With one it is easy, with many it gets complicated:
Understanding Channel Security for Groups

Giorgia Azzurra Marson and Bertram Poettering

Ruhr University Bochum, Germany
{giorgia.marson,bertram.poettering}@rub.de

Abstract. Secure messaging systems such as TextSecure and Signal aim, among others, at providing authenticated and confidential channels between two or more communicating users. The general understanding seems to be that providing security in the sense of authenticated encryption (AE) for every point-to-point link suffices to make the constructed messaging systems secure, i.e., authentic and confidential. However, as recently shown (in FSE17/ToSC17), in the bidirectional (two-party) case, just requiring the two unidirectional links to provide security independently of each other *does not* lead to a secure communication channel in general. Briefly, the reason for this is that security notions need to take into account the increased level of interaction between users. This also applies, a fortiori, in a multi-user setting where many parties communicate concurrently and in multiple directions. We observe that in the current academic literature there is no rigorous definition of security for (broadcast) group communication. Applying the method of provable security, we fill this gap by defining security goals and showing how to provably achieve them.
Concretely, the contributions of this paper are as follows: We develop a set of formal definitions of security goals for broadcast communication, capturing targets like confidentiality and authenticity. Our notions in particular take into account the causal dependencies that exchanged messages might have. (Note that these have no counterpart in the much simpler unidirectional case.) We then switch to the constructive side, designing an efficient protocol that requires only reliable point-to-point links between users and a standard cryptographic building block (AEAD), achieving all security goals defined in this paper. Our work thus paves the ground towards understanding the exact level of security that current secure messaging systems achieve.

Keywords: secure messaging, group communication, confidentiality, integrity, causality preservation

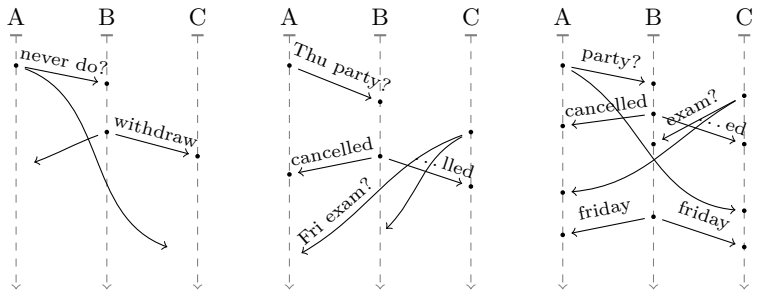
1 Introduction

Secure channels One of the fundamental applications of cryptography is secure point-to-point communication, more precisely establishing a *secure channel* to transport messages over an untrusted medium. Prominent examples of secure channels (a.k.a. cryptographic channels) that protect a reliable connection over TCP/IP are the SSL/TLS protocol suite [7] and the SSH remote shell protocol [27]. Due to their widespread deployment, both TLS and SSH have been extensively studied in the cryptographic literature (for instance, in [3,18,10,11,1]). The resulting analyses identify confidentiality, integrity, and protection against reordering and replay attacks as the main security goals of these types of channel. While the above goals are suitable for channel types allowing to reliably transmit a sequence of messages from a sender to a receiver *in one direction*, they do not fit the more realistic scenario in which two parties communicate simultaneously *in both directions*. This mismatch between how TLS and SSH are modeled in theory and how they are used in practice was recently pointed out in [15]. Beyond pinpointing surprising results concerning the bidirectional security of channels, [15] extends confidentiality and integrity notions from the unidirectional setting to the bidirectional case. As it turns out, explicitly including the bidirectional flavor of interactiveness in the model is crucial for understanding and reaching security.

Motivated by the above observations, in this work we make a further step towards understanding channel security in the context of interactive communication: we consider a *broadcast* setting in which two or more parties interact (by exchanging messages) with all other parties.

Group conversations A broadcast channel allows a group of participants to exchange messages in a broadcast fashion: All participants may transmit, and all transmissions target and reach the whole group

(i.e., all individuals). Such a setting is typical, e.g., in group messaging systems, but also automated communication systems like interconnected bank computers rely on such infrastructure. Ideally, we would like to lift the security notions of security for bidirectional channels to the broadcast setting. What we immediately notice is that, if more than two users participate in a conversation, the usual requirement of sequential delivery (i.e., that messages originating from each given user are received in the same order they were sent) may not be sufficient to guarantee that users have a faithful view on the conversation. We discuss below, and illustrate in Figure 1, some problematic situations that may occur in a three-party conversation, even though messages are transmitted reliably according to the sending order.



(a) Scenario #1 (b) Scenario #2 (c) Scenario #3

Fig. 1. Misunderstandings caused by causality violations. Vertical dashed lines symbolize per-party timelines (time progresses top-down), bullets represent broadcast and receiving actions.

Consider a chatroom situation like the one illustrated in Figure 1a. Alice starts a conversation by asking around what the other participants would never do in their life; Bob, being proud of his current submission to his favorite conference, would never withdraw his paper and answers with "withdraw the submission!"; Charlie, a co-author of Bob, misses Alice's initial question, by consequence misunderstands Bob, and ultimately withdraws their joint paper against Bob's will. A slightly different issue is illustrated in Figure 1b. This time Alice asks Bob for the exact time of a party planned at his place that is losely scheduled for Thursday night. Due to an exam on Friday, Bob decides to cancel the party and notifies his friends. Meanwhile, Charlie asks when the exam on Friday will be, and understands from Bob's answer that the exam (and not the party) has been canceled. In the same context, the situation illustrated in Figure 1c is even more severe. Alice asks for the day of the party and learns from Bob that it has been canceled. Charlie instead makes an inquiry about the exam, which will be on Friday as Bob announces. However, Charlie first receives Bob's first message, then Alice's question, and finally Bob's second message. In the end, not only Charlie misses a chance to take the exam, he even shows up on Friday night at Bob's place expecting a party. Note that although these situations may look artificial, they are perfectly in line with the delivery guarantees offered by point-to-point TCP connections. In particular, network adversaries have full control over message delivery and can easily arrange the corresponding delays.

Consistency through causality preservation Situations like the ones described in Figure 1 are ordinary in the context of group communication. The technical solution to misunderstandings of this type, classic in the domain of distributed systems, is to enforce deliveries that preserve the *causality* relation among messages. With other words, no participant shall receive a message if they have not yet received all messages sent before (here, 'before' is understood in a global sense).[1] Concretely, in all situations depicted in Figure 1, if causality was preserved, Charlie would receive Alice's message before receiving anything from Bob, and no misunderstanding would occur. Protocols that efficiently realize a causal broadcast infrastructure are well-known in the literature (we reproduce a standard construction in Figure 4). However, such protocols typically do not give guarantees on the causal delivery of messages in the face of adversaries that control the network. (Also, they do not provide confidentiality and authenticity against such adversaries.)

[1] The notion of time considered in causality is abstract and defined independently of physical time.

We are not the first to identify aspects of causality preservation as relevant security properties for group messaging protocols. For instance, security issues related to participants having inconsistent views about the ongoing conversation were recognized in [8]. (While their work does not explicitly mention causality, the informal description of a 'consistent view' suggests that causality preservation may be an implicit goal.) A protocol for secure group communication that explicitly targets a kind of causal delivery[2] appears in [20]. More recently, [16] refers to 'transcript consistency' as one of the ideal properties that the TextSecure v2 messaging protocol should fulfill. Causality preservation is also listed as one of the relevant 'conversation security' properties in [26]. More precisely, [26] proposes a taxonomy of desirable properties for two-party and group conversations, and indicates which ones among well-known protocols for secure messaging (claim to) achieve these properties. According to [26], specific configurations of OTR [17], GOTR [13], OldBlue [9], KleeQ [20], Axolotl [19], and TextSecure [14] offer some form of causality preservation (however, not on formal grounds).

Despite the increasing interest in preserving causality in secure messaging, there is yet no formal treatment of this notion as a security property.[3] Moreover, we note that all above-mentioned works consider causality aspects in secure messaging as *independent* of the other security requirements such as confidentiality and integrity. We argue instead that 'standard' security notions shall be adapted to incorporate the concept of causality within.

1.1 Contribution and organization

In the face of the wide-spread use of group chat apps, we feel that the security of causal broadcast channels in general, and of causality-preserving secure messaging in particular, deserves a closer look. Understanding causality from the perspective of cryptography as well as constructing corresponding secure schemes is the main goal of the present work. We treat secure broadcast channels as a generalization of bidirectional channels to the multi-user setting and, following the tradition of provable security, give rigorous formalizations of the novel functionality requirements (causality preservation) and of appropriate security notions. Notably, our extended notions of confidentiality and integrity incorporate aspects related to causality that have no counterpart in the simpler, bidirectional case.

The paper is organized as follows. After introducing some notation in Section 2, in Section 3 we specify a generalized syntax for (not necessarily cryptographic) broadcast channels. We further give a formal functionality definition and show how causal broadcast can be achieved from simpler network primitives. We proceed with defining integrity and confidentiality notions for broadcast channels in Section 4, where we also investigate how our confidentiality and integrity notions are related to each other. Finally, in Section 5 we propose a cryptographic protocol that provably meets the strongest confidentiality and integrity properties proposed in this paper.

1.2 Related work

Causality. Starting with Lamport's groundbreaking work on (distributed) logical clocks [12], the role of causality in communication systems has been extensively investigated in the distributed systems community. For a survey on logical time, its connections to causality, and related notions we suggest the work by Schwarz and Mattern [25]. Also cryptographic challenges related to causality have been recognized and studied extensively. Although targeting more general problems arising in the context of secure replication of services, Reiter and Birman [21] consider attacks on causality that specifically exploit causality violations among service requests, and propose countermeasures. In the same vein, Reiter and Gong [22] highlight the importance of detecting causal relations between messages exchanged by distributed processes; they introduce the notions of *causality denial* (making a server believe that a pair of causally related messages is not in such a relation) and *causality forgery* (convincing a server of a causality relation which does not exist). More recently, Cachin et al. [6] combined different techniques from modern cryptography and distributed computing to improve secure solutions of service replication. Among several variants of reliable broadcast primitives they propose a *causal broadcast* protocol which

[2] In fact, it targets a stronger property that implies causality.

[3] The concept of causality is well-understood in distributed systems. Indeed, broadcast protocols that enforce causality have existed for decades. However, we stress that these protocols do not, and in fact are not meant to, also enforce causality in the presence of active adversaries that control the network.

tolerates a Byzantine adversary and offers *input causality*—a property introduced in [21] ensuring that honest servers deliver client requests in the right order and that exchanged messages remain secret until delivery.

