This article is a minor revision of the version published in the proceedings of IEEE CSF 2018. The full version is available in the IACR eprint archive as article 2017/786. Note that a prior version of this article appeared with a different title and only two of the three authors.

A Cryptographic Look at Multi-Party Channels

Patrick Eugster
University of Lugano
Lugano, Switzerland
patrick.thomas.eugster @ usi.ch

Giorgia Azzurra Marson
NEC Laboratories Europe
Heidelberg, Germany
giorgia.marson @ neclab.eu

Bertram Poettering
Royal Holloway, University of London
Egham, Surrey, United Kingdom
bertram.poettering @ rhul.ac.uk

***Abstract*—Cryptographic channels aim to enable authenticated and confidential communication over the Internet. The general understanding seems to be that providing security in the sense of authenticated encryption for every (unidirectional) point-to-point link suffices to achieve this goal. As recently shown (in FSE17/ToSC17), however, the security properties of the unidirectional links do not extend, in general, to the bidirectional channel as a whole. Intuitively, the reason for this is that the increased interaction in bidirectional communication can be exploited by an adversary. The same applies, *a fortiori*, in a multi-party setting where several users operate concurrently and the communication develops in more directions. In the cryptographic literature, however, the targeted goals for group communication in terms of channel security are still unexplored. Applying the methodology of provable security, we fill this gap by defining exact (game-based) authenticity and confidentiality goals for broadcast communication, and showing how to achieve them. Importantly, our security notions also account for the causal dependencies between exchanged messages, thus naturally extending the bidirectional case where causal relationships are automatically captured by preserving the sending order. On the constructive side we propose a modular and yet efficient protocol that, assuming only point-to-point links between users, leverages (non-cryptographic) broadcast and standard cryptographic primitives to a full-fledged broadcast channel that provably meets the security notions we put forth.**

***Index Terms*—secure channels, broadcast communication, causality preservation, integrity, confidentiality**

I. Introduction

One of the fundamental applications of cryptography is secure end-to-end 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 reliable TCP/IP-based connections are the TLS protocol suite [1] and the SSH remote shell protocol [2]. Due to their widespread deployment, both TLS and SSH have been extensively studied in the cryptographic literature. In such analyses, confidentiality, integrity, and the protection against reordering and replay attacks were identified as the main security goals.

Somewhat surprisingly, the classic formalizations of the mentioned security properties (e.g., the definitions in [3]) consider only channel types that allow for transmitting messages *in one direction*, from a specific sender to a specific receiver. In particular they do not cover the (in the TCP/IP setting more realistic) scenario in which two parties communicate simultaneously *in both directions*. This mismatch between how TLS and SSH are classically modeled (theory) and how they are used (practice) was pointed out only recently, in [4]. In more detail, [4] extends confidentiality and integrity notions from the unidirectional setting to the bidirectional case, it identifies what can go wrong with security when constructing a bidirectional channel from two unidirectional ones, and it concludes with noting that explicitly including the bidirectional flavor of interactivity in the model is crucial for understanding and reaching security.

While [4] is a first step in an important direction, it does not reach beyond the case of *two*-party channels. Meanwhile, so called *multi-party* or *broadcast* communication is being considered to continue increasing in relevance in the future, with certain estimates in 2006 already placing it at 90% of the entire Internet traffic [5]. In such broadcast transmissions, predominant for instance in group messaging systems but also in automated communication systems like interconnected bank computers, all participants may transmit, and all transmissions target and are expected to eventually reach all other participants. Given the clearly more involved case of such broadcast communication and corresponding channels, and the previous above-mentioned experiences with uni-directional vs. bi-directional unicast channels, attempting to directly infer the security of broadcast channels from the security of their underlying two-party sub-channels seems a naive choice that might lead to fatal results.

Intuitively, when more than two users participate in a conversation, the usual requirement of sequential delivery (i.e., that messages originating from a given user are received by the other users in the same order they were sent) is unlikely to be sufficient to guarantee that users have a faithful view on the conversation. Below we pinpoint some specific problematic situations, illustrated in Figure 1, that may occur in a (three-party) conversation even though messages are transmitted and delivered according to the sending order.

Consider a situation like the one illustrated in Figure 1a. Alice asks Bob for the exact time of a party planned at his place that is loosely 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 1b 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.

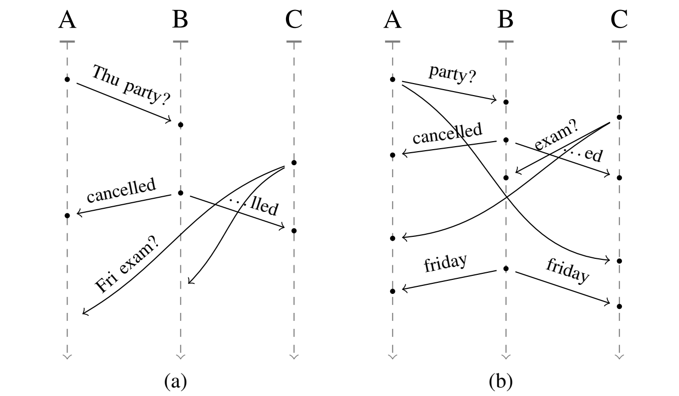

Fig. 1. Misunderstandings caused by causality violations. Vertical dashed lines symbolize per-party timelines (time progresses top-down), bullets represent broadcast and delivery actions.

Due to some delay affecting the network—possibly organized by an adversary—Charlie, however, receives Bob's message first, and only then Alice's question. In the end, not only Charlie misses the exam, he even shows up at Bob's place on Friday. 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 are typically assumed to have full control over message delivery and can easily arrange the corresponding delays.

Situations like the ones described in Figure 1 are common in the context of multi-party communication. The technical solution to misunderstandings of this type, classic in the domain of distributed systems, is to enforce that deliveries preserve the *causality* relation among messages. With other words, no participant shall receive a message m if it has not yet received every message sent before m (i.e., whose content may have influenced that of m). Concretely, in both situations from Figure 1, causality preservation would ensure that Charlie receives Alice's message before receiving anything from Bob, thus avoiding misunderstandings. Maintaining causality is thus highly desired, if not crucial, in many applications such as publish-subscribe systems, online stores, booking systems, (event-based) intrusion-detection systems, and stock trading. Fortunately, several protocols that efficiently realize a causal broadcast network exist in the distributed system literature (e.g., [6], [7], [8]). However, such protocols do not provably 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.)

A. Contribution

Considering the wide-spread use of the above-mentioned applications, and their relevance, the security of causal broadcast channels needs to be looked at more closely. Understanding causality from the perspective of cryptography as well as constructing corresponding secure schemes is the main goal of the present work. Concretely, we treat secure broadcast channels as a generalization of bidirectional channels to the multi-user setting and, to capture a realistic scenario, we let such channels run on top of a point-to-point network (implemented for example with TCP/IP connections). Following the approach of provable security, we give rigorous formalizations of the novel functionality requirements (causality preservation) as well as of appropriate security notions for this channel type, and propose a secure construction that provably meets these goals.

