Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness

Andrea Caforio† and F. Betül Durak⋆ and Serge Vaudenay†

PKC 2021, May 11

†LASEC, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
⋆Robert Bosch LLC - Research and Technology Center, Pittsburgh PA, USA

State of the Art i

- Secure messaging as a proper cryptographic sub-discipline has elevated itself into a frenzy throughout the past years.

State of the Art i

- Secure messaging as a proper cryptographic sub-discipline has elevated itself into a frenzy throughout the past years.
- Several ratcheting protocols tackling different areas of the security spectrum have been proposed.

State of the Art ii

The fundamental goal, however, is the same for all protocols.

How to secure an **asynchronous** channel between two participants that **arbitrarily** switch their roles (sender/receiver) in the presence of state exposures.

LASEC BOSCH

State of the Art iii

- **Forward Security:** Prevent the decryption of past messages by deleting old states through one-way functions.

LASEC BOSCH

State of the Art iii

- **Forward Security:** Prevent the decryption of past messages by deleting old states through one-way functions.
- **Post-Compromise Security:** Prevent the decryption of future communication by introducing some form of randomness into the states (ratcheting).

State of the Art iv

OTR [BGB04]: The earliest ratcheting protocol. Superseded by the omnipresent **Signal** whose security was formally assessed in 2017 [CCD+17].

- Synchronized ratcheting protocol by Cohn-Gordon et al. [CCG16].
- Unidirectional, no forward-security protocol by Bellare et al. [BSJ+17].

LASEC BOSCH

State of the Art v

PR [PR18]: Poettering and Rösler proposed a protocol in the random oracle model with *optimal* security that relies on HIBE but does not consider a potential leakage of random coins.

- **JS** [JS18]: A similar construction by Jaeger and Stepanovs also in the random oracle model utilizing HIBE but incorporates coin leakages before their usage.

DV [DV19]: Durak and Vaudenay put forward a highly efficient protocol with slightly lower security that is based on a public-key cryptosystem, a digital signature scheme, a one-time symmetric encryption construction and a collision-resistant hash function.

- Introduces **r-RECOVER** security.

LASEC BOSCH

State of the Art vi

DV [DV19]: Durak and Vaudenay put forward a highly efficient protocol with slightly lower security that is based on a public-key cryptosystem, a digital signature scheme, a one-time symmetric encryption construction and a collision-resistant hash function.

- Introduces **r-RECOVER** security.
- Post-compromise security implies public-key cryptography.

LASEC BOSCH

JMM [JMM19]: Jost, Maurer and Mularczyk proposed another protocol in the random oracle model with a security level somewhere between PR and DV with coin leakage resilience after their usage.

- Also based on *ordinary* primitives but not as efficient as DV.

LASEC BOSCH

State of the Art viii

ACD [ACD19]: Alwen, Coretti and Dodis proposed a reinterpretation of **Signal** with immediate decryption and security against adversarially chosen random coins. However, when the direction of communication does not change, the construction only relies on symmetric cryptography.

- **ACD-PK** is a tweak that does offer post-compromise security.

EtH [YV20]: The most efficient protocol (Encrypt-then-Hash) to date was recently proposed by Yan and Vaudenay but does not offer any post-compromise security.

LASEC BOSCH

Crux

In all proposed constructions the users are oblivious to the actual protocols. Active attacks may occur undetected as communication progresses. Additionally, there is no way to modulate the security level. In some scenarios, better performance is warranted at the cost of reduced security guarantees. The known protocols are either strongly secure but impractical or the other way around.

LASEC BOSCH

Contributions i

- Formal definition of the *security awareness* notion, in which users are able to detect active attacks by noticing a communication breakdown. Consequently, every acknowledged message is deemed confidential.

LASEC BOSCH

Contributions i

- Formal definition of the *security awareness* notion, in which users are able to detect active attacks by noticing a communication breakdown. Consequently, every acknowledged message is deemed confidential.
- Users can deduce from incoming messages which of the outgoing ones were actually delivered (*acknowledgment extractor*).

LASEC BOSCH

Contributions i

- Formal definition of the *security awareness* notion, in which users are able to detect active attacks by noticing a communication breakdown. Consequently, every acknowledged message is deemed confidential.
- Users can deduce from incoming messages which of the outgoing ones were actually delivered (*acknowledgment extractor*).
- Given a transcript of sent and received messages alongside potential state exposures we want to pinpoint which messages remain private (*cleanness extractor*).

LASEC BOSCH

Contributions ii

- Generic toolbox that allows the composition of *any* two protocols with different security levels. When a strongly secure protocol is paired with a weaker but more efficient protocol, we obtain the notion of *ratchet on-demand*.

Contributions ii

- Generic toolbox that allows the composition of *any* two protocols with different security levels. When a strongly secure protocol is paired with a weaker but more efficient protocol, we obtain the notion of *ratchet on-demand*.
- Hybrid system of two identical protocols that allows to reinstantiate broken communication.

LASEC BOSCH

Contributions iii

- Comprehensive implementation benchmark of all discussed schemes, i.e., PR, JS, DV, JMM, ACD, ACD-PK and EtH.

LASEC BOSCH

Definition (ARCAD)

An asynchronous ratcheted communication with associated data consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$

LASEC BOSCH

Preliminaries i

Definition (ARCAD)

An asynchronous ratcheted communication with associated data consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
- $\mathsf{Gen}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$

LASEC BOSCH

Preliminaries i

Definition (ARCAD)

An asynchronous ratcheted communication with associated data consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
- $\mathsf{Gen}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$
- $\mathsf{Init}(1^\lambda, \mathsf{pp}, \mathsf{sk}_P, \mathsf{pk}_{\overline{P}}) \rightarrow \mathsf{st}_P$

LASEC BOSCH

Preliminaries i

Definition (ARCAD)

An asynchronous ratcheted communication with associated data consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
- $\mathsf{Gen}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$
- $\mathsf{Init}(1^\lambda, \mathsf{pp}, \mathsf{sk}_P, \mathsf{pk}_{\overline{P}}) \rightarrow \mathsf{st}_P$
- $\mathsf{Send}(\mathsf{st}_P, \mathsf{ad}, \mathsf{pt}) \xrightarrow{\$} (\mathsf{st}'_P, \mathsf{ct})$

