On Active Attack Detection in Messaging with Immediate Decryption

Khashayar Barooti, Daniel Collins, Simone Colombo,
Loïs Huguenin-Dumittan, Serge Vaudenay

EPFL

CRYPTO 2023, NYU Crypto Reading Group

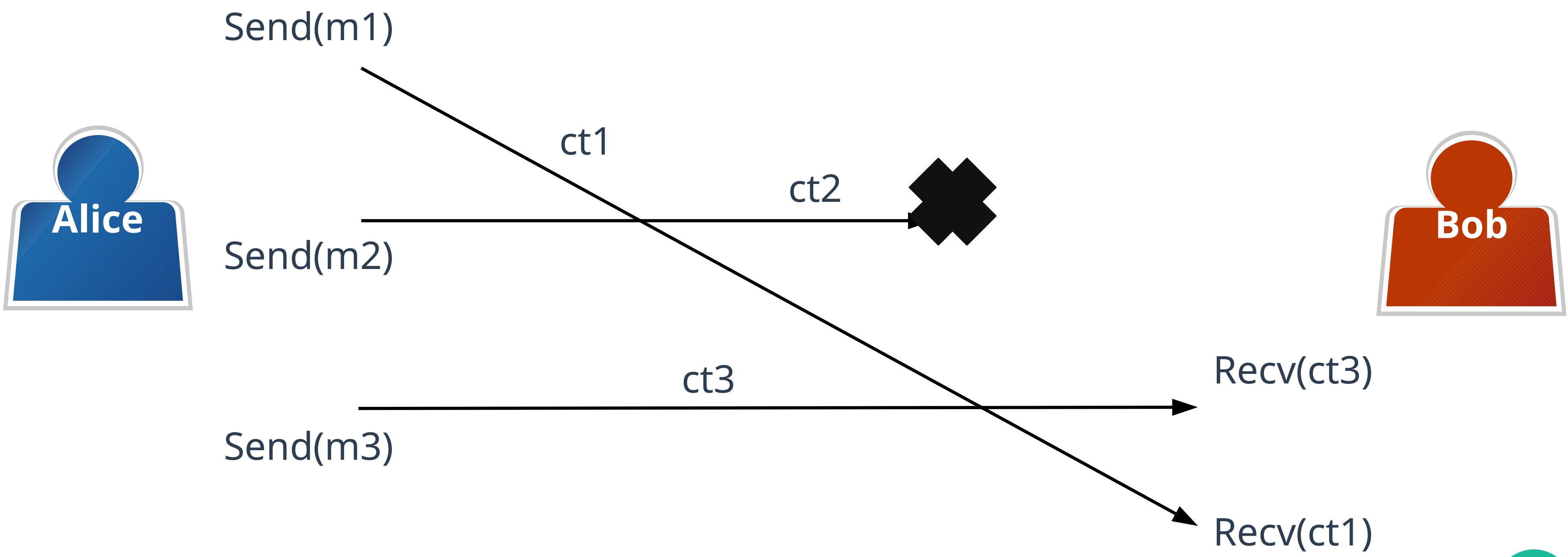
On Active Attack Detection in Messaging with Immediate Decryption

- Messaging apps are used by billions daily.
- We consider two–party chats between Alice and Bob.
- The Signal protocol is used by WhatsApp, Signal, …
- The Double Ratchet core offers forward security and post–compromise security.

On Active Attack Detection in Messaging with Immediate Decryption

- The Double Ratchet provides *immediate decryption* [ACD19].
- On the protocol level, messages can be dropped/reordered without stalling future communication.
- Helpful in demanding network settings and for performance.
- [PP22, BRT23, CZ24] (two–party) and MLS (group) also consider immediate decryption.

On Active Attack Detection in Messaging with Immediate Decryption



On Active Attack Detection in Messaging with Immediate Decryption

- **We consider an adversary that can *compromise* parties.**
- **After compromise, the adversary can trivially impersonate them and inject messages.**

