On Active Attack Detection in Messaging with Immediate Decryption

Khashayar Barooti[1], Daniel Collins[2], Simone Colombo[3], Loïs Huguenin-Dumittan[4], and Serge Vaudenay[5]

[1]Aztec Labs, London, United Kingdom
kashbrti@gmail.com
[2]Texas A&M University, College Station, United States
danielpatcollins@gmail.com
[3]King's College London, London, United Kingdom
simone.colombo@kcl.ac.uk
[4]Tune Insight, Lausanne, Switzerland
lois@tuneinsight.com
[5]EPFL, Lausanne, Switzerland
serge.vaudenay@epfl.ch

February 25, 2025

Abstract

The widely used Signal protocol provides protection against state exposure attacks through forward security (protecting past messages) and post-compromise security (for restoring security). It supports *immediate decryption*, allowing messages to be re-ordered or dropped at the protocol level without affecting correctness. In this work, we consider strong *active attack detection for secure messaging with immediate decryption*, where parties are able to immediately detect active attacks under certain conditions. We first consider in-band active attack detection, where participants who have been actively compromised but are still able to send a single message to their partner can detect the compromise. We propose two complementary notions to capture security, and present a compiler that provides security with respect to both notions. Our notions generalise existing work (RECOVER security) which only supported in-order messaging. We also study the related out-of-band attack detection problem by considering communication over out-of-band, authenticated channels and propose analogous security notions. We prove that one of our two notions in each setting imposes a linear communication overhead in the number of sent messages and security parameter using an information-theoretic argument. This implies that each message must information-theoretically contain all previous messages and that our construction, that essentially attaches the entire message history to every new message, is asymptotically optimal. We then explore ways to bypass this lower bound and highlight the feasibility of practical active attack detection compatible with immediate decryption.

This is the full version of a paper with the same title appearing at CRYPTO 2023. See Appendix H for how this document has been revised over time. This work was mainly completed while the authors were working at EPFL.

Contents

1 Introduction

Since the Snowden revelations and given the unprecedented rise of mass surveillance, several messaging solutions strengthened their security guarantees. The susceptibility to state exposure attacks pushed both researchers and practitioners to develop ratcheting-based schemes, which enables forward secrecy—confidentiality of messages sent before a state exposure—and post-compromise security—automatic healing of confidentiality upon compromise [15].

The asynchronicity of messaging and the unreliability of some network protocols have driven the design of ratcheting-based schemes with *immediate decryption* [1, 16, 32], which supports of out-of-order delivery and message loss at the *protocol level*. This property ensures that legitimate messages can be immediately decrypted by the receiver upon arrival and placed correctly among other received messages, even if some previous messages are delayed. Furthermore, communication can continue even if some messages are permanently lost. The Signal protocol, the current de-facto messaging standard, supports immediate decryption. By contrast, many schemes in the literature fail if even a single message is lost (see [7, 11, 14, 21, 33] for a non-exhaustive list).

The aforementioned security notions do not guarantee message authentication when the adversary impersonates parties, e.g., through state compromise. The lack of authentication implies that parties cannot *detect* active attacks. A recent phishing attack against Signal's phone number verification service enabled attackers to re-register accounts to another device, demonstrating the practicality of impersonation attacks via secret state compromise [37]. Similar attacks that steal verification codes to hijack accounts affect a plethora of messaging applications. The proliferation of spyware such as Pegasus represents an additional—and worrying—threat for secret exfiltration [35].

The most widely used mechanisms for detecting active attacks use an *out-of-band* authenticated channel. All such mechanisms we are aware of, either deployed in practice [29] or proposed in the literature [17, 19] assume such a channel. Solutions like Signal's safety numbers [29] enables parties to authenticate long-term keys by comparing QR codes in person. However, as observed by Dowling and Hale [18, 19], Signal's approach—and all similar methods to our knowledge—fails to provide guarantees after a user's state is exposed, since safety numbers only authenticate initial keys (for Signal, the keys that the X3DH key agreement protocol generates).

To remedy this situation, Dowling and Hale [19] proposed to add an additional authentication key to each iteration of Signal's asymmetric ratchet for on-demand use in out-of-band authentication. Their construction allows parties to immediately—that is, without additional communication rounds—authenticate their entire *asymmetric ratchet* out of band. However messages forged under symmetric keys will never be detected. The only other construction in the literature to our knowledge, proposed by Dowling, Günter and Poirrier [17], requires *three rounds* of in-band communication before an out-of-band hash comparison can take place. Contrary to Dowling and Hale's solution, this approach authenticates all messages (albeit does not formally treat out-of-order messages), but

imposes additional rounds, which is especially problematic in the presence of an active adversary. This raises our first research question:

1. Can we authenticate *all* messages in a *single round* of out-of-band communication to detect active attacks in the immediate decryption setting?

Out-of-band authentication is not always practical or even possible. A convenient alternative is to detect active attacks *in-band*, i.e., using the same channel as the messaging protocol. The adversary can, in the worst case, block all messages sent by honest parties, thereby forcing users to resort to out-of-band communication, but mounting such a persistent attack requires considerable resources. Durak and Vaudenay [21] introduce RECOVER security: if a party receives a forgery, then this party does not accept subsequent messages sent honestly by his counterpart. Caforio et al. [11] extend RECOVER security to enable a party to detect whether their *partner* was compromised, i.e., whether they received a forgery. By contrast to out-of-band authentication, no additional messages are required to support attack detection: in-band ciphertexts contain the authentication information. However, these notions and the corresponding constructions assume in-order message delivery and fail on message dropping. This raises a second question, first suggested by Alwen et al. [1]:

2. Can we achieve extended RECOVER security—immediate in-band active attack detection—while supporting immediate decryption?

To detect active attacks, parties need to authenticate their *entire* message history: each message may be a forgery, i.e., the result of an active attack. With immediate decryption, parties cannot be sure which messages their partner has received until they receive an honest reply from them. Intuitively, each message needs to "contain" the message history up until when it was sent. We prove this intuition, which motivates the exploration of performance/security trade-offs and optimisations. In this regard, existing protocols for both in-band and out-of-band active attack detection represent only a subset of the potential design space. Consequently we also ask:

3. What are the *communication costs* of in- and out-of-band active attack detection for messaging with immediate decryption, and what useful performance/security *trade-offs* can be made?

1.1 Our contributions

In this paper, we explore the aforementioned questions. In more detail:

- We introduce (Section 3) a new primitive that we call *authenticated ratcheted communication*, which captures immediate decryption and models communication through both insecure in-band and authentic out-of-band channels.
- In Section 4, we formalise in-band active attack detection for immediate decryption, by extending RECOVER security [11, 21], with two notions,

namely r-RID and s-RID security, for detecting active attacks towards the receiver and on reception of messages from the sender after they were attacked; combined, these two notions comprise RID (recover with immediate decryption) security. We propose a scheme secure under these notions.

- We consider *out-of-band* active attack detection for immediate decryption messaging in Section 5. We introduce notions r-UNF and s-UNF (which combine to UNF for unforgeable), which are analogous to the notions for in-band detection. Demonstrating their similarity, we construct an UNF-secure scheme from a RID-secure scheme. We also construct an UNF-secure ARC scheme given a RC scheme.

- In Section 6, we prove with an information-theoretic argument that ciphertexts in a scheme satisfying either r-RID or r-UNF security must grow *linearly* in the number of messages sent. As our constructions demonstrate, s-RID and s-UNF security are comparatively cheaper and practical.

- In Section 7, we consider different ways to bypass the aforementioned lower bounds. First we discuss ways to optimize the s-RID-secure scheme. We show how one can drastically reduce the overhead as long as the communication between the two parties is balanced. We present the details in **??** and we believe is the most suitable scheme for use in practice that we propose. We also focus on optimizing the *authenticated* out-of-band channel variant, by exploring pruning-based optimizations, where parties securely prune messages as soon as they are authenticated. We finally discuss the performance advantages resulting from a three-move authentication procedure.

1.2 Paper overview

We assume a network where parties communicate over two types of channels: insecure channels and out-of-band authenticated channels. The adversary has full control over insecure channels. In particular, she can read, deliver, modify and delay messages. In the authenticated channels, the integrity and authenticity of the messages are protected, that is, the adversary can read, deliver, duplicate and delay messages but not modify them. In the Signal application, the insecure channel is the usual network, whereas the out-of-band channel is that used for safety number verification [29], typically in-person.

(Authenticated) ratcheted communication. We introduce a syntax for ratcheted communication (RC) in which sent and received messages are associated with totally ordered *ordinals* (epoch/index pairs in Signal [1]). Ordinals enable our protocol to support *immediate decryption* [1], i.e., message loss and re-ordering on the network. We build on this syntax to define *authenticated ratcheted communication*, or ARC, which comprises two additional functions AuthSend and AuthReceive. A party can use AuthSend to send an authentication tag through the out-of-band channel that the counterpart processes with

AuthReceive. AuthSend outputs an ordinal that is at least as large as the last *sent* ordinal for that party. AuthReceive, if successful, should authenticate all messages up to that ordinal; this is captured in UNF security. Our notion ORDINALS enforces these semantics even in presence of forgeries.

RID security. We revisit the definitions of RECOVER security [11, 21] in the immediate decryption setting. We define two complementary security notions for RID security:

- r-RID ensures that the receiver of a forgery does not accept honest messages with ordinals *larger* than that of the forgery.
- s-RID security enables a party to detect if their counterpart has ever received a forgery (i.e., a forgery from the sender).

If a scheme is both r-RID- and s-RID-secure, then it is RID-secure. These notions are orthogonal to forward security and post-compromise security.

We propose a construction that transforms any ratcheted communication scheme into a provably RID-secure one. In the construction, both parties keep track of messages they have sent and received. Every time they send a message, they attach *all* messages (i.e., the ciphertexts from the underlying RC) they have sent and received so far to their ciphertext. When a party receives a message that "contradicts" what it has sent or received, it can deduce that an active attack took place.

To reduce bandwidth, parties send ordinals and hashes of messages, instead of complete ciphertexts. For r-RID security, a receiver $\overline{\mathcal{P}}$ compares the input message and the sender $\mathcal{P}$'s supposed set of sent messages with what $\overline{\mathcal{P}}$ has received previously. For s-RID, it suffices for a receiver $\overline{\mathcal{P}}$ (who knows exactly what it sent) to check whether the sender claims to have received anything that $\overline{\mathcal{P}}$ did not send. Here, $\mathcal{P}$ only needs to send a *single* hash alongside the set of received ordinals (which are generally smaller than hashes), since $\overline{\mathcal{P}}$ can recompute the hash locally. Since the channel is insecure, parties need to perform a series of checks on the ciphertexts to prevent the adversary from tampering with the sets of sent and received messages sent. Both r-UNF- and s-UNF-security build on the collision resistance of the hash function.

UNF security. We define notions analogous to r-RID and s-RID for *authenticated* ratcheted communication schemes. The r-UNF (receiver unforgeable) notion ensures that a party does not accept authentication tags after receiving a forgery, whereas s-UNF (sender unforgeable) ensures that a party does not accept authentication tags coming from a counterpart that received a forgery. We show that a RID-secure scheme can be turned into a UNF-secure scheme. The transformation highlights the similarity between RID and UNF security. In the former, parties authenticate all messages they have sent and received in band, whereas in the latter the messages are authenticated out of band. Concretely, the transformation uses the ciphertext of a RID-secure RC scheme as the authentication tag for an ARC scheme, by moving authentication material to the out-of-band channel.

Communication costs. We prove a linear lower bound on the ciphertext size of any r-RID-secure RC: each ciphertext must capture all information contained in previously sent ones. In fact, the security notion requires that the receiving party is able to immediately detect if *any* subset of previous ciphertexts contains a forgery, since the sender does not know what has been received and ciphertexts can be arbitrarily re-ordered or dropped.

For the proof, we construct an (inefficient) encoder/decoder pair for a list of input messages $(m_1, \ldots, m_n)$ and randomness R, that uses the r-RID RC to compress the input. More precisely, the encoder uses the RC to send messages $(m_1, \ldots, m_n)$ with randomness R to get a list of ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$, and outputs only (ct_n, R). Next, the decoder uses the RC to receive ciphertext ct_n, generates every possible ct_1 for all possible messages m_1 with randomness R and attempts to successfully receive one of them. If this succeeds, it means m_1 was the same message as the one input to the encoder (i.e., the "honest" one). Then, the decoder continues with m_2 and so on, eventually outputting $(m_1, \ldots, m_n, R)$. Finally, by setting the distributions of the messages and randomness as uniform, one can argue by Shannon's theorem that the ciphertext space must be exponentially large in $n \cdot |m_i|$. The formal proof is actually more complicated as the r-RID security does not need to be perfect and many ciphertexts might be successfully received when decoding.

Then, following a nearly identical proof, we prove a linear lower bound on the authentication tag size of any r-UNF-secure ARC. These proofs might be of independent interest. Following similar arguments we show that the state (i.e. secret key) of any public key puncturable encryption (PKPE) scheme must grow linearly with the number of punctures.

Practical active attack detection. We explore how to overcome the linear communication complexity that r-RID and r-UNF impose. We first observe that ciphertexts can be much smaller to achieve s-RID or s-UNF security. As noted above, it suffices for $\mathcal{P}$ to send a single hash of all received messages with the corresponding ordinals, since partner $\overline{\mathcal{P}}$ can recompute the hash if it stores all messages it sends. Assuming each ordinal uses c space, ciphertexts reduce in size from $O(n(\lambda + c))$ to $O(\lambda + nc)$ given $\mathcal{P}$ has received n messages.

Motivating this comparison, we observe that parties achieve r-RID/r-UNF-like guarantees after one round of communication from s-RID/s-UNF security. If $\mathcal{P}$ detects that their partner $\overline{\mathcal{P}}$ has received a forgery, $\mathcal{P}$ can let $\overline{\mathcal{P}}$ know, and thus $\overline{\mathcal{P}}$ can learn that they have received a forgery (which is what r-RID/r-UNF guarantee). We formalise this by proposing a lightweight three-move protocol, and a corresponding security model, over the out-of-band channel which provides bidirectional message authentication. Participant $\mathcal{P}$ (resp. $\overline{\mathcal{P}}$) sends their set of received messages to their partner in the first and second moves. In the second and third moves, $\overline{\mathcal{P}}$ (resp. $\mathcal{P}$) sends a bit that indicates whether the set of received messages was consistent with what they actually sent.

In the aforementioned protocols, performance degrades *linearly* in the number of exchanged messages, even for s-RID/s-UNF security. We observe that

for UNF security the authentication tags can be compressed over time by including acknowledgements in tags. Since the out-of-band channel is authentic, parties are sure that the authentication information—that is, the sets of sent and received messages—coming from the counterpart is correct. This enables parties to prune already authenticated messages. For r-RID security, pruning-based optimizations are more difficult to obtain, since parties do not know which messages are authentic, i.e., have really been sent or received by the counterpart.

1.3 Additional related work

A growing line of work considers the performance and security of messaging in both the two-party [6, 7, 11, 21, 26, 34] and more general group settings [2–4] settings. Some of these works provide similar [26] and sometimes weaker [27] guarantees for in-band active attack detection assuming in-order communication. To our knowledge, in-band active attack detection is not explicitly explored in group messaging, but schemes like MLS ensure that if the state of two parties is forked then their states become incompatible, in some protocol-specific sense.

Naor et al. [30] introduced the concept of immediate key delivery for key exchange: if one goes offline, the remaining ones should be able to complete it successfully by returning a shared secret. This property is orthogonal to immediate decryption as it focuses on keys instead of messages.

Apart from Durak and Vaudenay and Caforio et al. who introduced the RECOVER notions, Dowling et al. [20] provide r-RECOVER, but not s-RECOVER security via signatures, while providing anonymity guarantees even upon state exposure. Dowling et al. [17] frame their authentication guarantees as follows: if no long-term keys are compromised, then all messages exchanged are authentic. Otherwise, active attacks can be detected out-of-band. They achieve this by signing all messages with long-term keys. Our protocols and security notions can be adapted to achieve these guarantees. In distributed computing, the problem is formalised in terms of accountability, which enables parties to detect faulty (Byzantine) nodes [12, 23]. In multi-party computation, a line of work has explored security with *identifiable abort* [24] which ensures that if parties fail to compute a given function, they can identify the party that caused the failure.

The encoder/decoder technique that we use to prove the lower bounds in Section 6 and Appendix G have been used before in cryptography [25, 28]. While the basic idea is the same, the technical details of the proofs are not comparable as the primitives are different. Related work in group messaging achieves communication lower bounds in a symbolic model of execution [9] and in a black-box impossibility setting [8].

2 Notation

We consider two parties, A and B. Let $\mathcal{P}$ be one party (A resp. B) and $\overline{\mathcal{P}}$ be their partner (B resp. A). We use maps, or associative arrays, which associate keys

with values: $m[\cdot] \leftarrow x$ defines a new map with values initially set to x and $m[k]$ returns the element indexed by key k. Keys are tuples of any length $n \geq 1$. We index maps with integers starting from 1; in this case, $m[a : b]$ returns the list of elements whose keys are between a and b. We access the element of a tuple using the dot notation. The function $\mathsf{length}(m)$ returns the number of keys in map m. The empty string is denoted by ε. Given a set S, S^* (respectively S^n) is the set of all strings of arbitrary length (resp. of length n) whose elements are in S. PPT abbreviates 'probabilistic polynomially bounded', which we use in the context of algorithms bounded in terms of the security parameter λ.

3 (Authenticated) ratcheted communication

In this section we introduce the *ratcheted communication* (RC) cryptographic primitive and an extension *authenticated ratcheted communication* (ARC) supporting out-of-band authentication. These primitives augment classic ratcheted secure messaging schemes [1, 11, 27] in two ways: (i) sent and received messages are associated with *ordinals*, and, for ARC, (ii) the syntax encompasses two additional stateful algorithms $\mathsf{AuthSend}$ and $\mathsf{AuthReceive}$.

Ordinals associated with messages enable a party to (1) order incoming messages immediate decryption setting; (2) keep track of how many messages have passed through the communication channel; and (3) infer which messages have been authenticated using the out-of-band channel. Ordinals of the form num can be elements of any set on which a total order is defined. We assume that $\perp$ is always the smallest num, regardless of the set to which num belongs. In Alwen et al.'s [1] and Bienstock et al.'s [10] modelling of the Signal protocol, an ordinal num is defined as a pair of integers (e, c) such that $(\mathsf{e}, \mathsf{c}) < (\mathsf{e}', \mathsf{c}')$ if $\mathsf{e} < \mathsf{e}'$ or both $\mathsf{e} = \mathsf{e}'$ and $\mathsf{c} < \mathsf{c}'$. We formally define an RC scheme below.

Definition 1 (Ratcheted communication (RC))**.** A *ratcheted communication* (RC) scheme comprises the following PPT algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$ takes the security parameter $\lambda \in \mathbb{N}$, expressed in unary, and outputs public parameters pp.
- $\mathsf{Init}(\mathsf{pp}) \to (\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z)$ takes public parameters pp and outputs a state $\mathsf{st}_\mathcal{P}$ for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$, and public information z.
- $\mathsf{Send}(\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{pt}) \to (\mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{ct})$ takes a state $\mathsf{st}_\mathcal{P}$, associated data ad and a plaintext pt and outputs a new state $\mathsf{st}'_\mathcal{P}$, an ordinal num and ciphertext ct.
- $\mathsf{Receive}(\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{ct}) \to (\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt})$ takes a state $\mathsf{st}_\mathcal{P}$, associated data ad and ciphertext ct and outputs an acceptance bit $\mathsf{acc} \in \{\mathsf{true}, \mathsf{false}\}$, state $\mathsf{st}'_\mathcal{P}$, ordinal num and plaintext pt.

The $\mathsf{Receive}$ algorithm returns dummy $\mathsf{st}'_\mathcal{P}$, num, pt which are ignored when $\mathsf{acc} = \mathsf{false}$.

Remark 1. Signal can be viewed as an RC. In the work of Alwen et al. [1], a secure messaging scheme consists of an initialisation algorithm and party-specific Send and Receive algorithms with no associated data. The Receive algorithms, but not the Send algorithms, output an epoch/index pair $(e, i) \in \mathbb{N}^2$ which plays the role of an ordinal. Signal as defined by Alwen et al. [1] can thus be considered an RC by modifying its Send algorithm to output each (e, i) pair as an ordinal and enforcing that $\mathsf{ad} = \perp$ is always input to Send and Receive.

In an ARC, parties rely on AuthSend and AuthReceive to authenticate the communication using a (possibly narrowband) out-of-band authenticated channel. AuthSend inputs state and outputs an updated state, an authentication tag and an ordinal, whereas AuthReceive takes a state and an authentication tag to output an authentication bit, an updated state and an ordinal. Intuitively, the authentication tag is sent via the out-of-band authenticated channel and it enables the receiver to detect active attacks using the AuthReceive algorithm. Participants can decide when to invoke the algorithms and thus use the authentication tag on-demand, e.g., when an out-of-band channel is available.

AuthSend and AuthReceive outputs ordinals with the same semantics as Send and Receive. Namely, the num that AuthSend outputs is greater or equal to the last num that Send outputs; besides ordering authentication tags with respect to messages the party has sent or received, the ordinal indicates which messages (all up until num) the authentication tag authenticates. Similarly, for AuthReceive, the ordinal num indicates that all messages with $\mathsf{num}' \leq \mathsf{num}$ have been authenticated with the received tag.

Definition 2 (Authenticated ratcheted communication (ARC))**.** An *authenticated ratcheted communication* (ARC) scheme comprises the following PPT algorithms:

- Setup, Init, Send, Receive defined as in RC (Definition 1).
- $\mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}}) \to (\mathsf{st}'_{\mathcal{P}}, \mathsf{num}, \mathsf{at})$ takes a state $\mathsf{st}_{\mathcal{P}}$ and outputs a new state $\mathsf{st}'_{\mathcal{P}}$, an ordinal num and an authentication tag at.
- $\mathsf{AuthReceive}(\mathsf{st}_{\mathcal{P}}, \mathsf{at}) \to (\mathsf{auth}, \mathsf{st}'_{\mathcal{P}}, \mathsf{num})$ takes a state $\mathsf{st}_{\mathcal{P}}$ and authentication tag at and outputs an authentication bit $\mathsf{auth} \in \{\mathsf{true}, \mathsf{false}\}$, an updated state $\mathsf{st}'_{\mathcal{P}}$ and an ordinal num.

The AuthReceive algorithm returns dummy $\mathsf{st}'_{\mathcal{P}}$, num which the scheme ignores when $\mathsf{auth} = \mathsf{false}$.

One could alternatively define AuthSend/AuthReceive to output sets of ordinals, rather than single ones, corresponding to which messages have been authenticated. Our security notions ensure that this information can be efficiently computed by parties using the ordinal that the algorithms output.

We define *correctness* for an RC and ARC scheme with the CORRECT game presented in Fig. 1. The game takes a security parameter and a schedule sched as inputs. We use a schedule to model the message flow between the participants, which can (1) send a message, (2) receive a message, and for ARC only, (3) send

an authentication tag, or (4) receive a sent authentication tag. More precisely, sched is an ordered list of instructions either of the form $(\mathcal{P}, \text{“}\mathsf{send}\text{”}, \mathsf{ad}, \mathsf{pt})$, $(\mathcal{P}, \text{“}\mathsf{rec}\text{”}, j)$, and for ARC only, $(\mathcal{P}, \text{“}\mathsf{authsend}\text{”})$, and $(\mathcal{P}, \text{“}\mathsf{authrec}\text{”}, j)$, where $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$, ad denotes associated data, pt denotes a plaintext, and $j \in \mathbb{N}$ indicates either the $(\mathsf{ad}, \mathsf{ct})$ pair or the at to be received—that is, to be processed by $\mathsf{Receive}$ or $\mathsf{AuthReceive}$ respectively.