We observe that while many of the works discussed above employ cryptographic techniques for solving problems from distributed computing, none of them addresses what we are interested in: the confidential and authentic exchange of messages in multi-directional channels.

Secure messaging (for groups). In their systematization of knowledge (SoK) paper, Unger et al. [26] list security properties that are targeted (or claimed) by several protocols for secure messaging. Among these, the properties that are closest to ours are *causality preservation*, meaning that 'implementations can avoid displaying a message before messages that causally precede it,' and *global transcript*, requiring that 'all participants see all messages in the same order.' As [26] only provide informal descriptions of these properties and seem to address functionality rather than security, a close comparison with our notions is out of reach.

In the context of improving the off-the-record protocol to support group communication, Goldberg et al. [8] introduce a property that resembles the global transcript requirement of Unger et al. This property, refered to as *consensus*, is concerned with the participants' view of the ongoing conversation (where the view contains, among others, also the exchanged messages in the 'right order'). The corresponding definition given in [8] is rather informal, and it is unclear to us whether the message ordering is meant to also respect causality, or only weaker delivery guarantees.

Another work that targets a *global transcript* property is by Reardon et al. [20]. These authors propose a protocol, called KleeQ, that is meant to provide secure group communication in ad-hoc networks with limited connectivity. As for the above-mentioned papers, the lack of formal (functionality and) security notions makes this work incomparable to ours.

2 Notation

To initialize an empty array $\mathbf{X}$ we write $\mathbf{X}[\,] \leftarrow \emptyset$. For $N \in \mathbb{N}$ we denote by $\mathbf{0}_N$ the all-zero vector of length N. If $\mathbf{v}$ is a vector of length N and if $i \in [1\,..\,N]$, then $\mathbf{v}[i]$ denotes the component of $\mathbf{v}$ at position i. If also $\mathbf{w}$ is a vector of length N, we write $\mathbf{v} \leq \mathbf{w}$ if $\forall i\colon \mathbf{v}[i] \leq \mathbf{w}[i]$, and we write $\mathbf{v} \not\leq \mathbf{w}$ if $\exists i\colon \mathbf{v}[i] > \mathbf{w}[i]$. We denote boolean values by T (true) and F (false). Given a condition C we may use the shortcuts C and $\neg C$ for the expressions $C = \mathrm{T}$ and $C = \mathrm{F}$, respectively. If $\mathcal{A}$ is a deterministic algorithm, $y \leftarrow \mathcal{A}(x)$ indicates that $\mathcal{A}$ is run on input x and its output is assigned to variable y. If $\mathcal{A}$ is randomized and its coins are uniformly picked, we write $y \leftarrow_{\$} \mathcal{A}(x)$.

Our security definitions are in the game-based tradition. A game G is a randomized procedure that runs internally an adversary $\mathcal{A}$ and eventually outputs a bit. Within an algorithmic specification of a game G we write 'Stop with b', for $b \in \{0, 1\}$, to indicate that G halts and outputs b, and we denote by $G(\mathcal{A}) \Rightarrow b$ the corresponding event. The adversary may also call subroutines that are provided as oracles. Within a subroutine we write 'Give x to $\mathcal{A}$' to indicate that the adversary obtains value x when the subroutine terminates. We write 'Return' to terminate the execution of an algorithm/subroutine that does not produce any output visible to the adversary.

3 Causal Broadcast Channels

3.1 Broadcast Channels

We consider multiple parties[4] that exchange messages in a broadcast fashion using two dedicated algorithms: bc (broadcast) and rcv (receive). Conceptually, these algorithms connect the application and the network layers in the top-down and bottom-up directions, respectively: When a sender wishes to broadcast a message to the other users they invoke the bc algorithm with that message, and when a participant receives a datagram[5] from the network they invoke the rcv algorithm with the datagram and an identifier

[4] We use the words party, participant, and user synonymously.

[5] Datagrams should be understood as encapsulations of messages or, more pragmatically, as the data that is actually transmitted over the network. In the cryptographic literature, these objects are referred to as 'ciphertexts' (as they coincide, most often, with the output of an encryption primitive).

of the alleged sender. Internally, the two algorithms may invoke the abstract subroutines snd (send) and dlv (deliver), where snd initiates the transport of a given datagram to an indicated other user, and dlv delivers a given message to the local user, indicating also the (alleged) sender of that message. Both the bc and rcv algorithms may be randomized and keep state between invocations.

We formalize the above ideas as follows. Let N denote the total number of participants, and let $\mathcal{M}$ be a message space and $\mathcal{D}$ be a datagram space. Let $i, j \in [1\,..\,N]$ indicate two users. Subroutine snd is invoked as per $\mathsf{snd}(i, j, D)$ if datagram $D \in \mathcal{D}$ shall be sent by user i to user j. Similarly, subroutine dlv is invoked as per $\mathsf{dlv}(i, j, m)$ if message $m \in \mathcal{M}$ shall be delivered at user i, and the message was broadcast (allegedly) by user j. Both subroutines, snd and dlv, require $i \neq j$ and have no output. A helpful shorthand form for expressing the two syntactical conventions is

$$[1\,..\,N] \times [1\,..\,N] \times \mathcal{D} \to \mathsf{snd} \qquad \text{and} \qquad [1\,..\,N] \times [1\,..\,N] \times \mathcal{M} \to \mathsf{dlv}\ .$$

To broadcast messages $m \in \mathcal{M}$, users invoke the bc algorithm as per $st' \leftarrow \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st; m)$, where st, st' are the original and the updated state, respectively. Similarly, users can receive datagrams $D \in \mathcal{D}$ by invoking $st' \leftarrow \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st; j, D)$, where $j \in [1\,..\,N]$ indicates from which other user the datagram originates. If $\mathcal{S}$ is the state space, the shorthand forms for algorithms $\mathsf{bc}, \mathsf{rcv}$ are thus

$$\mathcal{S} \times \mathcal{M} \to \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}} \to \mathcal{S} \qquad \text{and} \qquad \mathcal{S} \times [1\,..\,N] \times \mathcal{D} \to \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}} \to \mathcal{S}\ .$$

The interplay of the $\mathsf{bc}, \mathsf{rcv}, \mathsf{snd}, \mathsf{dlv}$ algorithms is also illustrated in Figure 2.

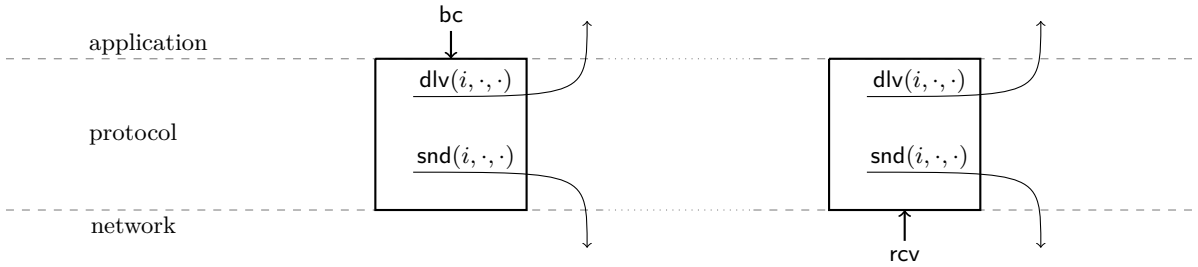


Fig. 2. Overview of the broadcast and receiving algorithms.

Before starting with the communication, the state of users has to be initialized. The corresponding procedure may choose identifiers for the users, initialize buffers, establish cryptographic keys, etc. Formally, we require that there be a randomized algorithm init that, on input a number of users N, outputs for each user $i \in [1\,..\,N]$ an individual initial state st_i. For expressing that N users shall be initialized we correspondingly write $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$.

Definition 1. *A* broadcast channel Ch *for a message space* $\mathcal{M}$ *consists of a datagram space* $\mathcal{D}$*, a state space* $\mathcal{S}$*, and algorithms* init*,* bc*, and* rcv*, that follow the syntax specified above. Note that* bc *and* rcv *are defined in respect to abstract subroutines* snd *and* dlv*.*

3.2 Vector clocks and causality

We aim at designing secure broadcast channels that preserve causal relations between broadcast and delivered messages. Informally, causal delivery means that no party can deliver a given message if they have not yet delivered all messages that were broadcast before (where 'before' is meant globally, i.e., according to Lamport's *happened before* relation [12]). For instance, in a situation like the one of Figure 1a, causal delivery would guarantee that Charlie does not deliver Bob's message until he has delivered Alice's message.

A standard technique, developed in the domain of distributed systems, for checking whether causal delivery is maintained is through *vector clocks* (or vector timestamps). The idea is that each user $i \in [1\,..\,N]$ maintains a vector $\mathbf{vc}_i$ of counters that is initialized to $\mathbf{0}_N$ and updated upon bc and dlv invocations. Concretely, element $\mathbf{vc}_i[i]$ is incremented whenever user i performs a broadcast operation, and element $\mathbf{vc}_i[j]$

is incremented whenever i delivers a message with alleged originator j. Whether a delivery occurs in the right causal order is then verified as follows: Let $\mathbf{vc}_i$ be the vector clock of user i immediately before delivering message m from user j, and let $\mathbf{vc}_j$ be the vector clock of user j right before broadcasting m. Then the delivery is according to causal order if and only if $\mathbf{vc}_j \leq \mathbf{vc}_i$, i.e., if party i has already delivered all messages broadcast before. Explained in more detail, the condition means that user i, when delivering m, (i) did not deliver from j more often than j broadcast until m (this is expressed by $\mathbf{vc}_j[j] \leq \mathbf{vc}_i[j]$), (ii) did not deliver from any other user $k \notin \{i, j\}$ more often than j did at the moment of broadcasting m (formally, $\mathbf{vc}_j[k] \leq \mathbf{vc}_i[k]$), and, in addition, (iii) that user j did not deliver from i more often than i had broadcast (formally, $\mathbf{vc}_j[i] \leq \mathbf{vc}_i[i]$). Note that if the underlying network offers reliability and FIFO delivery, (i) and (iii) are always fulfilled. The vector clock formalism provides us with a handy tool to verify that specific causality relations are met. In the rest of this section we will make use of this tool to define the functionality (a.k.a. correctness) requirement of a causal broadcast protocol.