B. Organization

The paper is organized as follows. In Section II we discuss closely related works. The technical contribution of our paper starts in Section III by introducing some notation, specifying a general syntax for (not necessarily cryptographic) broadcast channels, and recalling some concepts related to causality preservation from the distributed systems literature. We give a formal functionality definition and show how causal broadcast can be achieved from simpler network primitives in Section IV, and proceed with defining integrity and confidentiality notions for broadcast channels in Section V, where we also investigate how our notions are related to each other. Finally, in Section VI we propose a cryptographic protocol that provably meets the strongest confidentiality and integrity properties proposed in this paper. Section VII concludes with future directions.

II. Related work

A. Cryptography

Academic works studying the properties of secure channels abound in the cryptographic literature. Our security notions naturally extend the channel model of Bellare, Kohno, and Namprempre [3], introduced for the analysis of the SSH channel component and later adopted (and adapted) by numerous followups (e.g., [9], [10], [11], [4], [12]). We note that all of these works are restricted to the two-party scenario, thus ignoring potential security issues related to (the alteration of) causal dependencies among exchanged messages.

Following that line of work, our security notions are game-based. An alternative approach for defining security of cryptographic protocols is via simulation-based notions such as those employed in the frameworks of universal composability (UC) by Canetti [13] and of constructive cryptography (CC) by Maurer and Renner [14]. Both UC and CC aim to facilitate the modular design of complex protocols by explicitly targeting the appealing feature of composability, which enforces the conditions in which a given scheme can be safely used within any (possibly unknown) larger system. While it turns out that certain cryptographic primitives are less accessible to simulation-based modeling, we believe that this is not the case for causal broadcast channels and that there exist natural analogues of our notions in the UC and CC worlds. (In particular we expect that the schemes proposed in this paper will also be secure in their simulation-based counterparts.)

B. Distributed systems

Starting with Lamport's groundbreaking work on (distributed) logical time [15], 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 [16].

Security challenges related to causality preservation have been recognized in the context of state machine replication (SMR), a popular technique that implements a fault-tolerant service by replicating it across multiple servers. In terms of (functionality and) security, a replicated service should behave identically to a centralized one in the eye of clients, even if some of the servers may crash prematurely or be adversarially controlled. Among other measures to realize secure replicated services, the works of Reiter, Birman, and Gong [17] and of Reiter and Birman [18] unveil attacks that exploit causality violations based on compromised servers possibly colluding with clients, and propose heuristic countermeasures. In the same vein Reiter and Gong [19] identify security goals related to the detection of causality violations and discuss a number of techniques to achieve such goals. These works pioneer the treatment of causality preservation as a security property, dubbed *input causality* in subsequent works. Informally speaking, input causality means that no honest server can deliver a request until it has delivered all causally preceding requests (i.e., all requests that may have contributed to that request).

The seminal work of Cachin et al. [20] tackles SMR from a cryptographic perspective. Most importantly for the present work, Cachin et al. introduce the notion of *secure causal broadcast* which specifically targets input causality, and show how to realize it by combining atomic broadcast (a.k.a. total order broadcast) with public-key primitives. Building on [20], Duan et al. [21] propose improved constructions for secure causal broadcast using more generic and efficient cryptographic building blocks (secret sharing and non-malleable commitment schemes).

While also explicitly targeting causality preservation, these works around SMR address a different application scenario and thus technical problem and adversary model than the present work. First off, the SMR work distinguishes between clients and servers, with processes potentially exhibiting arbitrary (Byzantine) failures. In contrast the present work strives for direct peer-based (server-less) communication among mutually trusting participants in an adversarial network. In order to keep the states of replicas from diverging, SMR also requires total ordering among messages in addition to causal ordering. On the one hand this may seem to make the problem harder, as achieving such a total order is as hard as solving the popular consensus problem [22], [23], which in turn is provably unachievable in an asynchronous setting [24] like the Internet. On the other hand, total order can be used to enforce causal order [7], and most of the above-mentioned works simply assume the existence of a total order broadcast substrate, focusing on retaining causal order in some corner cases (which may also exist in a benign setting depending on the exact properties of the total order broadcast cf. [8]).

Since neither we seek protection against Byzantine behavior (i.e., corrupt participants) nor we require total order delivery of messages, in our setting we achieve provably secure constructions from milder network assumptions, namely FIFO links rather than atomic broadcast (i.e., more directly than via total order), and symmetric cryptographic building blocks (i.e., authenticated encryption with associated data). Last but not least, our work is the first to the best of our knowledge to define and prove security of causal communication rigorously in the well-established game-base framework of modern cryptography.

C. Secure messaging systems

Group messaging applications provide a natural motivation for the need of causality preservation. We are not the first to identify causality preservation as a relevant (security) property for group messaging protocols. This property is indicated in [25] as an explicit target of the TextSecure v2 protocol, and according to the systematization of knowledge (SoK) paper of Unger et al. [26] specific configurations of OTR [27], GOTR [28], OldBlue [29], KleeQ [30], Axolotl [31], and TextSecure [32] offer some form of causality preservation (however, not on formal grounds).

The already mentioned SoK paper on secure messaging by Unger et al. [26] lists several properties that are targeted or claimed by popular messengers, among which *causality preservation*. According to the authors, the latter means that "implementations can avoid displaying a message before messages that causally precede it." So stated, this property resembles the functionality of a causal broadcast channel, but it does not address security. Our work provides formal definitions, in the style of modern cryptography, of causality preservation as a correctness property as well as corresponding security notions of integrity and confidentiality for causal communication. A closely-related notion, strictly stronger than causality preservation (in the sense that it implies the latter), is that of *global transcript*—"all participants see all messages in the same order" as per [26]. This very notion was also listed by Goldberg et al. [33] (under the name of *consensus*) as a desired target for improving the off-the-record protocol [27] to support group communication. The corresponding definitions given in [33], [26] are rather informal, and it is unclear to us whether in [33] the message ordering is meant to also respect causality (i.e., whether only 'atomic broadcast' or 'causal atomic broadcast' is meant). Another work targeting the *global transcript* property is by Reardon et al. [30]. These authors propose a heuristic protocol, named KleeQ, to provide secure group communication in ad-hoc networks with limited connectivity. In contrast to our work, [30] explicitly targets total order in addition to causality preservation, and achieves both goals by letting participants complete the sequence of messages seen so far by including all missing messages that their peers have already received. This approach clearly incurs a high communication complexity. Nevertheless, a performance comparison between KleeQ and our secure construction from Section VI is unfair given the discrepancy between network assumptions and targeted goals. To some extent, the goal of KleeQ is closer to atomic causal broadcast (although [30] does not consider fault tolerance) rather than to provide a secure causal channel as we define it. In addition to that, KleeQ allows participants to dynamically join (but not leave), while our setting considers a static set of participants.