Game $\mathsf{CORRECT}(1^\lambda, \mathsf{sched})$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z) \leftarrow \mathsf{Init}(\mathsf{pp})$
2 : $\mathsf{ad}_*[\cdot], \mathsf{pt}_*[\cdot], \mathsf{ct}_*[\cdot], \mathsf{at}_*[\cdot] \leftarrow \bot$; $\mathsf{received}[\cdot], \mathsf{sent}[\cdot] \leftarrow \mathsf{false}$; $\mathsf{sent\text{-}num}_* \leftarrow 0$
3 : **for** $i = 1$ **to** $\mathsf{length}(\mathsf{sched})$ **do**
4 : **if** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“}\mathsf{send}\text{”}, \mathsf{ad}, \mathsf{pt})$ for $\mathsf{ad}, \mathsf{pt}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
5 : $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{ct}) \leftarrow \mathsf{Send}(\mathsf{st}_\mathcal{P}, \mathsf{ad}, \mathsf{pt})$
6 : **if** $i > 1 \wedge \mathsf{num} \leq \mathsf{sent\text{-}num}_P$ **then return** 1
7 : $\mathsf{sent}[i] \leftarrow \mathsf{true}$; $\mathsf{ad}_\mathcal{P}[i] \leftarrow \mathsf{ad}$; $\mathsf{pt}_\mathcal{P}[i] \leftarrow (\mathsf{num}, \mathsf{pt})$; $\mathsf{ct}_\mathcal{P}[i] \leftarrow \mathsf{ct}$; $\mathsf{sent\text{-}num}_\mathcal{P} \leftarrow \mathsf{num}$
8 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“}\mathsf{rec}\text{”}, j)$ for $j \in \mathbb{N}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
9 : **if** $\neg\mathsf{sent}[j] \vee \mathsf{received}[j] \vee \mathsf{at}_{\overline{\mathcal{P}}}[j] \neq \bot$ **then continue**
10 : $(\mathsf{acc}, \mathsf{st}'_\mathcal{P}, \mathsf{num}, \mathsf{pt}) \leftarrow \mathsf{Receive}(\mathsf{st}_\mathcal{P}, \mathsf{ad}_{\overline{\mathcal{P}}}[j], \mathsf{ct}_{\overline{\mathcal{P}}}[j])$
11 : **if** $\neg\mathsf{acc} \vee ((\mathsf{num}, \mathsf{pt}) \neq \mathsf{pt}_{\overline{\mathcal{P}}}[j])$ **then return** 1
12 : $\mathsf{received}[j] \leftarrow \mathsf{acc}$; $\mathsf{st}_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
13 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“}\mathsf{authsend}\text{”})$ for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
14 : $(\mathsf{st}_\mathcal{P}, \mathsf{num}, \mathsf{at}) \leftarrow \mathsf{AuthSend}(\mathsf{st}_\mathcal{P})$
15 : **if** $\mathsf{num} < \mathsf{sent\text{-}num}_\mathcal{P}$ **then return** 1
16 : $\mathsf{sent}[i] \leftarrow \mathsf{true}$; $\mathsf{at}_\mathcal{P}[i] \leftarrow (\mathsf{num}, \mathsf{at})$
17 : **elseif** $\mathsf{sched}[i]$ parses as $(\mathcal{P}, \text{“}\mathsf{authrec}\text{”}, j)$ for $j \in \mathbb{N}, \mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$ **then**
18 : **if** $\neg\mathsf{sent}[j] \vee \mathsf{received}[j] \vee \mathsf{at}_{\overline{\mathcal{P}}}[j] = \bot$ **then continue**
19 : $(\mathsf{num}_{\overline{\mathcal{P}}}, \mathsf{at}_{\overline{\mathcal{P}}}) \leftarrow \mathsf{at}_{\overline{\mathcal{P}}}[j]$
20 : $(\mathsf{auth}, \mathsf{st}'_\mathcal{P}, \mathsf{num}) \leftarrow \mathsf{AuthReceive}(\mathsf{st}_\mathcal{P}, \mathsf{at}_{\overline{\mathcal{P}}})$
21 : **if** $\neg\mathsf{auth} \vee \mathsf{num} \neq \mathsf{num}_{\overline{\mathcal{P}}}$ **then return** 1
22 : $\mathsf{received}[j] \leftarrow \mathsf{true}$; $\mathsf{st}_\mathcal{P} \leftarrow \mathsf{st}'_\mathcal{P}$
23 : **return** 0

Figure 1: Correctness game for an RC/ARC scheme. Highlighted statements are only executed for when considering an ARC.

A correct (A)RC scheme must recover the correct plaintext from the corresponding associated data/ciphertext pair. Moreover, the scheme must satisfy the following properties.

- Subsequent calls to the Send algorithm outputs increasing ordinals (line 6 in Fig. 1).[1]
- Ordinals are equal for corresponding calls to Send (resp. $\mathsf{AuthSend}$ for ARC) and $\mathsf{Receive}$ (resp. $\mathsf{AuthReceive}$ for ARC) (lines 11 and 21).

[1]This could instead require Send to output strictly increasing ordinals with respect to $\mathcal{P}$'s calls to Send *and* $\mathsf{Receive}$, which is satisfied by Alwen et al.'s work [1], but we opted against this for generality's sake.

- For ARC, AuthSend outputs an ordinal greater or equal to the ordinal returned by the last call to Send (line 15).

We encode these properties in the CORRECT game for clarity. We require that these properties hold in the adversarial setting (in particular when forgeries are received) and enforce them in the ORDINALS game presented in Fig. 3. We formally define correctness for an (A)RC scheme in Definition 3.

Definition 3 (CORRECT)**.** Consider the correctness game CORRECT presented in Fig. 1. An RC (resp. ARC) scheme is *correct* if, for all $\lambda \in \mathbb{N}$, and all sequences of the form sched with elements of the form $(\mathcal{P}, \text{“send”}, \mathsf{ad}, \mathsf{pt})$, $(\mathcal{P}, \text{“rec”}, j)$, (resp. also of the form $(\mathcal{P}, \text{“authsend”})$, $(\mathcal{P}, \text{“authrec”}, j)$), for $\mathcal{P} \in \{\mathsf{A}, \mathsf{B}\}$, we have $\Pr[\mathsf{CORRECT}(1^\lambda, \mathsf{sched}) \Rightarrow 1] = 0$.

Remark 2. Correctness states that AuthSend must output an ordinal greater or equal to the ordinal that the last call to Send returned. If AuthSend does not increase the ordinal, then it is clear which messages are authenticated; if the ordinal increases in AuthSend, the application designer must keep track of the last num that Send returned to infer what the tag authenticates. Nonetheless, the latter case may be desirable to ensure that all ordinals output by Send and AuthSend are distinct.

Our security notions for RC and ARC schemes build on a common set of oracles, introduced in Fig. 2. The SEND (resp. RECEIVE) oracle enables the adversary to send (resp. receive) a message on behalf of a party $\mathcal{P}$. In SEND, the caller can specify the randomness used by Send or let the challenger sample randomness uniformly. For ARC, AUTHSEND enables the adversary to send an authentication tag on behalf of a party $\mathcal{P}$, whereas AUTHRECEIVE handles AuthReceive. The oracles $\mathsf{EXP}_{\mathsf{pt}}(j)$ and $\mathsf{EXP}_{\mathsf{st}}(j)$ expose plaintexts and states, respectively.

The oracles of Fig. 2 models a communication network composed of insecure in-band and authentic out-of-band channels. The SEND and RECEIVE oracles enable the adversary to read, deliver, modify and delay messages, but AUTHSEND and AUTHRECEIVE do not allow the modification of authentication tags.

We assume an *always-authentic* out-of-band channel. To our knowledge, all deployed solution for out-of-band authentication and relevant literature [17, 18] assume this. One can define a stronger model where the out-of-band channel is authentic only in some cases, e.g., the tampering rate is bounded, or multiple out-of-band channels exist but the adversary can compromise only a subset of them. As a not-always-authentic out-of-band channel is a stronger version of an insecure in-band channel, the discussions in Section 4 apply.

For RC and ARC schemes we require that even in the presence of an adversary that injects forgeries, the Send and Receive (as well as AuthSend and AuthReceive for ARC schemes) algorithms output correct ordinals. An RC or ARC scheme has *correct ordinals* if (1) the Send algorithm always outputs increasing ordinals with respect to all previously sent or received ordinals; (2) corresponding calls to Send and Receive (resp. to AuthSend and AuthReceive)

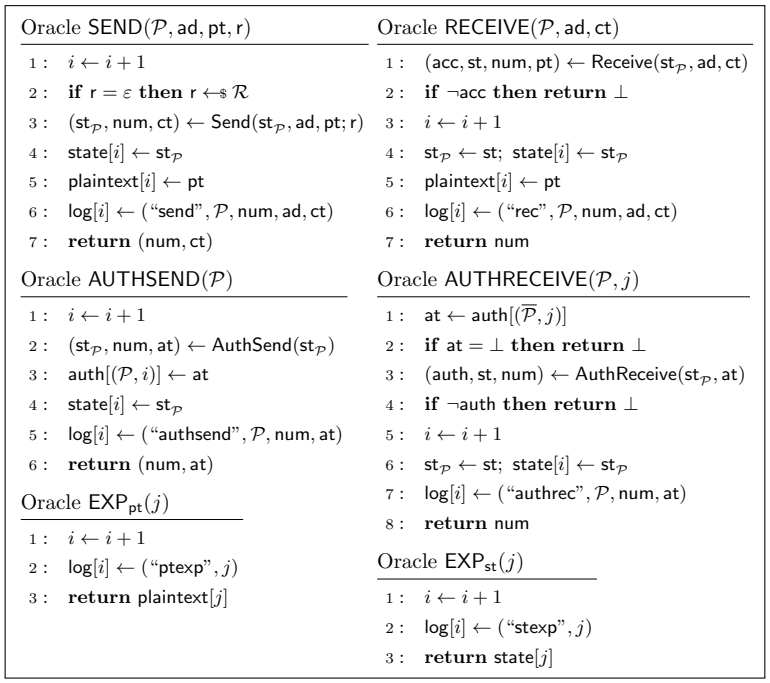


Figure 2: Oracles which use variables state, plaintext, log, auth, st_* and i, all initialized in games where the oracles are used. AUTHSEND and AUTHRECEIVE are only used when considering ARC.

output the same ordinal; and (3) for an ARC scheme, AuthSend outputs an ordinal greater or equal to the ordinal returned by the last call to Send. We consider these properties in CORRECT (Fig. 1), but they must hold also in presence of forgeries. We formalize this notion with the ORDINALS game in Fig. 3.

In this game the challenger verifies three predicates, which correspond to the conditions for correct ordinals presented above. In Definition 4 we formalize ORDINALS-security for (A)RC schemes.

Definition 4 (ORDINALS)**.** Consider the ORDINALS game in Fig. 3. We say that an (authenticated) ratcheted communication scheme is ORDINALS secure if, for all possibly unbounded adversaries $\mathcal{A}$ we have $\Pr[\mathsf{ORDINALS}^{\mathcal{A}}(1^{\lambda}) \Rightarrow 1] = 0$.

Remark 3. The ORDINALS game in Fig. 3 is not suited to the case where ordinals can be arbitrary and in particular collide between parties. Thus, each party must be associated with disjoint ordinals: practical schemes like the Signal protocol do this by associating one party with even epochs and the counterpart with odd epochs.

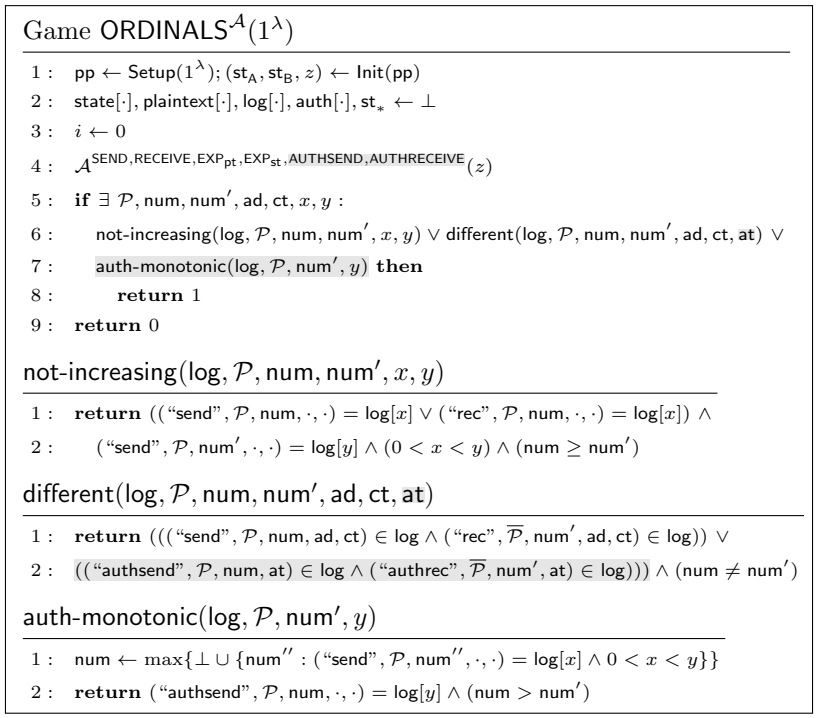


Figure 3: ORDINALS game. Highlighted statements are only considered for an ARC.

4 In-band active attack detection: RID

In this section we consider *in-band* active attack detection in the immediate decryption setting.

Caforio et al. [11] define RECOVER security, which encompasses both r-RECOVER security and s-RECOVER security, by assuming that the channel ensures in-order message delivery. Intuitively, r-RECOVER security prevents a party from being able to deliver an honest message *after* delivering a forgery, and s-RECOVER security allows a party to detect and stop communication when their partner has delivered a forgery. We extend these notions to handle out-of-order message delivery by introducing r-RID and s-RID, which we present in Fig. 4 and illustrate in Fig. 5. Combined, these two properties ensure RID security. Note that these definitions are orthogonal to the usual forward and post-compromise security notions that the ratcheting literature considers [1, 7].

The winning condition in RID consists of three predicates:

- forgery verifies whether a forgery was accepted by one of the participants by taking into account both injection and modification of messages. In the predicate, we denote the impersonated party as $\mathcal{P}$ and the recipient of the forgery as $\overline{\mathcal{P}}$.

- $\mathsf{bad\text{-}\overline{P}}$ checks whether the recipient of the forgery manages to detect the attack. This predicate corresponds to r-RID security.
- bad-P establishes whether $\mathcal{P}$, i.e., the participant that the adversary impersonates to send the forgery, fails to detect the attack. Since $\overline{\mathcal{P}}$ is the recipient of the forgery, the detection of the attack by $\mathcal{P}$ relies on a ciphertext sent by $\overline{\mathcal{P}}$ and honestly delivered. This predicate corresponds to s-RID security.

The game imposes that if forgery returns true, then at least one between bad-P and $\mathsf{bad\text{-}\overline{P}}$ must return true for the adversary to win the game.

Definition 5 (RID)**.** A RC is (q, t, ϵ)-r-RID (resp. s-RID) secure, if for all adversaries $\mathcal{A}$ which make at most q oracle queries and which run in time at most t, we have: $\Pr[\mathsf{r\text{-}RID}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon$ (resp. $\Pr[\mathsf{s\text{-}RID}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon$), where game $\mathsf{r\text{-}RID}^{\mathcal{A}}$ (resp. $\mathsf{s\text{-}RID}^{\mathcal{A}}$) is defined in Figure 4.

Game $\mathsf{r\text{-}RID}^{\mathcal{A}}(1^\lambda)$ / $\boxed{\mathsf{s\text{-}RID}^{\mathcal{A}}(1^\lambda)}$

1 : $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{st_A}, \mathsf{st_B}, z) \leftarrow \mathsf{Init}(\mathsf{pp})$
2 : $\mathsf{state}[\cdot], \mathsf{plaintext}[\cdot], \mathsf{log}[\cdot] \leftarrow \bot$
3 : $\mathsf{auth}[\cdot], \mathsf{st}_* \leftarrow \bot$
4 : $i \leftarrow 0$
5 : $\mathcal{A}^{\mathcal{O}}(z)$
6 : **if** $\exists\ \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}, \mathsf{ct}, \mathsf{ad}', \mathsf{ct}', x, y$:
7 : $\quad \mathsf{forgery}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x) \wedge$
8 : $\quad \mathsf{bad\text{-}\overline{P}}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ **then**
9 : $\quad \boxed{\mathsf{bad\text{-}P}(\mathsf{log}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y)}$ **then**
10 : $\quad\quad$ **return** 1
11 : **return** 0

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

1 : **return** $\mathsf{r\text{-}RID}^{\mathcal{A}}(1^\lambda) \vee \mathsf{s\text{-}RID}^{\mathcal{A}}(1^\lambda)$

$\mathsf{forgery}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$

1 : **return** $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \mathsf{log} \wedge$
2 : $\quad (\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \mathsf{log}[x]$

$\mathsf{bad\text{-}\overline{P}}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$

1 : **return** $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \mathsf{log} \wedge$
2 : $\quad (\text{“send”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \mathsf{log} \wedge$
3 : $\quad (\mathsf{num} < \mathsf{num}')$

$\mathsf{bad\text{-}P}(\mathsf{log}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y)$

1 : **return** $(y > x) \wedge$
2 : $\quad (\text{“send”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') = \mathsf{log}[y] \wedge$
3 : $\quad (\text{“rec”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \mathsf{log}$

Figure 4: r-RID, $\boxed{\text{s-RID}}$ and RID games for $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{EXP_{pt}}, \mathsf{EXP_{st}}\}$.

Although r-RID seems to be a stronger than s-RID at first glance, the two notions are not comparable. There exist schemes which provide r-RID and not s-RID security and vice versa, e.g., the scheme proposed in Figure 6 if the checks for either r-RID or s-RID are removed from the checks subroutine given the underlying RC is not r-RID or s-RID secure, respectively (the Double Ratchet is neither, for example).

However, we note the following link between both notions. Suppose we use a s-RID scheme. This means that $\mathcal{P}$ is able to know that $\overline{\mathcal{P}}$ received a forged message. Then, if $\mathcal{P}$ sends an “abort” message to $\overline{\mathcal{P}}$, $\overline{\mathcal{P}}$ would be able to detect

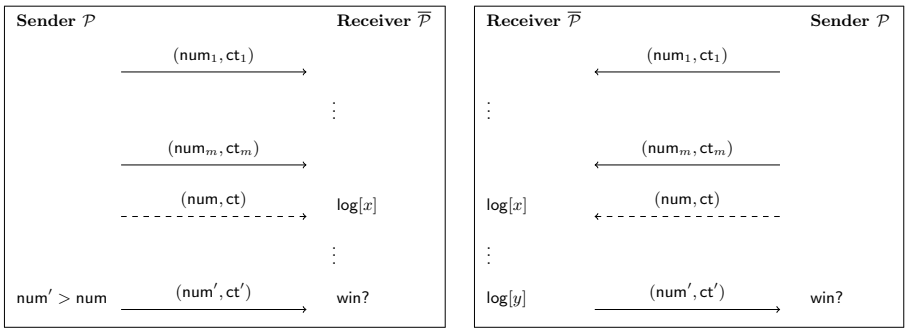


Figure 5: Visualizing r-RID (left) and s-RID (right). Each figure showcases an adversary's winning condition in the respective game. The dashed arrows are forged messages. If $\overline{\mathcal{P}}$ accepts the message at time "win?" then the adversary wins.

the forgery after *one* honest round-trip of messages. In other words, s-RID RC schemes can be transformed (by adding an "abort" message) into RC schemes with a weak variant of r-RID security: r-RID after a honest round-trip.

Remark 4. Suppose A sends five messages with $\mathsf{num} \in \{1, \ldots, 5\}$, B receives a forgery with $\mathsf{num} = 1000$, and then A sends five messages with $\mathsf{num} \in \{6, \ldots, 10\}$. If B never sends, i.e., A is the sender and B the receiver, RID-security only guarantees that the forgery might be detected when A sends the honest message with $\mathsf{num}' = 1001$. (cf. the condition "$\mathsf{num} < \mathsf{num}'$" in predicate bad-$\overline{\mathsf{P}}$ in Fig. 4.). Intuitively, B should be able to detect the forgery on receipt of honest message with $\mathsf{num} = 6$ since this message is "independent" of the forgery with $\mathsf{num} = 1000$. By the not-increasing predicate of the ORDINALS security, all messages that A sends after one round-trip will have $\mathsf{num} > 1000$, so such an attack will nevertheless be eventually detected. Fine-grained security capturing these scenarios can be formalised by tracking state exposures and message delivery timing at the cost of greater definitional complexity; we leave it open to do so. Some forgeries will be defeated by our construction below but it is likely required to leverage the security of the underlying RC to build a secure scheme. Looking ahead, this remark also applies to UNF ARC schemes defined in Section 5.

4.1 A RID-secure RC

In this section we build a RID-secure RC scheme given a correct and ORDINALS-secure RC scheme and a collision-resistant hash function $\mathcal{H}$ (Definition 8). We present our transformation in Figure 6.

Scheme description. Each party $\mathcal{P}$ keeps track of every message it has sent and received (in S and R, respectively). This information is communicated to $\overline{\mathcal{P}}$ every time $\mathcal{P}$ calls Send (via variables S and R′).

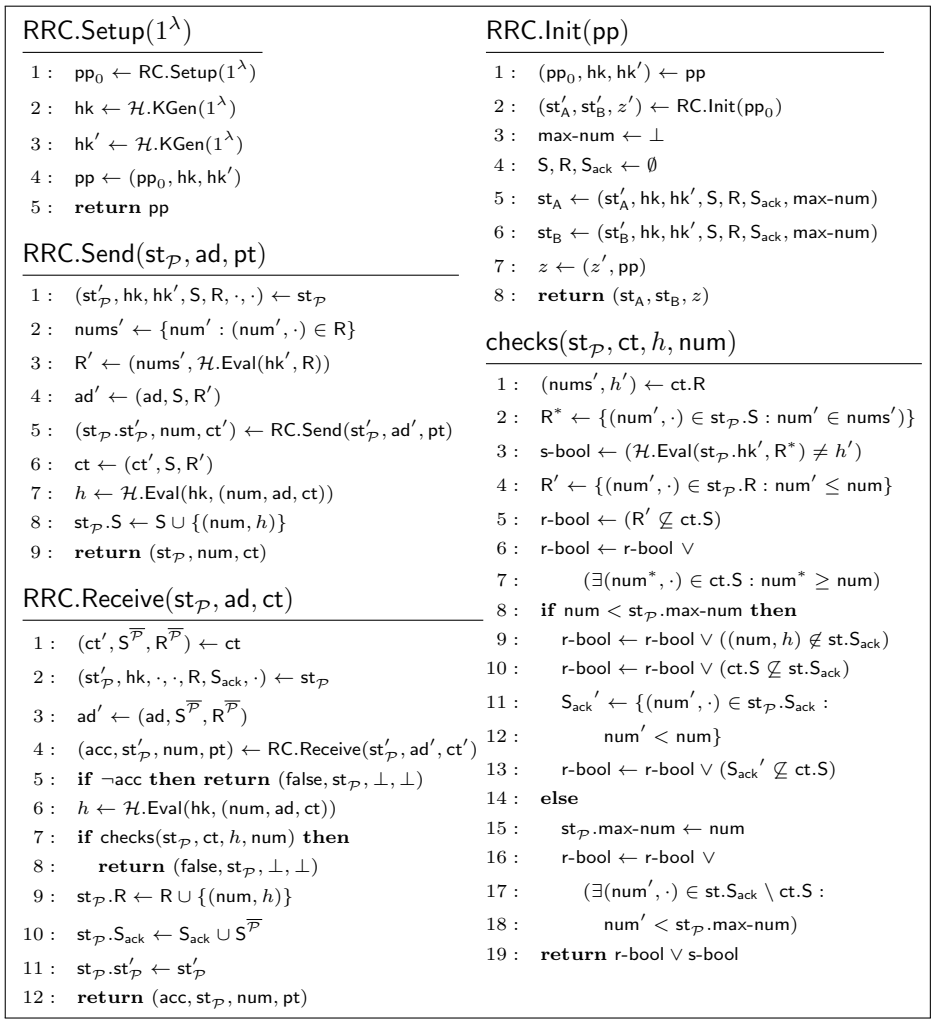


Figure 6: RID-secure RC scheme RRC based on a RC scheme RC (Definition 1) and a hash function $\mathcal{H}$ (Definition 7). RRC requires the following variables: max-num represents the largest received num. S is the set of (num, h) pairs; R is the set of received (num, h) pairs; S_{ack} is the set of (num, h) which are expected to be received (according to the received ciphertext ct). All sets are append-only.