3.3 Causal broadcast from reliable point-to-point connections

In this section we specify the functionality goal of causal broadcast channel protocols built on top of a FIFO network, i.e., reliable point-to-point links with first-in-first-out delivery. Intuitively, the requirement is that if the conditions assumed of the network are met then the constructed channel provides causal delivery. That is, if datagrams are received (via rcv) in the same order they were sent (via snd) then all delivered (via dlv) messages have been previously broadcast (via bc) by the alleged senders; moreover, deliveries happen according to the *causal* order among messages.

To formalize the above intuition we follow the game-based approach. We describe a functionality game FUNC, specified in Figure 3, in which an adversary $\mathcal{A}$ is given access to the broadcast algorithms bc and rcv through oracles bc and rcv. The goal of the adversary in this game is to violate correctness, i.e., to deliver messages with wrong content or in wrong order. One can view $\mathcal{A}$ as a scheduler that triggers the communication between users by requesting the execution of bc and rcv for users on input of messages and datagrams of $\mathcal{A}$'s choosing.

When oracle bc is queried on input (i, m), it invokes the broadcast algorithm on input message m and the current state st_i (i.e., of user i). Similarly, if $\mathcal{A}$ queries rcv on input (i, j, D) the oracle invokes the receiving algorithm on input datagram D, alleged originator j, and the current state st_i. These oracles, beyond executing the channel algorithms on adversarial request, also inform $\mathcal{A}$ of any sending and delivery operation that occurs within the execution of bc and rcv. To this end, for every call $\mathsf{snd}(i, j, D)$ the oracle gives a tuple (s, i, j, D) to the adversary as an indication that datagram D has been sent by user i to user j. This reflects that the adversary controls the network and in particular knows which datagram is transmitted by whom and to whom. Similarly, whenever $\mathsf{dlv}(i, j, m)$ is invoked, the oracle gives to $\mathcal{A}$ a tuple (d, i, j, m) to report that message m has been delivered to user i with alleged originator user j. This captures the adversary's ability to observe the reaction of applications upon delivery of specific messages.[6]

During the execution of the game, two specific events have to be considered: whether the adversary remains *passive*, meaning that it only schedules rcv operations that are consistent with the guarantees provided by the underlying network; and whether the adversary violates *correctness*.

Passiveness requires $\mathcal{A}$ to schedule rcv operations in a way that datagrams are transmitted faithfully and sequentially, according to the sending order. This property is tested within every call to oracle rcv (in line 16). For this, the experiment keeps boolean variables psv_i, one for each user $i \in [1 \,..\, N]$, indicating whether an active measure of the adversary against user i took place. Initially all flags are set to $psv_1 = \cdots = psv_N = \mathrm{T}$. The flag of user i is set to $psv_i \leftarrow \mathrm{F}$ when the adversary causes user i to receive from some user j a datagram D that either does not originate from that user, or is not received according to the sequential order (in contrast to the guarantees offered by the point-to-point reliable connection), or, transitively, was sent by j after the latter itself was exposed to an active measure of the adversary. To detect if any of the above conditions is satisfied, the game records for each pair of users (i, j) (equivalently, for each point-to-point connection) the number s_{ij} of send operations performed by i to j as well as the corresponding sequence $\mathbf{D}_{ij}$ of sent datagrams, and the number r_{ij} of receive operations by i with alleged

[6] In the functionality game, in contrast to the security games defined below, this ability actually does not give any advantage to the adversary (by functionality, datagrams received faithfully will cause the delivery of previously sent messages, which the adversary already knows).

originator j. Using these variables, the game sets $psv_i \leftarrow \mathrm{F}$ in line 17 (meaning that user i is actively attacked) when i receives from j more often than j sent ($s_{ji} \leq r_{ij}$) or if they receive a datagram that deviates from the genuine sequence ($\mathbf{D}_{ji}[r_{ij}] \neq D$). (See lines 21–23 for how the transitivity of setting $psv_i \leftarrow \mathrm{F}$ is implemented.)

Likewise, the game checks (in line 27) if there was a correctness violation in any invocation of dlv and, if so, it declares the adversary successful and terminates the game with output 1. For the correctness test, the game records the number b_i of broadcast operations performed by i, the corresponding sequence $\mathbf{M}_i$ of broadcast messages, as well as the number d_{ij} of delivery operations performed by i with alleged originator j (note that $\mathbf{vc}_i[i] = b_i$ and $\mathbf{vc}_i[j] = d_{ij}$ for all $j \in [1\,..\,N]$ throughout the game), the current vector clock $\mathbf{vc}_i$ of user i, and the sequence $\mathbf{VC}_i$ of all vector clocks registered for user i immediately before each of their broadcast operations (i.e., $\forall n : 0 \leq n < b_i$ vector $\mathbf{VC}_i[n]$ is the vector clock associated to i before the n-th invocation of bc). Then, whenever an operation of type $\mathsf{dlv}(i, j, m)$ is performed, the game flags a correctness violation in case the adversary is still passive and message m does not match the genuine sequence of messages broadcast by j ($b_j \leq d_{ij}$ or $\mathbf{M}_j[d_{ij}] \neq m$), or its delivery does not preserve causality ($\mathbf{VC}_j[d_{ij}] \not\leq \mathbf{vc}_i$).

For a broadcast channel Ch, we define the advantage of an adversary $\mathcal{A}$ in the FUNC game as $\mathrm{Adv}^{\mathrm{func}}_{\mathsf{Ch}}(\mathcal{A}) = \Pr[\mathrm{FUNC}(\mathcal{A}) \Rightarrow 1]$, where the probability is taken over the randomnesses of init, bc, and rcv, and over $\mathcal{A}$'s randomness. In this paper we demand perfect correctness, i.e., $\mathrm{Adv}^{\mathrm{func}}_{\mathsf{Ch}}(\mathcal{A}) = 0$ for every (even unbounded) adversary $\mathcal{A}$.

Game $\mathrm{FUNC}_{\mathsf{Ch},N}(\mathcal{A})$
00 For $i \leftarrow 1$ to N:
01 $psv_i \leftarrow \mathrm{T}$
02 $b_i \leftarrow 0$; $\mathbf{vc}_i \leftarrow \mathbf{0}_N$
03 $\mathbf{M}_i[\,] \leftarrow \emptyset$; $\mathbf{VC}_i[\,] \leftarrow \emptyset$
04 For $j \leftarrow 1$ to N, $j \neq i$:
05 $s_{ij}, r_{ij}, d_{ij} \leftarrow 0$
06 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$
07 $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$
08 $\mathcal{A}^{\mathrm{bc,rcv}}$
09 Stop with 0

Oracle $\mathrm{bc}(i, m)$
10 $\mathbf{M}_i[b_i] \leftarrow m$
11 $\mathbf{VC}_i[b_i] \leftarrow \mathbf{vc}_i$
12 $st_i \leftarrow_{\$} \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st_i; m)$
13 $b_i \leftarrow b_i + 1$
14 $\mathbf{vc}_i[i] \leftarrow b_i$
15 Return

Oracle $\mathrm{rcv}(i, j, D)$
16 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
17 $psv_i \leftarrow \mathrm{F}$
18 $st_i \leftarrow_{\$} \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st_i; j, D)$
19 $r_{ij} \leftarrow r_{ij} + 1$
20 Return

Proc $\mathsf{snd}(i, j, D)$
21 If psv_i:
22 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D$
23 $s_{ij} \leftarrow s_{ij} + 1$
24 Give $(\mathtt{s}, i, j, D)$ to $\mathcal{A}$
25 Return

Proc $\mathsf{dlv}(i, j, m)$
26 If psv_i:
27 If $b_j \leq d_{ij}$ or $\mathbf{M}_j[d_{ij}] \neq m$ or $\mathbf{VC}_j[d_{ij}] \not\leq \mathbf{vc}_i$:
28 Stop with 1
29 $d_{ij} \leftarrow d_{ij} + 1$
30 $\mathbf{vc}_i[j] \leftarrow d_{ij}$
31 Give $(\mathtt{d}, i, j, m)$ to $\mathcal{A}$
32 Return

Fig. 3. Game for correctness of (cryptographic) causal broadcast channels. Note that $\mathbf{vc}_i$ always reflects the current contents of b_i and d_{ij}: $\mathbf{vc}_i[i] = b_i$ and $j \neq i \Rightarrow \mathbf{vc}_i[j] = d_{ij}$.

We point out that the network guarantees—here FIFO ordering—are reflected by the operations snd and rcv (these are the interfaces between protocol and network layers), while the delivery properties that applications expect—causal ordering—concern operations bc and dlv (which provide interfaces between application and protocol layers).

3.4 Construction of causal broadcast channels

We reproduce a standard construction of a causal broadcast protocol, known in the distributed systems literature as *waiting causal broadcast* [5], built on top of reliable point-to-point connections like TCP/IP. As we assume reliability and FIFO delivery from the underlying network, the goal is to leverage these properties to causal delivery guarantees at the application layer. The core idea is to let each user store incoming messages in $N - 1$ queues, one for each possible sender, and to wait until the time to deliver these messages has come. The 'right' delivery time is determined by counting how many messages have been broadcast and delivered so far (concretely, this is done using vector clocks). Intuitively, the FIFO property is reflected in the use of queues to store incoming messages, while causal delivery is achieved by keeping and comparing vector clocks. The details of the construction are given in Figure 4. We give a proof of correctness (as per game FUNC) in Appendix A.

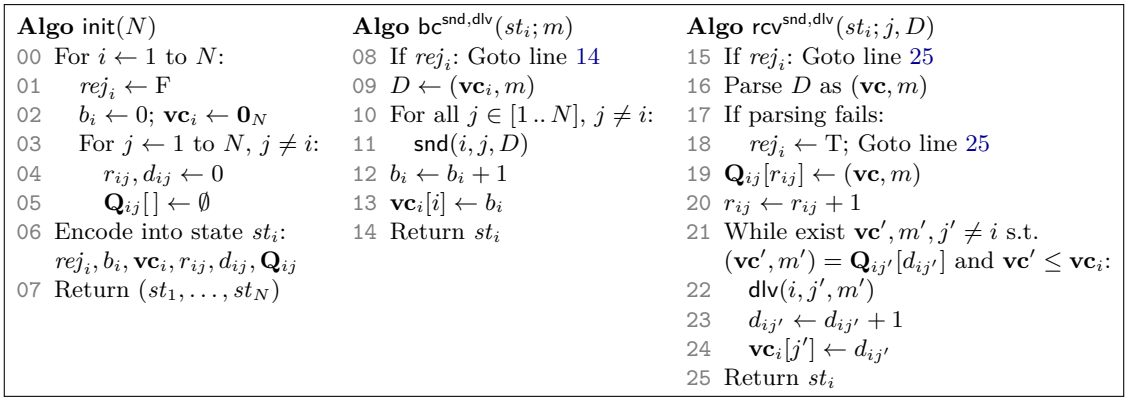


Fig. 4. Waiting causal broadcast. State variables b_i, r_{ij}, d_{ij} are broadcast, receive, and delivery counters, respectively. Data structure $\mathbf{Q}_{ij}[\,]$ implements a queue in which incoming datagrams are stored, in the order of their arrival, until they are eventually delivered (and implicitly removed).