III. Preliminaries

A. Notation

To initialize an empty associative array (dictionary) $\mathbf{X}$ 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, notation $y \leftarrow \mathcal{A}(x)$ 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 framework. 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 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.

B. Broadcast channels

We consider multiple parties[1] that exchange messages in a broadcast fashion using two dedicated algorithms: bcast (broadcast) and recv (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 bcast algorithm with that message, and when a participant receives a datagram[2] from the network they invoke the recv algorithm with the datagram and an identifier for the sender. Internally, the two algorithms may invoke the abstract subroutines send (send) and delv (deliver), where send initiates the transport of a given datagram to an indicated other user, and delv delivers a given message to the local user, indicating also the (alleged) sender of that datagram/message. Both the bcast and recv algorithms may be randomized and keep state between invocations.

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

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

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 send is invoked as per $\mathsf{send}(i, j, D)$ if datagram $D \in \mathcal{D}$ shall be sent by user i to user j. Similarly, subroutine delv is invoked as per $\mathsf{delv}(i, j, m)$ if message $m \in \mathcal{M}$ shall be delivered at user i, and the message was broadcast (allegedly) by user j. Both subroutines, send and delv, require $i \neq j$ and have no (explicit) output. A helpful shorthand form for expressing the two syntactical conventions is

$$[1\,..\,N] \times [1\,..\,N] \times \mathcal{D} \to \mathsf{send}$$
$$[1\,..\,N] \times [1\,..\,N] \times \mathcal{M} \to \mathsf{delv}$$

To broadcast a message $m \in \mathcal{M}$, users invoke the bcast algorithm as per $st' \leftarrow \mathsf{bcast}^{\mathsf{send},\mathsf{delv}}(st; m)$, where st, st' are the original and the updated state, respectively. Similarly, users can receive datagrams $D \in \mathcal{D}$ by invoking $st' \leftarrow \mathsf{recv}^{\mathsf{send},\mathsf{delv}}(st; j, D)$, 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 bcast, recv are thus

$$\mathcal{S} \times \mathcal{M} \to \mathsf{bcast}^{\mathsf{send},\mathsf{delv}} \to \mathcal{S}$$
$$\mathcal{S} \times [1\,..\,N] \times \mathcal{D} \to \mathsf{recv}^{\mathsf{send},\mathsf{delv}} \to \mathcal{S}$$

The interplay of the bcast, recv, send, delv algorithms and routines is also illustrated in Figure 2.

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, bcast, *and* recv, *that follow the syntax specified above. Note that* bcast *and* recv *are defined in respect to abstract subroutines* send *and* delv.

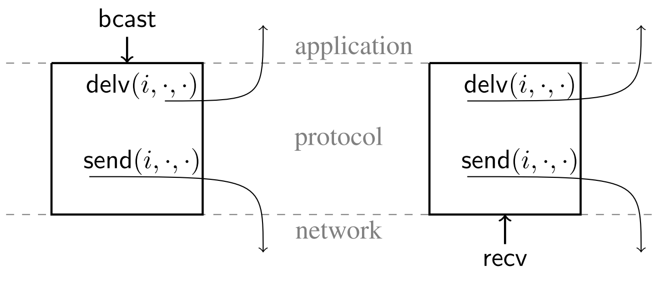

Fig. 2. Overview of broadcast and receiving algorithms and corresponding sending and delivery subroutines.

C. FIFO order, causal order, and logical timestamping

While in the syntactical specification of broadcast channels from Definition 1 the network interfaces (send and recv) in principle support different network types, in this work we assume reliable (point-to-point) FIFO links. With 'reliable FIFO' we mean that datagrams are received according to the sending order and without loss to the intended recipient, but without any timing guarantee (think of TCP). In particular, one can only expect that datagrams are *eventually* received (however, datagrams that do reach the recipients are guaranteed to arrive in the right order). Concretely, we formalize the (reliable) FIFO delivery condition by demanding that, at any point in time, for any two users $i, j \in [1 \mathinner{..} N]$ the sequence of datagrams that i receives from j is a prefix of the sequence of datagrams that j sent to i.

Our goal is to build broadcast channels that, in addition to FIFO delivery, also preserve causal relations among broadcast and delivered messages. Informally, causal delivery means that no party can deliver a given message unless they delivered all messages that have contributed to (the creation of) that message (i.e., they have been broadcast "before" the message according to Lamport's *happened before* relation [15]). For instance, in situations like the ones of Figure 1, causal delivery would guarantee that Charlie does not deliver Bob's message until he has delivered Alice's message.

To verify that messages are delivered in a causal order we borrow a technique developed in the domain of distributed systems that uses so-called *vector clocks* (or vector timestamps) for realizing waiting causal broadcast [34]. The idea is that each user $i \in [1 \mathinner{..} N]$ maintains a vector $\mathbf{vc}_i$ of counters that is initialized to $\mathbf{0}_N$ and updated upon bcast and delv invocations. Concretely, $\mathbf{vc}_i[i]$ is incremented whenever user i performs a broadcast operation, and $\mathbf{vc}_i[j]$ 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 $\mathbf{vc}_j$ be the vector clock of user j right before broadcasting m and let $\mathbf{vc}_i$ be the vector clock of user i immediately before delivering m from user j. Then the delivery of m at user i happens according to the causal order only once $\mathbf{vc}_j \leq \mathbf{vc}_i$, i.e., once user i has delivered all messages broadcast before. With a little unpacking, the condition above precisely means that, at the moment of delivering m, user i has already delivered all messages previously broadcast by user j (this is expressed by $\mathbf{vc}_j[j] \leq \mathbf{vc}_i[j]$), and has already delivered all messages that user j received so far from the other users (formally: $\mathbf{vc}_j[k] \leq \mathbf{vc}_i[k]$ for $k \notin \{i, j\}$).

The vector clock formalism provides us with a handy tool to verify that specific causality relations are met. In the next section we will make use of this tool to define the functionality (a.k.a. correctness) requirement of a causal broadcast protocol. *Lamport*('s own) *clocks* [15] are a well-known alternative to vector clocks. Roughly speaking, the former clocks collapse all the dimensions of the latter clocks into a single one. As a result, Lamport clocks only have a single component, which makes them "cheaper" to transmit in or along with messages. By confounding the clock spaces of all broadcasters, Lamport clocks however lead to false positives. More precisely, while Lamport clocks do capture all actual causal relationships and thus can be used for enforcing causal order, they also establish an order among unrelated messages, i.e., messages of which neither influenced the other. As a consequence, when used for enforcing causal order in broadcasting in a way similar to what was described above for vector clocks, Lamport clocks will lead to artificially delaying the delivery of certain messages which could be safely delivered without violating causal order. (In a more mathematical sense, causal order induces a partial order on events, whereas Lamport clocks yield a total order.)