LASEC BOSCH

Preliminaries i

Definition (ARCAD)

An asynchronous ratcheted communication with associated data consists of the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
- $\mathsf{Gen}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{sk}, \mathsf{pk})$
- $\mathsf{Init}(1^\lambda, \mathsf{pp}, \mathsf{sk}_P, \mathsf{pk}_{\overline{P}}) \rightarrow \mathsf{st}_P$
- $\mathsf{Send}(\mathsf{st}_P, \mathsf{ad}, \mathsf{pt}) \xrightarrow{\$} (\mathsf{st}'_P, \mathsf{ct})$
- $\mathsf{Receive}(\mathsf{st}_P, \mathsf{ad}, \mathsf{ct}) \rightarrow (\mathsf{acc}, \mathsf{st}'_P, \mathsf{pt})$

LASEC BOSCH

Preliminaries ii

```
Oracle RATCH(P, "rec", ad, ct)
  ct_P ← ct
  ad_P ← ad
  (acc, st'_P, pt_P) ← Receive(st_P, ad_P, ct_P)
  if acc then
      st_P ← st'_P
      append (ad_P, pt_P) to received^P_pt
      append (ad_P, ct_P) to received^P_ct
  end if
  return acc

Oracle RATCH(P, "send", ad, pt)
  pt_P ← pt
  ad_P ← ad
  (st'_P, ct_P) ← Send(st_P, ad_P, pt_P)
  st_P ← st'_P
  append (ad_P, pt_P) to sent^P_pt
  append (ad_P, ct_P) to sent^P_ct
  return ct_P
```

```
Game Correctness(sched)
  set all sent^*_* and received^*_* to ∅
  Setup(1^λ) →$ pp
  Initall(1^λ, pp) →$ (st_A, st_B, z)
  initialize two FIFO lists incoming_A, incoming_B ← ∅
  i ← 0
  loop
      i ← i + 1
      if sched_i of form (P, "rec") then
          if incoming_P is empty then return 0
          pull (ad, ct) from incoming_P
          acc ← RATCH(P, "rec", ad, ct)
          if acc = false then return 1
      else
          parse sched_i = (P, "send", ad, pt)
          ct ← RATCH(P, "send", ad, pt)
          push (ad, ct) to incoming_P̄
      end if
      if received^A_pt not prefix of sent^B_pt then return 1
      if received^B_pt not prefix of sent^A_pt then return 1
  end loop
```

Preliminaries iii

Definition (Matching Status [DV19])

P is in a matching status at time t for P if

1. at any moment of the game before time t for P, $\mathbf{received}_{\mathbf{ct}}^{P}$ is a prefix of $\mathbf{sent}_{\mathbf{ct}}^{\overline{P}}$;

LASEC BOSCH

Preliminaries iii

Definition (Matching Status [DV19])

P is in a matching status at time t for P if

1. at any moment of the game before time t for P, $\mathbf{received}_{\mathbf{ct}}^{P}$ is a prefix of $\mathbf{sent}_{\mathbf{ct}}^{\overline{P}}$;
2. at any moment of the game before time $\bar{t}$ for $\overline{P}$, $\mathbf{received}_{\mathbf{ct}}^{\overline{P}}$ is a prefix of $\mathbf{sent}_{\mathbf{ct}}^{P}$.

LASEC BOSCH

Preliminaries iv

In order to simply the analysis of various security we would like to tuck away trivial attacks behind a cleanness predicate $\mathcal{C}_{\textsf{clean}}$ such that the games for other security notions become easier to parse.

- FORGE

Preliminaries iv

In order to simply the analysis of various security we would like to tuck away trivial attacks behind a cleanness predicate $\mathcal{C}_{\mathsf{clean}}$ such that the games for other security notions become easier to parse.

- FORGE
- r-RECOVER

In order to simply the analysis of various security we would like to tuck away trivial attacks behind a cleanness predicate $\mathcal{C}_{\mathsf{clean}}$ such that the games for other security notions become easier to parse.

- FORGE
- r-RECOVER
- PREDICT

LASEC BOSCH

Preliminaries iv

In order to simply the analysis of various security we would like to tuck away trivial attacks behind a cleanness predicate $\mathcal{C}_{\text{clean}}$ such that the games for other security notions become easier to parse.

- FORGE
- r-RECOVER
- PREDICT
- INC-CCA

LASEC BOSCH

Game $\mathsf{FORGE}^{\mathcal{A}}_{\mathsf{C}_{\mathsf{clean}}}(1^\lambda)$

$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
$\mathsf{Initall}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{st}_A, \mathsf{st}_B, z)$
$(\mathsf{P}, \mathsf{ad}, \mathsf{ct}) \leftarrow \mathcal{A}^{\mathsf{RATCH}, \mathsf{EXP}_{\mathsf{st}}, \mathsf{EXP}_{\mathsf{pt}}}(z)$
$\mathsf{RATCH}(\mathsf{P}, \text{“rec”}, \mathsf{ad}, \mathsf{ct}) \to \mathsf{acc}$
if $\mathsf{acc} = \mathsf{false}$ **then return** 0
if $\neg \mathsf{C}_{\mathsf{clean}}$ **then return** 0
if $(\mathsf{ad}, \mathsf{ct})$ is not a forgery (Def. 32) for P **then return** 0
return 1

Game $\mathsf{r\text{-}RECOVER}^{\mathcal{A}}(1^\lambda)$

$\mathsf{win} \leftarrow 0$
$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
$\mathsf{Initall}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{st}_A, \mathsf{st}_B, z)$
set all sent^*_* and $\mathsf{received}^*_*$ variables to $\emptyset$
$\mathsf{P} \leftarrow \mathcal{A}^{\mathsf{RATCH}, \mathsf{EXP}_{\mathsf{st}}, \mathsf{EXP}_{\mathsf{pt}}}(z)$
if we can parse $\mathsf{received}^{\mathsf{P}}_{\mathsf{ct}} = (\mathsf{seq}_1, (\mathsf{ad}, \mathsf{ct}), \mathsf{seq}_2)$ and $\mathsf{sent}^{\overline{\mathsf{P}}}_{\mathsf{ct}} = (\mathsf{seq}_3, (\mathsf{ad}, \mathsf{ct}), \mathsf{seq}_4)$ with $\mathsf{seq}_1 \neq \mathsf{seq}_3$ (where $(\mathsf{ad}, \mathsf{ct})$ is a single message and all seq_i are finite sequences of single messages) **then** $\mathsf{win} \leftarrow 1$
return win