The Send procedure prepares the set R', which contains the ordinals and a hash of all received messages (line 3). This step can be optimized by using an incremental hash function as we discuss in Section 7.1. Next, it calls RC.Send with input (ad', pt) where $ad' = (ad, S, R')$ is the associated data. The ciphertext ct contains both ct′ and sets S and R'. Finally, it adds the pair (num, h) to S (line 8), where the hash h is computed as $\mathcal{H}.Eval(st_\mathcal{P}.hk, (ad, ct))$, where ct =

$(\mathsf{ct}', \mathsf{S}, \mathsf{R}')$. Intuitively, (num, h) acts as a summary of $\mathcal{P}$'s state after calling RC.Send which can be checked by $\overline{\mathcal{P}}$ for inconsistency.

When $\overline{\mathcal{P}}$ invokes Receive, the procedure calls RC.Receive, which outputs $\mathsf{num} \neq \perp$ if the call is successful. Since ct contains $\mathsf{R}^{\overline{\mathcal{P}}}$, $\overline{\mathcal{P}}$ checks that what $\mathcal{P}$ received so far was correct (line 3 in checks). In addition, using the S set contained in the ciphertext ct, $\overline{\mathcal{P}}$ can further check whether the ciphertexts it received so far have indeed been sent by $\mathcal{P}$. This is verified from lines 5 to 18 of checks. Some checks detect tampering of ct by the adversary (e.g. ct.S should not contain ordinals larger than the one of the current ciphertext, or if ct was sent earlier than another ciphertext already received, ct.S should be consistent with messages already acknowledged, etc.). If everything verifies, Receive stores (num, h) in R and adds ct.S to the set of acknowledged messages (lines 9 and 10).

Remark 5. The sets S and R′ included in the ciphertext are also included in the authenticated data passed to the underlying RC: the tuple $(\mathsf{S}, \mathsf{R}')$ is always authenticated in Fig. 6. This is actually not needed for RID security, but for authentication and confidentiality. Even if we do not define authentication and confidentiality for (A)RC, we use authenticated encryption for completeness.

Remark 6. Our scheme outputs a generic error symbol $\perp$ in all cases. In particular, it returns the same symbol regardless of whether the error was due to detecting active attack, or from the situation where the adversary did not expose any states and simply send a malformed ciphertext. Treating both scenarios identically could lead to a denial-of-service attack vector, so in practice, they should be distinguished. We leave this differentiation to future work.

Security analysis. Correctness of the RRC scheme follows from the correctness of the underlying scheme RC and the fact that the checks always outputs false when only honest messages are received. Similarly, ORDINALS-security follows from the ORDINALS-security of RC, as RRC outputs the num that RC outputs. As the next theorems state, the construction of Fig. 6 is r-RID-secure (Theorem 1) and s-RID-secure (Theorem 2). The construction is therefore RID-secure.

Theorem 1. Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function. Then RRC (defined in Figure 6) is a (q, t, ϵ_{cr})-r-RID-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

The r-RID security of RRC reduces to the collision resistance of the hash function $\mathcal{H}$. We present the complete proof in Appendix B.1.

Theorem 2. Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function. Then RRC (defined in Figure 6) is a (q, t, ϵ_{cr})-s-RID-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

The s-RID security of RRC reduces to the collision resistance of the hash function $\mathcal{H}$. We present the complete proof in Appendix B.2.

Optimization. The s-RID notion imposes less average overhead than r-RID: the construction can be further optimized and still provide s-RID security. We give here an intuition about this optimization and present the details in Section 7.2. In the optimized version, parties keep track of epochs. Party A starts with epoch $= 0$ and B with epoch $= 1$. While sending each message parties attach the epoch alongside. If $\mathsf{epoch}_\mathsf{A} = t$, A does not accept any messages with epoch $> t + 1$ and if A receives a message with epoch $= t + 1$ they update their epoch $\leftarrow t + 2$. The observation is that epoch values only increase when both parties have received a message. Using this fact, it can be shown that it suffices to convey information about the messages received in the last two epochs to provide s-RID security. If an honest message was sent from A to B after A received a forged message, either this forgery was received in the last 2 epochs, or there was another forgery and honest message pair after it, as otherwise the epoch values would be out of sync. Although this optimization does not change the worst-case complexity of Fig. 6, if the direction of the conversation changes frequently enough the overhead significantly decreases.

5 Out-of-band active attack detection: UNF

In-band active attack detection is not always possible, as an adversary may block all honest messages in the network. For example, modern messaging solutions use a possibly malicious third party server to relay messages between participants, thereby introducing a single point of failure for in-band communication. This brings us to consider out-of-band active attack detection.

An ARC scheme is *unforgeable* if, as soon as one of the two parties accepts a forgery, both parties can detect this out-of-band. We formalize this security notion through the UNF game (Fig. 7), which, similarly to RID, encompasses r-UNF and s-UNF. The winning condition in UNF consists of three predicates: forgery, bad-$\overline{\mathsf{P}}$ (corresponding to r-UNF) and bad-P (corresponding to s-UNF) that are essentially the same as the predicates that we use to define RID security (Definition 5), except they rely on authentication tags instead of ciphertexts for forgery detection.

Definition 6 (UNF)**.** Consider the r-UNF (resp. s-UNF) game in Fig. 7. We say that an ARC scheme is (q, t, ϵ)-r-UNF (resp. s-UNF) secure if, for all adversaries $\mathcal{A}$ which make at most q oracle queries, and which run in time at most t, we have: $\Pr[\mathsf{r\text{-}UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon$ (resp. $\Pr[\mathsf{s\text{-}UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon$).

Remark 7. As for RC schemes, we do not define message indistinguishability for ARC schemes. All the schemes include in the authentication tag only *public* material, i.e., messages that have already transited through the insecure channel. Since the adversary already has access to the entire transcript of the insecure channel, the authentication material does not give any additional advantage for distinguishing.

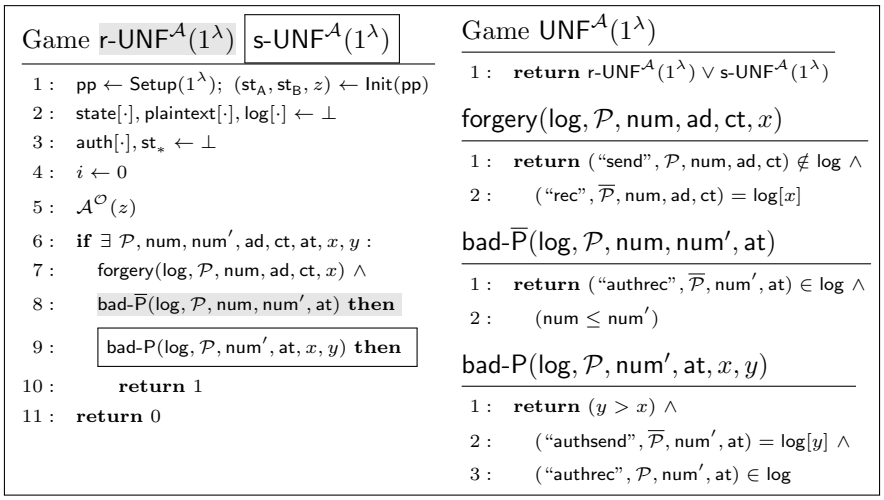


Figure 7: r-UNF, s-UNF and UNF games for $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{EXP}_{\mathsf{pt}}, \mathsf{EXP}_{\mathsf{st}}, \mathsf{AUTHSEND}, \mathsf{AUTHRECEIVE}\}$.

5.1 UNF-secure ARC from a RID-secure RC

We show in this section that RID-secure RC schemes imply UNF-secure ARC ones. More precisely, one can easily build an UNF-secure ARC scheme from a RID-secure RC. The ARC scheme uses the Setup, Gen, Init, Send, Receive function of RC. To send an authentication tag with AuthSend, the ARC scheme calls the Send function on a dummy message to generate a ciphertext ct that acts as the authentication tag. The function AuthReceive is then implemented as a Receive call on the authentication tag/ciphertext. The construction is detailed in Figure 8.

Then, we can show the following theorem, which also implies that the scheme of Fig. 8 is also r-UNF- and s-UNF-secure.

Theorem 3. Let RRC be a RC scheme and RC-ARC be the ARC scheme built out of RRC as shown in Figure 8. If RRC is RID, ORDINALS-secure and correct, then RC-ARC is UNF-, ORDINALS-secure and correct.

Proof. Correctness follows from the correctness of the underlying scheme RRC and the use of domain separation for tags and ciphertexts.

Now, we sketch the proof that RID security of RRC implies UNF security of RC-ARC. For any adversary $\mathcal{A}$ playing the UNF game with RC-ARC, we build a RID adversary $\mathcal{B}$ for RRC. Each query made by $\mathcal{A}$ to the oracles SEND, RECEIVE, $\mathsf{EXP}_{\mathsf{pt}}$, $\mathsf{EXP}_{\mathsf{st}}$ are forwarded by $\mathcal{B}$ to its own corresponding oracles (and domain separation is correctly implemented where needed). Queries of the form AUTHSEND($\mathcal{P}$) are simulated by $\mathcal{B}$ querying $\mathsf{at}' \leftarrow \mathsf{SEND}(\mathcal{P}, 0, 0)$ and setting $\mathsf{at} \leftarrow (1, \mathsf{at}')$, which perfectly simulates the generation of a tag in ARC. Finally, AUTHRECEIVE queries are simulated using the RECEIVE oracle on the tag/ciphertext. $\mathcal{B}$ can perfectly simulate the UNF game for $\mathcal{A}$.

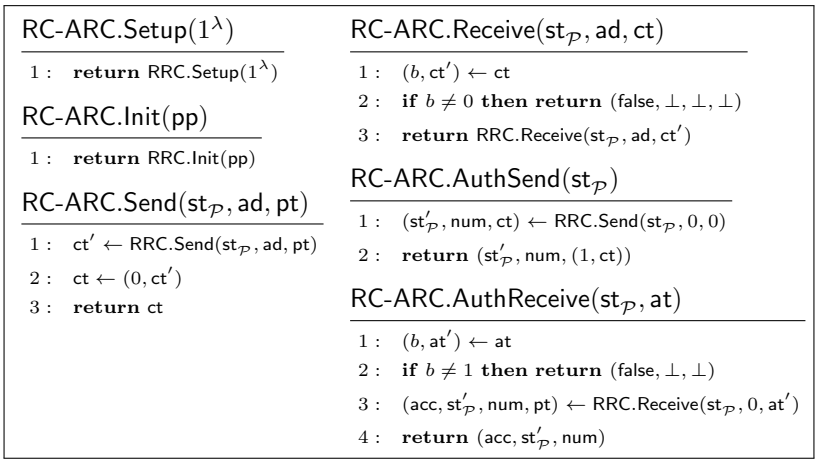


Figure 8: UNF-secure ARC scheme based on a RID-secure RC scheme RRC.

Now, let us assume that the UNF adversary $\mathcal{A}$ wins with the forgery and bad-P predicates evaluating to true. It means a forgery was received by a party $\mathcal{P}$, then, later, that party sent a tag (i.e. a ciphertext in the RID game played by $\mathcal{B}$) that is honestly and successfully delivered to a party $\overline{\mathcal{P}}$. That implies that in the RID game played by $\mathcal{B}$, a party received a forgery, then sent a message that was successfully delivered, which is a winning condition for $\mathcal{B}$.

The second case is when the UNF adversary $\mathcal{A}$ wins with the forgery and bad-$\overline{\mathsf{P}}$ predicates evaluating to true. This means that a forgery was received by a party $\mathcal{P}$ with ordinal num, then a tag with ordinal $\mathsf{num}' \geq \mathsf{num}$ was successfully received. Note that in our RC-ARC construction the tags are ciphertexts, thus the ordinals are strictly increasing, i.e., $\mathsf{num}' > \mathsf{num}$. Therefore, in the RID game played by $\mathcal{B}$, a forgery with ordinal num was received by $\mathcal{P}$, then later a honest ciphertext with ordinal $\mathsf{num}' > \mathsf{num}$ was successfully delivered to $\mathcal{P}$, making the bad-$\overline{\mathsf{P}}$ predicate in the RID game true.

Hence, for any adversary $\mathcal{A}$ winning the UNF game, there exists a RID adversary $\mathcal{B}$ that wins with at least the same probability.

ORDINALS-security follows from the construction. □

5.2 UNF-secure ARC from any RC

We present a non-optimized UNF-secure ARC scheme given a RC scheme (Definition 1), i.e. we define the two additional algorithms AuthSend and AuthReceive. We present our scheme in Fig. 9.

Scheme description. The Send and Receive procedures call the respective procedures of the underlying RC scheme. The Send procedure stores the hash of (ad, ct) for the message being sent, together with the corresponding num that the underlying RC.Send algorithm returns. The tuple composed of num

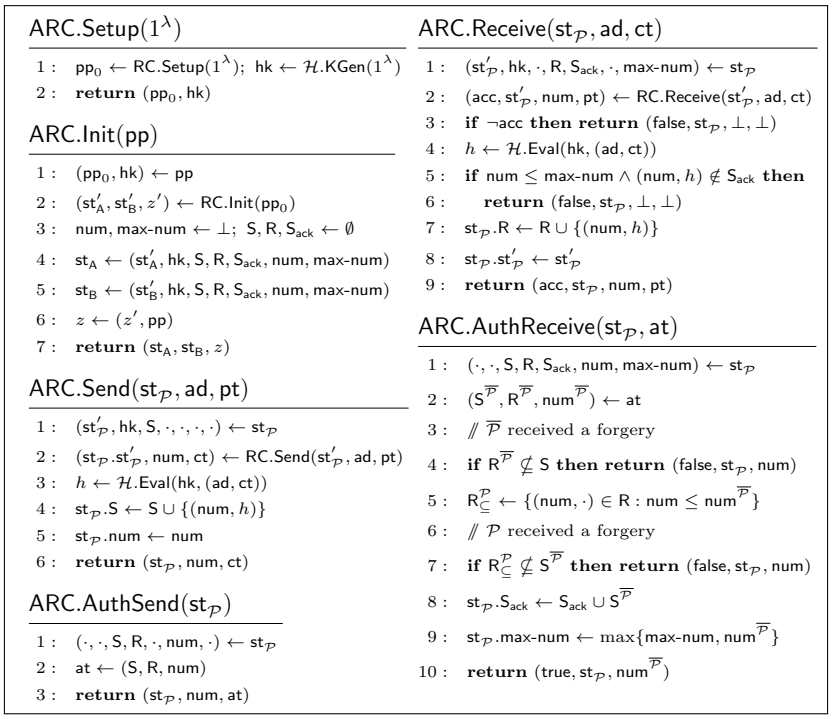


Figure 9: UNF-secure ARC scheme based on a RC scheme RC (Definition 1). The scheme uses four additional variables compared to RC: S is the set of sent (num, h); R is the set of received (num, h); S_ack is the set of (num, h) which are expected to be received (according to the received authentication tag at); max-num represents the largest num received in an at. All sets are append-only. For simplicity of exposition, we omit the optimisation where R is sent as a single hash and n ordinals as done in Fig. 6 for RID security.

and the hash is stored in a set S, which is in turn stored in the internal state of the party. The Send algorithm also updates the ordinal num in the state. The Receive procedure verifies if the RC.Receive algorithm accepts the inputs and that the received message is not a forgery on a previously authenticated message, which is by construction contained in S_ack. If both checks pass, Receive stores the hash of (ad, ct) together with the ordinal num returned by RC.Receive in a set R.

AuthSend puts in the authentication tag at the hashes of the sent and received messages together with the last num returned by RC.Send. Intuitively, the num in the authentication tag at indicates which messages are authenticated in the S messages. Since the adversary can reorder messages both in the normal channel and in the out-of-band channel, the ordinal indicates to the recipient of the authentication tag which messages they should compare against at. AuthReceive parses the authentication tag and checks whether the messages received by the

counterpart are in the local S set. Then it verifies whether the local set of received messages, without the messages not encompassed by at, is a subset of the messages sent by the counterpart. If one of these conditions is not satisfied, then a forgery is detected. The sent messages authenticated by the counterpart are stored in a set $\mathsf{S}_{\mathsf{ack}}$. $\mathsf{Receive}$ uses this set to avoid forgeries on already authenticated num's.

Remark 8. The size of the authentication tags and the state of each party in the scheme of Fig. 9 is linear in the number of sent and received. Messages can nevertheless be efficiently exchanged out-of-band in practice, e.g., using Bluetooth. Otherwise, parties can send authentication information over the insecure channel and authenticate it using the out-of-band channel by hashing and comparing digests [31]. If we assume that the underlying network channel is ordered (e.g., by using TCP), then the hashes of the last sent and received messages suffice to detect forgeries [11].

Security analysis. We now analyze the security properties of the scheme in Fig. 9. Correctness of the scheme follows from the correctness of the underlying RC scheme. Similarly, $\mathsf{ORDINALS}$ security follows from the $\mathsf{ORDINALS}$ security of RC, as the scheme of Fig. 9 outputs the same num that RC outputs.

The UNF-security of the ARC scheme presented in Fig. 9, lies in the collision resistance of the hash function that the scheme uses. When one party $\mathcal{P}$ wants to authenticate the communication it produces an authentication tag containing the hashes of all the messages inboxed and outboxed by $\mathcal{P}$. These hashes can be compared with the counterpart $\overline{\mathcal{P}}$ to detect if any forgery has been received and accepted by one of the participants. In what follows we prove that the scheme of Fig. 9 is UNF-secure.

Theorem 4 (Unforgeability of ARC)**.** Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 7). Then the ARC scheme, that we present in Fig. 9, is (q, t, ϵ_{cr})-UNF secure ARC scheme where $t \approx t_{cr}$.

We reduce the UNF-security of the scheme described in Fig. 9 to the collision resistance of the hash function $\mathcal{H}$. The complete proof is given in Appendix C.

6 Communication costs for attack detection

We study in this section the size of both ciphertexts and authentication tags of $\mathsf{r\text{-}RID}$ RC and $\mathsf{r\text{-}UNF}$ ARC schemes, respectively. In particular, all our constructions of such schemes imply a linear growth of ciphertexts (and tags) in the number of messages sent, in the worst case. We show here that one cannot hope for better by proving two lower bounds. More precisely, we show that the ciphertext space (resp. tag space) of a $\mathsf{r\text{-}RID}$ RC (resp. $\mathsf{r\text{-}UNF}$ ARC) grows exponentially in the number of messages sent. Note that we cannot prove a lower bound on the ciphertext size directly as it is always possible that *some* ciphertext is small. However, our bounds imply that at least n bits are required

to represent any ciphertext or tag in their respective domain after the n-th message.

6.1 Communication cost of r-RID RC

In what follows, we consider a RC that is perfectly correct: for all randomness r, valid states $\mathsf{st}_{\mathcal{P}}$ and associative data ad, the function $\mathsf{Send}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \cdot; \mathsf{r})$ mapping a plaintext to a ciphertext is injective.

The next theorem proves that ciphertext size in a r-RID RC grows linearly in the number of messages (times either the security parameter or message size).

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 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: $\Pr[\mathsf{r\text{-}RID}^{\mathcal{A}}_{\Pi}(1^\lambda) \Rightarrow 1] \leq \frac{1}{2^\gamma}$. Let $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \{0,1\}^k$ be the plaintext and ciphertext space associated to Π, respectively. Then,

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

A third lower bounds gives

$$k \geq nn_s - \frac{1}{1 - \frac{2^n n_s}{2^\gamma}},$$

which is tighter for low values of n (e.g. $n = 1, 2$) and when $\gamma > n + \mathsf{log}(n_s)$.

Proof. We show that if k is smaller than the given bounds, one can build an encoder/decoder for a uniform source s.t. the expected bit-length of a codeword is strictly lower than the entropy (i.e. the log of the size of the sampling set), contradicting Shannon's source coding theorem.

More formally, we consider a source that samples uniformly at random from the set $\{0,1\}^{n \times n_s} \times \{0,1\}^r$, where r is the maximal number of bits (i.e. random coins) needed by the two procedures Setup and Init of Π and n_s invocations of Send. We present an encoder and decoder for such a source in Figure 11 (the non-boxed instructions in the encoder, and the decoder shown on the left). First assume that the RC used in the encoder/decoder has perfect r-RID security. Then, the sender sends n_s honestly generated ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_{n_s}$, and the receiver receives the last ciphertext ct_{n_s}. By perfect r-RID security, for any $i < n_s$, any ciphertext $\mathsf{ct}'_i \neq \mathsf{ct}_i$ should be rejected by the receiver. Thus, one can build an (inefficient) decoder that tests all ct'_i to find the correct one and recovers the corresponding message. In a sense, all ct_i must be encoded in the last ciphertext ct_{n_s}. The actual encoding is more complicated as if the r-RID security is not perfect, there will be a number of false positives (i.e. $\mathsf{ct}'_i \neq \mathsf{ct}_i$ but ct'_i is accepted by the receiver). Note that w.l.o.g., we omit the associated data throughout the proof (or assume $\mathsf{ad} = \perp$) as it plays no role.

Lemma 1. Our encoder (Figure 11) is perfectly correct, i.e.

$$\Pr[\mathsf{Decode}(\mathsf{Encode}(m_1,\ldots,m_{n_s},R)) = (m_1,\ldots,m_{n_s},R)] = 1\ .$$

Proof. The value R output by Decode is the same as the one input in Encode. Since the initial states depend only on R and Π is correct, ct_{n_s} will decrypt to m_{n_s}. The states st_A^i will be identical in both the encoding and decoding procedures as they are generated from st_A^{i-1}, the previously recovered message m_{i-1} and randomness R_i. This implies that the sets of accepting messages S_i will be the same as they depend only on st_B^1 and st_A^{i-1}. In addition, by the perfect correctness of Π, each message m_i will be in the corresponding set S_i. Hence, the decoder can recover each message m_i by reading S_i at the index given in the input. □

Lemma 2. Let C be the random variable corresponding to the codeword length output by Encode. In addition, let $F_i := S_i \setminus \{m_i\}$ be the set of false positives, where S_i and m_i are as in Encode. Then, $\mathbb{E}[C] \leq k+r+\sum_{i=1}^{n_s-1} 1+\mathsf{log}(\mathbb{E}[|F_i|]+1)$.