4 Cryptographic Broadcast Channels

After defining broadcast channels and their functionality, we have seen how they can be constructed. However, quite obviously, the protocol in Figure 4 does not provide any resilience against active network adversaries that are interested in learning exchanged message contents or in altering them. In this section we study *cryptographic broadcast channels*. Syntactically and functionally such channels are like regular broadcast channels, but they also give security guarantees. Concretely, we formalize four security properties: integrity of plaintexts and integrity of ciphertexts as authentication notions, and indistinguishability against chosen-plaintext attacks and indistinguishability against chosen-ciphertext attacks as confidentiality notions. We then study how these notions relate to each other. Our notions are in the style of [2], [4], and [3] for symmetric (authenticated) encryption.

4.1 Notions of Integrity

We define two notions of authenticity: integrity of plaintexts (INT-PTXT) and integrity of ciphertexts (INT-CTXT). Intuitively, the former guarantees that all messages delivered to users are authentic in the sense that they were broadcast by some other user before (in the causal sense), while the latter ensures that once a user's rcv algorithm is exposed to a manipulated datagram the algorithm is isolated from user and network so that it cannot do harm to anybody. The two notions are different in spirit in that while INT-PTXT is formulated from the point of view of the application, which cares about the integrity of what it sees (messages) and not of what is transferred on the wire (datagrams), INT-CTXT cares about what happens on the wire and not what is delivered to the application. Importantly, only INT-PTXT explicitly requires that deliveries happen in causal order. (However, as we shall prove, causal delivery is *implicitly* also ensured by INT-CTXT.)

We start with the formalization of plaintext integrity. The corresponding experiment is in Figure 5. It is best explained by comparing it with the broadcast functionality game from Figure 3: Recall that in the FUNC game the adversary wins by making the dlv algorithm deliver a message that was either never sent by the alleged sender or is delivered out of order (in the causal sense), and all this preconditioned on the adversary remaining passive. For defining the INT-PTXT notion we drop the latter requirement and allow the adversary to be active. To a channel Ch and a number of users N we assign the INT-PTXT advantage of an adversary $\mathcal{A}$ as $\mathrm{Adv}^{\text{int-ptxt}}_{\mathsf{Ch},N}(\mathcal{A}) = \Pr[\mathrm{INT}^{\text{ptxt}}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]$, where the probability is over the game's randomness including over $\mathcal{A}$'s coins. Intuitively, channel Ch offers plaintext integrity if $\mathrm{Adv}^{\text{int-ptxt}}_{\mathsf{Ch},N}(\mathcal{A})$ is small for all realistic N and $\mathcal{A}$.

We next formalize ciphertext integrity, based on the experiment in Figure 6. Here the rcv oracle is as in the FUNC game, watching out for the adversary performing an active attack by injecting an out-of-order datagram (in the FIFO sense). The INT-CTXT notion demands that actively attacked users refuse such manipulated datagrams and become inoperative. The latter is formalized by requiring the channel not to invoke that party's snd and dlv procedures any further. We define the advantage of an

Game $\mathrm{INT}^{\mathrm{ptxt}}_{\mathsf{Ch},N}(\mathcal{A})$
00 For $i \leftarrow 1$ to N:
01 $b_i \leftarrow 0$; $\mathbf{vc}_i \leftarrow \mathbf{0}_N$
02 $\mathbf{M}_i[\,] \leftarrow \emptyset$; $\mathbf{VC}_i[\,] \leftarrow \emptyset$
03 For $j \leftarrow 1$ to N, $j \neq i$:
04 $d_{ij} \leftarrow 0$
05 $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$
06 $\mathcal{A}^{\mathrm{bc},\mathrm{rcv}}$
07 Stop with 0

Oracle $\mathrm{bc}(i, m)$
08 $\mathbf{M}_i[b_i] \leftarrow m$
09 $\mathbf{VC}_i[b_i] \leftarrow \mathbf{vc}_i$
10 $st_i \leftarrow_{\$} \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st_i; m)$
11 $b_i \leftarrow b_i + 1$
12 $\mathbf{vc}_i[i] \leftarrow b_i$
13 Return

Oracle $\mathrm{rcv}(i, j, D)$
14 $st_i \leftarrow_{\$} \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st_i; j, D)$
15 Return

Proc $\mathsf{snd}(i, j, D)$
16 Give (s, i, j, D) to $\mathcal{A}$
17 Return

Proc $\mathsf{dlv}(i, j, m)$
18 If $b_j \leq d_{ij}$ or $\mathbf{M}_j[d_{ij}] \neq m$ or $\mathbf{VC}_j[d_{ij}] \not\leq \mathbf{vc}_i$:
19 Stop with 1
20 $d_{ij} \leftarrow d_{ij} + 1$
21 $\mathbf{vc}_i[j] \leftarrow d_{ij}$
22 Give (d, i, j, m) to $\mathcal{A}$
23 Return

Fig. 5. Game for INT-PTXT security of cryptographic causal broadcast channels.

adversary $\mathcal{A}$ as $\mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{A}) = \Pr[\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]$. Intuitively, channel Ch offers ciphertext integrity if $\mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{A})$ is small for all realistic N and $\mathcal{A}$.

Game $\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}(\mathcal{A})$
00 For $i \leftarrow 1$ to N:
01 $psv_i \leftarrow \mathrm{T}$
02 For $j \leftarrow 1$ to N, $j \neq i$:
03 $s_{ij}, r_{ij} \leftarrow 0$
04 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$
05 $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$
06 $\mathcal{A}^{\mathrm{bc},\mathrm{rcv}}$
07 Stop with 0

Oracle $\mathrm{bc}(i, m)$
08 $st_i \leftarrow_{\$} \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st_i; m)$
09 Return

Oracle $\mathrm{rcv}(i, j, D)$
10 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
11 $psv_i \leftarrow \mathrm{F}$
12 $st_i \leftarrow_{\$} \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st_i; j, D)$
13 $r_{ij} \leftarrow r_{ij} + 1$
14 Return

Proc $\mathsf{snd}(i, j, D)$
15 If $\neg psv_i$: Stop with 1
16 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D$
17 $s_{ij} \leftarrow s_{ij} + 1$
18 Give (s, i, j, D) to $\mathcal{A}$
19 Return

Proc $\mathsf{dlv}(i, j, m)$
20 If $\neg psv_i$: Stop with 1
21 Give (d, i, j, m) to $\mathcal{A}$
22 Return

Fig. 6. Game for INT-CTXT security of cryptographic causal broadcast channels.

4.2 Notions of Confidentiality

We define the confidentiality notions IND-CPA and IND-CCA that consider passive and active adversaries, respectively. Our games, in Figures 7 and 8, use the left-or-right indistinguishability approach: If the adversary queries the bc oracle on messages m_0 and m_1, then message m_b is picked and broadcast, and the resulting datagrams are made available to the adversary, where b is a secret challenge bit. Intuitively, the scheme is confidential if no adversary can distinguish the $b = 0$ from the $b = 1$ world. The adversary further has access to a rcv oracle to advance the state of the corresponding participant. The difference between the notions IND-CPA and IND-CCA is about the kind of datagrams that can be submitted to rcv: The former notion considers passive adversaries, i.e., those that provide the rcv oracle with exclusively those datagrams that were output by the snd procedure before (see lines 09 and 10 of Figure 7 on how this type of passive behavior is enforced), while the latter notion considers active adversaries and has no such restriction.

Formally, for any causal broadcast channel Ch and any number N we define the IND-CPA advantage of an adversary $\mathcal{A}$ as $\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},N}(\mathcal{A}) = |\Pr[\mathrm{IND}^{\mathrm{cpa},1}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\mathrm{cpa},0}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]|$. Intuitively, channel Ch offers indistinguishability under chosen-plaintext attacks if the advantage $\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},N}(\mathcal{A})$ is small for all realistic N and $\mathcal{A}$. The IND-CCA advantage $\mathrm{Adv}^{\text{ind-cca}}_{\mathsf{Ch},N}(\mathcal{A})$ for the security notion of indistinguishability under chosen-ciphertext attacks is defined analogously.

4.3 Relations Among Security Notions

We gave four security definitions for causal broadcast channels: two formalizing confidentiality notions and two formalizing authenticity notions. We next prove three implications between these notions, showing that they are not independent of each other: (1) IND-CCA security implies IND-CPA security,

Game $\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}(\mathcal{A})$
00 For $i \leftarrow 1$ to N:
01 For $j \leftarrow 1$ to N, $j \neq i$:
02 $s_{ij}, r_{ij} \leftarrow 0$
03 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$
04 $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$
05 $b' \leftarrow_{\$} \mathcal{A}^{\mathrm{bc,rcv}}$
06 Stop with b'

Oracle $\mathrm{bc}(i, m_0, m_1)$
07 $st_i \leftarrow_{\$} \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st_i; m_b)$
08 Return

Oracle $\mathrm{rcv}(i, j, D)$
09 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
10 Stop with 0
11 $st_i \leftarrow_{\$} \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st_i; j, D)$
12 $r_{ij} \leftarrow r_{ij} + 1$
13 Return

Proc $\mathsf{snd}(i, j, D)$
14 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D$
15 $s_{ij} \leftarrow s_{ij} + 1$
16 Give $(\mathtt{s}, i, j, D)$ to $\mathcal{A}$
17 Return

Proc $\mathsf{dlv}(i, j, m)$
18 Give $(\mathtt{d}, i, j, \diamond)$ to $\mathcal{A}$
19 Return

Fig. 7. Game for IND-CPA security of cryptographic causal broadcast channels.