Game $\mathsf{PREDICT}^{\mathcal{A}}(1^\lambda)$

$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$
$\mathsf{Initall}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{st}_A, \mathsf{st}_B, z)$
$(\mathsf{P}, \mathsf{ad}, \mathsf{pt}) \leftarrow \mathcal{A}^{\mathsf{RATCH}, \mathsf{EXP}_{\mathsf{st}}, \mathsf{EXP}_{\mathsf{pt}}}(z)$
$\mathsf{RATCH}(\mathsf{P}, \text{“send”}, \mathsf{ad}, \mathsf{pt}) \to \mathsf{ct}$
if $(\mathsf{ad}, \mathsf{ct}) \in \mathsf{received}^{\overline{\mathsf{P}}}_{\mathsf{ct}}$ **then return** 1
return 0

Preliminaries vi

$$\mathsf{Adv}(\mathcal{A}) = \left|\Pr\left[\text{IND-CCA}^{\mathcal{A}}_{0,\mathcal{C}_{\text{clean}}}(1^\lambda) \rightarrow 1\right] - \Pr\left[\text{IND-CCA}^{\mathcal{A}}_{1,\mathcal{C}_{\text{clean}}}(1^\lambda) \rightarrow 1\right]\right|$$

Game $\text{IND-CCA}^{\mathcal{A}}_{b,C_{\text{clean}}}(1^\lambda)$	Oracle $\text{CHALLENGE}(P, ad, pt)$
1: $\text{Setup}(1^\lambda) \xrightarrow{\$} pp$	1: **if** $t_{test} \neq \perp$ **then return** $\perp$
2: $\text{Initall}(1^\lambda, pp) \xrightarrow{\$} (st_A, st_B, z)$	2: **if** $b = 0$ **then**
3: set all sent^*_* and received^*_* variables to $\emptyset$	3: replace pt by a random string of same length
4: set t_{test} to $\perp$	4: **end if**
5: $b' \leftarrow \mathcal{A}^{\text{RATCH},\text{EXP}_{st},\text{EXP}_{pt},\text{CHALLENGE}}(z)$	5: $ct \leftarrow \text{RATCH}(P, \text{“send”}, ad, pt)$
6: **if** $\neg C_{\text{clean}}$ **then return** $\perp$	6: $(t, P, ad, pt, ct)_{test} \leftarrow (time, P, ad, pt, ct)$
7: **return** b’	7: **return** ct
Oracle $\text{EXP}_{st}(P)$	Oracle $\text{EXP}_{pt}(P)$
1: **return** st_P	1: **return** pt_P

LASEC BOSCH

Security Awareness i

- **r-RECOVER** security averts that a users P continues to accepts genuine **ct** from $\overline{P}$ after having received a forgery. However, $\overline{P}$ is still capable of receiving messages from P.

LASEC BOSCH

Security Awareness i

- **r-RECOVER** security averts that a users P continues to accepts genuine **ct** from $\overline{P}$ after having received a forgery. However, $\overline{P}$ is still capable of receiving messages from P.
- A communication breakdown should reveal any forgery and receiving genuine messages should indicate that no forgeries took place.

LASEC BOSCH

Security Awareness ii

Game $\mathsf{s\text{-}RECOVER}^{\mathcal{A}}(1^\lambda)$

$\mathsf{win} \leftarrow 0$

$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}$

$\mathsf{Initall}(1^\lambda, \mathsf{pp}) \xrightarrow{\$} (\mathsf{st}_A, \mathsf{st}_B, z)$

set all sent_*^* and $\mathsf{received}_*^*$ variables to $\emptyset$

$\mathsf{P} \leftarrow \mathcal{A}^{\mathsf{RATCH}, \mathsf{EXP}_{\mathsf{st}}, \mathsf{EXP}_{\mathsf{pt}}}(z)$

if $\mathsf{received}_{\mathsf{ct}}^{\mathsf{P}}$ is a prefix of $\mathsf{sent}_{\mathsf{ct}}^{\overline{\mathsf{P}}}$ **then**

  set $\bar{t}$ to the time when $\overline{\mathsf{P}}$ sent the last message in $\mathsf{received}_{\mathsf{ct}}^{\mathsf{P}}$

  if $\mathsf{received}_{\mathsf{ct}}^{\overline{\mathsf{P}}}(\bar{t})$ is not a prefix of $\mathsf{sent}_{\mathsf{ct}}^{\mathsf{P}}$ **then** $\mathsf{win} \leftarrow 1$

end if

return win

Security Awareness iii

Lemma

*If an **ARCAD** is **r-RECOVER**, **s-RECOVER** and **PREDICT** secure, whenever P receives a genuine message from $\overline{P}$ (i.e., a (ad,ct) pair sent by $\overline{P}$ is accepted by P), P is in a matching status, except with negligible probability.*

LASEC BOSCH

Security Awareness iv

- In addition to RECOVER security, we want to give a user the power to verify whether a message has been accepted by his counterpart and check whether some cleanness predicate $\mathcal{C}_{\textsf{clean}}$ has been violated.

Security Awareness iv

- In addition to **RECOVER** security, we want to give a user the power to verify whether a message has been accepted by his counterpart and check whether some cleanness predicate $\mathcal{C}_{\textsf{clean}}$ has been violated.
- Let $\mathsf{T}_P(t)$ be the chronological partial transcript up to time t of send, receive, exposure and challenge calls involving user P alongside the $(\mathsf{ad}, \mathsf{ct})$ pairs corresponding to the send, receive invocations. Let $\mathsf{T}_P^{\mathsf{RATCH}}(t)$ be the transcript that only contains send, receive and challenge calls.