IV. Causal broadcast channels

Functionality (and security) notions for cryptographic protocols are tighly coupled with the delivery properties provided by the underlying network. For this reason, specifying the network assumptions is the very first step in the modeling process. Typically, a cryptographic channel preserves the functionality of the network—loosely speaking, the only role of the cryptographic layer is to protect the communication over the otherwise insecure network. For instance, the functionality of a secure channel like TLS (which runs on top of TCP/IP) guarantees that messages are delivered to the intended recipient reliably and according to the sending order, as if they were sent directly over the network. Following this line of thought, a natural approach to build a cryptographic causal channel would be to start with a causal broadcast network, which already achieves on its own the expected functionality of this channel type, and then add a cryptographic layer on top of it to also provide security, e.g., to enforce that causality relations are met.

In this work we explore another option and only assume that the network offers unprotected FIFO links, leaving the task of preserving causal relations to the cryptographic protocol run on top. That is, we demand the cryptographic channel to provide, beyond security, also an extended functionality (effectively making the network and crypto layers overlap). We anticipate (and discuss further in Section VI-A) that letting the cryptographic channel handle the causality preservation results in saving half the management work. Notice also that our network assumption is met by the widely-deployed TCP/IP network protocol. Thus, the notion of cryptographic causal channel that we propose is suitable for a wide variety of applications running on the Internet.

A. Functionality

In this section we specify the functionality of causal broadcast channels built on top of (point-to-point) FIFO links. Intuitively, a 'proper' channel of this type should guarantee that if datagrams are received (via recv) in the same order they were sent (via send) then all delivered (via delv) messages have been previously broadcast (via bcast) by the reported senders; moreover, deliveries happen according to the *causal* order among messages.

We formalize this property through the (functionality) game FUNC in Figure 3. In this game a scheduler $\mathcal{A}$ is given access to algorithms bcast and recv through oracles bcast and recv so that it controls the messages broadcast and the datagrams received by all parties. Such a scheduler's goal

```
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 $\mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
09 Stop with 0

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

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

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

Proc $\mathsf{delv}(i, j, m)$
26 If $psv_i$:
27     If $b_j \leq d_{ij}$ or $\mathbf{M}_j[d_{ij}] \neq m$
       or $\mathbf{VC}_j[d_{ij}] \nleq \mathbf{vc}_i$:
28         Stop with 1
29 $d_{ij} \leftarrow d_{ij} + 1$
30 $\mathbf{vc}_i[j] \leftarrow d_{ij}$
31 Give $(\mathsf{d}, i, j, m)$ to $\mathcal{A}$
32 Return
```

Fig. 3. Game for functionality/correctness of (cryptographic) causal broadcast channels (running over FIFO links). State variables $b_i, s_{ij}, r_{ij}, d_{ij}$ are broadcast, send, receive, and delivery counters, respectively. Note that the current contents of b_i and d_{ij} are (redundantly) also reflected in $\mathbf{vc}_i$: we always have $\mathbf{vc}_i[i] = b_i$ and $j \neq i \Rightarrow \mathbf{vc}_i[j] = d_{ij}$.

might be to achieve a situation in which messages are delivered with wrong content or in (causally) wrong order (in this sense, $\mathcal{A}$ can be seen as an adversary).[3] A 'functional' causal broadcast channel is one for which no $\mathcal{A}$ can do that.

We proceed with explaining the functionality game in detail. When oracle bcast is queried on input (i, m), it invokes the broadcast algorithm on input message m and the current state st_i (of user i). Similarly, if $\mathcal{A}$ queries recv on input (i, j, D) the oracle invokes the receiving algorithm on input datagram D, alleged originator j, and the current state st_i. The two 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 bcast and recv. To this end, for every invocation $\mathsf{send}(i, j, D)$ the oracle gives the 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 $\mathsf{delv}(i, j, m)$ is invoked, the oracle gives to the adversary the tuple (d, i, j, m) to report that message m has been delivered to user i with reported originator user j. This captures the adversary's ability to observe the reaction of applications upon delivery of messages.[4]

During the execution of the game, two specific events have to be considered: whether $\mathcal{A}$ remains *passive*, meaning that it only schedules recv operations that are consistent with the FIFO guarantees expected by the network; and whether $\mathcal{A}$ achieves the 'bad' situation (thus violating *correctness*).

Passiveness requires $\mathcal{A}$ to schedule recv 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 recv (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 FIFO links), 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 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 delv and, if so, it declares the adversary "successful" and terminates 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, the current vector clock $\mathbf{vc}_i$ of user i (note that $\mathbf{vc}_i[i] = b_i$ and $\mathbf{vc}_i[j] = d_{ij}$ for all $j \in [1\,..\,N]$ throughout the game), 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 bcast). Then, whenever an

[3]In distributed systems such a 'bad' situation would be a *safety* violation.

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

```
Algo init($N$)
00 For $i \leftarrow 1$ to $N$:
01     $rej_i \leftarrow \mathrm{F}$
02     $b_i \leftarrow 0$; $\mathbf{vc}_i \leftarrow \mathbf{0}_N$
03     For $j \leftarrow 1$ to $N$, $j \neq i$:
04         $r_{ij}, d_{ij} \leftarrow 0$
05         $\mathbf{Q}_{ij}[\,] \leftarrow \emptyset$
06 Encode into state $st_i$:
   $rej_i, b_i, \mathbf{vc}_i, r_{ij}, d_{ij}, \mathbf{Q}_{ij}$
07 Return $(st_1, \ldots, st_N)$
```

```
Algo bcast^{send,delv}($st_i$; $m$)
08 If $rej_i$: Goto line 14
09 $D \leftarrow (\mathbf{vc}_i, m)$
10 For all $j \in [1\,..\,N]$, $j \neq i$:
11     send($i, j, D$)
12 $b_i \leftarrow b_i + 1$
13 $\mathbf{vc}_i[i] \leftarrow b_i$
14 Return $st_i$
```

```
Algo recv^{send,delv}($st_i$; $j, D$)
15 If $rej_i$: Goto line 25
16 Parse $D$ as $(\mathbf{vc}, m)$
17 If parsing fails:
18     $rej_i \leftarrow \mathrm{T}$; Goto line 25
19 $\mathbf{Q}_{ij}[r_{ij}] \leftarrow (\mathbf{vc}, m)$
20 $r_{ij} \leftarrow r_{ij} + 1$
21 While exist $\mathbf{vc}', m', j' \neq i$ s.t.
   $(\mathbf{vc}', m') = \mathbf{Q}_{ij'}[d_{ij'}]$ and $\mathbf{vc}' \leq \mathbf{vc}_i$:
22     delv($i, j', m'$)
23     $d_{ij'} \leftarrow d_{ij'} + 1$
24     $\mathbf{vc}_i[j'] \leftarrow d_{ij'}$
25 Return $st_i$
```

Fig. 4. Waiting causal broadcast (from FIFO links). State variables b_i, r_{ij}, d_{ij} are broadcast, receive, and delivery counters, respectively. Note that the current contents of b_i and d_{ij} are (redundantly) also reflected in $\mathbf{vc}_i$: we always have $\mathbf{vc}_i[i] = b_i$ and $j \neq i \Rightarrow \mathbf{vc}_i[j] = d_{ij}$. 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).