Proof. By design, the encoder outputs a codeword of $k + r + \sum_{i=1}^{n_s-1} \lceil \mathsf{log}(|S_i|)\rceil$ bits. Therefore, we have

$$\mathbb{E}[C] = k + r + \sum_{i=1}^{n_s-1} \mathbb{E}[\lceil \mathsf{log}(1+|F_i|)\rceil] \leq k + r + \sum_{i=1}^{n_s-1} 1 + \mathbb{E}[\mathsf{log}(1+|F_i|)]$$

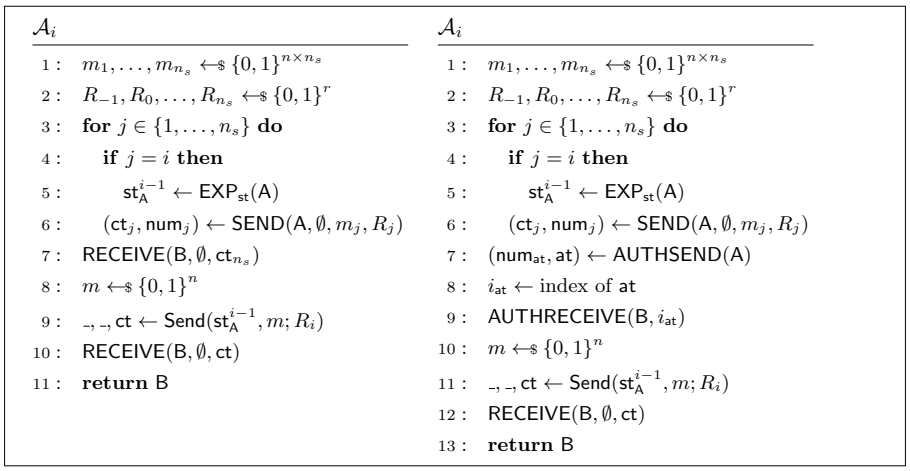
which is upper bounded by $k + r + \sum_{i=1}^{n_s-1} 1 + \mathsf{log}(\mathbb{E}[|F_i|] + 1)$, by the linearity of expectation and the definition of F_i, the fact that $\lceil x\rceil \leq 1 + x$, and Jensen's inequality, respectively. □

Finally, we show the following key lemma.

Lemma 3. Let F_i, $i \in [n_s - 1]$ be defined as above and n, γ as in the statement of the Theorem. Then, $\mathbb{E}[|F_i|] \leq 2^{n-\gamma}$.

Proof. We proceed by contradiction. That is, we show that if $\mathbb{E}[|F_i|] > 2^{n-\gamma}$, then there exists an adversary $\mathcal{A}_i$ s.t. $\Pr[\mathsf{r\text{-}RID}_\Pi^{\mathcal{A}_i}(1^\lambda) \Rightarrow 1] > \frac{1}{2^\gamma}$.

We present such an adversary $\mathcal{A}_i$ on the left of Figure 10. The adversary samples n_s messages $m_1,\ldots,m_{n_s}$ at random and makes A send these with randomness $R_1,\ldots,R_{n_s}$, respectively. Then, $\mathcal{A}_i$ makes B receive the last ciphertext ct_{n_s}. Next, $\mathcal{A}_i$ samples a random message m, sends it using state $\mathsf{st}_\mathsf{A}^{i-1}$ and randomness R_i to get a ciphertext ct and makes B receive it. Now, as ct_{n_s} is sent after ct (ct_{n_s} is sent with $\mathsf{st}_\mathsf{A}^{n_s-1}$ and ct with $\mathsf{st}_\mathsf{A}^{i-1}$), ct_{n_s} and ct will decrypt respectively to num_{n_s} and num_i s.t. $\mathsf{num}_{n_s} > \mathsf{num}_i$ by correctness. Then, if $m \neq m_i$, then ct is different from the i-th ciphertext ct_i (as we assume $\mathsf{Send}(\mathsf{st}_\mathsf{A}^{i-1},\cdot;R_i)$ is injective). Therefore, if ct is accepted and $m \neq m_i$, then ct and num_i satisfy the $\mathsf{forgery}$ predicate of the $\mathsf{r\text{-}RID}$ game in Figure 4. In addition, as ct_{n_s} is sent and delivered honestly, the conditions on lines 7 and 8 of the $\mathsf{r\text{-}RID}$ game always hold for num_{n_s} and ct_{n_s}, and the adversary wins (i.e. the $\mathsf{bad\text{-}\overline{P}}$ predicate is satisfied). We call this event win.



$\mathcal{A}_i$

1 : $m_1, \ldots, m_{n_s} \leftarrow\$ \{0,1\}^{n \times n_s}$
2 : $R_{-1}, R_0, \ldots, R_{n_s} \leftarrow\$ \{0,1\}^r$
3 : **for** $j \in \{1, \ldots, n_s\}$ **do**
4 : **if** $j = i$ **then**
5 : $\mathsf{st}_{\mathsf{A}}^{i-1} \leftarrow \mathsf{EXP}_{\mathsf{st}}(\mathsf{A})$
6 : $(\mathsf{ct}_j, \mathsf{num}_j) \leftarrow \mathsf{SEND}(\mathsf{A}, \emptyset, m_j, R_j)$
7 : $\mathsf{RECEIVE}(\mathsf{B}, \emptyset, \mathsf{ct}_{n_s})$
8 : $m \leftarrow\$ \{0,1\}^n$
9 : $\_, \_, \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{st}_{\mathsf{A}}^{i-1}, m; R_i)$
10 : $\mathsf{RECEIVE}(\mathsf{B}, \emptyset, \mathsf{ct})$
11 : **return** B

$\mathcal{A}_i$

1 : $m_1, \ldots, m_{n_s} \leftarrow\$ \{0,1\}^{n \times n_s}$
2 : $R_{-1}, R_0, \ldots, R_{n_s} \leftarrow\$ \{0,1\}^r$
3 : **for** $j \in \{1, \ldots, n_s\}$ **do**
4 : **if** $j = i$ **then**
5 : $\mathsf{st}_{\mathsf{A}}^{i-1} \leftarrow \mathsf{EXP}_{\mathsf{st}}(\mathsf{A})$
6 : $(\mathsf{ct}_j, \mathsf{num}_j) \leftarrow \mathsf{SEND}(\mathsf{A}, \emptyset, m_j, R_j)$
7 : $(\mathsf{num}_{\mathsf{at}}, \mathsf{at}) \leftarrow \mathsf{AUTHSEND}(\mathsf{A})$
8 : $i_{\mathsf{at}} \leftarrow$ index of at
9 : $\mathsf{AUTHRECEIVE}(\mathsf{B}, i_{\mathsf{at}})$
10 : $m \leftarrow\$ \{0,1\}^n$
11 : $\_, \_, \mathsf{ct} \leftarrow \mathsf{Send}(\mathsf{st}_{\mathsf{A}}^{i-1}, m; R_i)$
12 : $\mathsf{RECEIVE}(\mathsf{B}, \emptyset, \mathsf{ct})$
13 : **return** B

Figure 10: Adversary for the proof of Theorem 5 (resp. Theorem 6) on the left (resp. on the right).

We now compute the probability that win happens, which is the probability that $\mathsf{ct}_i \neq \mathsf{ct}$ and B accepts ct. Let $m_1, \ldots, m_{n_s}$ and the whole randomness R ($R = R_{-1}, R_0, \ldots, R_{n_s}$) be fixed. As before, let S_i be the set of messages m s.t. $\mathsf{Receive}(\mathsf{st}_B^1, \mathsf{ct})$ accepts, for $\mathsf{ct} = \mathsf{Send}(\mathsf{st}_A^{i-1}, m; R_i)$. Note that since S_i depends only $(m_1, \ldots, m_{n_s}, R)$ (which are now fixed), it is deterministic. Therefore, conditioned on $m_1, \ldots, m_{n_s}, R$, we have

$$\Pr_m[\mathsf{win}] = \Pr_m[m \in S_i \wedge m \neq m_i] = \Pr_m[m \in F_i] = \frac{|F_i|}{2^n}$$

as m is sampled uniformly at random. Hence, overall

$$\Pr_{m, m_1, \ldots, m_{n_s}, R}[\mathsf{win}] = \mathbb{E}_{m_1, \ldots, m_{n_s}, R}[\Pr_m[m \in F_i]] = \frac{\mathbb{E}[|F_i|]}{2^n} \ .$$

Note that both the source and the adversary sample $m_1, \ldots, m_{n_s}, R$ uniformly at random. Finally, if $\mathbb{E}[|F_i||] > 2^{n-\gamma}$, then $\Pr[\mathsf{win}] > \frac{2^{n-\gamma}}{2^n} = 2^{-\gamma}$, which leads to the contradiction. □

By the previous lemma, we have $\mathsf{log}(\mathbb{E}[|F_i|]+1) \leq \mathsf{log}(2^{n-\gamma}+1) \leq \max(0, n-\gamma)+1$. Plugging this result into Lemma 2, we get $\mathbb{E}[C] \leq k+r+(n_s-1)(\max(0, n-\gamma)+2)$. In addition, as our encoder outputs a uniquely decodable code, we know that $n_s n + r \leq \mathbb{E}[C]$ by Shannon's source coding theorem. Hence, we get

$$k + r + (n_s - 1)(n - \gamma + 2) \geq n_s n + r \iff k \geq n + (n_s - 1)(\gamma - 2)$$

if $\gamma \leq n$ and otherwise

$$k + r + (n_s - 1)2 \geq n_s n + r \iff k \geq 2 + n_s(n - 2).$$

Now that the first two lower bounds have been shown, we prove the final bound in the following lemma.

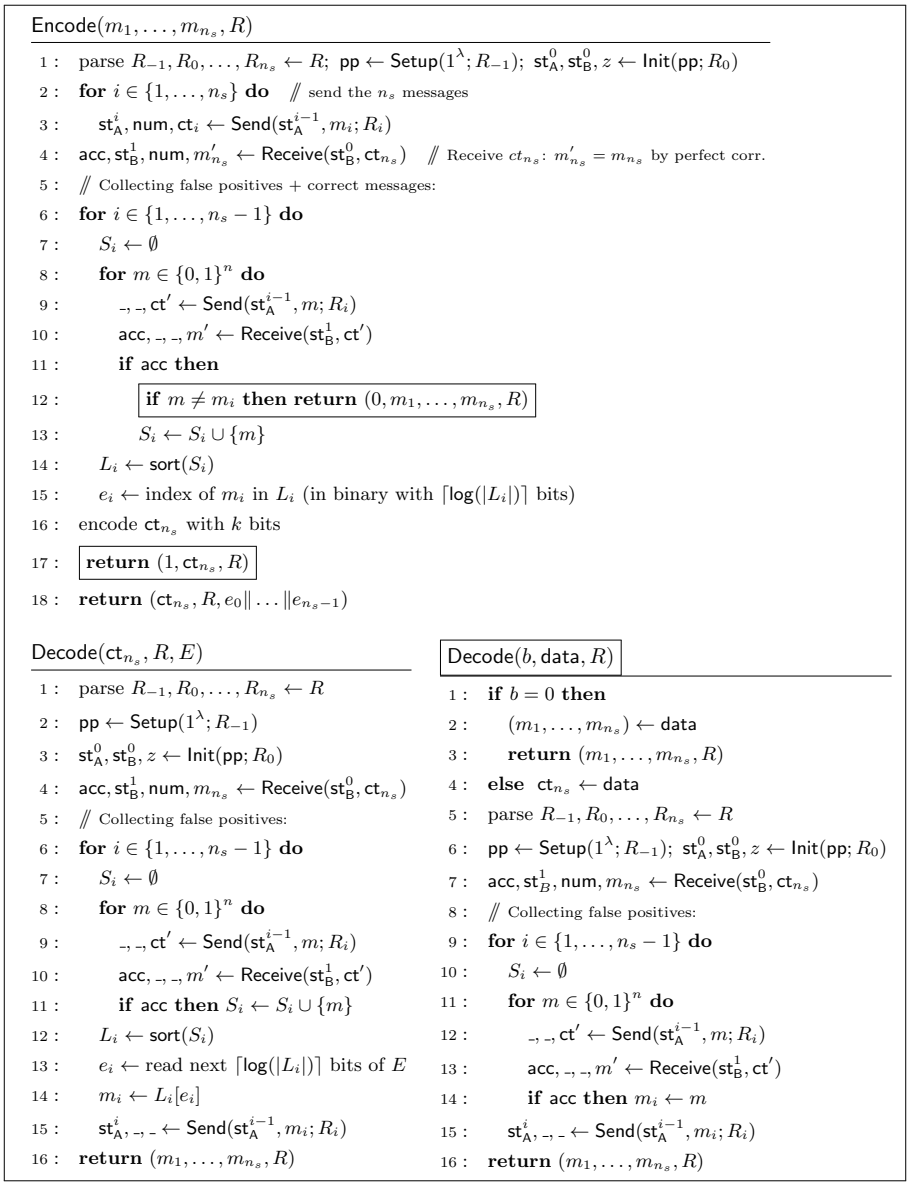


Figure 11: Encoder without (resp. with) boxed instructions and decoder on the left (resp. right) for proving the first 2 (resp. third) lower bound(s) in Theorem 5.

Lemma 4. Let k, n, n_s, γ as in the statement of the theorem. Then,

$$k \geq nn_s - \frac{1}{1 - \frac{2^n n_s}{2^\gamma}} \ .$$

Proof. In order to prove this lemma, we build another encoder/decoder pair very similar to the previous one. They are shown in Figure 11 (*with* the boxed instructions for the encoder and the boxed decoder on the right). The only difference in the encoder is that if *one* false positive is found, the encoder outputs a bit set to zero and the trivial encoding of the input. Let's call this event fail. If fail does not occur, a bit set to 1, the last ciphertext and the randomness are output.

In the decoder, either the first bit of the input is set to 0 and the input is returned straightaway, or $b = 1$ and the decoder proceeds as before. However, as there are no false positives, the m_i can be recovered without using the indices e_i (i.e. the correct message would be the only element of S_i). Overall, the expected codeword length is $\mathbb{E}[C] = 1+\alpha nn_s+(1-\alpha)k+r$, where $\alpha := \Pr[\mathsf{fail}]$, as if fail occurs a trivial encoding (on $1+nn_s+r$ bits) is used, and otherwise the encoder outputs $1 + k + r$ bits. By Shannon source coding theorem, we obtain

$$1 + \alpha nn_s + (1 - \alpha)k + r \geq nn_s + r \iff k \geq nn_s - \frac{1}{1-\alpha}$$

In addition, we have $\alpha := \Pr[\mathsf{fail}] = \Pr[\cup_{i=1}^{n_s-1}\{|F_i| \geq 1\}] \leq \sum_{i=1}^{n_s-1} \Pr[|F_i| \geq 1]$ as fail occurs if at least one of the sets of false positives F_i contains an element. Then, we have $\Pr[|F_i| \geq 1] \leq \mathbb{E}[|F_i|] \leq 2^{n-\gamma}$, where the first inequality follows from Markov's inequality and the second from Lemma 3. Overall, we get $\alpha \leq \frac{n_s 2^n}{2^\gamma}$. Hence,

$$k \geq nn_s - \frac{1}{1-\alpha} \geq nn_s - \frac{1}{1-\frac{2^n n_s}{2^\gamma}} \ .$$

Finally, some algebra shows that this bound is tighter than the second one when

$$2^\gamma \geq 2^n n_s \frac{2-2n_s}{3-2n_s} \ ,$$

that is, when γ is larger than $\approx n + \mathsf{log}(n_s)$. □

□

On non-perfect correctness. For simplicity in the proof, we only considered RC schemes which are perfectly correct. Note, however, that it should be possible to obtain a slightly worse bound for RC schemes that are *not* perfectly (computationally or statistically) correct. In more detail, perfect correctness is used twice in the proof of Theorem 5: 1. in the encoder to argue that the encoded messages will decrypt properly and 2. in the reduction to argue that $m \neq m_i \Rightarrow \mathsf{ct} \neq \mathsf{ct}_i$. Then, if the probability that a correctness error arises is at most δ, we can argue as follows. In case 1., we can simply use the trick of the 3rd bound (i.e. output the trivial encoding if the encoded message does not decrypt properly) to get the same bounds $-1/(1 - \delta \cdot n_s)$. Then, in case 2., we will have $\Pr[\mathsf{win}] > 2^{-\gamma} - \delta$ as we need to take into account the probability that m triggers a correctness error. This might incur an additional $\approx -\mathsf{log}(\delta)$ loss in the bound. Overall the proof still holds with $\delta > 0$, and if it is small then the bounds remain nearly identical.

6.2 Communication cost of r-UNF ARC

We consider a perfectly correct ARC (i.e. the function $\mathsf{Send}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}, \cdot; \mathsf{r})$ is injective for all randomness r, valid states $\mathsf{st}_{\mathcal{P}}$ and associative data ad). The following theorem states that the tag size of a secure ARC grows linearly in the number of messages (times either the security parameter or message size).

Theorem 6. Let Π be a perfectly correct ARC, n_s and λ be fixed, and T_{λ,n_s} be the time complexity of the (efficient) adversary given on the right of Figure 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: $\Pr[\mathsf{r\text{-}UNF}^{\mathcal{A}}_{\Pi}(1^\lambda) \Rightarrow 1] \leq \frac{1}{2^\gamma}$. Let $\mathcal{M} = \{0,1\}^n$ and $\mathcal{T} = \{0,1\}^k$ be the plaintext and tag space associated to Π, respectively. Then, $k \geq n_s(\gamma - 2)$, if $\gamma \leq n$, and $k \geq n_s(n-2)$, if $\gamma > n$. A third lower bounds gives

$$k \geq nn_s - \frac{1}{1 - \frac{2^n(n_s+1)}{2^\gamma}},$$

which is tighter for low values of n (e.g. $n = 1, 2$) and when $\gamma > n + \mathsf{log}(n_s)$.

The proof is nearly identical to the one of Theorem 5; we present the differences in Appendix D.

7 Performance and security trade-offs

In Section 6 we showed that r-RID and r-UNF impose a linear communication complexity for RC and ARC schemes. In this section we explore ways to bypass these lower bounds and propose practical protocols for active attack detection. We first argue that s-RID/s-UNF security can be achieved at a much lower cost than the counterparts r-RID/r-UNF. Motivating this analysis, we propose a lightweight three-move protocol that authenticates communication in both direction over the out-of-band channels. Noting that ciphertexts size is unbounded in the schemes that we presented up to this point, we then propose an optimised scheme for UNF security that prunes unnecessary messages included in authentication tags.

7.1 On the practicality of s-RID and s-UNF security

Security notion	r-RID	s-RID
Overhead	$\mathcal{O}(n\lambda + c_n)$	$\mathcal{O}(\lambda + c_n)$
Optimized overhead (Section 7.2)	N/A	$\mathcal{O}(\lambda + c_{n_{\mathsf{fresh}}})$

Table 1: Overhead induced by the two RID security notions. We assume that n messages are received and c_i is the space needed to encode i ordinals. The variable n_{fresh} refers to the number of messages received in the last two epochs.

We focus here on s-RID security, but similar arguments hold for s-UNF security. The RRC scheme in Fig. 6 achieves s-RID security by sending to the counter part the list of received nums and an *hash* of the R set. Informally, this suffices for security because a party can immediately detect when their counterpart has received a forgery (by keeping in state their sent messages and recomputing the hash). Table 1 summarizes the overheads of the two notions together with the optimization presented in Section 7.2, where we show that it is enough to only send information about messages received during the two last epochs. This significantly reduces the overhead for the scenarios in which the communication is "balanced".

One can reduce ciphertext size further by optimising for the "good case" scenario where messages are delivered in-order; in this case, ordinals can be encoded in ranges. For epochs with no lost messages, it suffices to encode only the last index. In any case, as the size of a single message in today's secure messaging applications can be several kilobytes, the overhead that s-RID imposes seems reasonable. We leave a deeper and concrete analysis to determine the impact of RID/UNF security in practice to future work.

In Fig. 6, the entire set of received messages is hashed (using a regular hash function) every time a message is sent by $\mathcal{P}$. Consequently, when $\mathcal{P}$ receives a new message, the entire hash must be re-computed when $\mathcal{P}$ sends their next message. To avoid this, the scheme can use an *incremental hash function* (Definition 9) such that, when a message is received, an efficient operation only depending on the new message and the previous digest can be executed to derive the new digest. Hash digests can be as small as a group element [13]. This enables parties to prune their set of sent/received messages in state. For example, if $\mathcal{P}$ receives a message m claiming that $\overline{\mathcal{P}}$ has received the first k messages from $\mathcal{P}$, and $\mathcal{P}$ has received messages for all possible ordinals that precede the ordinal of m, then $\mathcal{P}$ can safely store just the incrementally-hashed value corresponding to the first k messages, since $\overline{\mathcal{P}}$ can no longer claim to have only received a strict subset of the k messages.

Remark 9. The RID-secure RC of Fig. 6 sends the set of *received* ordinals for authentication (line 2 of the RRC.Send algorithm). Since ordinals are elements of a set on which a total order is defined, a simple optimization—that could reduce the overhead by up to 50%—consists in sending the smallest set among the set of received ordinals *or* the set of not received ordinals alongside a bit indicating which type of set has been sent. This optimization applies to all schemes that send sets of ordinals.

7.2 Epoch-based optimisation for s-RID security

In this section we show how to design an optimized s-RID-secure RC scheme given a correct and ORDINALS-secure RC scheme. A formal description of the optimized s-RID-secure RC scheme is given in Fig. 13. We start with a high-level description of the optimization.

The parties start at epoch $= 0$ and epoch $= 1$, respectively. Each time a party sends a message they also attach their current epoch to the message.

Upon receiving a message, a party $\mathcal{P}$ with $\mathsf{epoch} = t$ checks whether the epoch attached to the message received from $\overline{\mathcal{P}}$ is at most $t+1$. If the $\overline{\mathcal{P}}$'s epoch is exactly $t+1$ the party $\mathcal{P}$ updates their epoch to $t+2$. The epoch value of the parties are always one apart at each time.

To achieve s-RID security the sender does the following. Whenever sending a message with $\mathsf{epoch} = t$, the sender attaches the num and the accumulated hash of all messages they received during the time their epoch was t and $t-2$. Meaning, the parties do the same as the original s-RID construction, but only for the messages they have received in the last 2 epochs.

We continue by stating the main theorem of this section and providing a proof sketch.

Theorem 7. Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr}) collision resistant hash function. Then s-RID-RC (defined in Fig. 13) is a (q, t, ϵ_{cr})-s-RID-secure RC where $t_{cr} \approx t$ and q is upper bounded by t.

Proof sketch. Let $[(\mathsf{ct}_f, \cdot, t_f), (\mathsf{ct}_h, \cdot, t_h)]$, be the closest pair of sent-received messages contradicting the s-RID condition. Meaning $(\mathsf{ct}_f, \cdot, t_f)$ is a forgery received by $\overline{\mathcal{P}}$ before they sent the honest message $(\mathsf{ct}_h, \cdot, t_h)$ which was received by $\mathcal{P}$. We consider the time when $(\mathsf{ct}_h, \cdot, t_h)$ was sent by $\overline{\mathcal{P}}$. As mandated by the construction $(\mathsf{ct}_h, \cdot, t_h)$ contained num values and accumulated hash of all messages $\overline{\mathcal{P}}$ has received at epochs t_h and $t_h - 2$. Following the same argument as the proof of Theorem 2 one can show that no forgeries, including $(\mathsf{ct}_f, \cdot, t_f)$, were received by $\overline{\mathcal{P}}$ while $\mathsf{epoch}_{\overline{\mathcal{P}}} \in \{t_h, t_h - 2\}$.