On Active Attack Detection in Messaging with Immediate Decryption

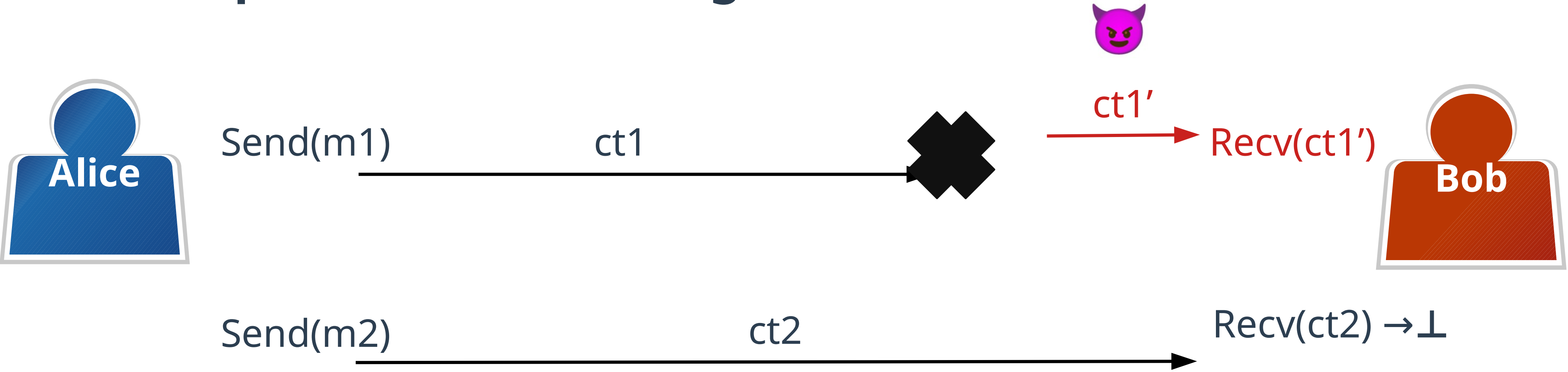
- In the worst case, the adversary can continue impersonating a party *forever*.

On Active Attack Detection in Messaging with Immediate Decryption

- Two main settings: in-band and out-of-band.
- In-band: if a honest message gets through after an active attack, *detection is possible*.
- Out-of-band: authentic out-of-band channel (e.g. QR code) to detect any attack.
- We focus first on in-order detection.

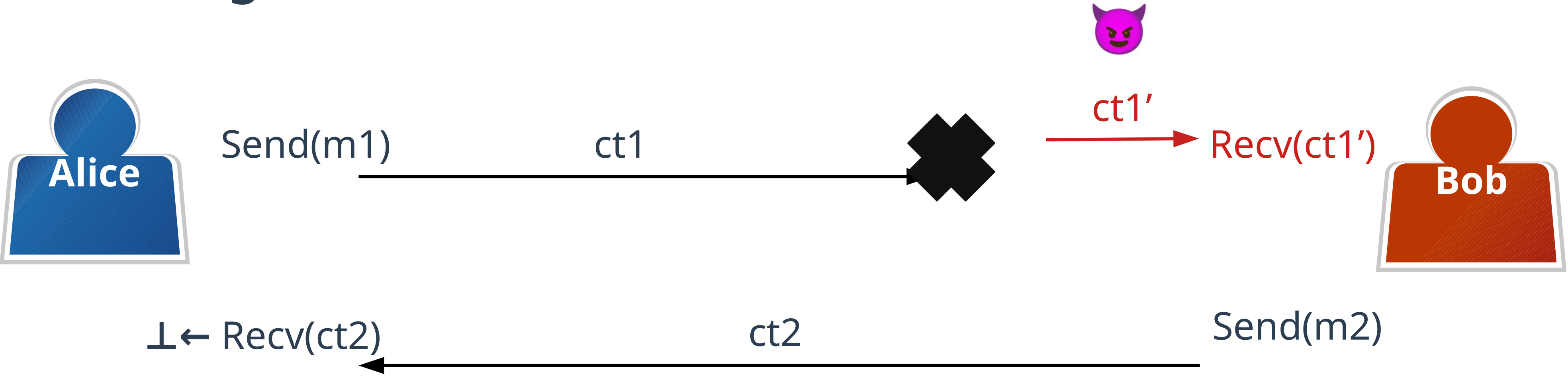
r–RECOVER Security (in–order communication) [DV19]

- **If Bob receives a forgery, <u>he must stop accepting subsequent honest messages</u>.**



s–RECOVER Security (in–order communication) [CDV21]

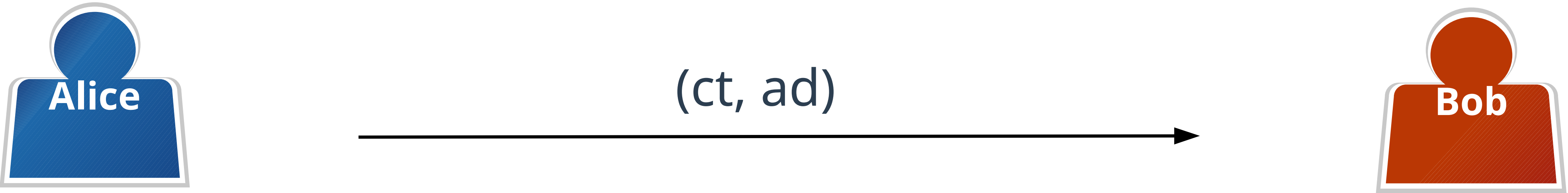
- **If Bob receives a forgery, Alice <u>must stop accepting future messages *from Bob*</u>.**



Our Contributions

- Define RECOVER with immediate decryption (RID) notions.
- A first construction satisfying r–RID and s–RID security.
- Linear communication lower bound for r–RID.
- Circumventing the lower bound: optimisations for s–RID.
- Out–of–band constructions with different trade–offs and security notions.