LASEC BOSCH

Definition (Acknowledgment Extractor)

Given $\mathsf{T}_P(t)$ and a message $(\mathsf{ad}, \mathsf{ct})$ successfully received by P at time t that was sent by $\overline{P}$ at time $\bar{t}$, let $(\mathsf{ad}', \mathsf{ct}')$ be the last message received by $\overline{P}$ before time $\bar{t}$.

An acknowledgment extractor is an efficient function f such that $f\left(\mathsf{T}_P^{\mathsf{RATCH}}(t)\right) = (\mathsf{ad}', \mathsf{ct}')$ for any time t when P is a matching status.

LASEC BOSCH

Definition (Cleanness Extractor)

Let $\mathsf{T}_P(t)$ and $\mathsf{T}_{\overline{P}}(\bar{t})$ be two partial transcripts, there is a cleanness extractor for $\mathcal{C}_{\mathsf{clean}}$ if there is an efficient function g such that $g(\mathsf{T}_P(t), \mathsf{T}_{\overline{P}}(\bar{t}))$ has the following properties: if there is one challenge in $\mathsf{T}_P(t)$, and either P received $(\mathsf{ad}_{\mathsf{test}}, \mathsf{ct}_{\mathsf{test}})$ or there is a round trip $P \to \overline{P} \to P$ starting with P sending $(\mathsf{ad}_{\mathsf{test}}, \mathsf{ct}_{\mathsf{test}})$, then $g(\mathsf{T}_P(t), \mathsf{T}_{\overline{P}}(\bar{t})) = \mathcal{C}_{\mathsf{clean}}$.

LASEC BOSCH

Security Awareness vii

We give a generic construction that elevates any secure ARCAD_0 into a *security-aware* $\mathsf{ARCAD}_1 = \mathsf{chain}(\mathsf{ARCAD}_0)$.

Security Awareness viii

- Intuitively, a blockchain-like structure that contains a hash chain of sent ciphertexts that is sent alongside each message suffices to associate each message to the digest of the chain.

LASEC BOSCH

Security Awareness viii

- Intuitively, a blockchain-like structure that contains a hash chain of sent ciphertexts that is sent alongside each message suffices to associate each message to the digest of the chain.
- Hsent: Hash of all sent ciphertexts. Computed by the sender, sent along each message and updated with a hashing key hk.

LASEC BOSCH

Security Awareness viii

- Intuitively, a blockchain-like structure that contains a hash chain of sent ciphertexts that is sent alongside each message suffices to associate each message to the digest of the chain.
- **Hsent**: Hash of all sent ciphertexts. Computed by the sender, sent along each message and updated with a hashing key **hk**.
- **Hreceived**: Hash of all received ciphertexts. Also updated with **hk** upon every reception.

LASEC BOSCH

Security Awareness viii

- Intuitively, a blockchain-like structure that contains a hash chain of sent ciphertexts that is sent alongside each message suffices to associate each message to the digest of the chain.
- **Hsent**: Hash of all sent ciphertexts. Computed by the sender, sent along each message and updated with a hashing key **hk**.
- **Hreceived**: Hash of all received ciphertexts. Also updated with **hk** upon every reception.
- **Hsent** and **Hreceived** are enough to ensure **r-RECOVER** security.

LASEC BOSCH

Security Awareness ix

- **Areceived**: Counter of received messages that need to be reported in the next send call where the last **Hreceived** is attached to **ct** to acknowledge received messages and **Areceived** $= 0$.

Security Awareness ix

- **Areceived**: Counter of received messages that need to be reported in the next send call where the last **Hreceived** is attached to **ct** to acknowledge received messages and **Areceived** $= 0$.
- **Asent**: List of the hashes of sent ciphertexts that are waiting for an acknowledgment.

Security Awareness ix

- **Areceived**: Counter of received messages that need to be reported in the next send call where the last **Hreceived** is attached to **ct** to acknowledge received messages and **Areceived** $= 0$.
- **Asent**: List of the hashes of sent ciphertexts that are waiting for an acknowledgment.
- Any impersonation of a participant leads to an immediate cut of communication as the intrusion is detected.

LASEC BOSCH

Security Awareness x

$\mathsf{ARCAD.Setup}(1^\lambda)$ 1: $\mathsf{ARCAD}_0.\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{pp}_0$ 2: $\mathsf{H.Gen}(1^\lambda) \xrightarrow{\$} \mathsf{hk}$ 3: $\mathsf{pp} \leftarrow (\mathsf{hk}, \mathsf{pp}_0)$ 4: **return** pp $\mathsf{ARCAD.Gen} = \mathsf{ARCAD}_0.\mathsf{Gen}$	$\mathsf{ARCAD.Init}(1^\lambda, \mathsf{pp}, \mathsf{sk}_P, \mathsf{pk}_{\overline{P}}, P)$ 1: parse $\mathsf{pp} = (\mathsf{hk}, \mathsf{pp}_0)$ 2: $\mathsf{ARCAD}_0.\mathsf{Init}(1^\lambda, \mathsf{pp}_0, \mathsf{sk}_P, \mathsf{pk}_{\overline{P}}, P) \xrightarrow{\$} \mathsf{st}'_P$ 3: $\mathsf{Hsent}, \mathsf{Hreceived} \leftarrow \perp$ 4: $\mathsf{Asent} \leftarrow [\,], \mathsf{Areceived} \leftarrow 0$ 5: $\mathsf{st}_P \leftarrow (\mathsf{st}'_P, \mathsf{hk}, \mathsf{Hsent}, \mathsf{Hreceived}, \mathsf{Asent}, \mathsf{Areceived})$ 6: **return** st_P

