{(\mathsf{sfAEAD}, \mathsf{G})_h.\mathsf{Upd}(st):}$

$st'.st \leftarrow \mathsf{sfAEAD.Upd}(st.st)$
$st'.u \leftarrow st'.u + 1$
return st'

$\underline{(\mathsf{sfAEAD}, \mathsf{G})_h.\mathsf{E}(K, ad, m, st_E)}$

$i \leftarrow \lfloor st_E.u/h \rfloor$
$K_i \leftarrow \mathsf{G.N}(K, i)$
$c \leftarrow \mathsf{sfAEAD.E}(K_i, ad, m, st_E.st)$
return c

$\underline{(\mathsf{sfAEAD}, \mathsf{G})_h.\mathsf{D}(K, ad, c, st_D)}$

$i \leftarrow \lfloor st_E.u/h \rfloor$
$K_i \leftarrow \mathsf{G.N}(K, i)$
$m \leftarrow \mathsf{sfAEAD.D}(K_i, ad, m, st_D.st)$
return m

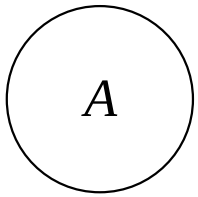
Theorem

Let G *be a generator and* sfAEAD *be a stateful AEAD-scheme of type MtE-AD and the following conditions hold:*

- *the IV-based encryption scheme* SE *is a* CRD*-scheme;*
- *the Message authentication scheme* MA *is such that the set* $\mathcal{T}$ *is* $\{0,1\}^{\tau}$*;*
- Next *is a bijective function such that* $\alpha_{\min} = \min\limits_{st \in \mathcal{S}} \alpha(st)$*;*
- StateToIV *is an injective function with according to* Next*;*
- $\mathsf{encode}_{\mathsf{MA}}$ *is an r-adding collision free function with according to* Next*;*
- $\mathsf{decode}_{\mathsf{SE}}$ *is injective.*

Theorem

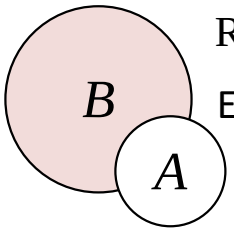
For any adversary A for $(\mathsf{sfAEAD}, \mathsf{G})_h$, $h \leqslant \alpha_{\min}$, in the IND-sfCCSA *model, there exist an adversary B for SE in the* ROR-CPNA *model, an adversary C for MA in the* PRF *model and an adversary D for G in the* PRG *model, such that*