Messaging Primitive



Send(st, ad, pt) → (st', num, ct)

Recv(st, ad, ct) → (acc, st', num, pt)

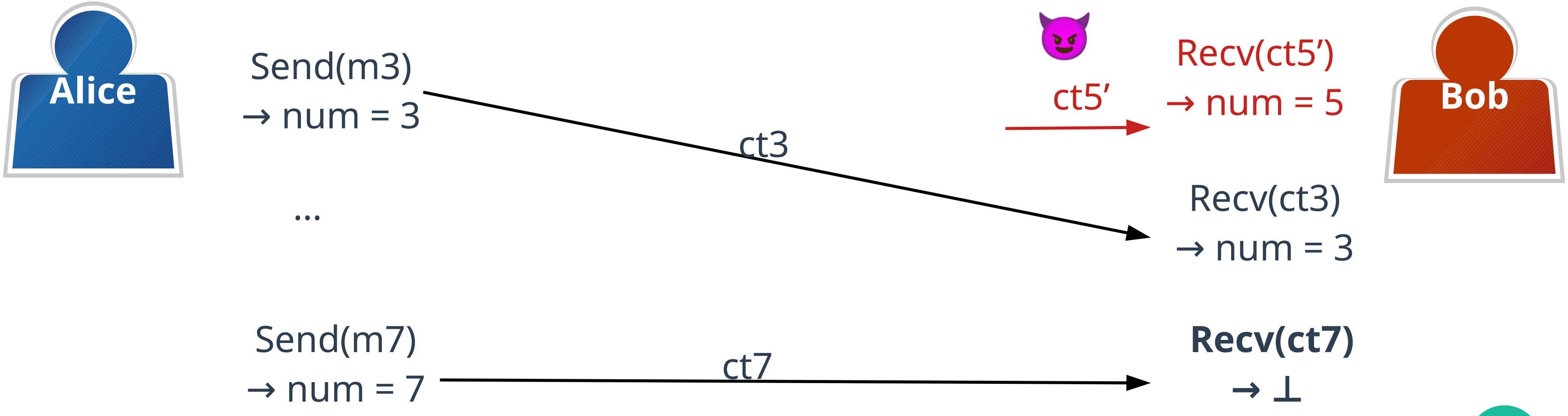
- **We assume ordinals of the form num are totally ordered.**
- **Ordinals in the Double Ratchet [ACD19]: (epoch, index) pair.**

RECOVER with Immediate Decryption (RID) Notions

- Generalises RECOVER notions from [DV19] and [CDV21] to the out-of-order setting.
- RID = r-RID + s-RID
- The adversary can freely expose states, control randomness and invoke Send/Recv via oracles.

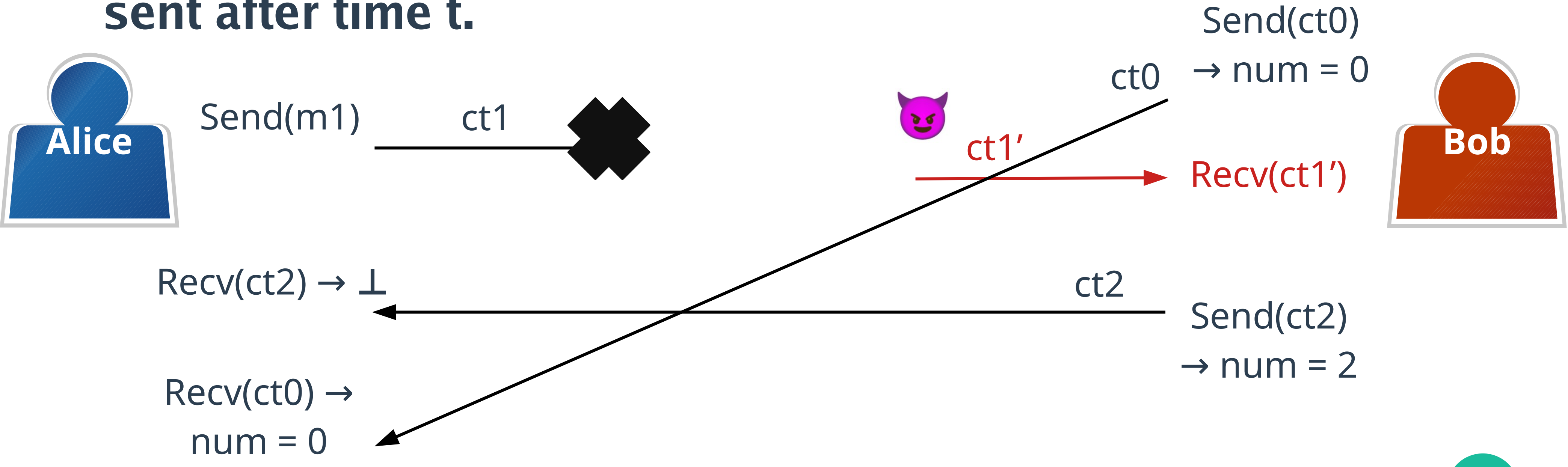
r–RID Security

- **If Bob receives a forgery for num, then the adversary wins if Bob ever receives an honest message with num’ > num.**



s–RID Security

- **If Bob receives a forgery for num at time t, then the adversary wins if Alice ever receives an honest message sent after time t.**