LASEC BOSCH

Security Awareness xi

$\mathsf{ARCAD.Send}(\mathsf{st}_P, \mathsf{ad}, \mathsf{pt})$

- parse st_P as $(\mathsf{st}'_P, \mathsf{hk}, \mathsf{Hsent}, \mathsf{Hreceived}, \mathsf{Asent}, \mathsf{Areceived})$
- **if** $\mathsf{Areceived} = 0$ **then** $\mathsf{ack} \leftarrow \perp$ **else** $\mathsf{ack} \leftarrow \mathsf{Hreceived}$
- $\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{Hsent}, \mathsf{ack})$
- $\mathsf{ARCAD}_0.\mathsf{Send}(\mathsf{st}'_P, \mathsf{ad}', \mathsf{pt}) \xrightarrow{\$} (\mathsf{st}'_P, \mathsf{ct}')$
- $\mathsf{ct} \leftarrow (\mathsf{ct}', \mathsf{Hsent}, \mathsf{ack})$
- $\mathsf{Areceived} \leftarrow 0$
- $\mathsf{Hsent} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, \mathsf{Hsent}, \mathsf{ad}, \mathsf{ct})$
- $\mathsf{Asent} \leftarrow (\mathsf{Asent}, \mathsf{Hsent})$
- $\mathsf{st}_P \leftarrow (\mathsf{st}'_P, \mathsf{hk}, \mathsf{Hsent}, \mathsf{Hreceived}, \mathsf{Asent}, \mathsf{Areceived})$
- **return** $(\mathsf{st}_P, \mathsf{ct})$

$\mathsf{ARCAD.Receive}(\mathsf{st}_P, \mathsf{ad}, \mathsf{ct})$

- parse st_P as $(\mathsf{st}'_P, \mathsf{hk}, \mathsf{Hsent}, \mathsf{Hreceived}, \mathsf{Asent}, \mathsf{Areceived})$
- parse ct as $(\mathsf{ct}', \mathsf{h}, \mathsf{ack})$
- **if** $\mathsf{h} \neq \mathsf{Hreceived}$ or $\mathsf{ack} \notin \{\perp\} \cup \mathsf{Asent}$ **then**
  - **return** $(\mathsf{false}, \mathsf{st}_P, \perp)$
- **end if**
- $\mathsf{ad}' \leftarrow (\mathsf{ad}, \mathsf{h}, \mathsf{ack})$
- $\mathsf{ARCAD}_0.\mathsf{Receive}(\mathsf{st}'_P, \mathsf{ad}', \mathsf{ct}') \rightarrow (\mathsf{acc}, \mathsf{st}'_P, \mathsf{pt}')$
- **if** acc **then**
  - $\mathsf{Hreceived} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, \mathsf{Hreceived}, \mathsf{ad}, \mathsf{ct})$
  - $\mathsf{Areceived} \leftarrow \mathsf{Areceived} + 1$
  - **if** $\mathsf{ack} \neq \perp$ **then** remove in Asent all elements of Asent until ack (included)
  - $\mathsf{st}_P \leftarrow (\mathsf{st}'_P, \mathsf{hk}, \mathsf{Hsent}, \mathsf{Hreceived}, \mathsf{Asent}, \mathsf{Areceived})$
- **end if**
- **return** $(\mathsf{acc}, \mathsf{st}_P, \mathsf{pt}')$

Lemma

If $\mathcal{H}$ is collision-resistant, chain($ARCAD_0$) *is* ***RECOVER****-secure (for both* ***r-RECOVER*** *and* ***s-RECOVER****).*

Lemma

chain($ARCAD_0$) *has an acknowledgment extractor.*

Security Awareness xiii

Lemma

chain($ARCAD_0$) *has a cleanness extractor for the following predicates:*

$$\mathcal{C}_{\text{leak}},\ \mathcal{C}^{\text{A,B}}_{\text{trivialforge}},\ \mathcal{C}^{\text{P}_{\text{test}}}_{\text{trivialforge}},\ \mathcal{C}^{\text{A,B}}_{\text{forge}}, \mathcal{C}^{\text{P}_{\text{test}}}_{\text{forge}},\ \mathcal{C}_{\text{leak}},\ \mathcal{C}_{\text{noexp}}.$$

Lemma

If $ARCAD_0$ is PREDICT-secure, then chain($ARCAD_0$) *is PREDICT-secure.*

LASEC BOSCH

On-Demand Ratcheting i

- Combine a strongly secure protocol (using public-key cryptography) with a weaker (only symmetric-key cryptography) but more efficient construction.

On-Demand Ratcheting i

- Combine a strongly secure protocol (using public-key cryptography) with a weaker (only symmetric-key cryptography) but more efficient construction.
- Use the weak protocol for frequent exchanges (no post-compromise security) and periodically ratchet with the strong one.

LASEC BOSCH

On-Demand Ratcheting i

- Combine a strongly secure protocol (using public-key cryptography) with a weaker (only symmetric-key cryptography) but more efficient construction.
- Use the weak protocol for frequent exchanges (no post-compromise security) and periodically ratchet with the strong one.
- Ratcheting could be administered at the application level or even by the user.

LASEC BOSCH

On-Demand Ratcheting ii

- Denote by $\mathsf{ARCAD}_{\mathsf{main}}$ the strong protocol and by $\mathsf{ARCAD}_{\mathsf{sub}}$ the weak protocol.

LASEC BOSCH

On-Demand Ratcheting ii

- Denote by $\mathsf{ARCAD}_{\mathsf{main}}$ the strong protocol and by $\mathsf{ARCAD}_{\mathsf{sub}}$ the weak protocol.
- An on-demand ratcheting is scheme is denoted by $\mathsf{hybrid}(\mathsf{ARCAD}_{\mathsf{main}}, \mathsf{ARCAD}_{\mathsf{sub}})$.

LASEC BOSCH

On-Demand Ratcheting ii

- Denote by $\mathsf{ARCAD}_{\mathsf{main}}$ the strong protocol and by $\mathsf{ARCAD}_{\mathsf{sub}}$ the weak protocol.
- An on-demand ratcheting is scheme is denoted by $\mathsf{hybrid}(\mathsf{ARCAD}_{\mathsf{main}}, \mathsf{ARCAD}_{\mathsf{sub}})$.
- Use **flag** that is sent together with **ad** to instigate ratcheting.

LASEC BOSCH

On-Demand Ratcheting iii

- The protocol proceeds in epochs, first defined in PR. Intuitively, an epoch designates a badge of sent/received messages until the direction of communication changes.

LASEC BOSCH

On-Demand Ratcheting iii

- The protocol proceeds in epochs, first defined in PR. Intuitively, an epoch designates a badge of sent/received messages until the direction of communication changes.
- Each user stores two counters $\mathbf{e}^{P}_{\mathsf{send}}$, $\mathbf{e}^{P}_{\mathsf{rec}}$ indicating the epoch of the last sent and received message.