Game $\mathrm{IND}^{\mathrm{cca},b}_{\mathsf{Ch},N}(\mathcal{A})$
00 For $i \leftarrow 1$ to N:
01 $psv_i \leftarrow \mathrm{T}$
02 For $j \leftarrow 1$ to N, $j \neq i$:
03 $s_{ij}, r_{ij} \leftarrow 0$
04 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$
05 $(st_1, \ldots, st_N) \leftarrow_{\$} \mathsf{init}(N)$
06 $b' \leftarrow_{\$} \mathcal{A}^{\mathrm{bc,rcv}}$
07 Stop with b'

Oracle $\mathrm{bc}(i, m_0, m_1)$
08 $st_i \leftarrow_{\$} \mathsf{bc}^{\mathsf{snd},\mathsf{dlv}}(st_i; m_b)$
09 Return

Oracle $\mathrm{rcv}(i, j, D)$
10 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
11 $psv_i \leftarrow \mathrm{F}$
12 $st_i \leftarrow_{\$} \mathsf{rcv}^{\mathsf{snd},\mathsf{dlv}}(st_i; j, D)$
13 $r_{ij} \leftarrow r_{ij} + 1$
14 Return

Proc $\mathsf{snd}(i, j, D)$
15 If psv_i:
16 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D$
17 $s_{ij} \leftarrow s_{ij} + 1$
18 Give $(\mathtt{s}, i, j, D)$ to $\mathcal{A}$
19 Return

Proc $\mathsf{dlv}(i, j, m)$
20 If psv_i:
21 Give $(\mathtt{d}, i, j, \diamond)$ to $\mathcal{A}$
22 Else:
23 Give $(\mathtt{d}, i, j, m)$ to $\mathcal{A}$
24 Return

Fig. 8. Game for IND-CCA security of cryptographic causal broadcast channels.

(2) INT-CTXT security implies INT-PTXT security, (3) IND-CPA and INT-CTXT together imply IND-CCA security. Note that while the first implication might be very expected (the adversary in IND-CPA is more restricted than in IND-CCA), proving the second is more involved and leverages on the perfect correctness of the channel protocol. Also proving the third implication is involved; its result will be key in the analysis of our construction presented in Section 5.

Lemma 1 (IND-CCA $\Longrightarrow$ IND-CPA). *Let* Ch *be a broadcast channel that offers indistinguishability under chosen-ciphertext attacks (IND-CCA). Then* Ch *also offers indistinguishability under chosen-plaintext attacks (IND-CPA). More precisely, for every adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *such that*

$$\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},N}(\mathcal{A}) \leq \mathrm{Adv}^{\text{ind-cca}}_{\mathsf{Ch},N}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*. Further, the number of* bc *and* rcv *queries it poses is the same as that of* $\mathcal{A}$*.*

Proof. The proof is based on the observation that the IND-CPA game is a specifically restricted variant of the IND-CCA game, and that thus any adversary that breaks the former security notion also breaks the latter security notion. The formal argument builds on the fact that for any adversary $\mathcal{A}$ it is straightforward to construct an adversary $\mathcal{B}$ such that $\Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] = \Pr[\mathrm{IND}^{\mathrm{cca},b}_{\mathsf{Ch},N}(\mathcal{B}) \Rightarrow 1]$, for $b \in \{0,1\}$. (This is possible because public information is sufficient to check in oracle rcv of Figure 8 whether $\mathcal{A}$ is passive or not.) Ultimately this shows $|\Pr[\mathrm{IND}^{\mathrm{cpa},1}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\mathrm{cpa},0}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]| = |\Pr[\mathrm{IND}^{\mathrm{cca},1}_{\mathsf{Ch},N}(\mathcal{B}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\mathrm{cca},0}_{\mathsf{Ch},N}(\mathcal{B}) \Rightarrow 1]|$, and thus the claim. □

Lemma 2 (INT-CTXT $\Longrightarrow$ INT-PTXT). *Let* Ch *be a broadcast channel that offers integrity of ciphertexts (INT-CTXT). Then* Ch *also offers integrity of plaintexts (INT-PTXT). More precisely, for every adversary* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *such that*

$$\mathrm{Adv}^{\text{int-ptxt}}_{\mathsf{Ch},N}(\mathcal{A}) \leq \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*. Further, the number of* bc *and* rcv *queries it poses is the same as that of* $\mathcal{A}$*.*

Proof. Fix any N. Consider the game $G_0 := \mathrm{INT}^{\mathrm{ptxt}}_{\mathsf{Ch},N}$ from Figure 5. Derive from G_0 the game G_1 by replacing the main game body, the rcv oracle, and the snd procedure by the corresponding versions of game $\mathrm{FUNC}_{\mathsf{Ch},N}$ from Figure 3, leaving unmodified the bc oracle and the dlv procedure. Note that these are pure rewriting steps that do nothing more than introducing variables for tracking the internals of the game, in particular the psv_i flags. That is, the changes do not affect the winning probability of the adversary. Thus, $\Pr[G_1(\mathcal{A}) \Rightarrow 1] = \Pr[G_0(\mathcal{A}) \Rightarrow 1]$.

Derive now game G_2 from G_1 by adding as first lines of the snd and dlv procedures the conditional abort instruction 'If $\neg psv_i$: Stop with 0'. Compare G_2 with the game $\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}$ from Figure 6 and observe that the newly added instructions make a difference only for those adversaries $\mathcal{A}$ that are successful with (implicitly) breaking the INT-CTXT property. Formally, for any $\mathcal{A}$ there exists a reduction $\mathcal{B}$ such that $|\Pr[G_2(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1]| = \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B})$.

Observe finally that every adversary that wins in game G_2 also wins in game $\mathrm{FUNC}_{\mathsf{Ch},N}$. (This is because winning in G_2 is possible only by having dlv be invoked in a '$psv_i = \mathrm{T}$' state, and in this case the winning conditions of G_2 and game $\mathrm{FUNC}_{\mathsf{Ch},N}$ are the same.) Thus $\Pr[G_2(\mathcal{A}) \Rightarrow 1] \leq \Pr[\mathrm{FUNC}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]$. As we assume perfect correctness, all in all we have $\Pr[G_0(\mathcal{A}) \Rightarrow 1] = \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B})$, and thus the claim. □

Theorem 1 (IND-CPA + INT-CTXT $\Longrightarrow$ IND-CCA). *Let* Ch *be a broadcast channel that offers indistinguishability under chosen-plaintext attacks (IND-CPA) and integrity of ciphertexts (INT-CTXT). Then* Ch *also offers indistinguishability under chosen-ciphertext attacks (IND-CCA). More precisely, for every adversary $\mathcal{A}$ there exist adversaries $\mathcal{B}^0, \mathcal{B}^1, \mathcal{C}$ such that*

$$\mathrm{Adv}^{\text{ind-cca}}_{\mathsf{Ch},N}(\mathcal{A}) \leq \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B}^0) + \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B}^1) + \mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},N}(\mathcal{C}) \ .$$

The running times of $\mathcal{B}^0, \mathcal{B}^1, \mathcal{C}$ are about that of $\mathcal{A}$. Further, the number of bc *and* rcv *queries they pose is the same as that of $\mathcal{A}$.*

Proof. Fix any N. For $b \in \{0,1\}$ consider the games $G^b_0 := \mathrm{IND}^{\mathrm{cca},b}_{\mathsf{Ch},N}$ from Figure 8. Derive from G^b_0 the games G^b_1 by inserting in the snd and dlv procedures, right before lines 15 and 20, the conditional abort instruction 'If $\neg psv_i$: Stop with 0'. Compare G^b_1 with the game $\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}$ from Figure 6 and observe that the newly added instructions make a difference only for those adversaries $\mathcal{A}$ that are successful with (implicitly) breaking the INT-CTXT property. Formally, for any $\mathcal{A}$ there exist (the obvious) reductions $\mathcal{B}^b$ such that $|\Pr[G^b_1(\mathcal{A}) \Rightarrow 1] - \Pr[G^b_0(\mathcal{A}) \Rightarrow 1]| = \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{B}^b)$. Further, a comparison with Figure 7 shows that for any $\mathcal{A}$ there exists a reduction $\mathcal{C}$ such that $\Pr[G^b_1(\mathcal{A}) \Rightarrow 1] = \Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}(\mathcal{C}) \Rightarrow 1]$. (This holds because public information is sufficient to check in oracle rcv whether $\mathcal{A}$ is passive or not.) Using the triangle inequality (and using a shortcut notation that neither annotates $\mathcal{A}$ nor the probabilities) we have $|G^1_0 - G^0_0| \leq |G^1_0 - G^1_1| + |G^1_1 - G^0_1| + |G^0_1 - G^0_0|$. Together with the above this implies the claim. □

5 Construction of Cryptographic Causal Broadcast

After defining the security goals of cryptographic causal broadcast in the previous section, we now present a particular way to jointly achieve them. Our construction combines two ingredients: the (non-cryptographic) protocol from Figure 4 that achieves causal broadcast from point-to-point links, and, as a cryptographic primitive, an authenticated encryption with associated data (AEAD) scheme. For reference, we recall syntax, functionality, and security definitions of AEAD in Appendix B.

The algorithms of our construction are in Figure 9. As they need to achieve the functionality requirements of causal broadcast, not surprisingly their structure is similar to that of the algorithms from Figure 4. The design challenge was to augment the routines by AEAD invocations at the right spots and in the right dosage, so that we could reach two overall goals simultaneously: security (our construction provably meets all security notions defined in this paper), and efficiency (we aimed at minimizing the number of AEAD invocations per execution of bc/rcv.)

Let us compare the algorithms of our construction with the ones from Figure 4. The init algorithms are almost the same, the only difference being the fresh AEAD key K that is shared among all participants of a broadcast channel instance. In the bc algorithm we see a slightly different structure: While in Figure 4 one datagram D is computed and sent to all $N-1$ other users, in our design each user gets its individual datagram D_j. When creating it we include the identities of the sending and receiving users in

the associated data, as well as a transmission number, so that the adversary cannot replay datagrams or issue them in the wrong order. Our rcv algorithm reverses the encryption step and recovers the messages. Note that some of these might never be delivered to the corresponding user, as they might not have appeared in the correct causal order. We caution that leaking information on waiting messages to the user would likely harm the confidentiality of the scheme.

That our construction is correct (i.e., achieves the causal broadcast functionality) follows from the fact that the protocol from Figure 4 is correct, plus the correctness of the AEAD. For the security analysis, see below.