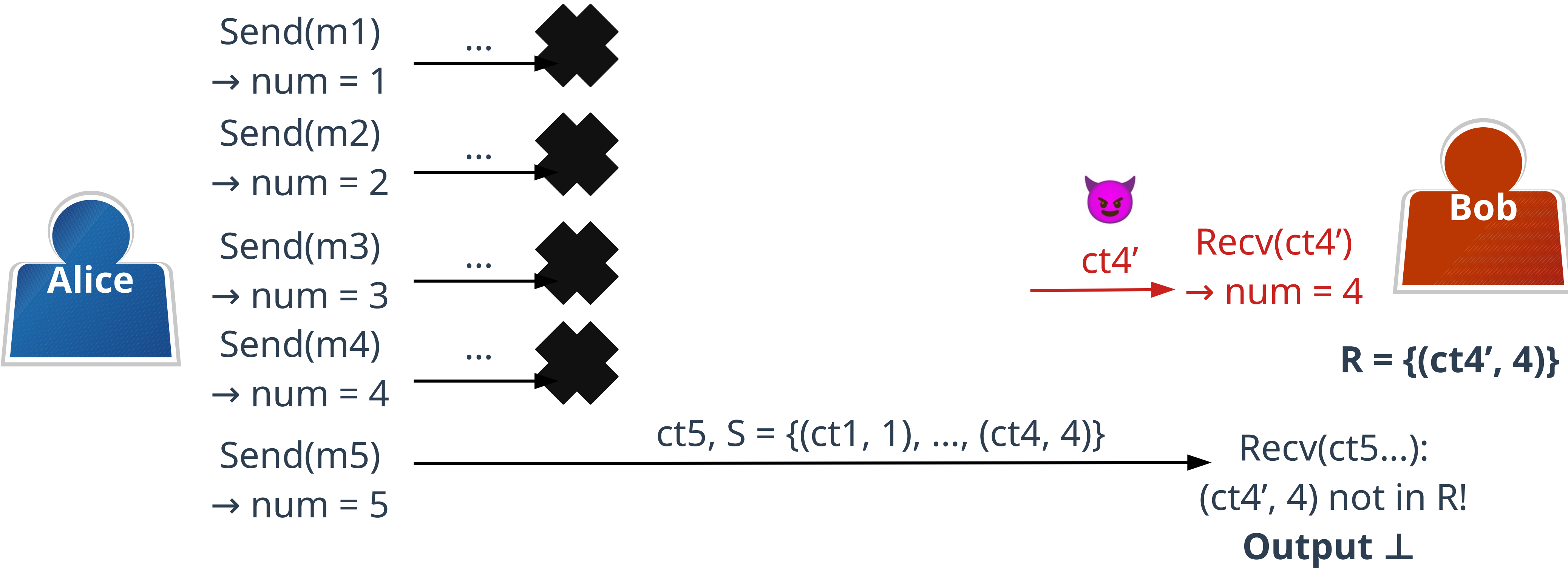


A First RID Construction

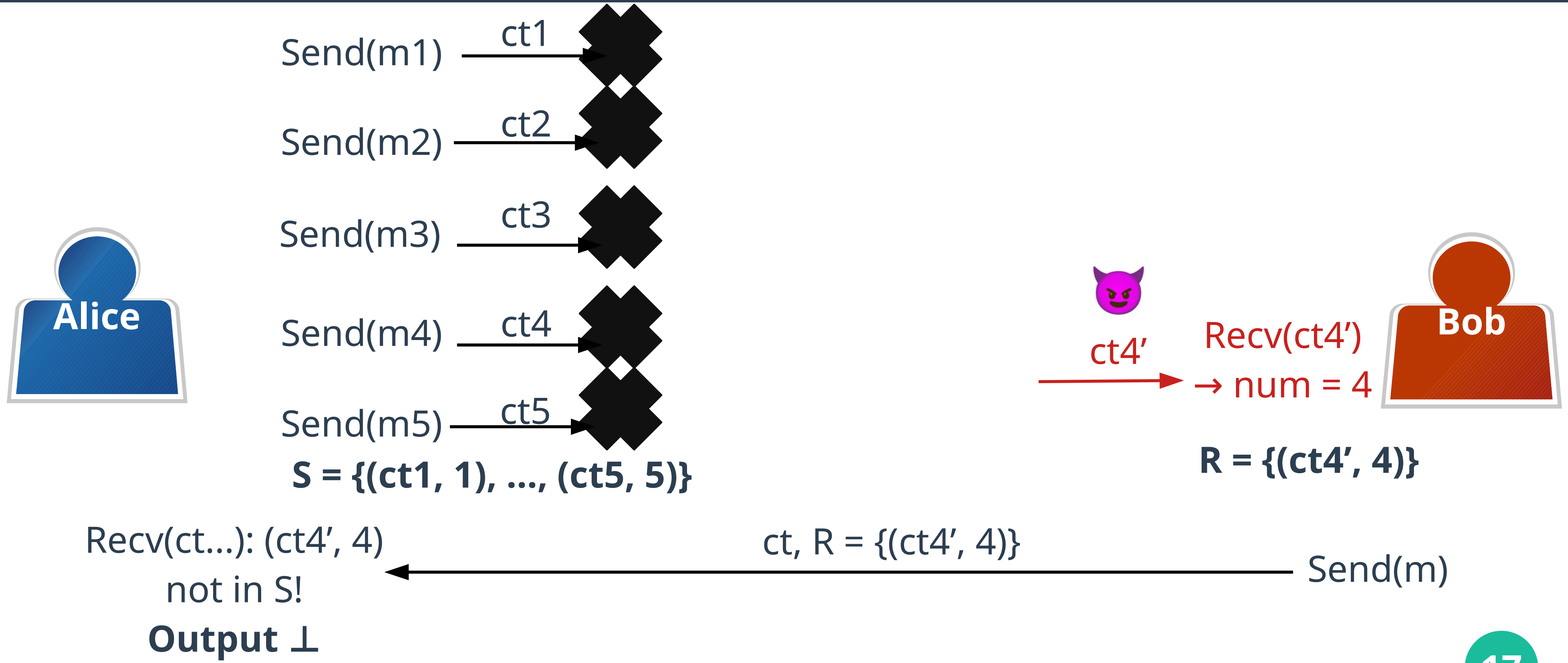
- We build a compiler on top of any Ch = (Send, Recv).
- Naive idea: attach all sent and received messages in every Send call; check for contradictions in every Recv call.
- We get r–RID from attaching *sent* messages.
- We get s–RID from attaching *received* messages.
- (Simplified:) Messages are stored as (ct, ad, num) tuples.

  [The checks are a bit delicate since the adversary can try to forge ciphertexts to bypass them.]

Example (r–RID)



Example (s–RID)



r–RID Lower Bound

- Our construction is very costly (linear growth).
- We show for r–RID that linear growth is unavoidable.
  – Intuitively, a ciphertext must 'contain' all previously sent messages.
  – If an honest ciphertext with ordinal num is delivered, all forgeries with num' < num should thereafter be rejected!

r–RID Lower Bound Statement

- Suppose Alice sends n_s messages of length $L \le n$ in a row.
- Then, the ciphertext space grows exponentially in $O(n + \lambda n_s)$ for security parameter λ.