The two messages that changes $\mathsf{epoch}_{\overline{\mathcal{P}}}$ from $t_h - 4$ to $t_h - 2$ and from $t_h - 2$ to t_h, were honest messages by $\mathcal{P}$. Let us call them $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ and $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$ respectively. Note that, both these messages were received after $(\mathsf{ct}_f, \cdot, t_f)$ was received and before $(\mathsf{ct}_f, \cdot, t_f)$ was sent, as otherwise $(\mathsf{ct}_h, \cdot, t_h)$ would contradict $(\mathsf{ct}_f, \cdot, t_f)$. Note that between sending the messages $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ and $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$, $\mathsf{epoch}_{\mathcal{P}}$ was changed meaning $\mathcal{P}$ received a message with $\mathsf{epoch} = t_h - 2$ that caused the change of $\mathsf{epoch}_{\mathcal{P}}$. Let us call this message $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$. We argue this message should have been forged.

Let us assume by contradiction that this message was honest. Note that $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ was sent after $(\mathsf{ct}_{t_h-3}, \cdot, t_h - 3)$ was received, hence after $(\mathsf{ct}_f, \cdot, t_f)$ was received. Now if $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ is honest it forms a pair with $(\mathsf{ct}_f, \cdot, t_f)$ which violates the s-RID condition and has less distance from the original pair which is a contradiction. So $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ must have been forged. A visualisation of the scenario can be found in Fig. 12.

One other observation is that, $(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2)$ was received before $(\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)$ was sent, hence, before $(\mathsf{ct}_h, \cdot, t_h)$ was received. This shows that the pair $[(\mathsf{ct}_{t_h-2}, \cdot, t_h - 2), (\mathsf{ct}_{t_h-1}, \cdot, t_h - 1)]$ also violates the s-RID (for $\overline{\mathcal{P}}$ and not $\mathcal{P}$) and has less distance than the original pair.

□

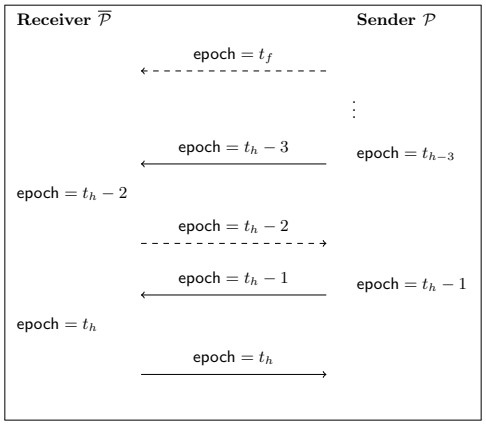


Figure 12: Visualising the proof sketch of Theorem 7. The dotted messages are forged and the others are honest messages. Intuitively we argue if the last message did not contradict the first message, the fourth message would have contradicted the third, therefore violating s-RID in the other direction.

7.3 Lightweight bidirectional authentication

We propose a three-move bidirectional authentication protocol. Fig. 14 describes the protocol at a high level. Parties include in the authentication tag only the set of *received* messages. The receiver of the tag compares then the set of received messages from the counterpart with the set of sent messages. We envision this approach to be used when participants meet in person or online and can both authenticate the respective views of the conversation at the same time. This is already required in Signal—with the verification of safety numbers [29]—and other messaging solutions. We defer our formal investigation of the scheme to Appendix E.1.

7.4 Reducing bandwidth for UNF security

In Figure 15, we present a scheme that optimises bandwidth consumption for UNF security. A complete description and security proof are given in Appendix E.2. Our scheme takes advantage of the fact that messages sent out-of-band cannot be forged. Suppose that $\mathcal{P}$ sends an authentication tag to $\overline{\mathcal{P}}$, then $\overline{\mathcal{P}}$ acknowledge the reception of the tag to $\mathcal{P}$. At this point, $\mathcal{P}$ no longer needs to send the information that $\overline{\mathcal{P}}$ has already obtained. As usual, our scheme supports out-of-order communication even on the authenticated channels. This approach is complicated by the fact that parties have to keep track of, e.g., which tags their partner has received to determine what is safe to prune from state (in S_{at}-Seen), which incurs relatively small overhead in typical executions.

The following theorem states that pruning preserves security.

Theorem 8. Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 7). Then the ARC-OP scheme (Fig. 15) is correct, ORDINALS secure, and (q, t, ϵ_{cr})-

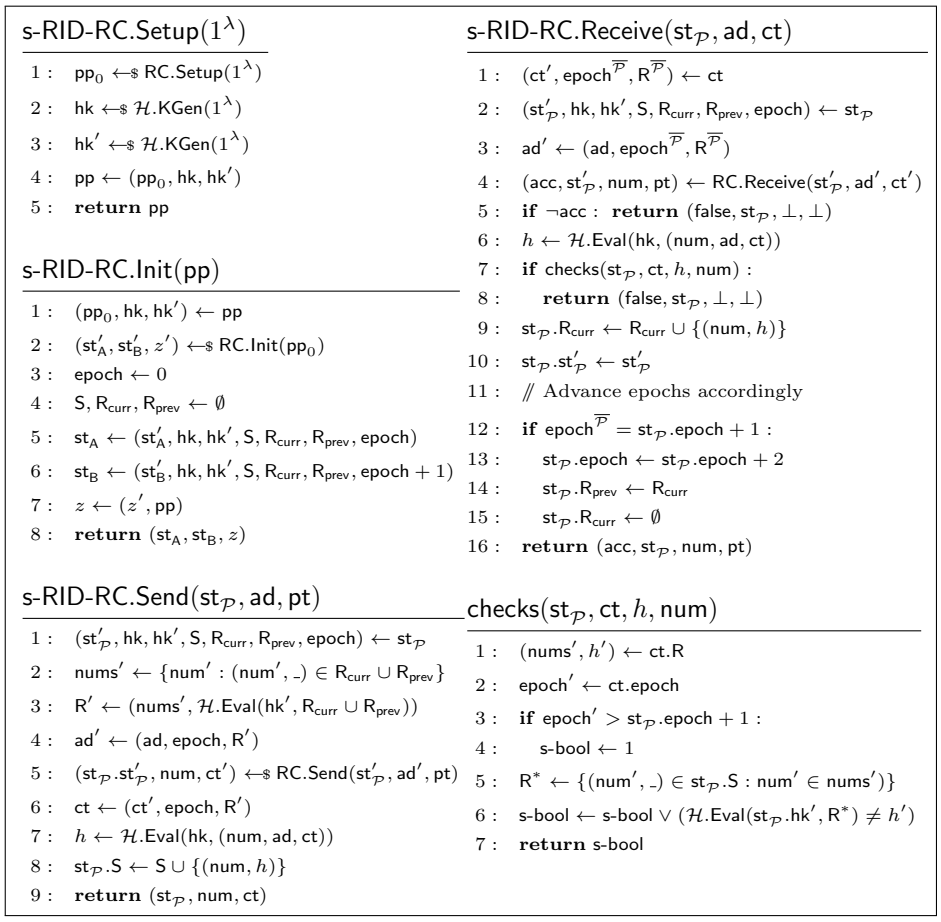


Figure 13: Optimized s-RID-secure RC scheme given a correct and ORDINALS-secure RC scheme.

UNF secure, where $t \approx t_{cr}$.

The collision resistance of the hash function implies UNF-security and this is not affected by the pruning operations. The complete proof is given in Appendix E.2.

8 Conclusion

This work considers active attack detection for secure messaging with immediate decryption, including its its inherent performance limitations and how to overcome them. We conclude with some avenues for future work: 1) Analyze the *practical* overhead of in- and out-of-band authentication; and 2) Define RID and UNF notions and corresponding constructions in the group setting.

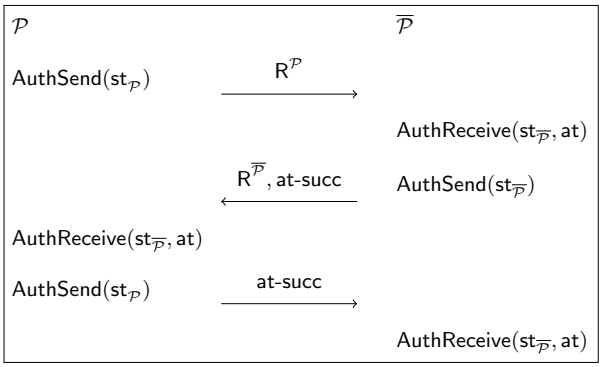


Figure 14: Description of the three-move authentication procedure. The boolean at-succ indicates whether the counterpart's set of received messages is a subset of the local set of sent messages.

Acknowledgements. Khashayar Barooti and Loïs Huguenin-Dumittan are supported by a grant (project no. 192364) of the Swiss National Science Foundation (SNSF). We thank Olivier Becker and Nathan Duchesne for pointing out bugs in our constructions and proofs, and the anonymous reviewers of this paper for their feedback, notably the reviewer who proposed the epoch-based s-RID optimisation.

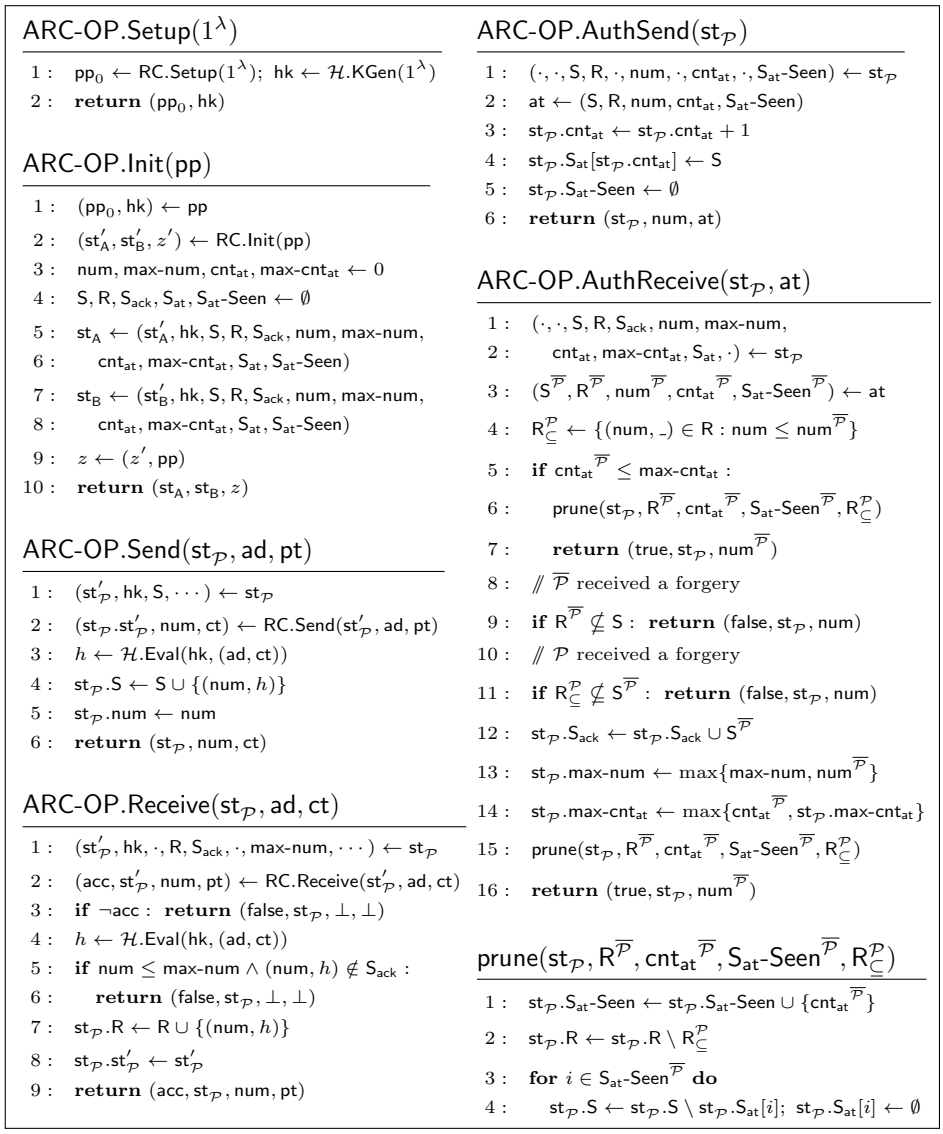


Figure 15: Optimised UNF-secure ARC scheme ARC-OP based on a RC scheme RC (Definition 1). The sets S, R and S_ack are as in Fig. 6. The variable max-num represents the largest num received in an at. The counters cnt_at and $\mathsf{max\text{-}cnt}_\mathsf{at}$ keep track of how many at have been sent and largest cnt_at received in an at, respectively. S_at-Seen is the list of cnt_at of received at since the last sent one; $\mathsf{S}_\mathsf{at}[i]$ contains the content of S sent in the ith at.

References

[1] Alwen, J., Coretti, S., Dodis, Y.: The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. In: EUROCRYPT (2019)

[2] Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Modular Design of Secure Group Messaging Protocols and the Security of MLS. In: CCS (2021)

[3] Alwen, J., Coretti, S., Jost, D., Mularczyk, M.: Continuous group key agreement with active security. In: TCC (2020)

[4] Alwen, J., Jost, D., Mularczyk, M.: On the insider security of mls. In: CRYPTO (2022)

[5] Balfanz, D., Smetters, D.K., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: NDSS (2002)

[6] Balli, F., Rösler, P., Vaudenay, S.: Determining the core primitive for optimally secure ratcheting. In: ASIACRYPT (2020)

[7] Bellare, M., Singh, A.C., Jaeger, J., Nyayapati, M., Stepanovs, I.: Ratcheted Encryption and Key Exchange: The Security of Messaging. In: CRYPTO (2017)

[8] Bienstock, A., Dodis, Y., Garg, S., Grogan, G., Hajiabadi, M., Rösler, P.: On the Worst-Case Inefficiency of CGKA. In: TCC (2022)

[9] Bienstock, A., Dodis, Y., Rösler, P.: On the price of concurrency in group ratcheting protocols. In: TCC (2020)

[10] Bienstock, A., Fairoze, J., Garg, S., Mukherjee, P., Raghuraman, S.: A More Complete Analysis of the Signal Double Ratchet Algorithm. In: CRYPTO (2022)

[11] Caforio, A., Durak, F.B., Vaudenay, S.: Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness. In: PKC (2021)

[12] Civit, P., Gilbert, S., Gramoli, V., Guerraoui, R., Komatovic, J., Milosevic, Z., Serendinschi, A.: Crime and punishment in distributed byzantine decision tasks. Cryptology ePrint Archive (2022)

[13] Clarke, D.E., Devadas, S., van Dijk, M., Gassend, B., Suh, G.E.: Incremental Multiset Hash Functions and Their Application to Memory Integrity Checking. In: ASIACRYPT (2003)

[14] Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: EuroS&P (2017)

[15] Cohn-Gordon, K., Cremers, C., Garratt, L.: On Post-compromise Security. In: CSF (2016)

[16] Cremers, C., Zhao, M.: Provably post-quantum secure messaging with strong compromise resilience and immediate decryption. IACR Cryptol. ePrint Arch. (2022)

[17] Dowling, B., Günther, F., Poirrier, A.: Continuous authentication in secure messaging. In: ESORICS (2022)

[18] Dowling, B., Hale, B.: There Can Be No Compromise: The Necessity of Ratcheted Authentication in Secure Messaging. IACR Cryptol. ePrint Arch. p. 541 (2020)

[19] Dowling, B., Hale, B.: Secure Messaging Authentication against Active Man-in-the-Middle Attacks. In: EuroS&P (2021)

[20] Dowling, B., Hauck, E., Riepel, D., Rösler, P.: Strongly anonymous ratcheted key exchange. In: ASIACRYPT (2022)

[21] Durak, F.B., Vaudenay, S.: Bidirectional Asynchronous Ratcheted Key Agreement with Linear Complexity. In: IWSEC (2019)

[22] Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: S&P (2015)

[23] Haeberlen, A., Kouznetsov, P., Druschel, P.: Peerreview: Practical accountability for distributed systems. SIGOPS **41**(6), 175–188 (2007)

[24] Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: CRYPTO (2014)

[25] Jacob, R., Larsen, K.G., Nielsen, J.B.: Lower bounds for oblivious data structures. In: SODA (2019)

[26] Jaeger, J., Stepanovs, I.: Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging. In: CRYPTO (2018)

[27] Jost, D., Maurer, U., Mularczyk, M.: Efficient Ratcheting: Almost-Optimal Guarantees for Secure Messaging. In: EUROCRYPT (2019)

[28] Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious ram lower bound! In: CRYPTO (2018)

[29] Marlinspike, M.: Safety number updates. `https://signal.org/blog/verified-safety-number-updates/` (2017), accessed: 22-05-2022

[30] Naor, M., Rotem, L., Segev, G.: Out-Of-Band Authenticated Group Key Exchange: From Strong Authentication to Immediate Key Delivery. In: ITC (2020)

[31] Pasini, S., Vaudenay, S.: An optimal non-interactive message authentication protocol. In: CT-RSA (2006)

[32] Pijnenburg, J., Poettering, B.: On Secure Ratcheting with Immediate Decryption. In: ASIACRYPT (2022)

[33] Poettering, B., Rösler, P.: Asynchronous ratcheted key exchange. IACR Cryptol. ePrint Arch. p. 296 (2018)

[34] Poettering, B., Rösler, P.: Towards Bidirectional Ratcheted Key Exchange. In: CRYPTO (2018)

[35] Scott-Railton, J., Campo, E., Marczak, B., Razzak, B.A., Anstis, S., Böcü, G., Solimano, S., Deibert, R.: CatalanGate: Extensive Mercenary Spyware Operation against Catalans Using Pegasus and Candiru. `https://citizenlab.ca/2022/04/catalangate-extensive-mercenary-spyware-operation-against-catalans-using-pegasus-candiru/` (2022), accessed: 22-05-2022

[36] Stäuble, M.: Data structures for puncturable encryption, bachelor thesis (2021)

[37] Support, S.: Twilio Incident: What Signal Users Need to Know. `https://support.signal.org/hc/en-us/articles/4850133017242` (2022), accessed: 03-10-2022

A Primitives

A.1 Hash function

Definition 7 (Hash function)**.** A hash function $\mathcal{H}$ consists of PPT algorithms Gen and Eval such that:

- $\mathsf{Gen}(1^\lambda) \to \mathsf{hk}$ takes a unary string 1^λ and outputs hash key hk.
- $\mathsf{Eval}(\mathsf{hk}, \mathsf{pt}) \to h$ inputs a hash key hk and a message $\mathsf{pt} \in \{0,1\}^*$ and outputs digest h.

Game $\mathsf{CR}^{\mathcal{A}}_{\mathcal{H}}(1^\lambda)$

1 : $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\lambda)$
2 : $m_1, m_2 \leftarrow \mathcal{A}(\mathsf{hk})$
3 : **if** $\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, m_1) = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, m_2) \wedge m_1 \neq m_2$ **then return** 1
4 : **return** 0

Figure 16: Collision resistance of a hash function $\mathcal{H}$.

Definition 8 (Collision resistance)**.** We say that a hash function $\mathcal{H}$ is (t, ϵ)-collision resistant if, for all adversaries $\mathcal{A}$ running in time at most t, we have:

$$\Pr[\mathsf{CR}^{\mathcal{A}}_{\mathcal{H}}(1^\lambda) \Rightarrow 1] \leq \epsilon,$$

where game $\mathsf{CR}^{\mathcal{A}}_{\mathcal{H}}$ is defined in Fig. 16.

A.2 Incremental hash function

Clarke et al. [13] define incremental *multiset* hash functions and multiset collision resistance. That is, a hash function which takes a set of elements as input, where the digest can be updated with an operation with complexity proportional to the number of elements added/removed. For our purposes, it suffices to consider an *incremental set hash function* and *set collision resistance*.

Definition 9 (Incremental set hash function)**.** An incremental set hash function $\mathcal{H}$ consists of the following PPT algorithms:

- $\mathsf{IncGen}(1^\lambda) \xrightarrow{\$} \mathsf{hk}$: This probabilistic algorithm takes a unary string 1^λ and outputs hash key hk.
- $\mathsf{IncEval}(\mathsf{hk}, S = \{m_1, \ldots, m_k\}) \to h_S$: This deterministic algorithm takes a hash key hk and set S and outputs digest h_S.
- $\mathsf{IncEval}(\mathsf{hk}, h, S_h = \{m'_1, \ldots, m'_{k'}\}, S = \{m_1, , \ldots, m_k\}) \to h_{S \cup S_h}$: This deterministic algorithm takes a hash key hk, digest h, set S_h (associated with h) and set S and outputs a digest $h_{S_h \cup S}$.

Game $\mathsf{SCR}^{\mathcal{A}}_{\mathcal{H}}(1^\lambda)$

$\mathsf{hk} \leftarrow\!\!\$\ \mathcal{H}.\mathsf{IncGen}(1^\lambda)$

$(S_1 = \{m_i\}_i, S_2 = \{m'_j\}_j) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{hk})$

if $\mathcal{H}.\mathsf{IncEval}(\mathsf{hk}, S_1) = \mathcal{H}.\mathsf{IncEval}(\mathsf{hk}, S_2) \wedge (S_1 \neq S_2)$:

  return 1

return 0

Figure 17: Set collision resistance of an incremental set hash function $\mathcal{H}$.

Definition 10 (Set collision resistance)**.** A family of incremental set hash function $\mathcal{H}$ is (t, ϵ)-set collision resistant, if for any adversary $\mathcal{A}$ running in at most time t the following holds

$$\Pr[\mathsf{SCR}^{\mathcal{A}}_{\mathcal{H}}(1^\lambda) \Rightarrow 1] \leq \epsilon$$

We refer the reader to Clarke et al. [13] for more details about incremental set hash functions.

B Proofs for Theorem 1 and Theorem 2

In this section we provide the omitted proofs and details in Section 4.1. We start by proving Theorem 1.

B.1 Proof for Theorem 1

Theorem 1. Let us assume there exists an adversary $\tilde{\mathcal{A}}$ playing the r-RID game, running in time $\tilde{t}$ and making at most $\tilde{q}$ queries. Let us call the advantage of this adversary $\tilde{\epsilon}$, hence we have

$$\Pr[\mathsf{r\text{-}RID}^{\tilde{\mathcal{A}}}_{\mathsf{RRC}}(1^\lambda) \Rightarrow 1] = \tilde{\epsilon}.$$

Let E be an event that occurs when $\mathsf{r\text{-}RID}_{\mathsf{RRC}}(\tilde{\mathcal{A}})$ outputs 1 The proof strategy is to construct an adversary $\mathcal{A}^*$, running in time $\approx \tilde{t}$ such that

$$\Pr[\mathsf{CR}^{\mathcal{A}^*}_{\mathcal{H}}(1^\lambda) \Rightarrow 1 \mid E] = 1.$$

By definition of r-RID, E occurring means that there exist $\mathcal{P}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}), (\mathsf{num}', \mathsf{ad}', \mathsf{ct}'), x$ such that $\mathsf{bad\text{-}\overline{P}}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ and $\mathsf{forgery}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$ are both true. This means that the message with ordinal num was not sent by $\mathcal{P}$ but received at some point by $\overline{\mathcal{P}}$ ($\mathsf{log}[x] = (\text{“}\mathsf{rec}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct})$), and the message with ordinal num' was also received and was actually sent by $\mathcal{P}$. Moreover, $\mathsf{num} < \mathsf{num}'$.