LASEC BOSCH

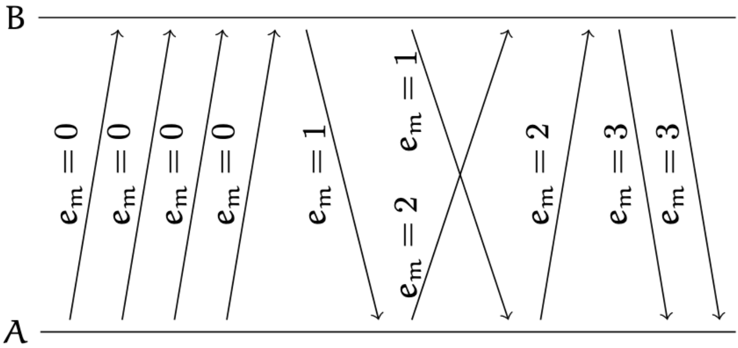
On-Demand Ratcheting iii

- The protocol proceeds in epochs, first defined in **PR**. Intuitively, an epoch designates a badge of sent/received messages until the direction of communication changes.
- Each user stores two counters $\mathbf{e}^{P}_{\mathbf{send}}, \mathbf{e}^{P}_{\mathbf{rec}}$ indicating the epoch of the last sent and received message.
- An epoch counter $\mathbf{e}_{\mathbf{m}}$ is sent along each message, i.e.,

$$\mathbf{e}_{\mathbf{m}} = \begin{cases} \mathbf{e}^{P}_{\mathbf{send}} & \text{if } \mathbf{e}^{P}_{\mathbf{rec}} < \mathbf{e}^{P}_{\mathbf{send}} \\ \mathbf{e}^{P}_{\mathbf{rec}} + 1 & \text{otherwise.} \end{cases}$$

LASEC BOSCH

On-Demand Ratcheting iv



On-Demand Ratcheting v

- In addition to the epoch counters, each user keeps count of the number of messages within one epoch, i.e., $\mathsf{ctr}[e_m]$.

LASEC BOSCH

On-Demand Ratcheting v

- In addition to the epoch counters, each user keeps count of the number of messages within one epoch, i.e., $\mathsf{ctr}[e_m]$.
- Every ARCAD_{main} call creates a new ARCAD_{sub} send/receive state pair. The sender stores the send state in $\mathsf{sub}[e_m, \mathsf{ctr}[e_m]]$ and transmits the receive state in the ciphertext.

LASEC BOSCH

On-Demand Ratcheting vi

$\mathsf{hybridARCAD.Setup}(1^\lambda)$

$\mathsf{pp_{main}} \leftarrow \mathsf{ARCAD_{main}.Setup}(1^\lambda)$

$\mathsf{pp_{sub}} \leftarrow \mathsf{ARCAD_{sub}.Setup}(1^\lambda)$

return $(\mathsf{pp_{main}}, \mathsf{pp_{sub}})$

$\mathsf{hybridARCAD.Gen}(1^\lambda, \mathsf{pp_{main}}, \mathsf{pp_{sub}})$

return $\mathsf{ARCAD_{main}.Gen}(1^\lambda, \mathsf{pp_{main}})$

$\mathsf{hybridARCAD.Init}(1^\lambda, (\mathsf{pp_{main}}, \mathsf{pp_{sub}}), \mathsf{sk_P}, \mathsf{pk_{\overline{P}}}, \mathrm{P})$

$\mathsf{ARCAD_{main}.Init}(1^\lambda, \mathsf{pp_{main}}, \mathsf{sk_P}, \mathsf{pk_{\overline{P}}}, \mathrm{P}) \rightarrow \mathsf{st_{main}}$

initialize array $\mathsf{st_{sub}}[]$ to empty

if $\mathrm{P} = \mathrm{A}$ **then** $(\mathsf{e_{send}}, \mathsf{e_{rec}}) \leftarrow (0, -1)$

else $(\mathsf{e_{send}}, \mathsf{e_{rec}}) \leftarrow (-1, 0)$

end if

initialize array ctr with $\mathsf{ctr}[0] = -1$

$\mathsf{st_P} \leftarrow (\lambda, \mathsf{pp_{sub}}, \mathsf{st_{main}}, \mathsf{st_{sub}}[], \mathsf{e_{send}}, \mathsf{e_{rec}}, \mathsf{ctr}[])$