Theorem 5. *Let Π be a perfectly correct RC, n_s and λ be fixed, and T_{λ,n_s} be the time complexity of the (efficient) adversary given on the left of Figure $\boxed{10}$. In addition, let $\gamma \in \mathbb{Z}$ be such that for all adversaries $\mathcal{A}$ running in at most time T_{λ,n_s} which send at most n_s messages, we have:* $\Pr[\textsf{r-RID}_{\Pi}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] \le \frac{1}{2^{\gamma}}$. *Let $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \{0,1\}^k$ be the plaintext and ciphertext space associated to Π, respectively. Then,*

$$k \ge n + (n_s - 1)(\gamma - 2), \text{ if } \gamma \le n$$
$$k \ge 2 + n_s(n - 2), \text{ if } \gamma > n.$$

r–RID Lower Bound Proof Sketch

- We construct an (inefficient) encoder/decoder pair E/D.
- Both take Alice and Bob's initial state as input.
- E additionally takes as input n_s messages and outputs ciphertext n_s.
- Invoking Shannon's source coding theorem we arrive at the bound on the ciphertext space.

r–RID Lower Bound Proof Sketch 2

- E takes as input n_s messages, sends the input messages using Alice's state and Send, and outputs ciphertext n_s.
- D uses Bob's state to deliver the n_sth ciphertext.