We separate the two cases where 1) the message with ordinal num (the forged message) is received before the message with ordinal num' (the honest

message), and 2) the message with ordinal num' is received first. We first analyse the former case.

We argue that unless the adversary found a collision, the message with ordinal num' (the honest message) would not have been delivered. Suppose that the honest message *was* delivered. Let $\mathsf{st}_{\mathcal{P}}$ be the state of the receiver $\mathcal{P}$ while receiving message num', and $\mathsf{st}_{\overline{\mathcal{P}}}$ be the state of the sender $\overline{\mathcal{P}}$ while sending the message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. As $(\text{“rec”}, \mathcal{P}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}') \in \mathsf{log}$, it means $\mathsf{RRC.Receive}(\mathsf{st}_{\mathcal{P}}, \mathsf{ad}', \mathsf{ct}') \rightarrow (\mathsf{true}, \mathsf{st}'_{\mathcal{P}}, \mathsf{num}', \mathsf{pt}')$, which implies $\mathsf{checks}(\mathsf{st}_{\mathcal{P}}, \mathsf{ct}', \mathsf{num}', \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}', \mathsf{ad}', \mathsf{ct}'))$ returned false.

Note that as $\mathsf{num} \leq \mathsf{num}'$, we have

$$(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))) \in \mathsf{R}' \text{ and } (\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))) \in \mathsf{ct}'.\mathsf{S},$$

as otherwise $\mathsf{r\text{-}bool}$ would have been set to true in line 5. Let $h_f := \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$. As $(\mathsf{num}, h_f) \in \mathsf{ct}'.\mathsf{S}$, we have

$$(\mathsf{num}, h_f) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S} \tag{1}$$

as ct' was sent by $\overline{\mathcal{P}}$. This would mean that $\overline{\mathcal{P}}$ did send a message with ordinal num, let us call it $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$. Hence, we have that,

$$(\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}_h, \mathsf{ct}_h, \mathsf{num})), \mathsf{num}) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S} \tag{2}$$

By combining (1) and (2) and the fact that num can appear only once in $\mathsf{st}_{\overline{\mathcal{P}}}$ (due to the $\mathsf{ORDINALS}$ security of RRC), we get that $\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$ which gives a collision. This is because $(\mathsf{ad}_h, \mathsf{ct}_h) \neq (\mathsf{ad}, \mathsf{ct})$ as $(\text{“send”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \mathsf{log}$ and $(\text{“send”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \mathsf{log}$.

Now we discuss the case where $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ is received before $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$. As $\mathsf{num} \leq \mathsf{num}'$, this would mean that while receiving $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$, $\mathsf{max\text{-}num} \geq \mathsf{num}' \geq \mathsf{num}$. This would mean $(\mathsf{num}, h) \in \mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{ack}}$, otherwise the condition on line 9 would have not been satisfied. As $\mathsf{S}_{\mathsf{ack}}$ is only updated by adding the elements in $\mathsf{S}^{\overline{\mathcal{P}}}$ when a message is received, and as (num, h_f) is not in $\mathsf{ct}^*.\mathsf{S}$, for any honest ct^*, there should exist a forged message $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ received before $(\mathsf{ad}, \mathsf{ct}, \mathsf{num})$ such that $(\mathsf{num}, h_f,) \in \mathsf{ct}''.\mathsf{S}$. As we considered $(\mathsf{num}, \mathsf{num}')$ to be the first pair of messages violating the $\mathsf{r\text{-}RID}$ property, we know that $\mathsf{num}'' > \mathsf{num}' > \mathsf{num}$.

We split the two cases where $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is received before $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ and the case where it is received after $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. Let us consider the first case. We argue in this case $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ (the honest message) would not be accepted. As num'' is received before num', $\mathsf{num}' < \mathsf{num}'' \leq \mathsf{max\text{-}num}$. And as $\mathsf{r\text{-}bool} = \mathsf{false}$, $\mathsf{S}'_{\mathsf{ack}} \subseteq \mathsf{ct}'.\mathsf{S}$ (line 12). However $(\mathsf{num}, h_f) \in \mathsf{S}'_{\mathsf{ack}}$, as it was in $\mathsf{ct}''.\mathsf{S}$, hence it should also be in $\mathsf{ct}'.\mathsf{S}$, which would mean $h_f = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, \mathsf{ad}_h, \mathsf{ct}_h, \mathsf{num})$ which is again a collision.

Now let us consider the case where $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is received after the message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. We argue that $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ should not have been accepted. We split the cases where $\mathsf{num}'' \geq \mathsf{max\text{-}num}$ and $\mathsf{num}'' < \mathsf{max\text{-}num}$. Let us consider the later first. As $(\mathsf{ad}'', \mathsf{ct}'', \mathsf{num}'')$ was accepted, and hence

$\mathsf{r\text{-}bool} = \mathsf{false}$, $\mathsf{ct}''.\mathsf{S} \subset \mathsf{S_{ack}}$ (line 10). Now $(\mathsf{num}, h_f) \in \mathsf{ct}''.\mathsf{S}$, so $(\mathsf{num}, h_f) \in \mathsf{S_{ack}}$. As without loss of generality we can imagine $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ being the first message vouching for (h_f, num), this would mean (num, h_f) was added to $\mathsf{S_{ack}}$ by an honest message, i.e. $h_f = h_h$ which leads to a collision again.

Finally for the case in which $\mathsf{num}'' \geq \mathsf{max\text{-}num}$, again, considering that $(\mathsf{num}'', \mathsf{ad}'', \mathsf{ct}'')$ is the first message vouching for (h_f, num), we have $(\mathsf{num}, h_f) \in \mathsf{S_{ack}} \setminus \mathsf{ct}''.\mathsf{S}$ (and so $\mathsf{r\text{-}bool}$ would be set to true) unless $h_f = h_h$. Moreover, at this point $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ has already been received so, $\mathsf{max\text{-}num} \geq \mathsf{num}' > \mathsf{num}$. Hence, unless $h_f = h_h$, $\mathsf{r\text{-}bool}$ would be set to true in line 7. This concludes the proof that $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$, $(\mathsf{num}, \mathsf{ad}, \mathsf{ct})$ are accepted if and only if $\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$.

Now we describe the CR adversary $\mathcal{A}^*$. $\mathcal{A}^*$ runs the initialisation of RRC by replacing the sampling step of hk with the hk given by the $\mathsf{CR}_{\mathcal{H}}$ game, then runs $\tilde{\mathcal{A}}$ as a subroutine, and computes $(\text{“}\mathsf{rec}\text{”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}_f, \mathsf{ct}_f) \in \mathsf{log}$, and $(\text{“}\mathsf{send}\text{”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \mathsf{log}$ such that $(\mathsf{ad}_f, \mathsf{ct}_f) \neq (\mathsf{ad}_h, \mathsf{ct}_h)$ and $h_f = h_h$ if possible. Given E, this pair always exists as we have $\Pr[\mathsf{CR}_{\mathcal{H}}(\mathcal{A}^*) \Rightarrow 1 \mid E] = 1$. Moreover, as $\mathcal{A}^*$ is just running $\tilde{\mathcal{A}}$ as a subroutine and not doing anything extra, the time it runs is also $\approx \tilde{t}$. Finally, we have

$$\begin{aligned}\Pr[\mathsf{CR}_{\mathcal{H}}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1] &\geq \Pr[\mathsf{CR}_{\mathcal{H}}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] \cdot \Pr[E] \\ &= \Pr[\mathsf{r\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1] = \tilde{\epsilon}\ . \end{aligned} \tag{3}$$

Hence, $\tilde{\epsilon} \leq \Pr[\mathsf{CR}_{\mathcal{H}}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1]$. This means that if $\mathcal{H}$ is (t_{cr}, ϵ_{cr})-collision resistant, then $\mathsf{PRC_{rs}}$ is (q, t, ϵ_{cr})-$\mathsf{r\text{-}RID}$ secure with $t_{cr} \approx t$ which concludes the proof. □

B.2 Proof for Theorem 2

We proceed by presenting the proof for $\mathsf{s\text{-}RID}$ security of the construction introduced in Figure 6.

Theorem 2. The proof strategy is identical to the one done for proof of Theorem 1. For any adversary $\tilde{\mathcal{A}}$ playing the $\mathsf{s\text{-}RID}$ game, we construct an adversary $\mathcal{A}^*$ playing the CR game with comparable complexity. We first describe the adversary $\mathcal{A}^*$ in terms of $\tilde{\mathcal{A}}$ and proceed by proving that $\mathcal{A}^*$ wins at least as often as $\tilde{\mathcal{A}}$.

As with the previous proof we define an event E that occurs only when $\mathsf{s\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1$, and we prove that $\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] = 1$.

The event $\mathsf{s\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1$, means $\exists \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, \mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y$ such that both $x < y$, $(\mathsf{num}, \mathsf{pt}, \mathsf{ct})$ is a forged message received by $\overline{\mathcal{P}}$ at time x, $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ is an honest message sent by $\overline{\mathcal{P}}$ at time y and received by $\mathcal{P}$. As $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ was received, $\mathsf{RRC.checks}$ returned 0.

Let us define $h_f = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}, \mathsf{ct}))$. When receiving the forged message, i.e. at time x, $\overline{\mathcal{P}}$ adds (num, h_f) to $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. As $y > x$ at time y, (num, h_f) is still in $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. Hence $\mathsf{num} \in \mathsf{nums}'$ for the honest message $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$. Now as $(\mathsf{num}', \mathsf{ad}', \mathsf{ct}')$ was accepted, we have that due to line 3 of checks,

$$\mathcal{H}.\mathsf{Eval}(\mathsf{hk}', \mathsf{R}^*) = \mathcal{H}.\mathsf{Eval}(\mathsf{hk}', \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}) \ .$$

If $\mathsf{R}^* \neq \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$ we have already found a collision. So let us assume $\mathsf{R}^* = \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$. Now as $(\mathsf{num}, h_f) \in \mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{R}$, we also have that $(\mathsf{num}, h_f) \in \mathsf{R}^* \subset \mathsf{st}_{\mathcal{P}}.\mathsf{S}$.

This would mean that there exists an honest message $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$ such that $\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)) = h_f$. But note that as $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h)$ is an honest message, $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \in \mathsf{log}$ but $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \mathsf{log}$ as the message was forged, hence $(\mathsf{num}, \mathsf{ad}_h, \mathsf{ct}_h) \neq (\mathsf{num}, \mathsf{ad}, \mathsf{ct})$, which again yields a collision pair.

Now the CR adversary $\mathcal{A}^*$ does the following: they run the $\mathsf{s\text{-}RID}$ adversary $\tilde{\mathcal{A}}$ as a subroutine with the hk given by the CR game. They later find $\mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}$, $\mathsf{num}', \mathsf{ad}', \mathsf{ct}', x, y$ satisfying the condition, in case they exist. Now by exposing the states of the parties at time y they can find the collision pairs described above. Hence we have,

$$\Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1] \geq \Pr[\mathsf{CR}^{\mathcal{A}^*}(1^\lambda) \Rightarrow 1 | E] \cdot \Pr[E] = \Pr[\mathsf{s\text{-}RID}^{\tilde{\mathcal{A}}}(1^\lambda) \Rightarrow 1] \quad (4)$$

Moreover the run-time of $\mathcal{A}^*$ is roughly the run-time of $\tilde{\mathcal{A}}$. This concludes the proof. □

Remark 10. This proof can be trivially extended to the optimized version of the compiler with respect to set collision resistance.

C Proof for Theorem 4

In this section we provide the complete omitted proof for Theorem 4.

Proof. Assume an adversary $\mathcal{A}$ playing the UNF game (Fig. 7), which makes at most q oracle queries and runs in time at most t. We assume the advantage of $\mathcal{A}$ is ϵ, hence by Definition 6 we have $\Pr[\mathsf{UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] = \epsilon$. We construct an adversary $\mathcal{B}$, running in time approximately equal to t, which, running $\mathcal{A}$ as a subroutine, wins the collision resistance game for $\mathcal{H}$ (Definition 8), that is $\Pr[\mathsf{CR}_{\mathcal{H}}^{\mathcal{B}^{\mathcal{A}}}(1^\lambda) \Rightarrow 1] = 1$.

The UNF adversary $\mathcal{A}$ wins when one party accepts a forgery (predicate $\mathsf{forgery}$) and at least one of the two parties fails to detect the forgery (predicates $\mathsf{bad\text{-}P}$ and $\mathsf{bad\text{-}}\overline{\mathsf{P}}$). Suppose there exist $\mathcal{P}$, num, num', ad, ct, at, x, y such that $\mathsf{forgery}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x) = \mathsf{true}$. We analyze the predicates $\mathsf{bad\text{-}P}$ and $\mathsf{bad\text{-}}\overline{\mathsf{P}}$ separately, starting with the latter.

The $\mathsf{forgery}$ predicate states that $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \in \mathsf{log}$ and $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \mathsf{log}[x]$ for some $x \in \mathbb{N}$, where $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$. This means that $(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}', \mathsf{ct}')) \in \mathsf{S}^{\mathcal{P}}$ and $(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct})) \in \mathsf{R}^{\overline{\mathcal{P}}}$, otherwise the forgery is trivially detected because $(\mathsf{num}, \cdot) \notin \mathsf{R}^{\overline{\mathcal{P}}}$. Moreover,

by the $\mathsf{bad\text{-}}\overline{\mathsf{P}}$ predicate we know that $\mathsf{R}^{\overline{\mathcal{P}}}_{\subseteq} \subseteq \mathsf{S}^{\mathcal{P}}$ for any $\mathsf{num} \leq \mathsf{num}'$, which implies that $(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}', \mathsf{ct}'))) = (\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct})))$. By correctness, $(\mathsf{num}, \cdot)$ can appear only once in $\mathsf{S}^{\mathcal{P}}$, respectively in $\mathsf{R}^{\overline{\mathcal{P}}}$, and by assumption $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$, therefore we have a collision for $\mathcal{H}.\mathsf{Eval}(\mathsf{hk}, \cdot)$.

We now analyze the $\mathsf{bad\text{-}P}$ predicate. The $\mathsf{forgery}$ predicate states that $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \in \mathsf{log}$ and $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \mathsf{log}[x]$ for some $x \in \mathbb{N}$, where $(\mathsf{ad}', \mathsf{ct}') \neq (\mathsf{ad}, \mathsf{ct})$, otherwise the forgery is trivially detected. This implies that $(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct})) \in \mathsf{R}^{\overline{\mathcal{P}}}$ when $\mathsf{ARC.Receive}(\cdot, \mathsf{ad}, \mathsf{ct}) \rightarrow (\mathsf{true}, \cdot, \mathsf{num}, \cdot)$. By the $\mathsf{bad\text{-}P}$ predicate we know that $\overline{\mathcal{P}}$ sends an authentication tag at after accepting $(\mathsf{ad}, \mathsf{ct})$, since $(\text{“authsend”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}) = \mathsf{log}[y]$ and $y > x$, which means that $(\mathsf{num}, \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, (\mathsf{ad}, \mathsf{ct}))$ is in the $\mathsf{R}^{\overline{\mathcal{P}}}$ that at contains. The rest of the argument follows the same approach as the previous paragraph.

We give in Fig. 18 the adversary $\mathcal{B}$ which plays against the collision resistance game. $\mathcal{B}$ runs the $\mathsf{ARC.Setup}$ procedure and replaces the hash key hk' that the procedure returns with the hk that the adversary receives from the CR challenger. After running $\mathcal{A}$ as a subroutine, $\mathcal{B}$ analyzes the log array to find the $(\mathsf{ad}, \mathsf{ct})$, $(\mathsf{ad}', \mathsf{ct}')$ pairs that represents a forgery and returns those. If $\mathcal{A}$ wins the UNF game, then $\mathcal{B}$ wins the CR game almost surely, that is

$$\Pr[\mathsf{CR}^{\mathcal{B}}_{\mathcal{H}}(1^\lambda) \Rightarrow 1] \geq \Pr[\mathsf{UNF}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] = \epsilon.$$

Moreover, $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine and executes an additional negligible amount of work, so we have $t \approx t_{cr}$.

□

CR adversary $\mathcal{B}^{\mathcal{A}}(\mathsf{hk})$

1 : $(1^\lambda, \mathsf{hk}_0) \leftarrow \mathsf{hk};\ \mathsf{pp} \leftarrow \mathsf{ARC.Setup}(1^\lambda);\ (\mathsf{pp}_0, \mathsf{hk}') \leftarrow \mathsf{pp};\ \mathsf{pp}' \leftarrow (\mathsf{pp}_0, \mathsf{hk})$
2 : $(\mathsf{st}_\mathsf{A}, \mathsf{st}_\mathsf{B}, z) \leftarrow \mathsf{ARC.Init}(\mathsf{pp}')$
3 : $\mathsf{state}[\cdot], \mathsf{plaintext}[\cdot], \mathsf{log}[\cdot], \mathsf{auth}[\cdot], \mathsf{st}_* \leftarrow \bot;\ i \leftarrow 0$
4 : $\mathcal{A}^{\mathcal{O}}(z)$
5 : Let $\mathsf{num}, \mathcal{P}, \mathsf{ad}, \mathsf{ct}, \mathsf{ad}', \mathsf{ct}'$:
6 : $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \in \mathsf{log} \wedge (\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}', \mathsf{ct}') \wedge (\mathsf{ad}, \mathsf{ct}) \neq (\mathsf{ad}', \mathsf{ct}')$
7 : **return** $(\mathsf{ad}, \mathsf{ct}), (\mathsf{ad}', \mathsf{ct}')$

Figure 18: Adversary $\mathcal{B}$ where $\mathcal{O} = \{\mathsf{SEND}, \mathsf{RECEIVE}, \mathsf{AUTHSEND}, \mathsf{AUTHRECEIVE}, \mathsf{EXP}_\mathsf{pt}, \mathsf{EXP}_\mathsf{st}\}$ for the proof of Theorem 4.

D Proof of Theorem 6

Proof. We only provide the idea of the proof, as it is nearly identical to the one of Theorem 5. The intuition is that if a party $\overline{\mathcal{P}}$ sends n_s messages and then an authentication tag at to $\mathcal{P}$, then at must contain information about all the

ciphertexts previously sent. This is because $\overline{\mathcal{P}}$ cannot know which message was ever received by $\mathcal{P}$.

More precisely, the only difference with the proof of Theorem 5 is that we use at instead of ct_{n_s}, the rest follows similarly. In particular, the encoder outputs the randomness, the tag and the indices of the encoded messages in the sets of correctly received messages. Then, the decoder receives the authentication tag and tries to receive all possible ciphertexts ct_1. Among the ones that are successfully received, it extracts the correct message using the index provided by the encoder. Next it moves to receiving all possible ciphertexts ct_2 and so on, until the n_s messages are received. In addition, we give on the right of Figure 10 the adversary that can be used to prove an upper bound on the number of messages that are correctly received (i.e. the number of messages in the sets S_i in the proof of Theorem 5). □

E Deferred optimisations

In this appendix, we detail some techniques that can be used to overcome the inherent inefficiency of r-RID/r-UNF security.

E.1 Lightweight three-move authentication

We formalise our three-move protocol by proposing an appropriate security model, describing the construction itself in detail alongside a security proof.

Security model. We modify the UNF game (Section 5) by requiring the adversary to run authentication sessions in sequence. We present the corresponding game in Figure 19.

When the adversary, through the AUTHSEND′ oracle, starts the authentication protocol with initial sender $\mathcal{P}$, the adversary's access to AUTH∗, SEND′ and RECEIVE′ oracles is restricted until the three-move authentication session is completed between $\mathcal{P}$ and $\overline{\mathcal{P}}$. This is encoded in the next-oob-op variable. This simplifies the exposition, but it is not necessary in particular to restrict SEND′ and RECEIVE′ calls in practice, even though parties who authenticate in person would generally not send messages during this time. To handle this, parties can simply buffer messages during authentication that are authenticated in the next authentication session. However, buffering messages implies that an attack carried out during the out-of-band authentication will not be detected until the next authentication protocol. For this reason we block in-band communication during the three-move protocol and we encourage this restriction to be maintained also in practice. We note also that parties are guaranteed slightly weaker security than in the UNF game. Namely, after receiving the first authentication message, the receiver can deduce that their counterpart has not received a forgery but not that they themselves have until they receive the third message in the protocol.

In Definition 11 we define 3M-UNFORGEABLE-security for ARC schemes.

Definition 11 (3M-UNFORGEABLE)**.** Consider the 3M-UNFORGEABLE game presented in Fig. 19. We say that an ARC scheme is (q, t, ϵ)-3M-UNFORGEABLE secure if, for all adversaries $\mathcal{A}$ which make at most q oracle queries, and which run in time at most t, we have: $\Pr[\mathsf{3M\text{-}UNFORGEABLE}^{\mathcal{A}}(1^\lambda) \Rightarrow 1] \leq \epsilon$.