return $\mathsf{st_P}$

On-Demand Ratcheting vii

$\mathsf{hybridARCAD.Send}(\mathsf{st}_P, \mathsf{ad}, \mathsf{pt})$

parse st_P as $(\lambda, \mathsf{pp}_{\mathsf{sub}}, \mathsf{st}_{\mathsf{main}}, \mathsf{st}_{\mathsf{sub}}[], \mathsf{e}_{\mathsf{send}}, \mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[])$
$e \leftarrow \max(\mathsf{e}_{\mathsf{send}}, \mathsf{e}_{\mathsf{rec}})$; $\mathsf{c} \leftarrow \mathsf{ctr}[e]$ ▷ current epoch
if $\mathsf{ad.flag}$ or $\mathsf{c} = -1$ **then**
  if $\mathsf{e}_{\mathsf{send}} < \mathsf{e}_{\mathsf{rec}}$ **then** $e \leftarrow \mathsf{e}_{\mathsf{rec}} + 1$; $\mathsf{c} \leftarrow 0$
  else $e \leftarrow \mathsf{e}_{\mathsf{send}}$; $\mathsf{c} \leftarrow \mathsf{ctr}[e] + 1$
  end if
  $\mathsf{ARCAD}_{\mathsf{sub}}.\mathsf{Initall}(1^\lambda, \mathsf{pp}_{\mathsf{sub}}) \xrightarrow{\$} (\mathsf{st}_S, \mathsf{st}_R, z)$ ▷ create a new sub-state.
  $\mathsf{st}_{\mathsf{sub}}[e, \mathsf{c}] \leftarrow \mathsf{st}_S$
  $\mathsf{pt}' \leftarrow (\mathsf{st}_R, \mathsf{pt})$; $\mathsf{ad}' \leftarrow (\mathsf{ad}, 1, e, \mathsf{c})$
  $\mathsf{ARCAD}_{\mathsf{main}}.\mathsf{Send}(\mathsf{st}_{\mathsf{main}}, \mathsf{ad}', \mathsf{pt}') \xrightarrow{\$} (\mathsf{st}_{\mathsf{main}}, \mathsf{ct}')$ ▷ send using the main state.
  $\mathsf{ct} \leftarrow (\mathsf{ct}', e, \mathsf{c})$
  $\mathsf{e}_{\mathsf{send}} \leftarrow e$; $\mathsf{ctr}[\mathsf{e}_{\mathsf{send}}] \leftarrow \mathsf{c}$
else
  $\mathsf{ad}' \leftarrow (\mathsf{ad}, 0, e, \mathsf{c})$
  $\mathsf{ARCAD}_{\mathsf{sub}}.\mathsf{Send}(\mathsf{st}_{\mathsf{sub}}[e, \mathsf{c}], \mathsf{ad}', \mathsf{pt}) \xrightarrow{\$} (\mathsf{st}_{\mathsf{sub}}[e, \mathsf{c}], \mathsf{ct}')$ ▷ send using the sub-state.
  $\mathsf{ct} \leftarrow (\mathsf{ct}', e, \mathsf{c})$
end if
clean-up: erase $\mathsf{st}_{\mathsf{sub}}[e, \mathsf{c}]$ for all (e, c) such that $(e, \mathsf{c}) < (\mathsf{e}_{\mathsf{send}}, \mathsf{ctr}[\mathsf{e}_{\mathsf{send}}])$ and $(e, \mathsf{c}) < (\mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[\mathsf{e}_{\mathsf{rec}}])$
clean-up: erase $\mathsf{ctr}[e]$ for all e such that $e < \mathsf{e}_{\mathsf{send}}$ and $e < \mathsf{e}_{\mathsf{rec}}$
$\mathsf{st}_P \leftarrow (\lambda, \mathsf{pp}_{\mathsf{sub}}, \mathsf{st}_{\mathsf{main}}, \mathsf{st}_{\mathsf{sub}}[], \mathsf{e}_{\mathsf{send}}, \mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[])$
return $(\mathsf{st}_P, \mathsf{ct})$

LASEC BOSCH

On-Demand Ratcheting viii

$\mathsf{hybridARCAD.Receive}(\mathsf{st}_P, \mathsf{ad}, \mathsf{ct})$

parse st_P as $(\lambda, \mathsf{pp}_{\mathsf{sub}}, \mathsf{st}_{\mathsf{main}}, \mathsf{st}_{\mathsf{sub}}[], \mathsf{e}_{\mathsf{send}}, \mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[])$
parse ct as $(\mathsf{ct}', \mathsf{e}, \mathsf{c})$
if $(\mathsf{e}, \mathsf{c}) < (\mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[\mathsf{e}_{\mathsf{rec}}])$ **then return** $(\mathsf{false}, \mathsf{st}_P, \bot)$ ▷ (e, c) must increase
if $\mathsf{ad.flag}$ or $(\mathsf{e} = 0$ and $\mathsf{ctr}[0] = -1)$ **then**
    $\mathsf{ad}' \leftarrow (\mathsf{ad}, 1, \mathsf{e}, \mathsf{c})$
    $\mathsf{ARCAD}_{\mathsf{main}}.\mathsf{Receive}(\mathsf{st}_{\mathsf{main}}, \mathsf{ad}', \mathsf{ct}') \rightarrow (\mathsf{acc}, \mathsf{st}_{\mathsf{main}}, \mathsf{pt}')$
    parse pt' as $(\mathsf{st}_R, \mathsf{pt})$
    if acc **then**
        $\mathsf{st}_{\mathsf{sub}}[\mathsf{e}, \mathsf{c}] \leftarrow \mathsf{st}_R$
        $\mathsf{e}_{\mathsf{rec}} \leftarrow \mathsf{e}$; $\mathsf{ctr}[\mathsf{e}] \leftarrow \mathsf{c}$
    end if
else
    $\mathsf{ad}' \leftarrow (\mathsf{ad}, 0, \mathsf{e}, \mathsf{c})$
    if $\mathsf{st}_{\mathsf{sub}}[\mathsf{e}, \mathsf{c}]$ undefined **then return** $(\mathsf{false}, \mathsf{st}_P, \bot)$
    $\mathsf{ARCAD}_{\mathsf{sub}}.\mathsf{Receive}(\mathsf{st}_{\mathsf{sub}}[\mathsf{e}, \mathsf{c}], \mathsf{ad}', \mathsf{ct}') \rightarrow (\mathsf{acc}, \mathsf{st}_{\mathsf{sub}}[\mathsf{e}, \mathsf{c}], \mathsf{pt})$
end if
clean-up: erase $\mathsf{st}_{\mathsf{sub}}[\mathsf{e}, \mathsf{c}]$ for all (e, c) such that $(\mathsf{e}, \mathsf{c}) < (\mathsf{e}_{\mathsf{send}}, \mathsf{ctr}[\mathsf{e}_{\mathsf{send}}])$ and $(\mathsf{e}, \mathsf{c}) < (\mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[\mathsf{e}_{\mathsf{rec}}])$
clean-up: erase $\mathsf{ctr}[\mathsf{e}]$ for all e such that $\mathsf{e} < \mathsf{e}_{\mathsf{send}}$ and $\mathsf{e} < \mathsf{e}_{\mathsf{rec}}$
$\mathsf{st}_P \leftarrow (\lambda, \mathsf{pp}_{\mathsf{sub}}, \mathsf{st}_{\mathsf{main}}, \mathsf{st}_{\mathsf{sub}}[], \mathsf{e}_{\mathsf{send}}, \mathsf{e}_{\mathsf{rec}}, \mathsf{ctr}[])$
return $(\mathsf{acc}, \mathsf{st}_P, \mathsf{pt})$

On-Demand Ratcheting ix

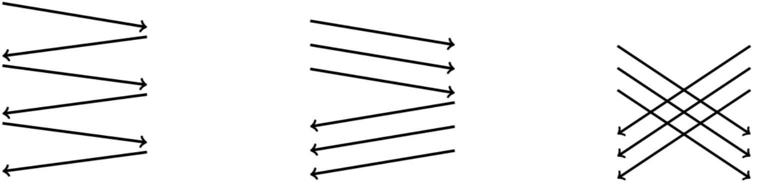
- Security is derived from $\mathsf{ARCAD}_{\mathsf{main}}$ and $\mathsf{ARCAD}_{\mathsf{sub}}$.