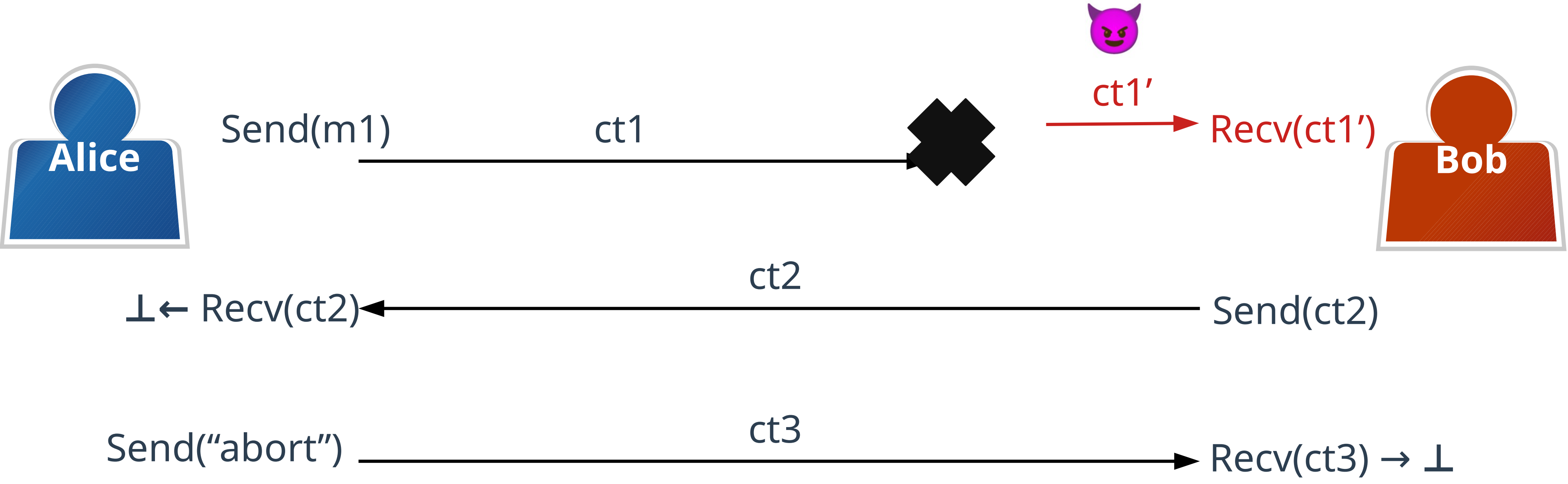
  Then, D iterates over all ciphertexts and tries to deliver the first $n_s - 1$ messages with Bob's state.
- Assuming perfect r–RID security, only the correct messages are successfully received by Bob!

Proof Sketch: Additional Details

- **To make the proof work, the encoder and decoder needs also to take as input and output the same randomness R.**
- **Since r–RID security is not perfect, sometimes Bob can decrypt the wrong messages.**
  - This is resolved by Alice by precomputing the false positives and encoding them as indices.
  - Bob uses these indices to recover the correct messages.

Overcoming the Lower Bound with s-RID

- **s-RID provides 'delayed' r-RID guarantees.**



s–RID Hashing Optimisation

- Recall for s–RID security, Alice attaches her received messages R = ((num_1, ad_1, ct_1), …, (num_n, ad_n, ct_n)).
- Alice can instead send R' = (H(R), num_1, …, num_n).
- Bob, who knows what he sent to Alice, can then recompute H(R) on message reception using the ordinals in R'.
- Can use an *incremental* hash function to compute H(R) more (asymptotically) efficiently.

Other optimisations are possible (ordinal encodings, ...)