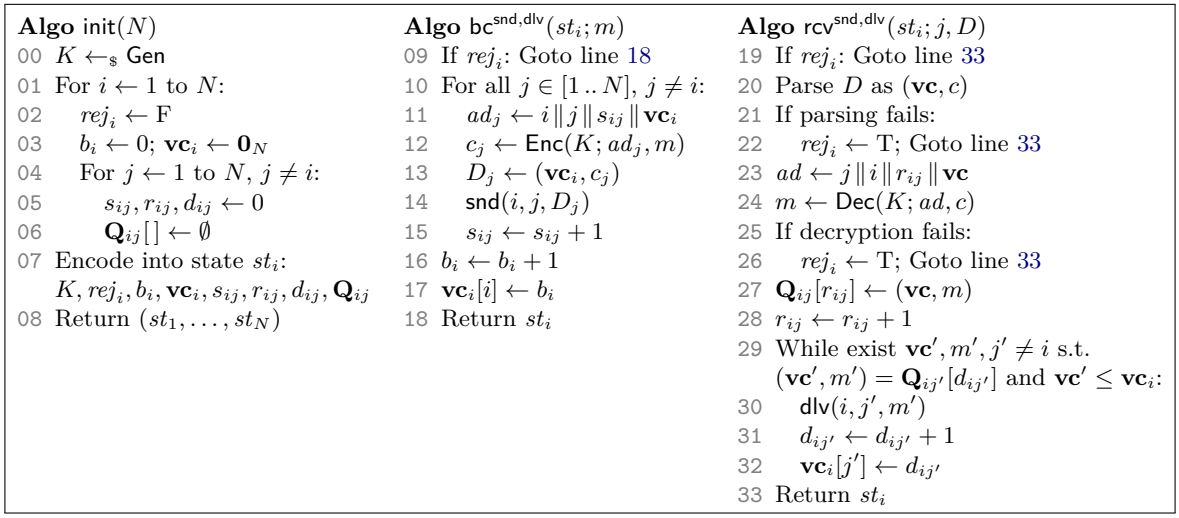


Fig. 9. Construction of cryptographic causal broadcast from reliable point-to-point connections.

5.1 Security analysis

We analyze the security of our broadcast channel constructed from reliable point-to-point connections and an AEAD scheme, as specified in Figure 9. Our argument consists of three steps: first we show that the IND-CPA security of the AEAD implies the IND-CPA security of the broadcast channel; we then show that the INT-CTXT security of the AEAD implies the INT-CTXT security of the broadcast channel; finally, as a corollary of the two results, we apply Theorem 1 to establish the IND-CCA security of the broadcast channel.

Theorem 2 (IND-CPA security). *Let* Ch *be the broadcast channel constructed in Figure 9 from reliable point-to-point connections and an AEAD scheme* AEAD*. If the AEAD scheme offers (one-time) indistinguishability under chosen-plaintext attacks (IND-CPA), also* Ch *offers indistinguishability under chosen-plaintext attacks (IND-CPA). More precisely, for every adversary $\mathcal{A}$ against* Ch *there exists an adversary $\mathcal{B}$ against* AEAD *such that*

$$\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{Ch},N}(\mathcal{A}) \leq \mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD}}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ is about that of $\mathcal{A}$, and $\mathcal{B}$ poses as many Enc *queries as $\mathcal{A}$ poses* bc *queries.*

Proof. For $b \in \{0, 1\}$, consider games G^b_0 from Figure 10. They are identical to games $\mathrm{IND}^{\text{cpa},b}_{\mathsf{Ch},N}$ from Figure 7, but with the following rewriting steps applied: (a) the abstract bc and rcv algorithms are instantiated with the ones from Ch, (b) as variables s_{ij} appear in both $\mathrm{IND}^{\text{cpa},b}$ and the specification of Ch, but during game execution they would always carry the same values, they were unified, (c) the variables r_{ij} from the Ch specification were renamed to r'_{ij} (the game variables r_{ij} were not renamed). Further, in line 13 we added an instruction that populates associative array $\mathbf{L}$ with entries that map associated-data–ciphertext pairs established by the AEAD algorithm Enc to the messages they decrypt

to. As none of the steps changes the output of the game we have $\Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] = \Pr[G^b_0(\mathcal{A}) \Rightarrow 1]$ for any $\mathcal{A}$ and $b \in \{0,1\}$.

Consider next the games G^b_1 in Figure 10. The difference to G^b_0 is that they replace the invocation of the AEAD algorithm Dec by a table look-up using associative array $\mathbf{L}$. The key argument of why this is possible is that in the IND-CPA setting the adversary remains passive, i.e., it only queries the rcv oracle on ciphertexts that were output by the bc oracle before. Inspection shows that the mechanics enforced by lines 21–22 indeed ensure that the vectors $(j, i, r'_{ij}, \mathbf{vc}, c)$ appearing in line 31 were first added to array $\mathbf{L}$ in line 13. Thus, by the perfect correctness of AEAD, we have $\Pr[G^b_0(\mathcal{A}) \Rightarrow 1] = \Pr[G^b_1(\mathcal{A}) \Rightarrow 1]$ for any $\mathcal{A}$ and $b \in \{0,1\}$.

Observe now that in the rcv oracle of games G^b_1 the messages m recovered in line 31 are never used. (That is, they are stored in $\mathbf{Q}_{ij}$ and then removed again, but not ever any game action depends on their value.) We thus define games G^b_2 that are like G^b_1 except that in line 13 the value $\diamond$ is stored in $\mathbf{L}$ instead of message m_b. We obtain $\Pr[G^b_1(\mathcal{A}) \Rightarrow 1] = \Pr[G^b_2(\mathcal{A}) \Rightarrow 1]$.

In games G^b_2 the Enc invocation of line 12 can be simulated using the Enc oracle provided by an IND-CPA challenger of the AEAD scheme. More precisely, there exists a straight-forward reduction $\mathcal{B}$ such that $\Pr[G^b_2(\mathcal{A}) \Rightarrow 1] = \Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{AEAD}}(\mathcal{B}) \Rightarrow 1]$, for any $\mathcal{A}$ and $b \in \{0,1\}$.

All in all we obtain $|\Pr[\mathrm{IND}^{\mathrm{cpa},1}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\mathrm{cpa},0}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]| = |\Pr[\mathrm{IND}^{\mathrm{cpa},1}_{\mathsf{AEAD}}(\mathcal{B}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\mathrm{cpa},0}_{\mathsf{AEAD}}(\mathcal{B}) \Rightarrow 1]|$, and thus the claim. □

Games $G^b_0(\mathcal{A}), G^b_1(\mathcal{A})$
00 $\mathbf{L}[\,] \leftarrow \emptyset$; $K \leftarrow_{\$} \mathsf{Gen}$
01 For $i \leftarrow 1$ to N:
02 $rej_i \leftarrow \mathrm{F}$
03 $b_i \leftarrow 0$; $\mathbf{vc}_i \leftarrow \mathbf{0}_N$
04 For $j \leftarrow 1$ to N, $j \neq i$:
05 $s_{ij}, r_{ij}, r'_{ij}, d_{ij} \leftarrow 0$
06 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$; $\mathbf{Q}_{ij}[\,] \leftarrow \emptyset$
07 $b' \leftarrow_{\$} \mathcal{A}^{\mathrm{bc,rcv}}$
08 Stop with b'

Oracle $\mathrm{bc}(i, m_0, m_1)$
09 If rej_i: Goto line 20
10 For all $j \in [1\,..\,N]$, $j \neq i$:
11 $ad_j \leftarrow i \,\|\, j \,\|\, s_{ij} \,\|\, \mathbf{vc}_i$
12 $c_j \leftarrow \mathsf{Enc}(K; ad_j, m_b)$
13 $\mathbf{L}[i, j, s_{ij}, \mathbf{vc}_i, c_j] \leftarrow m_b$
14 $D_j \leftarrow (\mathbf{vc}_i, c_j)$
15 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D_j$
16 $s_{ij} \leftarrow s_{ij} + 1$
17 Give $(\mathtt{s}, i, j, D_j)$ to $\mathcal{A}$
18 $b_i \leftarrow b_i + 1$
19 $\mathbf{vc}_i[i] \leftarrow b_i$
20 Return

Oracle $\mathrm{rcv}(i, j, D)$
21 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
22 Stop with 0
23 If rej_i: Goto line 38
24 Parse D as $(\mathbf{vc}, c)$
25 If parsing fails:
26 $rej_i \leftarrow \mathrm{T}$; Goto line 38
27 $ad \leftarrow j \,\|\, i \,\|\, r'_{ij} \,\|\, \mathbf{vc}$
28 Only G_0: $m \leftarrow \mathsf{Dec}(K; ad, c)$
29 Only G_0: If decryption fails:
30 Only G_0: $rej_i \leftarrow \mathrm{T}$; Goto line 38
31 Only G_1: $m \leftarrow \mathbf{L}[j, i, r'_{ij}, \mathbf{vc}, c]$
32 $\mathbf{Q}_{ij}[r'_{ij}] \leftarrow (\mathbf{vc}, m)$
33 $r'_{ij} \leftarrow r'_{ij} + 1$
34 While exist $\mathbf{vc}', m', j' \neq i$ s.t. $(\mathbf{vc}', m') = \mathbf{Q}_{ij'}[d_{ij'}]$ and $\mathbf{vc}' \leq \mathbf{vc}_i$:
35 Give $(\mathtt{d}, i, j', \diamond)$ to $\mathcal{A}$
36 $d_{ij'} \leftarrow d_{ij'} + 1$
37 $\mathbf{vc}_i[j'] \leftarrow d_{ij'}$
38 $r_{ij} \leftarrow r_{ij} + 1$
39 Return

Fig. 10. Games G^b_0, G^b_1, $b \in \{0,1\}$, used in the IND-CPA proof of Theorem 2. Games G^b_0 include all lines with exception of line 31, and games G^b_1 include all lines with exception of lines 28–30.