Game $\mathsf{3M\text{-}UNFORGEABLE}^{\mathcal{A}}(1^\lambda)$

1 : $\mathsf{auth\text{-}state}[\cdot] \leftarrow 0$; $\mathsf{next\text{-}oob\text{-}op} \leftarrow \bot$
2 : play UNF with $\mathcal{A}^{\mathcal{O}}(z)$

Oracle $\mathsf{SEND}'(\mathcal{P}, \mathsf{ad}, \mathsf{pt}, \mathsf{r})$

1 : **if** $\mathsf{auth\text{-}state}[\mathcal{P}] \neq 0$ **then return** $\bot$
2 : **return** $\mathsf{SEND}(\mathcal{P}, \mathsf{ad}, \mathsf{pt}, \mathsf{r})$

Oracle $\mathsf{RECEIVE}'(\mathcal{P}, \mathsf{ad}, \mathsf{ct})$

1 : **if** $\mathsf{auth\text{-}state}[\mathcal{P}] \neq 0$ **then return** $\bot$
2 : **return** $\mathsf{RECEIVE}(\mathcal{P}, \mathsf{ad}, \mathsf{ct})$

Oracle $\mathsf{AUTHSEND}'(\mathcal{P})$

1 : **if** $\mathsf{next\text{-}oob\text{-}op} \notin \{(\mathcal{P}, \text{“authsend”}), \bot\}$ **then**
2 : **return** $\bot$
3 : $i \leftarrow i + 1$
4 : $(\mathsf{st}_{\mathcal{P}}, \mathsf{num}, \mathsf{at}) \leftarrow \mathsf{AuthSend}(\mathsf{st}_{\mathcal{P}})$
5 : $\mathsf{auth}[(\mathcal{P}, i)] \leftarrow \mathsf{at}$; $\mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
6 : $\mathsf{init} \leftarrow \mathbb{1}\{\mathsf{auth\text{-}state}[\mathcal{P}] = 0\}$
7 : $\mathsf{auth\text{-}state}[\mathcal{P}] \leftarrow \mathsf{auth\text{-}state}[\mathcal{P}] + 1 \bmod 3$
8 : $\mathsf{log}[i] \leftarrow (\text{“authsend”}, \mathcal{P}, \mathsf{num}, \mathsf{at}, \mathsf{init})$
9 : $\mathsf{next\text{-}oob\text{-}op} \leftarrow (\overline{\mathcal{P}}, \text{“authrec”}, i)$
10 : **return** $(\mathsf{at}, \mathsf{num})$

$\mathsf{forgery}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}, x)$

1 : **return** $(\text{“send”}, \mathcal{P}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) \notin \mathsf{log} \wedge$
2 : $(\text{“rec”}, \overline{\mathcal{P}}, \mathsf{num}, \mathsf{ad}, \mathsf{ct}) = \mathsf{log}[x]$

Oracle $\mathsf{AUTHRECEIVE}'(\mathcal{P}, j)$

1 : **if** $\mathsf{next\text{-}oob\text{-}op} \neq (\mathcal{P}, \text{“authrec”}, j)$ **then**
2 : **return** $\bot$
3 : $\mathsf{at} \leftarrow \mathsf{auth}[(\overline{\mathcal{P}}, j)]$
4 : **if** $\mathsf{at} = \bot$ **then return** $\bot$
5 : $(\mathsf{auth}, \mathsf{st}, \mathsf{num}) \leftarrow \mathsf{AuthReceive}(\mathsf{st}_{\mathcal{P}}, \mathsf{at})$
6 : **if** $\neg\mathsf{auth}$ **then return** $\bot$
7 : $i \leftarrow i + 1$
8 : $\mathsf{auth\text{-}state}[\mathcal{P}] \leftarrow \mathsf{auth\text{-}state}[\mathcal{P}] + 1 \bmod 3$
9 : **if** $\mathsf{auth\text{-}state}[\mathcal{P}] = 0 \wedge$
10 : $\mathsf{auth\text{-}state}[\overline{\mathcal{P}}] = 0$ **then**
11 : $\mathsf{next\text{-}oob\text{-}op} \leftarrow \bot$
12 : **else**
13 : $\mathsf{next\text{-}oob\text{-}op} \leftarrow (\mathcal{P}, \text{“authsend”})$
14 : $\mathsf{st}_{\mathcal{P}} \leftarrow \mathsf{st}$; $\mathsf{state}[i] \leftarrow \mathsf{st}_{\mathcal{P}}$
15 : $\mathsf{log}[i] \leftarrow (\text{“authrec”}, \mathcal{P}, \mathsf{num}, \mathsf{at})$
16 : **return** num

$\mathsf{bad\text{-}P}(\mathsf{log}, \mathcal{P}, \mathsf{num}', \mathsf{at}, x, y)$

1 : **return** $(y > x) \wedge$
2 : $(\text{“authsend”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}, \cdot) = \mathsf{log}[y] \wedge$
3 : $(\text{“authrec”}, \mathcal{P}, \mathsf{num}', \mathsf{at}) \in \mathsf{log}$

$\mathsf{bad\text{-}\overline{P}}(\mathsf{log}, \mathcal{P}, \mathsf{num}, \mathsf{num}', \mathsf{at})$

1 : **return** $\mathsf{num} \leq \mathsf{num}' \wedge$
2 : $(\text{“authrec”}, \overline{\mathcal{P}}, \mathsf{num}', \mathsf{at}) \in \mathsf{log} \wedge$
3 : $(\text{“authsend”}, \mathcal{P}, \mathsf{num}', \mathsf{at}, \mathsf{false}) \in \mathsf{log}$

Figure 19: 3M-UNFORGEABLE game for $\mathcal{O} = \{\mathsf{SEND}', \mathsf{RECEIVE}', \mathsf{AUTHSEND}', \mathsf{AUTHRECEIVE}', \mathsf{EXP}_{\mathsf{pt}}, \mathsf{EXP}_{\mathsf{st}}\}$. Highlighted statements correspond to differences relative to the UNF game (Fig. 7).

Remark 11. The oracles in Figure 19 mandate that the participants send all messages in authentication via the out-of-band channel. By providing no security guarantees on the first message (i.e., delaying guarantees for the receiver until receiving the third message), it is possible to send the first message in the protocol over the in-band channel and then authenticate it in the second message with an additional hash [31]. Consequently, the protocol can be made essentially non-interactive out-of-band: the counterpart to the initiator can simply send the message out-of-band, and the bit can be determined easily via determining e.g. QR code scanning success/failure. By contrast, solutions like safety

numbers require *both* parties to scan QR-codes out-of-band.

Scheme description. We present a 3M-UNFORGEABLE-secure scheme in Figure 20. The AuthSend and AuthReceive procedures encode the three-move authentication protocol of Figure 14. To identify the different states of the bidirectional authentication, we borrow the terminology from TCP and refer to SYN, SYN-ACK, and ACK messages and roles. When a party $\mathcal{P}$ first calls AuthSend, it takes the SYN role and sends to $\overline{\mathcal{P}}$ the set of *received* messages and the current num, i.e., $\mathsf{at} \leftarrow (\mathsf{R}, \mathsf{num})$; this set is stored in a separate set R_{at}. As described below, we use R_{at} in AuthReceive to optimize the scheme. The counterpart $\overline{\mathcal{P}}$ replies with a SYN-ACK message, containing its set of received messages, the current ordinal num and the bit at-succ. The bit at-succ indicates whether $\mathcal{P}$'s set of received messages is included in $\overline{\mathcal{P}}$'s set of sent messages (line 10), i.e., at-succ indicates whether the authentication of $\mathcal{P}$'s set of received messages was successful. As the counterpart, $\overline{\mathcal{P}}$ stores the current set of received messages in R_{at}. Upon receiving the SYN-ACK message, the initiator $\mathcal{P}$ checks whether $\mathsf{at\text{-}succ}^{\overline{\mathcal{P}}} = \mathsf{true}$ and rejects the authentication tag otherwise. $\mathcal{P}$ then sends the ACK message $\mathsf{at} \leftarrow (\mathsf{num}, \mathsf{at\text{-}succ})$. Finally, $\overline{\mathcal{P}}$ calls AuthReceive to process the ACK message. The party checks the at-succ variable to verify that the set $\mathsf{R}^{\overline{\mathcal{P}}}$ is a subset of $\mathsf{S}^{\mathcal{P}}$. If the check passes, the authentication protocol ends.

The optimization of the scheme consists in pruning the set of received messages as soon as the counterpart authenticates them. This reduces the size of the authentication tags, since parties include in at only the received messages that have not been authenticated yet. The AuthReceive algorithm on line 11 checks whether the counterpart authenticated set of received messages R. If this is the case, all the authenticated messages are stored in $\mathsf{R}_{\mathsf{ack}}$—this set is used in the Receive algorithm to avoid replay attacks—and at the same time those messages are removed from R thanks to set R_{at}, thereby reducing the size of the next authentication tag and memory consumption. After the pruning, the R set contains only received messages that the counterpart *still* needs to authenticate.

Remark 12. Dowling et al. [17] propose a scheme that is broadly similar to ours. In particular, their protocol uses three moves in-band to allow parties to agree on a common set of respectively received messages R and R′. Then, to authenticate messages and detect active attacks, parties compare a hash $H(\mathsf{R}, \mathsf{R}')$ for hash function H out-of-band. Note however that they do not consider RID security and that they do not formally treat out-of-order message delivery.

The correctness of 3M-ARC can be shown using the underlying correctness of RC.

Security analysis. ORDINALS security is inherited from the underlying RC scheme. As usual, we argue that 3M-UNFORGEABLE security follows from the collision resistance of the underlying hash function.

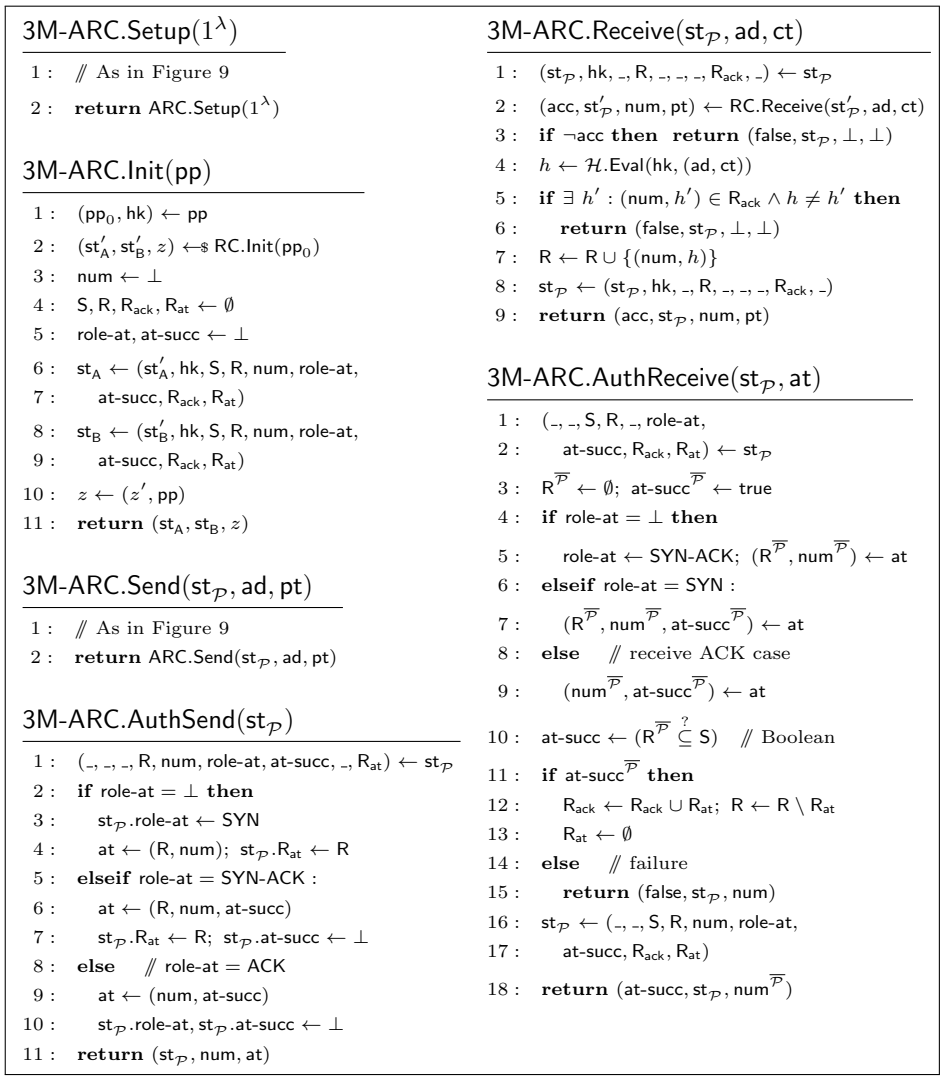


Figure 20: Optimised 3M-UNFORGEABLE-secure ARC scheme 3M-ARC based on a RC scheme RC (Definition 1). ARC refers to the unoptimised ARC defined in Figure 9. We assume ARC.Send updates local variable num. As before, the representation of R communicated can be optimised to contain only a single hash.

Theorem 9 (Unforgeability of 3M-ARC)**.** Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 7). Then the 3M-ARC scheme, that we present in Fig. 20 is (q, t, ϵ_{cr})-3M-UNFORGEABLE secure ARC scheme where $t \approx t_{cr}$.

Proof. We proceed similarly to the proof of Theorem 4. Without loss of generality, we analyze the authentication of $\mathcal{P}$, who we assume to be the initiator,

towards $\overline{\mathcal{P}}$. The adversary cannot call the SEND' and $\mathsf{RECEIVE}'$ oracles once the authentication process is started, therefore the sets S and R of both parties are fixed until the completion of the protocol.

To authenticate the set of received messages $\mathsf{R}^{\mathcal{P}}$, $\mathcal{P}$ first sends $\mathsf{at} \leftarrow (\mathsf{R}^{\mathcal{P}}, \mathsf{num})$ to $\overline{\mathcal{P}}$. To verify the authenticity of $\mathsf{R}^{\mathcal{P}}$, the party $\overline{\mathcal{P}}$ verifies whether $\mathsf{R}^{\mathcal{P}} \subseteq \mathsf{S}^{\overline{\mathcal{P}}}$. By the arguments of the proof for Theorem 4, this reduces to the collision-resistance of the hash function $\mathcal{H}$ where the reduction runs in time $t \approx t_{cr}$. After receiving the first tag, $\overline{\mathcal{P}}$ is able to detect forgeries received by $\mathcal{P}$ but not by itself. This is taken into account in the 3M-UNFORGEABLE game (line 3 in $\mathsf{bad}\text{-}\overline{\mathsf{P}}$), which states that a forgery received by $\overline{\mathcal{P}}$ is valid only if it is not detected after receiving the second or third tag in the authentication process. Then, when receiving the second tag $(\mathsf{R}^{\overline{\mathcal{P}}}, \mathsf{num}^{\overline{\mathcal{P}}}, \mathsf{at}\text{-}\mathsf{succ}^{\overline{\mathcal{P}}})$ from $\overline{\mathcal{P}}$, $\mathcal{P}$ is able to tell if itself received a forgery if the $\mathsf{at}\text{-}\mathsf{succ}^{\overline{\mathcal{P}}} = \mathsf{false}$. By the same arguments as before, $\mathcal{P}$ can tell whether $\overline{\mathcal{P}}$ received a forgery by checking $\mathsf{R}^{\overline{\mathcal{P}}} \subseteq \mathsf{S}^{\mathcal{P}}$. Finally, upon receiving the third tag, $\overline{\mathcal{P}}$ can detect a forgery using $\mathsf{at}\text{-}\mathsf{succ}^{\mathcal{P}}$.

The optimization maintains 3M-UNFORGEABLE-security. Recall that the goal of the optimization (lines 11-13 in Fig. 20) is to reduce the size of the R set by storing authenticated messages in $\mathsf{R}_{\mathsf{ack}}$. To achieve this reduction, the party executing AuthReceive removes from R the set R_{at}, which is the set of received messages authenticated by the counterpart through the at-succ variable. Since by construction all the messages in R_{at} have been already authenticated by the counterpart, removing them from R does not remove unauthenticated messages from R. □

E.2 ARC pruning-based optimization

In Section 5 we argue that is is possible to build an UNF-secure ARC scheme from a correct and ORDINALS-secure RC scheme. We defer the scheme and its security analysis to Section 5.2, since the construction is very similar to Fig. 6, except that ARC schemes use authentication tags instead of ciphertexts to exchange authentication information.

In this section we explore the optimization opportunities that the authenticated out-of-band channel gives: the adversary cannot tamper with authentication tags, which enable parties to prune messages from memory—and therefore to reduce bandwidth—as soon as authenticated tags are received.

Scheme description (Figure 15.) The Send procedure stores the hash of $(\mathsf{ad}, \mathsf{ct})$ for the message being sent, together with the corresponding num that the underlying RC.Send algorithm returns. The algorithm stores (num, h) in a set S, which is in turn stored in the party's internal state. The Send algorithm also updates the ordinal num in the state.

The Receive procedure verifies whether the RC.Receive algorithm accepts the inputs and verifies that the received message is not a forgery on a previously authenticated message, which is by construction contained in $\mathsf{S}_{\mathsf{ack}}$. If both

checks pass, Receive stores the hash of $(\mathsf{ad}, \mathsf{ct})$ together with the ordinal num returned by RC.Receive in a set R.

The AuthSend procedure is similar to the unoptimized one, except that (1) it stores the set of sent messages S authenticated *within the current* at into an array S_{at}, indexed by counter $\mathsf{cnt}_{\mathsf{at}}$, and (2) it empties the set S_{at}-Seen, which is already in at and whose goal is to communicate to the other parties which authentication tags the AuthReceive function processed, as explained later.

AuthReceive behaves like ARC.AuthReceive, but with optimizations. It firstly verifies whether $\mathsf{cnt}_{\mathsf{at}}{}^{\overline{\mathcal{P}}} \leq \mathsf{max\text{-}cnt}_{\mathsf{at}}$. The goal of this check is to avoid processing old authentication tags, since AuthReceive already authenticated their content with the newer (in terms of $\mathsf{cnt}_{\mathsf{at}}{}^{\overline{\mathcal{P}}}$) tag. The max-num-at keeps indeed track of the most recent authentication tag that the procedure processed. In other words, older tags either contain less information than newer, already accepted tags or they contain outdated information that has already been verified and pruned. AuthReceive also performs garbage collection. It first stores the counter of the input at into S_{at}-Seen, which will be sent to the counterparty in the next call to AuthSend. Then it removes the already authenticated messages from memory. The party already authenticated the subset $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ and it can remove the corresponding messages from the set R, which represents now the set of currently unauthenticated received messages. Similarly, AuthReceive uses the set of authentication tags that the counterpart already processed to prune the set of sent messages. In detail, the pruning of sent messages works as follows.

When a party $\mathcal{P}$ receives a set of messages $\mathsf{S}_{\mathsf{at}}{}^{\overline{\mathcal{P}}}[\mathsf{cnt}_{\mathsf{at}}] \leftarrow \mathsf{S}$ sent by the counterpart with the authentication tag number $\mathsf{cnt}_{\mathsf{at}}$, it stores them in a set $\mathsf{S}_{\mathsf{ack}}$. Then, when $\mathcal{P}$ sends a subsequent authentication tag back to $\overline{\mathcal{P}}$, it informs $\overline{\mathcal{P}}$ that the authentication tag $\mathsf{cnt}_{\mathsf{at}}$ was received using the $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$ set. When this tag is delivered, $\overline{\mathcal{P}}$ can remove the acknowledged messages $\mathsf{S}_{\mathsf{at}}{}^{\overline{\mathcal{P}}}[\mathsf{cnt}_{\mathsf{at}}]$ for all counters in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$ from its set $\mathsf{S}^{\overline{\mathcal{P}}}$. This reduces the size of the authentication tag as, on every round-trip on the out-of-band channel, all authenticated messages can be removed from the sets S and R. To reduce the size even further, we can use hashing optimization for the received set. Instead of sending $\mathsf{R} = \{(\mathsf{num}_1, h_1), \ldots, (\mathsf{num}_k, h_k)\}$ in AuthSend, one can send $\mathsf{R}' = \{(\mathsf{num}_1, \ldots, \mathsf{num}_k), \mathcal{H}.\mathsf{Eval}(\mathsf{hk}, h_1, \ldots, h_k)\}$. On reception, AuthReceive can recompute the hashes of the single messages and authenticate R′. Pruning does not affect this optimization, since AuthReceive removes from S only messages that the counterpart already authenticated. A similar technique was employed by Dowling et al. [17].

Security analysis. We informally argue that the scheme prunes only messages that have already been authenticated. The procedures use sets $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ and $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ to detect active attacks. AuthReceive prunes $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ by removing elements in $\mathsf{R}^{\mathcal{P}}_{\subseteq}$; since the procedure authenticates the elements in $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ at line 11, it is safe to prune $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$. AuthReceive prunes $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ by removing elements in $\mathsf{st}_{\mathcal{P}}[i]$ for $i \in \mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$. The set $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$ contains counters of the authentication tags that $\mathcal{P}$ sent to $\overline{\mathcal{P}}$ and $\overline{\mathcal{P}}$ correctly received. Moreover, the AuthReceive

procedure updates $\mathsf{st}.\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$ at line 1, i.e., after the integrity checks. Since the AuthSend stores the set of sent messages S authenticated within the current at into the array S_{at}, pruning $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ only removes messages that have already been received and authenticated by $\overline{\mathcal{P}}$.

Recall that the adversary can only delete and replay authentication tags in the out-of-band channel. We informally discuss how the scheme handles these cases. Assume $\mathcal{P}$ and $\overline{\mathcal{P}}$ exchange some messages, $\mathcal{P}$ receives an authentication tag $\mathsf{at}^{\overline{\mathcal{P}}}$ from $\overline{\mathcal{P}}$ and then sends the authentication tag $\mathsf{at}^{\mathcal{P}}$; the adversary removes $\mathsf{at}^{\mathcal{P}}$ from the channel. Since the adversary removes $\mathsf{at}^{\mathcal{P}}$, $\mathcal{P}$ does not acknowledge the reception of $\mathsf{at}^{\overline{\mathcal{P}}}$ to $\overline{\mathcal{P}}$ (because AuthSend empties $\mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}$ at every invocation). Consequently, $\overline{\mathcal{P}}$ does not prune $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S}$: these messages will be authenticated with the next authentication tag and security is preserved. The AuthReceive procedure handles adversarial reordering of authentication tags with counters $\mathsf{cnt}_{\mathsf{at}}$ at line 5.

Formally, we state the security of ARC-OP in the next theorem.

Theorem 10. Let $\mathcal{H}$ be a (t_{cr}, ϵ_{cr})-collision resistant hash function (Definition 7). Then the ARC-OP scheme (Fig. 15) is correct, ORDINALS secure, and (q, t, ϵ_{cr})-UNF secure, where $t \approx t_{cr}$.

Proof. Correctness and ORDINALS-security for the transformation of Figure 15 follow from the scheme in Figure 9.

The scheme is the same as Figure 9 modulo the optimizations we introduced. The proof of the theorem thus reduces to showing that the optimizations preserve the security properties of the unoptimized ARC scheme ARC (Figure 9). Observe that $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ and $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ are used to detect active attacks. We start by showing that pruning these sets does not undermine UNF security.

- The set $\mathsf{st}_{\mathcal{P}}.\mathsf{R}$ is pruned by removing elements from $\mathsf{R}^{\mathcal{P}}_{\subseteq}$, which was authenticated on line 11 of AuthReceive. We therefore know that messages in $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ are honest. Thus, we can stop sending them to $\overline{\mathcal{P}}$ hereafter.

- $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ is pruned by all sets $\mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{at}}[i]$ for $i \in \mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$. By construction we know that $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$ contains counters for which $\overline{\mathcal{P}}$ *accepted* the authentication tags, since those are included in line 1 of prune. Therefore, $\{\mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{at}}[i]\}_{i \in \mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}}$ contains all messages in $\mathsf{st}_{\mathcal{P}}.\mathsf{S}$ that $\overline{\mathcal{P}}$ correctly received and authenticated, which the procedure stores in $\mathsf{st}_{\overline{\mathcal{P}}}.\mathsf{S}_{\mathsf{ack}}$ for future checks. Hence, $\mathcal{P}$ can safely stop sending those and prune S correspondingly.

We proceed by showing that UNF security still holds. By the arguments above, an authentication tag at that the AuthSend procedure generates *after* another authentication tag at′, will contain only messages that have *not* been authenticated in at′. Therefore the check on line 5 preserves security.

The check on line 9 verifies whether $\mathsf{R}^{\overline{\mathcal{P}}} \subseteq \mathsf{S}$. Without pruning, this property is met in the absence of forgeries as shown for ARC. Assume for contradiction that $\mathsf{R}^{\overline{\mathcal{P}}}$ contains a message not authenticated yet, but S does not contain this

message due to pruning. This means that the message was removed from S by removing one of the values in $\mathsf{st}_{\mathcal{P}}.\mathsf{S}_{\mathsf{at}}$ whose counter $\mathsf{cnt}_{\mathsf{at}}$ was present in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$. Since the counter is present in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\overline{\mathcal{P}}}$, we know that $\overline{\mathcal{P}}$ accepted the authentication tag containing $\mathsf{cnt}_{\mathsf{at}}$, i.e., $\overline{\mathcal{P}}$ correctly received and authenticated the message. But this means by construction that $\overline{\mathcal{P}}$ pruned the message from R on line 2 of prune, which leads to a contradiction. Therefore the check preserves UNF security.