IND-sfCCSA model

Encrypt: q_E $|ad| + |c| \leq l$

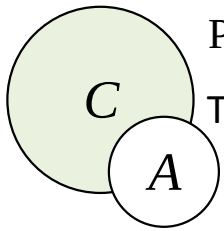
Decrypt: q_D $|ad| + |m| \leq l$



ROR-CPNA model

Encrypt: $q \leq \min(q_E, h)$

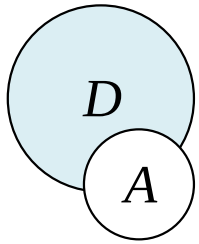
$|m| \leq l + \tau$



PRF model

TAG: $q \leq \min(q_E, h) + q_D$

$|m| \leq l + r$

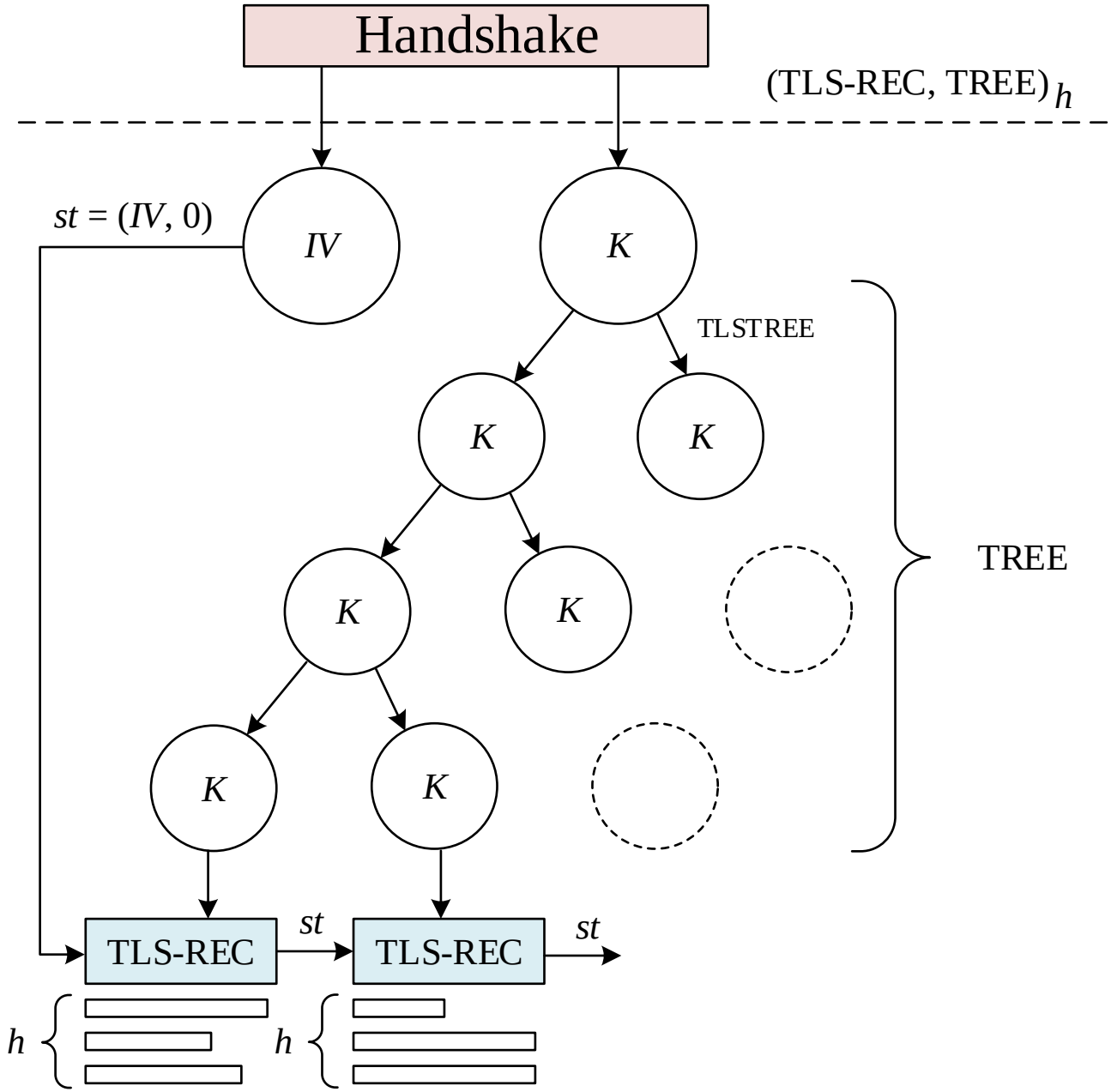


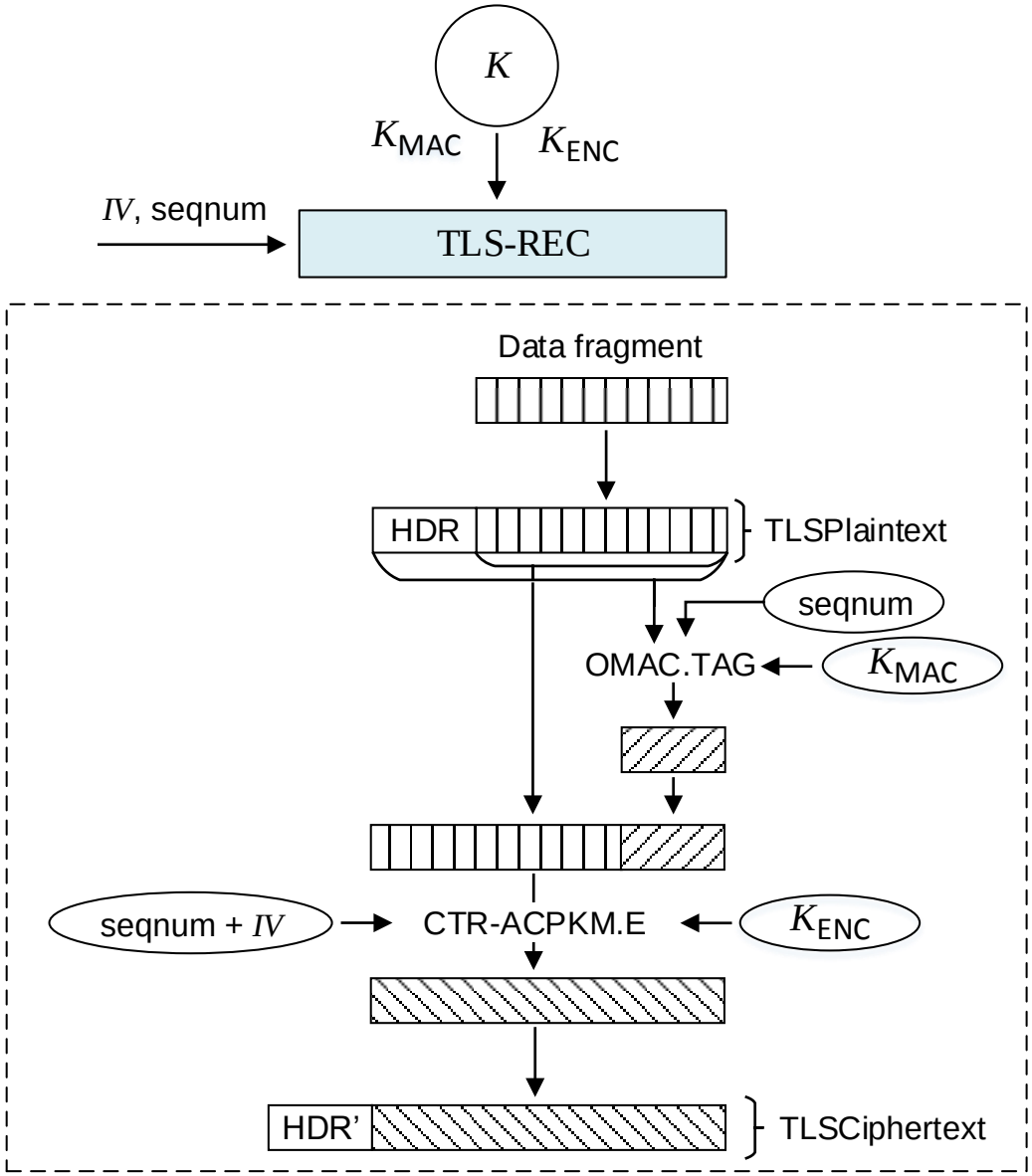
PRG model

q_E / h blocks

$$\mathsf{Adv}^{\text{IND-sfCCSA}}_{(\mathsf{sfAEAD},\mathsf{G})_h}(A) \leqslant N \cdot \mathsf{Adv}^{\text{ROR-CPNA}}_{\mathsf{SE}}(B) + 2N \cdot \mathsf{Adv}^{\text{PRF}}_{\mathsf{MA}}(C) + \frac{Nq_D}{2^n} + 2 \cdot \mathsf{Adv}^{\text{PRG}}_{\mathsf{G}}(D),$$

where $N = \lceil q_E/h \rceil$.


Handshake
(TLS-REC, TREE)$_h$
$st = (IV, 0)$
IV
K
TLSTREE
K
K
K
K
K
K
TREE
TLS-REC
st
TLS-REC
st
h
h


K
K_MAC
K_ENC
IV, seqnum
TLS-REC
Data fragment
HDR
TLSPlaintext
seqnum
OMAC.TAG