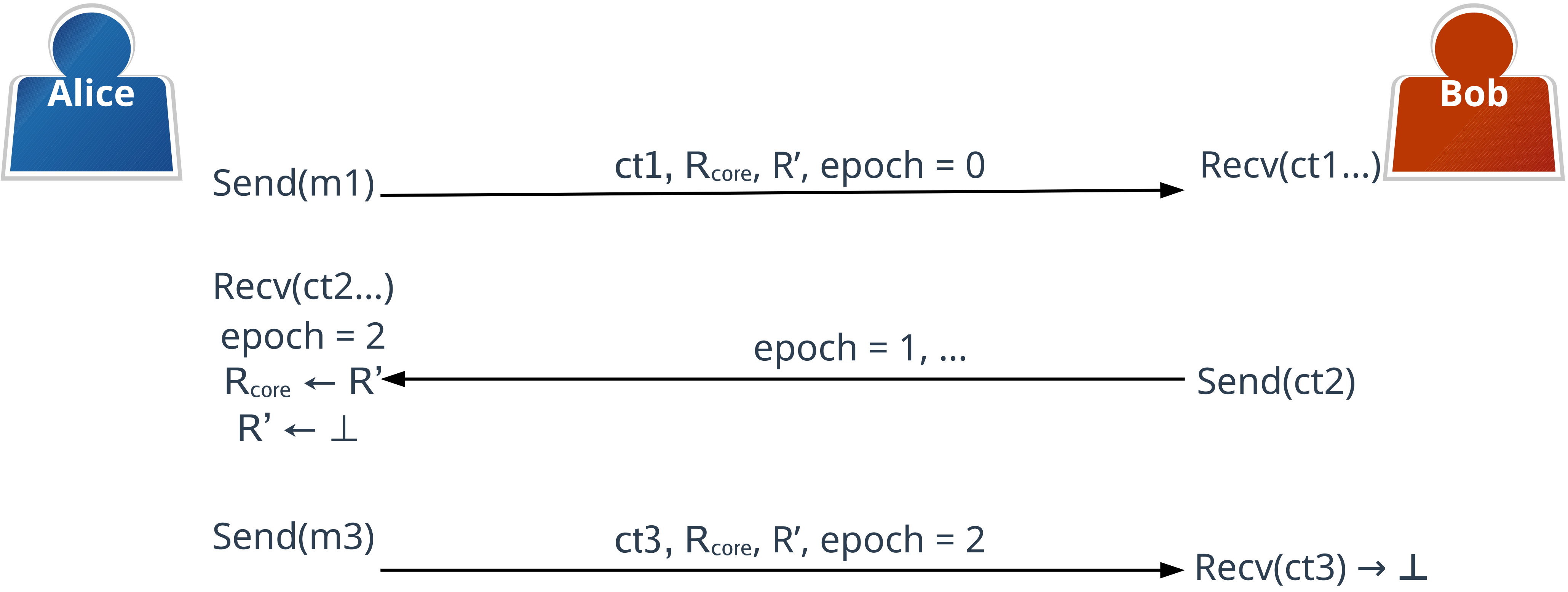
s-RID Epoch-Based Optimisation

- Alice is in epoch e when sending and Bob is in epoch e + 1.
- When Alice receives a message from e + 1, she moves to e + 2.
- The optimisation:

  – Each epoch e message contains R_{core} and R', where R' is initially $\perp$ and grows over time.

  – Upon epoch e + 2, Alice sets $R_{core} \leftarrow$ R' and R' $\leftarrow \perp$.
- Assuming honest delivery, Alice/Bob will definitely receive one message containing R_{core} in each epoch, by definition of epochs.
- Otherwise, a later honest message will contradict a forgery.

s–RID Epoch–Based Optimisation 2



Out–of–Band Messaging Primitive

- In addition to Send and Receive, we define:
  – AuthSend(st) → (st’, num, at);
  – AuthRecv(st, at) → (acc, st’, num).
- Authentication tag = at.
- We assume the channel is *authentic*.
- Examples: QR code scanning, Bluetooth, blockchain, several combined channels…

UNF Out–of–Band Security Notions

- We consider analogous security notions r–UNF and s–UNF to r–RID and s–RID.
- r–UNF: Bob will not accept a tag with ordinal num if it has received a forgery with ordinal num’ ≤ num.
- s–UNF: Bob will not accept a tag sent by Alice after she has received a forgery.
- Given an UNF–secure scheme, tags authenticate the message history:
  → With RID security alone this cannot be done in general.

From RID to UNF Security

- Suppose Ch = (Send, Recv) is RID–secure.
- Then we can construct an UNF–secure Ch’ = (Send, Recv, AuthSend, AuthRecv) as follows:

  – Send and Recv are as in Ch.

  – AuthSend invokes Send with special input; AuthRecv analogously receives.
- [Optimisation: unlike for RID, Alice and Bob only need to send their sets S and R in AuthSend for UNF].

Out-of-Band Performance/Security Trade-offs

- A 3-move protocol that allows parties to mutually authenticate messages (~delayed UNF security).
- First can be sent in-band, and the last is 1 bit, so it is ~non-interactive.

$\mathcal{P}$		$\overline{\mathcal{P}}$
$\mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}})$	$\xrightarrow{\mathsf{R}^{\mathcal{P}}}$	
		$\mathsf{AuthReceive}(\mathsf{st}_{\overline{\mathcal{P}}}, \mathsf{at})$
	$\xleftarrow{\mathsf{R}^{\overline{\mathcal{P}}},\ \mathsf{at\text{-}succ}}$	$\mathsf{AuthSend}(\mathsf{st}_{\overline{\mathcal{P}}})$
$\mathsf{AuthReceive}(\mathsf{st}_{\overline{\mathcal{P}}}, \mathsf{at})$		
$\mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}})$	$\xrightarrow{\mathsf{at\text{-}succ}}$	
		$\mathsf{AuthReceive}(\mathsf{st}_{\overline{\mathcal{P}}}, \mathsf{at})$

Related Work

- Signal safety numbers: QR codes for out-of-band comparison of long-term keys.
- [DH21, DH23]: Authenticates Signal's asymmetric ratchet.
- [DGP22]: Message authentication; different trade-offs to us.
- Apart from [DV19] and [CDV21] that define RECOVER:
  – [JS18] implicitly satisfies RECOVER security;
  – [DHRR22] explicitly considers r-RECOVER.

Conclusion

- Active attacks are worth defending against.
- r–RID is expensive.
- s–RID can be practical!
- Future work:

  – Group RECOVER and practical active attack notions and constructions;
  – Benchmarking and integration into e.g. Signal;
  – ...