Theorem 3 (INT-CTXT security). *Let* Ch *be the broadcast channel constructed in Figure 9 from reliable point-to-point connections and an AEAD scheme* AEAD*. If the AEAD scheme offers (one-time) integrity of ciphertexts (INT-CTXT), also* Ch *offers integrity of ciphertexts (INT-CTXT). More precisely, for every adversary* $\mathcal{A}$ *against* Ch *there exists an adversary* $\mathcal{B}$ *against* AEAD *such that*

$$\mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{Ch},N}(\mathcal{A}) \leq \mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{AEAD}}(\mathcal{B}) \ .$$

The running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*. Further,* $\mathcal{B}$ *poses at most as many* Enc *and* Dec *queries as* $\mathcal{A}$ *poses* bc *and* rcv *queries, respectively.*

Proof. Consider game G_0 from Figure 11. It is identical to game $\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}$ from Figure 6, but with the following rewriting steps applied: (a) the abstract bc and rcv algorithms are instantiated with the ones from Ch, (b) as variables s_{ij} appear in both $\mathrm{INT}^{\mathrm{ctxt}}$ and the specification of Ch, but during game execution they would always carry the same values, they were unified, (c) the variables r_{ij} from the Ch specification were renamed to r'_{ij} (the game variables r_{ij} were not renamed). Further, in line 13 we added an instruction that populates a set L with the associated-data–ciphertext pairs that emerge in the processing of the bc oracle. As none of the steps changes the output of the game we have $\Pr[\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] = \Pr[G_0(\mathcal{A}) \Rightarrow 1]$ for any $\mathcal{A}$.

Consider next the game G_1 in Figure 11. The difference to G_0 is that in lines 32–33 it has an added abort instruction that is executed if the Dec invocation in line 29 fails to reject a ciphertext that was not created by Enc before (in line 12). The probability that this condition is ever fulfilled is bounded by the INT-CTXT advantage of an AEAD adversary: There exists an obvious reduction $\mathcal{B}$ such that $|\Pr[G_0(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1]| \leq \Pr[\mathrm{INT}^{\mathrm{ctxt}}_{\mathsf{AEAD}}(\mathcal{B}) \Rightarrow 1]$.

Let us finally assess the probability $\Pr[G_1(\mathcal{A}) \Rightarrow 1]$. To stop with 1, game G_1 needs to run into either line 15 or line 37 with $psv_i = \mathrm{F}$ for some party $i \in [1\,..\,N]$. Note that the flags psv_i are initially set to T for all parties, and that they are cleared in exclusively line 23, namely when a datagram is provided to the rcv oracle that is not in synchrony with what the bc oracle output before. Consider thus a query (i, j, D) to rcv where D is not authentic such that the psv_i flag of user i is cleared. If the query is posed and condition $rej_i = \mathrm{T}$ is fulfilled, or if $rej_i = \mathrm{T}$ is set by lines 27 or 31 during the processing of the query, then by lines 09, 24, 27, and 31 the instructions in lines 15 and 37 become unreachable (within all further queries involving participant i). The one remaining possibility for stopping with 1 is that line 32 is reached during the query in which flag psv_i is cleared. But recall that the flag was cleared due to an unauthentic ciphertext. This means that line 33 will abort the game with outcome 0. The conclusion is that for no participant i the game will run into a 'Stop with 1' instruction. This means $\Pr[G_1(\mathcal{A}) \Rightarrow 1] = 0$. The claim follows. □

Games $G_0(\mathcal{A}), G_1(\mathcal{A})$
00 $L \leftarrow \emptyset$; $K \leftarrow_{\$}$ Gen
01 For $i \leftarrow 1$ to N:
02 $psv_i \leftarrow \mathrm{T}$; $rej_i \leftarrow \mathrm{F}$
03 $b_i \leftarrow 0$; $\mathbf{vc}_i \leftarrow \mathbf{0}_N$
04 For $j \leftarrow 1$ to N, $j \neq i$:
05 $s_{ij}, r_{ij}, r'_{ij}, d_{ij} \leftarrow 0$
06 $\mathbf{D}_{ij}[\,] \leftarrow \emptyset$; $\mathbf{Q}_{ij}[\,] \leftarrow \emptyset$
07 $\mathcal{A}^{\mathrm{bc,rcv}}$
08 Stop with 0

Oracle bc(i, m)
09 If rej_i: Goto line 21
10 For all $j \in [1\,..\,N]$, $j \neq i$:
11 $ad_j \leftarrow i \,\|\, j \,\|\, s_{ij} \,\|\, \mathbf{vc}_i$
12 $c_j \leftarrow \mathsf{Enc}(K; ad_j, m)$
13 $L \leftarrow L \cup \{(i, j, s_{ij}, \mathbf{vc}_i, c_j)\}$
14 $D_j \leftarrow (\mathbf{vc}_i, c_j)$
15 If $\neg psv_i$: Stop with 1
16 $\mathbf{D}_{ij}[s_{ij}] \leftarrow D_j$
17 $s_{ij} \leftarrow s_{ij} + 1$
18 Give (s, i, j, D_j) to $\mathcal{A}$
19 $b_i \leftarrow b_i + 1$
20 $\mathbf{vc}_i[i] \leftarrow b_i$
21 Return

Oracle rcv(i, j, D)
22 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
23 $psv_i \leftarrow \mathrm{F}$
24 If rej_i: Goto line 41
25 Parse D as $(\mathbf{vc}, c)$
26 If parsing fails:
27 $rej_i \leftarrow \mathrm{T}$; Goto line 41
28 $ad \leftarrow j \,\|\, i \,\|\, r'_{ij} \,\|\, \mathbf{vc}$
29 $m \leftarrow \mathsf{Dec}(K; ad, c)$
30 If decryption fails:
31 $rej_i \leftarrow \mathrm{T}$; Goto line 41
32 Only G_1: If $(j, i, r'_{ij}, \mathbf{vc}, c) \notin L$:
33 Only G_1: Stop with 0
34 $\mathbf{Q}_{ij}[r'_{ij}] \leftarrow (\mathbf{vc}, m)$
35 $r'_{ij} \leftarrow r'_{ij} + 1$
36 While exist $\mathbf{vc}', m', j' \neq i$ s.t.
 $(\mathbf{vc}', m') = \mathbf{Q}_{ij'}[d_{ij'}]$ and $\mathbf{vc}' \leq \mathbf{vc}_i$:
37 If $\neg psv_i$: Stop with 1
38 Give (d, i, j', m') to $\mathcal{A}$
39 $d_{ij'} \leftarrow d_{ij'} + 1$
40 $\mathbf{vc}_i[j'] \leftarrow d_{ij'}$
41 $r_{ij} \leftarrow r_{ij} + 1$
42 Return

Fig. 11. Games G_0, G_1 used in the INT-CTXT proof of Theorem 3. Game G_0 includes all lines with exception of lines 32–33, and game G_1 includes all lines.

Corollary 1 (IND-CCA security). *Let* Ch *be the broadcast channel constructed in Figure 9 from reliable point-to-point connections and an AEAD scheme* AEAD*. If the AEAD scheme offers both (one-time)*

indistinguishability under chosen-plaintext attacks (IND-CPA) and (one-time) integrity of ciphertexts (INT-CTXT), then Ch *offers indistinguishability under chosen-ciphertext attacks (IND-CCA).*

Proof. Combine the results of Theorems 1, 2 and 3. □

References

1. Badertscher, C., Matt, C., Maurer, U., Rogaway, P., Tackmann, B.: Augmented secure channels and the goal of the TLS 1.3 record layer. In: Au, M.H., Miyaji, A. (eds.) ProvSec 2015. LNCS, vol. 9451, pp. 85–104. Springer, Heidelberg, Germany, Kanazawa, Japan (Nov 24–26, 2015)
2. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS. pp. 394–403. IEEE Computer Society Press, Miami Beach, Florida (Oct 19–22, 1997)
3. Bellare, M., Kohno, T., Namprempre, C.: Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In: Atluri, V. (ed.) ACM CCS 02. pp. 1–11. ACM Press, Washington D.C., USA (Nov 18–22, 2002)
4. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg, Germany, Kyoto, Japan (Dec 3–7, 2000)
5. Cachin, C., Guerraoui, R., Rodrigues, L.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)
6. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 524–541. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
7. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard) (Aug 2008), http://www.ietf.org/rfc/rfc5246.txt, updated by RFCs 5746, 5878, 6176
8. Goldberg, I., Ustaoglu, B., Van Gundy, M., Chen, H.: Multi-party off-the-record messaging. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM CCS 09. pp. 358–368. ACM Press, Chicago, Illinois, USA (Nov 9–13, 2009)
9. Gundy, M.D.V., Chen, H.: OldBlue: Causal Broadcast In A Mutually Suspicious Environment. Tech. rep. (November 2012), http://matt.singlethink.net/projects/mpotr/oldblue-draft.pdf
10. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)
11. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: A systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)
12. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21(7), 558–565 (1978)
13. Liu, H., Vasserman, E.Y., Hopper, N.: Improved group off-the-record messaging. In: Sadeghi, A., Foresti, S. (eds.) Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013. pp. 249–254. ACM (2013)
14. Marlinspike, M.: Advanced cryptographic Ratcheting. Blog (2013), https://whispersystems.org/blog/advanced-ratcheting
15. Marson, G., Poettering, B.: Security Notions for Bidirectional Channels. IACR Transactions on Symmetric Cryptology 2017(1), 405–426 (2017), http://tosc.iacr.org/index.php/ToSC/article/view/602
16. Private group messaging (2014), https://whispersystems.org/blog/private-groups
17. Off-the-Record Messaging. http://otr.cypherpunks.ca (2016)
18. Paterson, K.G., Ristenpart, T., Shrimpton, T.: Tag size does matter: Attacks and proofs for the TLS record protocol. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 372–389. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)
19. Perrin, T.: Double Ratchet Algorithm. GitHub wiki (2016), https://github.com/trevp/double_ratchet/wiki
20. Reardon, J., Kligman, A., Agala, B., Goldberg, I.: KleeQ: Asynchronous key management for dynamic ad-hoc networks. Tech. Rep. CACR 2007-03 (January 2007), http://cacr.uwaterloo.ca/techreports/2007/cacr2007-03.pdf
21. Reiter, M.K., Birman, K.P.: How to securely replicate services. ACM TOPLAS 16(3), 986–1009 (1994)
22. Reiter, M.K., Gong, L.: Securing causal relationships in distributed systems. Comput. J. 38(8), 633–642 (1995), http://dx.doi.org/10.1093/comjnl/38.8.633
23. Rogaway, P.: Authenticated-encryption with associated-data. In: Atluri, V. (ed.) ACM CCS 02. pp. 98–107. ACM Press, Washington D.C., USA (Nov 18–22, 2002)

24. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006)
25. Schwarz, R., Mattern, F.: Detecting causal relationships in distributed computations: In search of the holy grail. Distributed Computing 7(3), 149–174 (1994), http://dx.doi.org/10.1007/BF02277859
26. Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M.: SoK: Secure messaging. In: 2015 IEEE Symposium on Security and Privacy. pp. 232–249. IEEE Computer Society Press, San Jose, CA, USA (May 17–21, 2015)
27. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard) (Jan 2006), http://www.ietf.org/rfc/rfc4251.txt

A Correctness of waiting causal broadcast

We prove that the causal broadcast protocol from Figure 4 (on page 8) fulfills the correctness property specified in Section 3.3.