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

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

B. *Construction of Causal Broadcast*

We reproduce a standard construction of a causal broadcast protocol, known in the distributed systems literature as *waiting causal broadcast* [34], built on top of point-to-point connections like TCP/IP. As we assume FIFO delivery from the underlying network, the goal is to leverage this property 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 that are included in datagrams). 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.

V. Security notions

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.[5] We then study how these notions relate to each other. Our notions are in the style of [35], [36], and [3] for symmetric (authenticated) encryption.

A. *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 recv 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

[5] In contrast to the naming convention introduced in the previous section, our security notions refer to 'ciphertexts' rather than 'datagrams' and to 'plaintexts' rather than 'messages'. The reason for this (conservative) choice it to stay close to the analogous integrity and confidentiality notions for authenticated encryption/secure channels that are widely-adopted in the literature.

```
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 $\mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
07 Stop with 0

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

Oracle $\mathrm{recv}(i, j, D)$
14 $st_i \leftarrow_\$ \mathsf{recv}^{\mathsf{send},\mathsf{delv}}(st_i; j, D)$
15 Return

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

Proc $\mathsf{delv}(i, j, m)$
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 $(\mathtt{d}, i, j, m)$ to $\mathcal{A}$
23 Return
```

Fig. 5. Game for INT-PTXT security of causal broadcast channels (over FIFO links).

```
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 $\mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
07 Stop with 0

Oracle $\mathrm{bcast}(i, m)$
08 $st_i \leftarrow_\$ \mathsf{bcast}^{\mathsf{send},\mathsf{delv}}(st_i; m)$
09 Return

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

Proc $\mathsf{send}(i, j, D)$
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 $(\mathtt{s}, i, j, D)$ to $\mathcal{A}$
19 Return

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

Fig. 6. Game for INT-CTXT security of causal broadcast channels (over FIFO links).

prove, causal delivery is *implicitly* also ensured by INT-CTXT as long as correctness is fulfilled.)

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 delv algorithm deliver a message that was either never sent by the reported 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 condition 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 $\mathrm{Adv}^{\text{int-ptxt}}_{\mathsf{Ch},N}(\mathcal{A}) = \Pr[\mathrm{INT}^{\mathrm{ptxt}}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1]$, 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 N and realistic $\mathcal{A}$.

We next formalize ciphertext integrity, based on the experiment in Figure 6. Here the recv 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 send and delv procedures any further. We define the advantage of an 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 N and realistic $\mathcal{A}$.

B. *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 bcast 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 recv oracle to advance the state of the corresponding participant. The difference between the notions IND-CPA and IND-CCA is the kind of datagrams that can be submitted to recv: The former notion considers passive adversaries, i.e., those that provide the recv oracle with exclusively the datagram sequences that were output by the send procedure (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. In addition to that, oracle recv lets $\mathcal{A}$ obtain messages delivered by users upon receiving manipulated datagrams (again, datagram manipulation refers to any violation of the network guarantees, including reordering), effectively realizing a decryption oracle in the broadcast channel scenario. Note that in both confidentiality games, messages delivered by a given user are not given to the

```
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 $b' \leftarrow_\$ \mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
06 Stop with $b'$

Oracle $\mathrm{bcast}(i, m_0, m_1)$
07 $st_i \leftarrow_\$ \mathsf{bcast}^{\mathsf{send},\mathsf{delv}}(st_i; m_b)$
08 Return

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

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

Fig. 7. Games for IND-CPA security of causal broadcast channels (over FIFO links).

```
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 $b' \leftarrow_\$ \mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
07 Stop with $b'$

Oracle $\mathrm{bcast}(i, m_0, m_1)$
08 $st_i \leftarrow_\$ \mathsf{bcast}^{\mathsf{send},\mathsf{delv}}(st_i; m_b)$
09 Return

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

Proc $\mathsf{send}(i, j, D)$
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 $\mathsf{delv}(i, j, m)$
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. Games for IND-CCA security of causal broadcast channels (over FIFO links).

adversary if no active measure took place against that user (in the experiments, $\mathcal{A}$ is given the distinguished symbol $\diamond$ instead, as in lines 18 and 21 of Figures 7 and 8, respectively). This is a standard technique to define sound confidentiality notions for encryption and secure channels.

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 N and realistic $\mathcal{A}$. 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.

C. Relations among notions

We proceed with establishing important relations among the notions just defined: (1) IND-CCA implies IND-CPA, (2) INT-CTXT implies INT-PTXT, (3) IND-CPA and INT-CTXT together imply IND-CCA. These implications are analogous to those for unidirectional channels from [3] and bidirectional channels [4] (and reflect the fact that broadcast channels are a generalization of two-party channels). 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 VI.

Theorem 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* bcast *and* recv *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 straight-forward 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 recv 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. $\square$

Theorem 2 (INT-CTXT $\Longrightarrow$ INT-PTXT)**.** *Let* Ch *be a (perfectly correct) 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* bcast *and* recv *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 recv oracle, and the send procedure by the corresponding versions of game $\mathrm{FUNC}_{\mathsf{Ch},N}$ from Figure 3, leaving unmodified the bcast oracle and the delv 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 send and delv 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 delv 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 3 (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* bcast *and* recv *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 send and delv 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 recv 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. □

VI. Constructions

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

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 two overall goals could be reached simultaneously: security (our construction provably meets all security notions defined in this paper), and efficiency (we aimed at minimizing the ciphertext size and the number of AEAD invocations per execution of bcast/recv).

Let us compare the algorithms of our construction with the ones from Figure 4 in more detail. The init algorithms are almost identical, the only difference being the fresh AEAD key K that is shared among all participants of a broadcast channel instance. In the bcast 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 change their order. Our recv 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.

Correctness of our construction follows by inspection (and comparison with Figure 4). In the rest of this section we analyze the security of our construction. 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