On-Demand Ratcheting ix

- Security is derived from $\mathsf{ARCAD}_{\mathsf{main}}$ and $\mathsf{ARCAD}_{\mathsf{sub}}$.
- If $\mathsf{ARCAD}_{\mathsf{main}} = \mathsf{ARCAD}_{\mathsf{sub}}$, then $\mathsf{ARCAD}_{\mathsf{main}}$ can be used to generate a new $\mathsf{ARCAD}_{\mathsf{sub}}$ session, making it possible to restore communication.

Benchmarks i

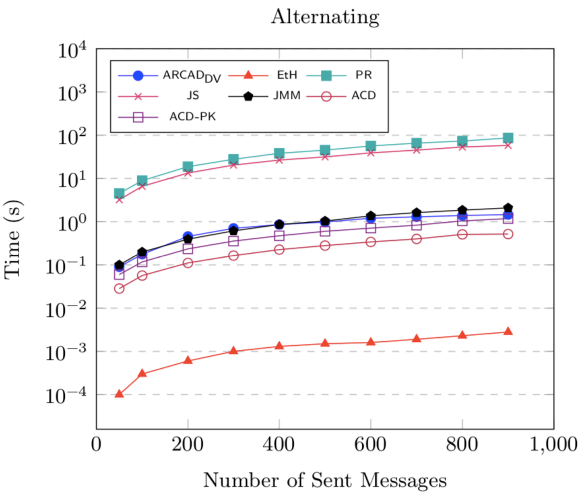
We implemented **PR**, **JS**, **DV**, **JMM**, **ACD**, **ACD-PK** and **EtH** on a machine comparable to a high-end smartphone in three different scenarios:



Alternating — Unidirectional — Def. Unidirectional

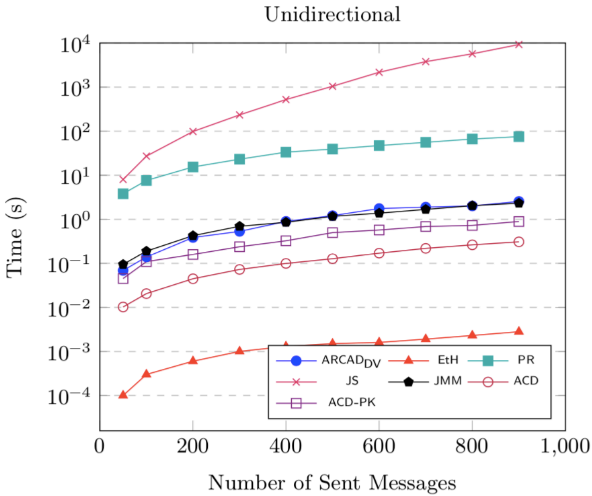
LASEC BOSCH

Benchmarks ii



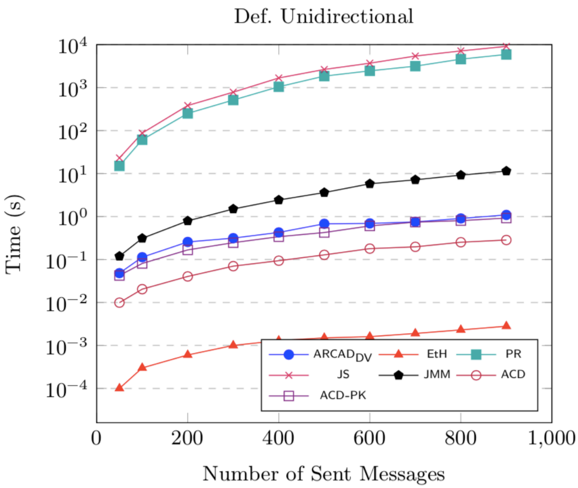
LASEC BOSCH

Benchmarks iii



LASEC BOSCH

Benchmarks iv



LASEC BOSCH

Joël Alwen, Sandro Coretti, and Yevgeniy Dodis.
The double ratchet: Security notions, proofs, and modularization for the signal protocol.
In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 129–158. Springer, 2019.

Nikita Borisov, Ian Goldberg, and Eric A. Brewer.
Off-the-record communication, or, why not to use PGP.
In Vijay Atluri, Paul F. Syverson, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 77–84. ACM, 2004.

Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs.
Ratcheted encryption and key exchange: The security of messaging.

In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 619–650. Springer, 2017.

Katriel Cohn-Gordon, Cas J. F. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila.
A formal security analysis of the signal messaging protocol.
In *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017*, pages 451–466. IEEE, 2017.

Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt.
On post-compromise security.
In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178. IEEE Computer Society, 2016.

F. Betül Durak and Serge Vaudenay.
Bidirectional asynchronous ratcheted key agreement with linear complexity.
In Nuttapong Attrapadung and Takeshi Yagi, editors, *Advances in Information and Computer Security - 14th International Workshop on Security, IWSEC 2019, Tokyo, Japan, August 28-30, 2019, Proceedings*, volume 11689 of

Lecture Notes in Computer Science, pages 343–362. Springer, 2019.

Daniel Jost, Ueli Maurer, and Marta Mularczyk.
Efficient ratcheting: Almost-optimal guarantees for secure messaging.
In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 159–188. Springer, 2019.

Joseph Jaeger and Igors Stepanovs.
Optimal channel security against fine-grained state compromise: The safety of messaging.
In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.

Bertram Poettering and Paul Rösler.
Towards bidirectional ratcheted key exchange.
In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2018.

Hailun Yan and Serge Vaudenay.
Symmetric asynchronous ratcheted communication with associated data.
In Kazumaro Aoki and Akira Kanaoka, editors, *Advances in Information and Computer Security - 15th International Workshop on Security, IWSEC 2020, Fukui, Japan, September 2-4, 2020, Proceedings*, volume 12231 of *Lecture Notes in Computer Science*, pages 184–204. Springer, 2020.