The check on line 11 verifies whether $\mathsf{R}^{\mathcal{P}}_{\subseteq} \subseteq \mathsf{S}^{\overline{\mathcal{P}}}$. Without pruning, this property is met in absence of forgeries as shown for ARC. Note that $\mathcal{P}$ removes from R only messages that have been authenticated (on line 2 of prune), therefore $\mathsf{R}^{\mathcal{P}}_{\subseteq}$ only contains unauthenticated messages. Similarly, by the argument presented in the paragraph above, $\mathsf{S}^{\overline{\mathcal{P}}}$ contains messages included in at's whose counter was not included in $\mathsf{S}_{\mathsf{at}}\text{-}\mathsf{Seen}^{\mathcal{P}}$, and therefore unauthenticated messages. We conclude that this check also preserves UNF security. □

Remark 13. $\mathsf{ARC\text{-}OP}$ (Fig. 15) sends all the authentication material through the out-of-band channel This might be impractical when the authenticated out-of-band channel is narrowband, e.g., if parties use QR-codes to authenticate the communication. We can improve the scheme by using both channels: use the insecure channel to send the authentication data and the possibly narrowband authenticated channel to verify the integrity of those data [5]. While the idea of using both channels for authentication is natural, some security risks might arise when the scheme does not correctly match the two messages. Since UNF-security depends on both the messages, and therefore on the messages being correctly matched, it might be safer to enforce this property at the scheme level. In Appendix F we propose $\mathsf{BW\text{-}UNFORGEABLE}$, a security game that enforces matching of the two authentication messages at the scheme level.

F Bandwidth-optimized UNF-security

We introduce $\mathsf{BW\text{-}UNFORGEABLE}$ game in Fig. 21. We say that an ARC scheme is $\mathsf{BW\text{-}UNFORGEABLE}$-secure if participants, after receiving *both* the message in the normal channel and the one in the out-of-band channel, can detect a forgery.

After an initialization phase, the adversary plays with a set of oracles. The $\mathsf{RECEIVE}$, $\mathsf{AUTHRECEIVE}$, $\mathsf{EXP}_{\mathsf{pt}}$ and $\mathsf{EXP}_{\mathsf{st}}$ oracles are defined in Fig. 2. The SEND' and $\mathsf{AUTHSEND}'$ oracles are very similar to the ones in Fig. 2, but they enforce the matching of authentication data, which the party sends in the normal channel, and the message that authenticates those data, that the party sends over the out-of-band channel.

The challenger of the $\mathsf{BW\text{-}UNFORGEABLE}$ game is very similar to the challenger of the UNF game. The $\mathsf{forgery}$ predicate is the same and we include it in Fig. 21 for clarity. The predicate $\mathsf{both\text{-}rec}$ enforces that the adversary must deliver both components of the authentication, i.e., data and authentication, to the receiving party. We modify the $\mathsf{bad\text{-}P}$ and $\mathsf{bad\text{-}}\overline{\mathsf{P}}$ predicates to determine

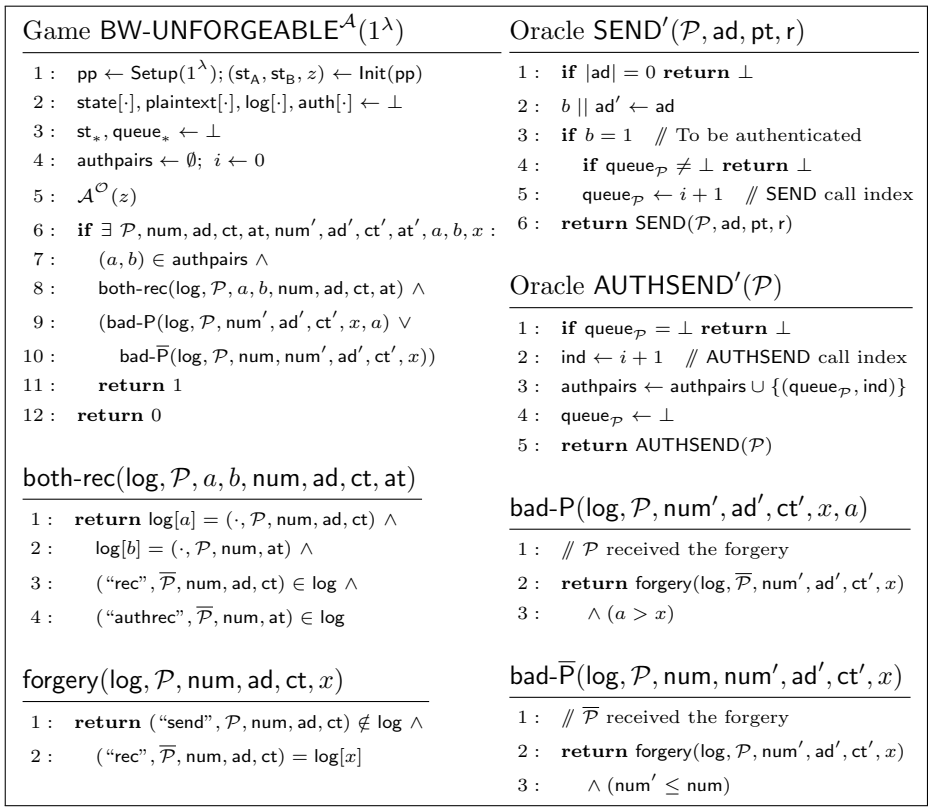


Figure 21: BW-UNFORGEABLE game for $\mathcal{O} = \{\mathsf{SEND}', \mathsf{RECEIVE}, \mathsf{AUTHSEND}', \mathsf{AUTHRECEIVE}, \mathsf{EXP}_\mathsf{pt}, \mathsf{EXP}_\mathsf{st}\}$.

which of the two parties $\mathcal{P}$ and $\overline{\mathcal{P}}$ accepts a forgery, since this is relevant to the both-rec predicate.

G Puncturable encryption lower bound

We first recall the notion of public key puncturable encryption as a slightly simplified version of the original definition by Green and Miers [22] (i.e. each ciphertext is associated to one tag only in our definition); our result still holds under their definition.

Definition 12 (PKPE)**.** A public key puncturable encryption (PKPE) scheme is a tuple of four efficient algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc})$ associated to a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, a tag space $\mathcal{T}$ and a state space $\mathcal{ST}$, defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \$\!\!\rightarrow (\mathsf{pk}, \mathsf{st})$: This probabilistic algorithm takes the security parameter λ and outputs the public key pk and a state $\mathsf{st} \in \mathcal{ST}$.

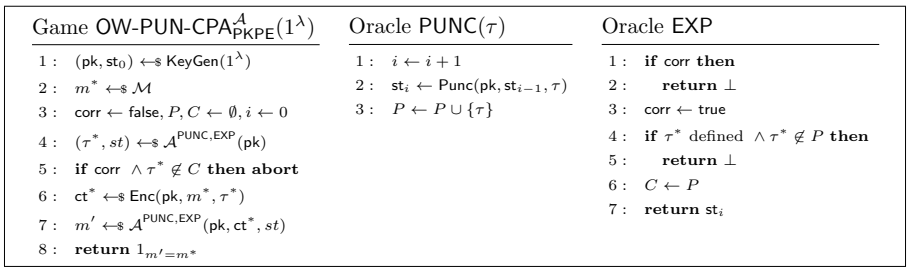


Figure 22: OW-PUN-CPA game.

- $\mathsf{Enc}(\mathsf{pk}, m, \tau)\$\!\!\rightarrow\!\mathsf{ct}$: This probabilistic algorithm takes a public key pk, message $m \in \mathcal{M}$ and tag $\tau \in \mathcal{T}$ and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{Punc}(\mathsf{pk}, \mathsf{st}, \tau) \rightarrow \mathsf{st}'$: This deterministic algorithm takes a public key pk, state $\mathsf{st} \in \mathcal{ST}$ and tag $\tau \in \mathcal{T}$ and outputs a new state $\mathsf{st}' \in \mathcal{ST}$.
- $\mathsf{Dec}(\mathsf{st}, \mathsf{ct}, \tau) \rightarrow m'$: This deterministic algorithm takes a state $\mathsf{st} \in \mathcal{ST}$, ciphertext ct and tag $\tau \in \mathcal{T}$ and outputs a message $m' \in \mathcal{M} \cup \{\perp\}$.

For the sake of simplicity, we will consider perfect correctness only, which is defined as follows.

Definition 13 (PKPE Correctness)**.** A PKPE is perfectly correct if for all λ, $n \in \mathbb{N}, m \in \mathcal{M}, (\tau_1, \ldots, \tau_n) \in \mathcal{T}$ and $\tau^* \in \mathcal{T} \setminus \{\tau_1, \ldots, \tau_n\}$, the sequence:

1. $(\mathsf{pk}, \mathsf{st}_0) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$
2. For $i \in [n]$: $\mathsf{st}_i \leftarrow \mathsf{Punc}(\mathsf{pk}, \mathsf{st}_{i-1}, \tau_i)$
3. $\mathsf{ct}^* \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, m, \tau^*)$

is s.t.

$$\Pr[\mathsf{Dec}(\mathsf{st}_n, \mathsf{ct}^*, \tau^*) = m] = 1 \ .$$

Finally, we introduce a security notion for PKPE, called OW-PUN-CPA, defined as follows.

Definition 14 (OW-PUN-CPA)**.** We consider the OW-PUN-CPA game presented in Figure 22. We say a PKPE scheme PKPE is (q, t, ϵ)-OW-PUN-CPA secure if for all adversaries $\mathcal{A}$ running in time at most t and making at most q queries to the PUNC oracle we have

$$\Pr[\mathsf{OW\text{-}PUN\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKPE}}(1^\lambda) \Rightarrow 1] \leq \epsilon \ .$$

Note that we omit the number of queries to EXP in the parameters as at most 1 such query is useful.

In the OW-PUN-CPA game, the adversary can puncture any tag and expose/corrupt the current state (at most once). Then, it outputs a challenge tag

τ^*. The game gives the encryption of a random message m^* under the tag τ^* and the adversary must recover m^*. In addition, it is ensured that the adversary cannot expose a state st_i if τ^* was not punctured before the state was created. This restriction is enforced by returning $\perp$ in the EXP oracle when $\tau^* \notin P$, or by making the adversary lose when $\tau^* \notin C$. Informally, this prevents trivial wins where an adversary gets a secret state st where τ^* was not punctured, and simply recovers m_b by decrypting ct^* with st.

Finally, we stress that if the message space is large enough, this notion is weaker than IND-PUN-CPA as defined by Green and Miers [22], in the same way that one-wayness for public key encryption is weaker than IND-CPA. However, OW-PUN-CPA will be sufficient (and necessary) for *our* lower bound proof to go through.

G.1 Lower bound

In order to find a lower bound on the state size of puncturable encryption schemes, one can apply the same idea used to derive the lower bound on ciphertext length in r-RID and r-UNF security (Theorem 5 and 6). Informally, one can even see puncturable encryption as a kind of dual to r-RID RC. Indeed, in the latter, each ciphertext should embed which past messages have been sent (i.e. which messages should be *valid*). In the former, each state update should embed which tags have been punctured (i.e. which tags should be *invalid*). Overall, as we use information-theoretical techniques in the proof (encoder/decoder can be inefficient), both constructions are somewhat equivalent. Hence, one can prove the following theorem.

Theorem 11. Let PKPE be a perfectly correct PKPE, λ be fixed, n_p be the maximal number of punctures and T_{λ,n_p} be the running time of the adversary $(\mathcal{A}_1, \mathcal{A}_2)$ given in Figure 24.
In addition, let $\gamma \in \mathbb{Z}$ be s.t. PKPE is $(n_p, T_{\lambda,n_p}, \frac{1}{2^\gamma})$-OW-PUN-CPA secure. Finally, let $\mathcal{T} = \{0,1\}^n$ and $\mathcal{ST} = \{0,1\}^k$ be the tag and state space associated to PKPE, respectively. Then,

$$k \geq \mathsf{log}\binom{2^n}{n_p} - \frac{2}{1-\frac{n_p}{2^\gamma}} \geq nn_p - n_p\mathsf{log}(n_p) - \frac{2}{1-\frac{n_p}{2^\gamma}}$$

where the rightmost expression becomes $\approx (n - \mathsf{log}(n_p))n_p - 2$ when $n_p \ll 2^\gamma$.

Proof. As in the proof of Theorem 5, we show the existence of an encoder/decoder that would break Shannon's coding theorem if the theorem does not hold.

The encoder and decoder are presented in Figure 23. We assume the encoder input is sampled from a source that outputs a random n_p-sized set of tags, a random sequence of coins R and a random message $M \in \mathcal{M}$. Let $\mathcal{S}$ be the random set of tags. The encoder punctures the initial state st_0 on all tags in $\mathcal{S}$ to obtain a state st_{n_p}. Note that encrypting any message M with any tag $\tau \notin \mathcal{S}$ and decrypting with st_{n_p} returns M by the perfect correctness of the PKPE

$\mathsf{Encode}(\{\tau_1, \ldots, \tau_{n_p}\}, R, M)$

1 : parse $R_0, R_1 \leftarrow R$
2 : $\mathsf{pk}, \mathsf{st}_0 \leftarrow \mathsf{KeyGen}(1^\lambda; R_0)$
3 : **for** $i \in \{1, \ldots, n_p\}$ **do** // puncture the n_p tags
4 : $\quad \mathsf{st}_i \leftarrow \mathsf{Punc}(\mathsf{pk}, \mathsf{st}_{i-1}, \tau_i)$
5 : // Checking whether encrypting and decrypting with punctured tag is succ.:
6 : **for** $i \in \{1, \ldots, n_p\}$ **do**
7 : $\quad \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, \tau_i; R_1)$
8 : $\quad$ **if** $\mathsf{Dec}(\mathsf{st}_i, \mathsf{ct}, \tau_i) = M$ **then**
9 : $\quad\quad \mathsf{fail} \leftarrow \mathsf{true}$ // if fail: output trivial encoding
10 : **if** fail **then**
11 : $\quad$ **return** $(0, \{\tau_1, \ldots, \tau_{n_p}\}, R, M)$
12 : **else**
13 : $\quad$ **return** $(1, \mathsf{st}_{n_p}, R, M)$

$\mathsf{Decode}(b, \mathsf{data}, R, M)$

1 : **if** $b = 0$ **then** // if encoding failed: the input is the trivial encoding
2 : $\quad$ **return** (data, R, M)
3 : parse $\mathsf{st}_{n_p} \leftarrow \mathsf{data}$ // st_{n_p} is on k bits
4 : parse $R_0, R_1 \leftarrow R$
5 : $\mathsf{pk}, \mathsf{st}_0 \leftarrow \mathsf{KeyGen}(1^\lambda; R_0)$
6 : $\mathcal{S} \leftarrow \emptyset$
7 : **for** $\tau \in \{0,1\}^n$ **do**
8 : $\quad \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, M, \tau; R_1)$
9 : $\quad$ **if** $\mathsf{Dec}(\mathsf{st}_{n_p}, \mathsf{ct}, \tau) \neq M$ **then** // if decryption failed, τ was punctured
10 : $\quad\quad \mathcal{S} \cup \{\tau\}$
11 : **return** (S, R, M)

Figure 23: Encoder/Decoder for Theorem 11 proof.

scheme. However, it might be that encrypting and decrypting with $\tau \in \mathcal{S}$ still returns M, we call this event fail. If that happens, the encoder outputs a bit set to 0 and the trivial encoding of the input. Otherwise, it outputs 1 and the punctured state st_{n_p} (and R, M).

If $b = 0$, the decoding is trivial. If $b = 1$, the decoder can find all tags in $\mathcal{S}$ by encrypting M with all possible tags and storing those for which the decryption under st_{n_p} does not hold M. As all tags in $\mathcal{S}$ would behave so (otherwise fail would have occurred in the encoder), $\mathcal{S}$ can be recovered. Thus, the encoder is correct and uniquely decodable (note that the bit indicating a failure is necessary for this).

We now prove the following lemma.

Lemma 5. Let fail be defined as above. Then,

$$\Pr[\mathsf{fail}] \leq \frac{n_p}{2^\gamma} \ .$$

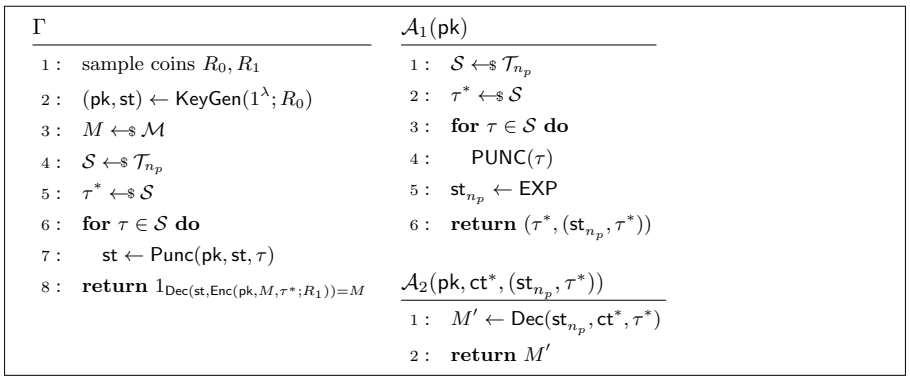


Figure 24: Intermediary game Γ (left) and adversary $\mathcal{A}$ (right) for Theorem 11 proof.

Proof. Let $\mathcal{T}_{n_p} := \{T : T \subseteq \mathcal{T}, |T| = n_p\}$ be the set of subsets of $\mathcal{T}$ of size n_p. Then, let's consider the game Γ defined on the left in Figure 24. The probability that this algorithm outputs 1 is the probability that $\mathcal{S}$ is a failing set (i.e. fail occurs) *and* τ^* is a tag that triggers fail. Therefore, overall the prob. that the process outputs 1 is

$$\Pr[\Gamma \Rightarrow 1] = \Pr[\tau^* \text{ triggers } \mathsf{fail} | \mathsf{fail}] \times \Pr[\mathsf{fail}] \geq \frac{\Pr[\mathsf{fail}]}{|\mathcal{S}|}$$

as τ^* is sampled uniformly at random from $\mathcal{S}$.

Then, one can see that Γ is exactly the $\mathsf{OW\text{-}PUN\text{-}CPA}$ game played by the adversary $(\mathcal{A}_1, \mathcal{A}_2)$ given on the right in Figure 24. Note that all elements in such a game have the same distribution as in Γ. Moreover, τ^* will be in C in the $\mathsf{OW\text{-}PUN\text{-}CPA}$ game (which is necessary for the adversary to win). Thus, we will have

$$\frac{\Pr[\mathsf{fail}]}{|\mathcal{S}|} \leq \Pr[\Gamma \Rightarrow 1] = \Pr[\mathsf{OW\text{-}PUN\text{-}CPA}^{\mathcal{A}}_{\mathsf{PKPE}}(1^\lambda) \Rightarrow 1] \leq 2^{-\gamma}$$

where the second inequality follows from the theorem assumption. Hence, $\Pr[\mathsf{fail}] \leq \frac{n_p}{2^\gamma}$ as $|\mathcal{S}| = n_p$. □

Now, we make the following claim.

Claim 1. Let $M \in \{0,1\}^\ell$, $R_0, R_1 \in \{0,1\}^r$, C be the random variable representing the length of the output of the encoder and $\alpha := \Pr[\mathsf{fail}]$. Then,

$$\mathbb{E}[C] = 1 + \alpha \left\lceil \mathsf{log}\binom{2^n}{n_p} \right\rceil + (1-\alpha)k + \ell + 2r \ .$$

Proof. If fail occurs, the set of tags $\mathcal{S}$ will be output by the encoder. As there are $\binom{2^n}{n_p}$ possible sets, one can represent each set with $\left\lceil \mathsf{log}\binom{2^n}{n_p} \right\rceil$ bits. If fail does not occur, the state st_{n_p} is output, which is represented on k bits. Finally,

regardless of fail, a bit, M and R_0, R_1 are output. These take $1 + \ell + 2r$ bits to represent. □ □

Then, $\mathbb{E}[C] \leq 2 + \alpha \mathsf{log}\binom{2^n}{n_p} + (1-\alpha)k + \ell + 2r$ and by Shannon's source coding theorem we obtain

$$\mathsf{log}\binom{2^n}{n_p} + \ell + 2r \leq \mathbb{E}[C] \leq 2 + \alpha \mathsf{log}\binom{2^n}{n_p} + (1-\alpha)k + \ell + 2r$$

as the entropy of the source is $\mathsf{log}\binom{2^n}{n_p} + \ell + 2r$. Finally, we obtain

$$k \geq \mathsf{log}\binom{2^n}{n_p} - \frac{2}{1-\alpha} \geq \mathsf{log}\binom{2^n}{n_p} - \frac{2}{1 - \frac{n_p}{2^\gamma}}$$

where the second inequality follows from Lemma 5. Using that $\binom{2^n}{n_p} \geq \frac{2^{nn_p}}{n_p^{n_p}}$, we can lower bound k further by □

Remarks. First, we note that the bound is symmetric due to the binomial coefficient. That is, if n_p grows larger than $\frac{2^n}{2}$ the lower bound decreases. However, this case is not really relevant as it implies that the number of supported punctures is larger than $\frac{2^n}{2}$, and thus the bound is maximal as $\mathcal{A}$ can make $\frac{2^n}{2}$ punctures. This is also consistent with the fact that one can store the tags that have already been punctured (as is implicitly done in e.g. pairing-based schemes [22]) or, on the contrary, store the tags that have not been punctured yet (e.g. the trivial construction which uses $|\mathcal{T}|$ instances of PKE). The state in the former case would outgrow the state in the latter once the number of tags punctured exceeds $\frac{2^n}{2}$.

Secondly, while the lower bound is shown to work for perfectly correct PKEs, it can be adapted to the non-perfect setting. One can use the same technique used for the first lower bounds in Theorem 5. That is, we collect all tags that make decryption fail in a list and output the index of the correct one in the encoding. If the probability of a correctness error is at most $\frac{1}{2^\delta}$, then each index should fit into $\approx (n - \delta)$ bits. Hence, overall, the lower bound in Theorem 11 would be worsened by a factor $\approx n_p(n - \delta)$.

Finally, we note that Stauble [36] gave a simple lower bound for a restricted type of puncturable encryption schemes, namely *perfectly correct non-hierarchical data-structure-based puncturable encryption*. That is, puncturable encryption schemes that store secret-keys (each one used to decrypt some tags) in non-hierarchical data structures such as lists (as opposed to hierarchical structures such as trees). They simply note that in such a setting, if the scheme is perfectly correct, there should be as many keys as there are tags. To the best of our knowledge, we are the first to give a lower bound for any PKPE.

H Changelog

H.1 December 6 2023

This revision includes some fixes and minor changes in both writing and figures, the most notable being:

- The UNF-secure ARC ARC-OP (Figure 15) now perform garbage collection when tags are input to AuthReceive out-of-order, and the line numbers in the security proof have been corrected.
- In the 3M-UNFORGEABLE-secure ARC 3M-ARC (Figure 20), AuthReceive now outputs the correct output bit auth, unused variables have been removed and missing/overloaded variables have been corrected.
- In the pruning-based s-RID-secure RC s-RID-RC (Figure 13), messages are now only acknowledged for two epochs (rather than four as written before), which improves performance. The figure has also been simplified.

A table of contents has also been added for convenience.

H.2 November 20 2024

This revision includes some fixes and minor changes in both writing and figures, the most notable being:

- The epoch-based optimization for s-RID security is now in the body of the paper (Section 7.2). We also fixed some typos in Fig. 13.
- The CORRECT game (Fig. 1) now correctly keeps track of ordinals for each party.
- The auth-monotonic predicate in the ORDINALS game (Fig. 3) is now simpler.