```
**Algo** $\mathsf{init}(N)$
00 $K \leftarrow_\$ \mathsf{Gen}$
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         $s_{ij}, r_{ij}, d_{ij} \leftarrow 0$
06         $\mathbf{Q}_{ij}[\,] \leftarrow \emptyset$
07 Encode into state $st_i$:
   $K, rej_i, b_i, \mathbf{vc}_i, s_{ij}, r_{ij}, d_{ij}, \mathbf{Q}_{ij}$
08 Return $(st_1, \ldots, st_N)$
```

```
**Algo** $\mathsf{bcast}^{\mathsf{send,delv}}(st_i; m)$
09 If $rej_i$: Goto line 18
10 For all $j \in [1 \mathinner{..} N]$, $j \neq i$:
11     $ad_j \leftarrow i \,\|\, j \,\|\, s_{ij} \,\|\, \mathbf{vc}_i$
12     $c_j \leftarrow \mathsf{Enc}(K; ad_j, m)$
13     $D_j \leftarrow (\mathbf{vc}_i, c_j)$
14     $\mathsf{send}(i, j, D_j)$
15     $s_{ij} \leftarrow s_{ij} + 1$
16 $b_i \leftarrow b_i + 1$
17 $\mathbf{vc}_i[i] \leftarrow b_i$
18 Return $st_i$
```

```
**Algo** $\mathsf{recv}^{\mathsf{send,delv}}(st_i; j, D)$
19 If $rej_i$: Goto line 33
20 Parse $D$ as $(\mathbf{vc}, c)$
21 If parsing fails:
22     $rej_i \leftarrow \mathrm{T}$; Goto line 33
23 $ad \leftarrow j \,\|\, i \,\|\, r_{ij} \,\|\, \mathbf{vc}$
24 $m \leftarrow \mathsf{Dec}(K; ad, c)$
25 If decryption fails:
26     $rej_i \leftarrow \mathrm{T}$; Goto line 33
27 $\mathbf{Q}_{ij}[r_{ij}] \leftarrow (\mathbf{vc}, m)$
28 $r_{ij} \leftarrow r_{ij} + 1$
29 While exist $\mathbf{vc}', m', j' \neq i$ s.t.
   $(\mathbf{vc}', m') = \mathbf{Q}_{ij'}[d_{ij'}]$ and $\mathbf{vc}' \leq \mathbf{vc}_i$:
30     $\mathsf{delv}(i, j', m')$
31     $d_{ij'} \leftarrow d_{ij'} + 1$
32     $\mathbf{vc}_i[j'] \leftarrow d_{ij'}$
33 Return $st_i$
```

Fig. 9. Construction of cryptographic causal broadcast from FIFO links. See Appendix A for an explanation of algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$, and Figures 3 and 4 for state variables $b_i, \mathbf{vc}_i[\,], s_{ij}, r_{ij}, d_{ij}, \mathbf{Q}_{ij}[\,]$.

(Theorem 4); we then show that the INT-CTXT security of the AEAD implies the INT-CTXT security of the broadcast channel (Theorem 5); finally, as a corollary of the two results, we apply Theorem 3 to establish the IND-CCA security of the broadcast channel.

Theorem 4 (IND-CPA security)**.** *Let* Ch *be the broadcast channel constructed in Figure 9 from FIFO links 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* bcast *queries.*

Proof. For $b \in \{0,1\}$, consider games G_0^b from Figure 10. They are identical to games $\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}$ from Figure 7, but with the following rewriting steps applied: (a) the abstract bcast and recv algorithms are instantiated with the ones from Ch, (b) as variables s_{ij} appear in both $\mathrm{IND}^{\mathrm{cpa},b}$ 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 these steps changes the output of the game we have $\Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{Ch},N}(\mathcal{A}) \Rightarrow 1] = \Pr[G_0^b(\mathcal{A}) \Rightarrow 1]$ for any $\mathcal{A}$ and $b \in \{0,1\}$.

Consider next the games G_1^b in Figure 10. The difference to G_0^b 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 recv oracle on ciphertexts that were output by the bcast 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 $\Pr[G_0^b(\mathcal{A}) \Rightarrow 1] = \Pr[G_1^b(\mathcal{A}) \Rightarrow 1]$ for any $\mathcal{A}$ and $b \in \{0,1\}$.

Observe now that in the recv oracle of games G_1^b 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_2^b that are like G_1^b except that in line 13 the value $\diamond$ is stored in $\mathbf{L}$ instead of message m_b. We obtain $\Pr[G_1^b(\mathcal{A}) \Rightarrow 1] = \Pr[G_2^b(\mathcal{A}) \Rightarrow 1]$.

In games G_2^b 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 $\Pr[G_2^b(\mathcal{A}) \Rightarrow 1] = \Pr[\mathrm{IND}^{\mathrm{cpa},b}_{\mathsf{AEAD}}(\mathcal{B}) \Rightarrow 1]$, 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. $\square$

Theorem 5 (INT-CTXT security)**.** *Let* Ch *be the broadcast channel constructed in Figure 9 from FIFO links 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*

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

```
**Games** $G_0^b(\mathcal{A}), G_1^b(\mathcal{A})$
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 $b' \leftarrow_\$ \mathcal{A}^{\mathrm{bcast,recv}}$
08 Stop with $b'$
```

```
**Oracle** $\mathrm{bcast}(i, m_0, m_1)$
09 If $rej_i$: Goto line 20
10 For all $j \in [1 \mathinner{..} N]$, $j \neq i$:
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** $\mathrm{recv}(i, j, D)$
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_0^b, G_1^b, $b \in \{0,1\}$, used in the IND-CPA proof of Theorem 4. Games G_0^b include all lines with exception of line 31, and games G_1^b include all lines with exception of lines 28–30.

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* bcast *and* recv *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 bcast and recv 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 bcast 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 $i \in [1 \mathinner{..} N]$. 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 recv oracle that is not in synchrony with what the bcast oracle had sent before. Consider thus a query (i, j, D) to recv 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. ☐

The following is a corollary of Theorems 3, 4, and 5.