K_MAC
seqnum + IV
CTR-ACPKM.E
K_ENC
HDR'
TLSCiphertext

$(\text{TLS-REC}, \text{TREE})_h$ scheme

- $\mathsf{MA} = \text{OMAC for } \mathcal{T} = \{0,1\}^n$
- $\mathsf{SE} = \text{CTR-ACPKM for } \mathcal{IV} = \{0,1\}^{n/2}$
- $\mathsf{G} = \text{TREE}$

$\mathcal{K} = \{0,1\}^k \times \{0,1\}^k, \mathcal{AD} = \{0,1\}^{40}, \mathcal{M} = \mathcal{C} = \{0,1\}^*, \mathcal{S} = \mathcal{IV} \times \mathbb{Z}_{2^{64}}$

- $\mathsf{encode}_{\mathsf{MA}}(ad, m, st) \to \widehat{m} = \mathsf{str}_{64}(st.sn) \| ad \| m;$
- $\mathsf{encode}_{\mathsf{SE}}(m, t) \to \widetilde{m} = m \| t,$
  $\mathsf{decode}_{\mathsf{SE}}(\widetilde{m}) \to (m, t) = (\mathsf{msb}_{|\widetilde{m}|-n}(\widetilde{m}), \mathsf{lsb}_n(\widetilde{m}));$
- $\mathsf{StateToIV}(st) \to IV = \mathsf{str}_{n/2}((st.sn + \mathsf{int}(st.IV)) \mod 2^{n/2});$
- $\mathsf{Next}(st) \to st' = (st.IV, (st.sn + 1) \mod 2^{64}).$