Theorem 4 (Correctness of waiting causal broadcast). *Let* $\mathsf{Ch} = (\mathsf{init}, \mathsf{bc}, \mathsf{rcv})$ *be a broadcast channel constructed according to Figure 4. If the network provides FIFO delivery then* Ch *provides a causal broadcast channel. More precisely, for every (even unbounded) scheduler* $\mathcal{A}$ *we have* $\mathrm{Adv}^{\mathrm{func}}_{\mathsf{Ch}}(\mathcal{A}) = 0$.

Proof. The core of the proof is to use the assumption of reliability of the network, i.e., that datagrams are received in FIFO order and without modification, to show that delivery of messages also happens without modification and according to a causal order. Specifically, we show that if pairs $(\mathbf{vc}, m)$ are enqueued in a FIFO way (in line 19 of Figure 4), they are dequeued in a causal way (in line 21).

Consider the FUNC game from Figure 3, and fix arbitrary $i, j \in [1\,..\,N]$, $i \neq j$. Note that the assumption of FIFO delivery on the network is fulfilled as long as $\mathcal{A}$ remains passive, i.e., $psv_i = \mathrm{T}$ for all $i \in [1\,..\,N]$. We can thus assume wlog that $\mathcal{A}$ never sets $psv_i \leftarrow \mathrm{F}$ in line 17—otherwise it would be declared active and could not win the game in the first place—and hence whenever $\mathrm{rcv}(i, j, D)$ is invoked it holds that $s_{ji} > r_{ij}$ and $\mathbf{D}_{ji}[r_{ij}] = D$, meaning precisely that every datagram D that user i receives from user j has been previously sent by j and that it arrives according to the corresponding sending order. In particular, queries $\mathrm{rcv}(i, j, *)$ have the effect that a prefix of the datagram sequence $\mathbf{D}_{ji}$ (i.e., datagrams sent by user j to user i) is stored in $\mathbf{Q}_{ij}$.

Now, given the above assumption on the network, we will show that messages are delivered according to causal order and without modification (formally: the condition specified in line 27 of Figure 3 never holds). Observe that, by construction, every $\mathsf{bc}(st_j; m)$ operation induces $\mathsf{snd}(j, i, D)$ operations, for all $i \neq j$, where $D = (\mathbf{vc}, m)$. Thus, the sequence of pairs stored in $\mathbf{Q}_{ij}$ contains a prefix of the sequence of messages m broadcast by user j together with their associated counter vectors $\mathbf{vc}$ (see line 19 of Figure 4).

Further, by construction, dlv operations happen as long as there exist $\mathbf{vc}'$, m', $j' \neq i$ such that $\mathbf{Q}_{ij'}[d_{ij'}] = (\mathbf{vc}', m')$ and $\mathbf{vc}' \leq \mathbf{vc}_i$ (see line 21 of Figure 4). Let us consider any i, j', $\mathbf{vc}'$, and m' that meet this property. Then, when invoking the corresponding delivery subroutine $\mathsf{dlv}(i, j', m')$—meaning that m' is on its way to be delivered from user j' to user i, but has not been delivered yet—in game FUNC we have $psv_i = \mathrm{T}$ (in line 26) by assumption, and thus

$$d_{ij'} < r_{ij'} \leq s_{j'i} = b_{j'} \ ,$$

where the first relation holds by construction (the numbers of dlv operations is dominated by the number of rcv operations), the second by the assumption of FIFO delivery, and the third again by construction.

From the assumption of FIFO delivery we also can conclude that pair $(\mathbf{vc}', m')$ coincides with datagram $\mathbf{D}_{j'}[d_{ij'}]$, and hence by construction we have $m' = \mathbf{M}_{j'}[d_{ij'}]$ and $\mathbf{vc}' = \mathbf{VC}_{j'}[d_{ij'}]$. Finally, note that $\mathbf{vc}' \leq \mathbf{vc}_i$ (this is again by construction, otherwise pair $(\mathbf{vc}', m')$ could not be dequeued from $\mathbf{Q}_{ij'}$ yet), hence in particular $\mathbf{VC}_{j'}[d_{ij'}] \leq \mathbf{vc}_i$. Putting all together we have $b_{j'} > d_{ij'}$ and $\mathbf{M}_{j'}[d_{ij'}] = m'$ and $\mathbf{VC}_{j'}[d_{ij'}] \leq \mathbf{vc}_i$, (as in line 21 of Figure 3,) meaning that user i delivers message m' from j' in correct causal order. □

Note that the statement above also implies the correctness of the construction of Figure 9, as long as the AEAD in use is also correct.

B Definitions of AEAD

We recall the definition of AEAD from [23], slightly adapting it to a new syntax. The security properties that we give interpolate the ones of [23,24].

Definition 2 (AEAD). *A scheme providing* authenticated encryption with associated data (AEAD) *for a message space $\mathcal{M}$ and an associated data space $\mathcal{AD}$ consists of a key space $\mathcal{K}$, a ciphertext space $\mathcal{C}$, and three algorithms,* $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$*, with the following syntax. The key generation algorithm* Gen *is randomized, takes no input, and outputs a key $K \in \mathcal{K}$. The encryption algorithm* Enc*, which may be randomized or deterministic, takes a key K, an associated data string $ad \in \mathcal{AD}$, and a message $m \in \mathcal{M}$; its output is a ciphertext $c \in \mathcal{C}$. Finally, the decryption algorithm* Dec *is deterministic and takes a key K, an associated data string ad, and a ciphertext c; its output is a message m, or an indication that decryption failed (the latter is often encoded by writing $m = \perp$). A helpful shorthand form for expressing this syntactical convention is*

$$\mathsf{Gen} \to \mathcal{K} \qquad \textit{and} \qquad \mathcal{K} \times \mathcal{AD} \times \mathcal{M} \to \mathsf{Enc} \to \mathcal{C} \qquad \textit{and} \qquad \mathcal{K} \times \mathcal{AD} \times \mathcal{C} \to \mathsf{Dec} \to \mathcal{M}/\perp \ .$$

For correctness we require that for all $K \in [\mathsf{Gen}]$, $ad \in \mathcal{AD}$, and $m \in \mathcal{M}$, if $c \in [\mathsf{Enc}(K, ad, m)]$ then $\mathsf{Dec}(K, ad, c) = m$.

Note that we do not require Enc to be a randomized algorithm. The reason is that randomization is not necessary in our application where the associated data input to Enc never repeats. Correspondingly, the security definitions we give for AEAD are one-time notions in the sense that they do not promise anything if the *ad* input is not fresh for each invocation of Enc. This is a weaker requirement than standard, and thus allows for more efficient instantiations. (Or, put differently, if 'only' a randomized or nonce-based AEAD scheme is at hand, it doesn't hurt to use it). We formalize IND-CPA security as a confidentiality notion and INT-CTXT security as an authenticity notion. It is a folklore result that an AEAD scheme that fulfills both notions is actually IND-CCA secure.

Definition 3 (IND-CPA). *We say scheme* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *provides (one-time) indistinguishability under chosen-plaintext attacks (IND-CPA) if it is hard to distinguish the encryptions of two messages in a passive attack. Formally, to an adversary $\mathcal{A}$ we assign the advantage* $\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD}}(\mathcal{A}) = |\Pr[\mathrm{IND}^{\text{cpa},1}_{\mathsf{AEAD}}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{IND}^{\text{cpa},0}_{\mathsf{AEAD}}(\mathcal{A}) \Rightarrow 1]|$*, where the games are in Figure 12. Intuitively, the scheme is secure if all realistic adversaries have small advantage.*

Game $\mathrm{IND}^{\text{cpa},b}_{\mathsf{AEAD}}(\mathcal{A})$	**Oracle** $\mathrm{Enc}(ad, m_0, m_1)$
00 $\mathbf{D} \leftarrow \emptyset$	04 If $(ad, \cdot) \in \mathbf{D}$: Stop with 0
01 $K \leftarrow_{\$} \mathsf{Gen}$	05 $c \leftarrow_{\$} \mathsf{Enc}(K; ad, m_b)$
02 $b' \leftarrow_{\$} \mathcal{A}^{\mathrm{Enc}}$	06 $\mathbf{D} \leftarrow \mathbf{D} \cup \{(ad, c)\}$
03 Stop with b'	07 Return c

Fig. 12. One-time IND-CPA security games for AEAD. Note that line 04 encodes the requirement for a fresh associated data string per Enc query.

Definition 4 (INT-CTXT). *We say scheme* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *provides (one-time) integrity of ciphertexts (INT-CTXT) if it is hard to find ciphertexts (beyond regularly created ones) that validly decrypt. Formally, to an adversary $\mathcal{A}$ we assign the advantage* $\mathrm{Adv}^{\text{int-ctxt}}_{\mathsf{AEAD}}(\mathcal{A}) = \Pr[\mathrm{INT}^{\text{ctxt}}_{\mathsf{AEAD}}(\mathcal{A}) \Rightarrow 1]$*, where the game is in Figure 13. Intuitively, the scheme is secure if all realistic adversaries have small advantage.*

Game $\mathrm{INT}^{\text{ctxt}}_{\mathsf{AEAD}}(\mathcal{A})$	**Oracle** $\mathrm{Enc}(ad, m)$	**Oracle** $\mathrm{Dec}(ad, c)$
00 $\mathbf{D} \leftarrow \emptyset$	04 If $(ad, \cdot) \in \mathbf{D}$: Stop with 0	08 $m \leftarrow \mathsf{Dec}(K; ad, c)$
01 $K \leftarrow_{\$} \mathsf{Gen}$	05 $c \leftarrow_{\$} \mathsf{Enc}(K; ad, m)$	09 If $(ad, c) \notin \mathbf{D}$ and $m \neq \perp$:
02 $\mathcal{A}^{\mathrm{Enc},\mathrm{Dec}}$	06 $\mathbf{D} \leftarrow \mathbf{D} \cup \{(ad, c)\}$	10 Stop with 1
03 Stop with 0	07 Return c	11 Return m

Fig. 13. One-time INT-CTXT security game for AEAD. Note that line 04 encodes the requirement for a fresh associated data string per Enc query.