Corollary 1 (IND-CCA security)**.** *Let* Ch *be the broadcast channel constructed in Figure 9 from FIFO links 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).*

A. Principles behind our design

Intuitively, any solution that realizes a secure causality-preserving broadcast channel on top of unprotected point-to-point connections will combine, as two ingredients, one primitive that constructs (non-cryptographic) causal broadcast from FIFO links, and a second primitive that contributes the cryptographic protection. It would add to clarity if these

```
Games $G_0(\mathcal{A}), G_1(\mathcal{A})$
00 $L \leftarrow \emptyset$; $K \leftarrow_\$ \mathsf{Gen}$
01 For $i \leftarrow 1$ to $N$:
02     $psv_i \leftarrow$ T; $rej_i \leftarrow$ 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 $\mathcal{A}^{\mathrm{bcast},\mathrm{recv}}$
08 Stop with 0

Oracle $\mathrm{bcast}(i, m)$
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 $(\mathtt{s}, i, j, D_j)$ to $\mathcal{A}$
19 $b_i \leftarrow b_i + 1$
20 $\mathbf{vc}_i[i] \leftarrow b_i$
21 Return

Oracle $\mathrm{recv}(i, j, D)$
22 If $s_{ji} \leq r_{ij}$ or $\mathbf{D}_{ji}[r_{ij}] \neq D$:
23     $psv_i \leftarrow$ F
24 If $rej_i$: Goto line 41
25 Parse $D$ as $(\mathbf{vc}, c)$
26 If parsing fails:
27     $rej_i \leftarrow$ 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     $rej_i \leftarrow$ 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 $(\mathtt{d}, i, j', m')$ 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 5. Game G_0 includes all lines with exception of lines 32–33, and game G_1 includes all lines.

components were combined generically, but our construction from Figure 9 interveawes the two components and is thus not an example for this. In the following we first describe two other, generic, options to realize secure causal broadcast channels; we then explain the particular advantage that we see in our (ad-hoc) design. Option 1 is to first use a protocol like the one from Figure 4 to achieve (unprotected) causal delivery and to then run a cryptographic layer on top of that; Option 2 is to first cryptographically protect the FIFO links and to then run a protocol like the one from Figure 4 on top. In principle both these approaches are sound, but Option 1 is likely the more expensive one, as both the (lower-level) causal broadcast protocol *and* the (higher-level) cryptographic layer will have to, redundantly, implement and maintain logical clocks (otherwise it will be impossible to check for causality violations). Our construction was thus designed in the spirit of Option 2, but it has one important difference: The described generic broadcast-over-crypto approach would not be able to make use of the associated-data input of the AEAD scheme for protecting control data like sequence numbers and user identifiers, and this ultimately would result in a higher computational load, longer ciphertexts, and a less efficient scheme in general. In contrast, our scheme fully exploits the functionality of the AEAD primitive and does not encrypt data that does not need to be confidential.

VII. Conclusion & Future work

In this work we define game-based security notions for cryptographic multi-party channels that allow a group of mutually trusted users to exchange messages in a confidential, authentic, and causality-preserving manner, and we propose a provably secure instantiation of this channel type. Our work is an important step towards understanding the security of causal multi-party communication, but it also suggests two further research questions: (1) Our model assumes a static set of users and hence does not capture applications that let users join and leave the conversation. While the problem of securely distributing and updating keys among a dynamic set of users has been investigated [37], [38], a study of confidentiality, authenticity, and causality preservation in such a dynamic scenario yet has to appear. (2) An orthogonal problem is that of relaxing the trust assumption of our model and tolerating the corruption of (a fraction of) participants. Again, broadcast protocols that tolerate Byzantine behavior appear in the distributed systems literature [17], [18], [19], [20], but they are designed for different/more specific applications than secure multi-party channels and, thus, a fresh look at the problem can lead to more efficient solutions.

Acknowledgments

We thank Marc Fischlin and Martijn Stam for valuable comments on earlier versions of this work, Felix Günther for helpful discussions, and the anonymous reviewers for their valuable comments.

Patrick Eugster was supported by ERC through grant 617805 (LiVeSoft). Giorgia Marson conducted part of the research at Technische Universität Darmstadt and at Ruhr University Bochum, where she was supported by DFG as part of the CRC 1119 CROSSING and as part of the priority program 1736 Algorithms for Big Data, respectively. Bertram Poettering conducted part of the work while working at Ruhr University Bochum supported by ERC Project ERCC (FP7/615074).

References

[1] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[2] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture," RFC 4251 (Proposed Standard), Internet Engineering Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4251.txt

[3] M. Bellare, T. Kohno, and C. Namprempre, "Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol," in *ACM CCS 02*, V. Atluri, Ed. Washington D.C., USA: ACM Press, Nov. 18–22, 2002, pp. 1–11.

[4] G. Marson and B. Poettering, "Security Notions for Bidirectional Channels," *IACR Transactions on Symmetric Cryptology*, vol. 2017, no. 1, pp. 405–426, 2017. [Online]. Available: http://tosc.iacr.org/index.php/ToSC/article/view/602

[5] Van Jacobson, "A New Way to look at Networking (Google Tech Talk)," https://www.youtube.com/watch?v=oCZMoY3q2uM, Aug. 2006.

[6] K. Birman, A. Schiper, and P. Stephenson, "Lightweight Causal and Atomic Multicast," *Transactions on Computer Systems*, vol. 9, no. 3, pp. 272–314, Aug. 1991.

[7] G. F. C. Toinard, "A New Way to Design Causally and Totally Ordered Multicast Protocols," *SIGOPS Operating Systems Review*, vol. 26, no. 4, pp. 77–83, 1992.

[8] V. Hadzilacos and S. Toueg, "A Modular Approach to Fault-Tolerant Broadcast and Related Problems," Cornell University, Computer Science, Tech. Rep., May 1994.

[9] K. G. Paterson, T. Ristenpart, and T. Shrimpton, "Tag size does matter: Attacks and proofs for the TLS record protocol," in *ASIACRYPT 2011*, ser. LNCS, D. H. Lee and X. Wang, Eds., vol. 7073. Seoul, South Korea: Springer, Heidelberg, Germany, Dec. 4–8, 2011, pp. 372–389.

[10] A. Boldyreva, J. P. Degabriele, K. G. Paterson, and M. Stam, "Security of symmetric encryption in the presence of ciphertext fragmentation," in *EUROCRYPT 2012*, ser. LNCS, D. Pointcheval and T. Johansson, Eds., vol. 7237. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 15–19, 2012, pp. 682–699.