$(\text{TLS-REC}, \text{TREE})_h$ scheme

- $\mathsf{MA} = \text{OMAC for } \mathcal{T} = \{0,1\}^n$
- $\mathsf{SE} = \text{CTR-ACPKM for } \mathcal{IV} = \{0,1\}^{n/2}$
- $\mathsf{G} = \text{TREE}$

$$\mathcal{K} = \{0,1\}^k \times \{0,1\}^k,\ \mathcal{AD} = \{0,1\}^{40},\ \mathcal{M} = \mathcal{C} = \{0,1\}^*,\ \mathcal{S} = \mathcal{IV} \times \mathbb{Z}_{2^{64}}$$

- $\mathsf{encode}_{\mathsf{MA}}(ad, m, st) \to \widehat{m} = \mathsf{str}_{64}(st.sn) \| ad \| m;$
- $\mathsf{encode}_{\mathsf{SE}}(m, t) \to \widetilde{m} = m \| t,$
  $\mathsf{decode}_{\mathsf{SE}}(\widetilde{m}) \to (m, t) = (\mathsf{msb}_{|\widetilde{m}|-n}(\widetilde{m}), \mathsf{lsb}_n(\widetilde{m}));$
- $\mathsf{StateToIV}(st) \to IV = \mathsf{str}_{n/2}((st.sn + \mathsf{int}(st.IV)) \mod 2^{n/2});$
- $\mathsf{Next}(st) \to st' = (st.IV, (st.sn + 1) \mod 2^{64}).$

$(\text{TLS-REC}, \text{TREE})_h$ scheme

- $\mathsf{MA} = \text{OMAC for } \mathcal{T} = \{0,1\}^n$
- $\mathsf{SE} = \text{CTR-ACPKM for } \mathcal{IV} = \{0,1\}^{n/2}$
- $\mathsf{G} = \text{TREE}$

$$\mathcal{K} = \{0,1\}^k \times \{0,1\}^k,\ \mathcal{AD} = \{0,1\}^{40},\ \mathcal{M} = \mathcal{C} = \{0,1\}^*,\ \mathcal{S} = \mathcal{IV} \times \mathbb{Z}_{2^{64}}$$

- $\mathsf{encode}_{\mathsf{MA}}(ad, m, st) \to \widehat{m} = \mathsf{str}_{64}(st.sn) \| ad \| m;$
- $\mathsf{encode}_{\mathsf{SE}}(m, t) \to \widetilde{m} = m \| t,$
  $\mathsf{decode}_{\mathsf{SE}}(\widetilde{m}) \to (m, t) = (\mathsf{msb}_{|\widetilde{m}|-n}(\widetilde{m}), \mathsf{lsb}_n(\widetilde{m}));$
- $\mathsf{StateToIV}(st) \to IV = \mathsf{str}_{n/2}((st.sn + \mathsf{int}(st.IV)) \mod 2^{n/2});$
- $\mathsf{Next}(st) \to st' = (st.IV, (st.sn + 1) \mod 2^{64}).$

Security bound for the Record layer

$$\mathsf{InSec}^{\text{IND-sfCCSA}}_{(\text{TLS-REC},\text{TREE})_h}(t, q_E, 1, l) \leqslant$$

$$\leqslant 2 \cdot \mathsf{InSec}^{\text{PRG}}_{\text{TREE}}(t_1, \left\lceil \frac{q_E}{h} \right\rceil) + \left\lceil \frac{q_E}{h} \right\rceil \cdot \Big(\mathsf{InSec}^{\text{ROR-CPNA}}_{\text{CTR-ACPKM}_S}(t_2, h, \lceil l/n \rceil + 1) +$$

$$+ 2 \cdot \mathsf{InSec}^{\text{PRF}}_{\text{OMAC}}(t_3, h+1, \lceil l/n \rceil + 1) + \frac{1}{2^n} \Big),$$

where

- n is a block size;
- q_E is a total number of encrypted records;
- h is a number of messages processed using one «leaf» key;
- l is a maximum message length (in bits);
- $t \approx t_1 \approx t_2 \approx t_3$ is computational resources of an adversary.

Security bound for the Record layer

Ciphersuite	s	n	h	Security bound	
				$q_E < h$	$q_E \geqslant h$
KUZNYECHIK	2^{15}	2^7	64	$q_E^2 \cdot (l+2^7)^2 \cdot 2^{-140}$	$q_E \cdot (l+2^7)^2 \cdot 2^{-126}$
MAGMA	2^{13}	2^6	4096	$q_E^2 \cdot (l+2^6)^2 \cdot 2^{-74}$	$q_E \cdot (l+2^6)^2 \cdot 2^{-62}$

Table: Security bounds for TLS-REC ciphersuites.

Thank you for your attention!

Questions?

Questions, comments:

- alekseev@cryptopro.ru
- lah@cryptopro.ru
- svs@cryptopro.ru