Full version: ia.cr/2023/880

X: @dcol97

Bluesky: @dcol

Bibliography

- [JS18]: Jaeger, Stepanovs: Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging. CRYPTO'18
- [ACD19]: Alwen, Coretti, Dodis: The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. EUROCRYPT'19
- [DV19]: Durak, Vaudenay: Bidirectional Asynchronous Ratcheted Key Agreement with Linear Complexity. IWSEC'19
- [CDV21]: Caforio, Durak, Vaudenay: Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness. PKC'21
- [DH21]: Dowling, Hale: Secure Messaging Authentication against Active Man-in-the-Middle Attacks. EuroS&P'21
- [DGP22]: Dowling, Günther, Poirrier: Continuous Authentication in Secure Messaging. ESORICS'22
- [PP22]: Pijnenburg, Poettering: On Secure Ratcheting with Immediate Decryption. ASIACRYPT'22
- [BRT23]: Bienstock, Rösler, Tang: ASMesh: Secure Messaging in Mesh Networks Using Stronger, Anonymous Double Ratchet. CCS'23 (to appear)
- [DH23]: Dowling, Hale: Authenticated Continuous Key Agreement: Active MitM Detection and Prevention. Preprint
- [CZ24]: Cremers, Zhao. Stronger Secure Messaging with Immediate Decryption and Constant-Size Overhead. S&P'24 (to appear)

Encoder/Decoder Algorithms

$\mathsf{Encode}(m_1, \ldots, m_{n_s}, R)$

parse $R_{-1}, R_0, \ldots, R_{n_s} \leftarrow R$; $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda; R_{-1})$; $\mathsf{st}_\mathsf{A}^0, \mathsf{st}_\mathsf{B}^0, z \leftarrow \mathsf{Init}(\mathsf{pp}; R_0)$
for $i \in \{1, \ldots, n_s\}$ **do** // send the n_s messages
  $\mathsf{st}_\mathsf{A}^i, \mathsf{num}, \mathsf{ct}_i \leftarrow \mathsf{Send}(\mathsf{st}_\mathsf{A}^{i-1}, m_i; R_i)$
$\mathsf{acc}, \mathsf{st}_\mathsf{B}^1, \mathsf{num}, m'_{n_s} \leftarrow \mathsf{Receive}(\mathsf{st}_\mathsf{B}^0, \mathsf{ct}_{n_s})$ // Receive ct_{n_s}: $m'_{n_s} = m_{n_s}$ by perfect corr.
// Collecting false positives + correct messages:
for $i \in \{1, \ldots, n_s - 1\}$ **do**
  $S_i \leftarrow \emptyset$
  for $m \in \{0,1\}^n$ **do**
    $\_, \_, \mathsf{ct}' \leftarrow \mathsf{Send}(\mathsf{st}_\mathsf{A}^{i-1}, m; R_i)$
    $\mathsf{acc}, \_, \_, m' \leftarrow \mathsf{Receive}(\mathsf{st}_\mathsf{B}^1, \mathsf{ct}')$
    if acc **then**
      $\boxed{\textbf{if } m \neq m_i \textbf{ then return } (0, m_1, \ldots, m_{n_s}, R)}$
      $S_i \leftarrow S_i \cup \{m\}$
  $L_i \leftarrow \mathsf{sort}(S_i)$
  $e_i \leftarrow$ index of m_i in L_i (in binary with $\lceil \log(|L_i|) \rceil$ bits)
encode ct_{n_s} with k bits
$\boxed{\textbf{return } (1, \mathsf{ct}_{n_s}, R)}$
return $(\mathsf{ct}_{n_s}, R, e_0 \| \ldots \| e_{n_s-1})$

$\mathsf{Decode}(\mathsf{ct}_{n_s}, R, E)$

parse $R_{-1}, R_0, \ldots, R_{n_s} \leftarrow R$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda; R_{-1})$
$\mathsf{st}_\mathsf{A}^0, \mathsf{st}_\mathsf{B}^0, z \leftarrow \mathsf{Init}(\mathsf{pp}; R_0)$
$\mathsf{acc}, \mathsf{st}_\mathsf{B}^1, \mathsf{num}, m_{n_s} \leftarrow \mathsf{Receive}(\mathsf{st}_\mathsf{B}^0, \mathsf{ct}_{n_s})$
// Collecting false positives:
for $i \in \{1, \ldots, n_s - 1\}$ **do**
  $S_i \leftarrow \emptyset$
  for $m \in \{0,1\}^n$ **do**
    $\_, \_, \mathsf{ct}' \leftarrow \mathsf{Send}(\mathsf{st}_\mathsf{A}^{i-1}, m; R_i)$
    $\mathsf{acc}, \_, \_, m' \leftarrow \mathsf{Receive}(\mathsf{st}_\mathsf{B}^1, \mathsf{ct}')$
    if acc **then** $S_i \leftarrow S_i \cup \{m\}$
  $L_i \leftarrow \mathsf{sort}(S_i)$
  $e_i \leftarrow$ read next $\lceil \log(|L_i|) \rceil$ bits of E
  $m_i \leftarrow L_i[e_i]$
  $\mathsf{st}_\mathsf{A}^i, \_, \_ \leftarrow \mathsf{Send}(\mathsf{st}_\mathsf{A}^{i-1}, m_i; R_i)$
return $(m_1, \ldots, m_{n_s}, R)$