[11] M. Fischlin, F. Günther, G. A. Marson, and K. G. Paterson, "Data is a stream: Security of stream-based channels," in *CRYPTO 2015, Part II*, ser. LNCS, R. Gennaro and M. J. B. Robshaw, Eds., vol. 9216. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 16–20, 2015, pp. 545–564.

[12] F. Günther and S. Mazaheri, "A formal treatment of multi-key channels," in *CRYPTO 2017, Part III*, ser. LNCS, J. Katz and H. Shacham, Eds., vol. 10403. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 20–24, 2017, pp. 587–618.

[13] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd FOCS*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 14–17, 2001, pp. 136–145.

[14] U. Maurer and R. Renner, "Abstract cryptography," in *ICS 2011*, B. Chazelle, Ed. Tsinghua University, Beijing, China: Tsinghua University Press, Jan. 7–9, 2011, pp. 1–21.

[15] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[16] R. Schwarz and F. Mattern, "Detecting causal relationships in distributed computations: In search of the holy grail," *Distributed Computing*, vol. 7, no. 3, pp. 149–174, 1994.

[17] M. K. Reiter, K. P. Birman, and L. Gong, "Integrating security in a group oriented distributed system," in *1992 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 1992, pp. 18–32.

[18] M. K. Reiter and K. P. Birman, "How to securely replicate services," *ACM TOPLAS*, vol. 16, no. 3, pp. 986–1009, 1994.

[19] M. K. Reiter and L. Gong, "Securing causal relationships in distributed systems," *Comput. J.*, vol. 38, no. 8, pp. 633–642, 1995.

[20] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "Secure and efficient asynchronous broadcast protocols," in *CRYPTO 2001*, ser. LNCS, J. Kilian, Ed., vol. 2139. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 19–23, 2001, pp. 524–541.

[21] S. Duan, M. K. Reiter, and H. Zhang, "Secure causal atomic broadcast, revisited," in *Dependable Systems and Networks (DSN)*. IEEE Computer Society, 2017, pp. 61–72.

[22] M. C. Pease, R. E. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, vol. 27, no. 2, pp. 228–234, 1980.

[23] L. Lamport, R. E. Shostak, and M. C. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[24] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[25] "Private group messaging," 2014. [Online]. Available: https://whispersystems.org/blog/private-groups

[26] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure messaging," in *2015 IEEE Symposium on Security and Privacy*. San Jose, CA, USA: IEEE Computer Society Press, May 17–21, 2015, pp. 232–249.

[27] "Off-the-Record Messaging," 2016. [Online]. Available: http://otr.cypherpunks.ca

[28] H. Liu, E. Y. Vasserman, and N. Hopper, "Improved group off-the-record messaging," in *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, A. Sadeghi and S. Foresti, Eds. ACM, 2013, pp. 249–254.

[29] M. D. V. Gundy and H. Chen, "OldBlue: Causal broadcast in a mutually suspicious environment," November 2012, (unpublished). [Online]. Available: http://matt.singlethink.net/projects/mpotr/oldblue-draft.pdf

[30] J. Reardon, A. Kligman, B. Agala, and I. Goldberg, "KleeQ: Asynchronous key management for dynamic ad-hoc networks," January 2007, Technical Report CACR 2007-03. [Online]. Available: http://cacr.uwaterloo.ca/techreports/2007/cacr2007-03.pdf

[31] T. Perrin, "Double Ratchet Algorithm," GitHub wiki, 2016. [Online]. Available: https://github.com/trevp/double_ratchet/wiki

[32] M. Marlinspike, "Advanced cryptographic ratcheting," Blog, 2013. [Online]. Available: https://whispersystems.org/blog/advanced-ratcheting

[33] I. Goldberg, B. Ustaoglu, M. Van Gundy, and H. Chen, "Multi-party off-the-record messaging," in *ACM CCS 09*, E. Al-Shaer, S. Jha, and A. D. Keromytis, Eds. Chicago, Illinois, USA: ACM Press, Nov. 9–13, 2009, pp. 358–368.

[34] C. Cachin, R. Guerraoui, and L. Rodrigues, *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

[35] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *38th FOCS*. Miami Beach, Florida: IEEE Computer Society Press, Oct. 19–22, 1997, pp. 394–403.

[36] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *ASIACRYPT 2000*, ser. LNCS, T. Okamoto, Ed., vol. 1976. Kyoto, Japan: Springer, Heidelberg, Germany, Dec. 3–7, 2000, pp. 531–545.

[37] Y. Kim, A. Perrig, and G. Tsudik, "Communication-efficient group key agreement," in *SEC*, ser. IFIP Conference Proceedings, vol. 193. Kluwer, 2001, pp. 229–244.

[38] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably authenticated group Diffie-Hellman key exchange – the dynamic case," in *ASIACRYPT 2001*, ser. LNCS, C. Boyd, Ed., vol. 2248. Gold Coast, Australia: Springer, Heidelberg, Germany, Dec. 9–13, 2001, pp. 290–309.

[39] P. Rogaway, "Authenticated-encryption with associated-data," in *ACM CCS 02*, V. Atluri, Ed. Washington D.C., USA: ACM Press, Nov. 18–22, 2002, pp. 98–107.

[40] P. Rogaway and T. Shrimpton, "A provable-security treatment of the key-wrap problem," in *EUROCRYPT 2006*, ser. LNCS, S. Vaudenay, Ed., vol. 4004. St. Petersburg, Russia: Springer, Heidelberg, Germany, May 28 – Jun. 1, 2006, pp. 373–390.

Appendix A
Definitions of AEAD

We recall the definition of AEAD from [39], slightly adapting it to a new syntax. The security properties that we give interpolate the ones of [39], [40].

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*

$$\mathsf{Gen} \rightarrow \mathcal{K}$$

$$\mathcal{K} \times \mathcal{AD} \times \mathcal{M} \rightarrow \mathsf{Enc} \rightarrow \mathcal{C}$$

$$\mathcal{K} \times \mathcal{AD} \times \mathcal{C} \rightarrow \mathsf{Dec} \rightarrow \mathcal{M}/\perp$$

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.*

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 $\mathrm{IND}^{\text{cpa},b}_{\mathsf{AEAD}}(\mathcal{A})$
00 $\mathbf{D} \leftarrow \emptyset$
01 $K \leftarrow_{\$} \mathsf{Gen}$
02 $b' \leftarrow_{\$} \mathcal{A}^{\mathrm{Enc}}$
03 Stop with $b'$

Oracle $\mathrm{Enc}(ad, m_0, m_1)$
04 If $(ad, \cdot) \in \mathbf{D}$: Stop with 0
05 $c \leftarrow_{\$} \mathsf{Enc}(K; ad, m_b)$
06 $\mathbf{D} \leftarrow \mathbf{D} \cup \{(ad, c)\}$
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.

```
Game $\mathrm{INT}^{\text{ctxt}}_{\mathsf{AEAD}}(\mathcal{A})$
00 $\mathbf{D} \leftarrow \emptyset$
01 $K \leftarrow_{\$} \mathsf{Gen}$
02 $\mathcal{A}^{\mathrm{Enc},\mathrm{Dec}}$
03 Stop with 0

Oracle $\mathrm{Enc}(ad, m)$
04 If $(ad, \cdot) \in \mathbf{D}$: Stop with 0
05 $c \leftarrow_{\$} \mathsf{Enc}(K; ad, m)$
06 $\mathbf{D} \leftarrow \mathbf{D} \cup \{(ad, c)\}$
07 Return $c$

Oracle $\mathrm{Dec}(ad, c)$
08 $m \leftarrow \mathsf{Dec}(K; ad, c)$
09 If $(ad, c) \notin \mathbf{D}$ and $m \neq \perp$:
10     Stop with 1
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.