# A Message Franking Channel

Loïs Huguenin-Dumittan and Iraklis Leontiadis

LASEC, EPFL, Switzerland
{lois.huguenin-dumittan, iraklis.leontiadis}@epfl.ch

**Abstract.** We pursue to formalize and instantiate a secure bidirectional channel with message franking properties. Under this model a sender may send an abusive message to the receiver and the latter wish to open it in a verifiable way to a third party. Potential malicious behavior of a sender requires message franking protocols resistant to sending messages that cannot be opened later by the receiver. An adversary impersonated by the receiver may also try to open messages that have not been sent by the sender. Wrapping a message franking protocol in a secure channel requires a more delicate treatment in order to avoid drops or replay of messages and out-of-order delivery. To the best of our knowledge we are the first to model the security of a message franking channel, which apart from integrity, confidentiality, resistance to drops, relays and out-of-order delivery is *sender* and *receiver binding*: a sender cannot send a message which cannot be opened in a verifiable way later by the receiver, and the receiver cannot claim a message that had not been truly sent by the receiver. Finally, we instantiate a bidirectional message franking channel from symmetric primitives and analyze its security.

**Keywords:** message franking channel, secure communication, channel security, abusive verifiable reports

## 1 Introduction

The most popular messaging services such as Facebook Messenger, Whatsapp, Telegram or Signal offer end-to-end encryption, preventing anyone apart from the recipients from reading the messages. While preserving privacy, such schemes increase the difficulty of filtering spam or reporting abusive messages. Indeed, without the capacity to read the plaintexts, the router (e.g. Facebook) cannot check for abusive content, malware, malicious links, etc. The problem of abuse reporting was recently tackled by Facebook, which introduced the concept of *message franking* [Fac16]. With this proposed protocol, a user can report abusive messages to Facebook and can prove that an abusive message was sent by another user. More recently, Grubbs et al. initiated the formal analysis of such schemes [GLR17]. In particular, they introduced a new cryptographic primitive called committing authenticated encryption with associated data (committing AEAD) along with new security definitions. In the same paper, they analyze the security of Facebook's scheme and present a more efficient construction. In a follow up

work Dodis, Grubbs, Ristenpart and Woodage [DGRW18] revealed a flaw in Facebook message franking protocol for attachment delivery. The authors showed compromisation of sender binding, letting a malicious sender to send messages that cannot be reported. To circumvent that flaw they suggest a new design for message franking based on hashing encryption which provides the requiring commitment properties on the ciphertext with only one pass.

The security of communication protocols though depends not only on the underlying cryptographic primitives but also on the behavior of the protocol itself. For instance, a protocol based on secure primitives accepting out-of-order messages or an adversary being able to drop or replay messages renders communication between two end points vulnerable to such malicious behaviors. More generally, traditional cryptographic primitives cannot model real-world attacks beyond basic confidentiality and integrity. In particular, the integrity of the communication channel (e.g. security against out-of-order messages, message drops or replay attacks) is not captured by traditional security definitions. These reasons led to the study of communication protocols as stateful encryption schemes, so called: cryptographic channels [BKN02,KPB03,Mar17,PR18].

**Cryptographic channel.** Consider a messaging protocol where several participants share a key with each other and want to send and receive end-to-end encrypted messages. Once all the keys are fixed, a channel with confidentality, integrity–which includes resistance to replay attacks, out-of-order delivery and drops–must be established between each pair of participants. The cryptographic primitive that models the channels and the interaction between the participants is called a *cryptographic channel*. A channel where the only actions available to the clients are *send* and *receive* (i.e. as in a traditionnal message exchange protocol) will be referred to as a *standard cryptographic channel*.

A standard cryptographic channel $\mathsf{Ch} = (\mathsf{init}, \mathsf{snd}, \mathsf{rcv})$ is a tuple of three efficient algorithms that allows the participants to send and receive encrypted messages. If there are two participants and only one can send and only one can receive, the channel is unidirectional while if both can send and receive, the channel is bidirectional.

One could imagine that a bidirectional channel made up of two secure unidirectional channels should be secure. However, as shown in [MP17] this does not hold. In the same paper, the authors show several security results on bidirectional channels constructed from unidirectional channels in a special construction called the *canonic composition*. The more interesting ones concerning confidentality, are the following:

$$\text{IND-1CPA} + \text{IND-1CPA} \iff \text{IND-2CPA} \tag{1}$$

$$\text{IND-1CCA} + \text{IND-1CCA} \impliedby \text{IND-2CCA} \tag{2}$$

$$\text{IND-1CCA} + \text{IND-1CCA} \not\Longrightarrow \text{IND-2CCA} \tag{3}$$

where IND-1CPA, IND-2CPA are the IND-CPA security for unidirectional and bidirectional channels, respectively, and IND-1CCA, IND-2CCA are the IND-CCA security for unidirectional and bidirectional channels, respectively [MP17]. One can see that if a unidirectional channel is CPA secure, a CPA secure

bidirectional channel can be constructed. On the other hand, two CCA secure unidirectional channels are not sufficient to create a CCA secure bidirectional channel. Intuitively, one can understand this result by considering the following example. We consider a bidirectional channel made up of two independent confidential unidirectional channels that do not guarantee integrity. Let the protocol be such that if an adversary sends a special ciphertext $c'$ to Alice, she sends her password to Bob without a handshake. Note that this contradicts integrity but not condidentiality. Then, since Bob does not expect this message, he outputs the message (i.e. the password) in clear for everyone. Obviously, this does not contradict the confidentiality of the $B \to A$ channel. Now, we assume that Alice sends her password to Bob without a handshake if and only if she receives $c'$. Then, the $A \to B$ channel can still be confidential since unidirectional confidentiality (IND-1CCA) does not model the fact that Alice can receive messages in the $A \to B$ channel, in particular $c'$. Thus, these results show the importance of considering protocols in bidirectional cryptographic channels.

Recent results analyze security of secure communication channel over TLS [BHMS15,BH17,GM17], but without sender and receiver binding guarantees. In this work we aim to close the gap in the existing literature with the definitions, design and analysis of a secure communication channel with message franking properties: sender and receiver binding.

**Our contributions.** The contributions of this work are summarized as follows:

1. We first define a message franking channel (MFC) that models a messaging protocol where users can report abusive messages.
2. Then, we present unidirectional and bidirectional security definitions for our construction. The most challenging are the uni/bi-directional sender and receiver binding notions, which were introduced by Grubbs et al. [GLR17]. Specifically, binding definitions guarantee that a delivered message can be reported and a forged message cannot be reported.
3. We prove that a special construction called hereafter the *canonical composition*, made from two binding unidirectional MFC, is sufficient to build a secure binding bidirectional MFC.
4. Finally we present an instantiation of a message franking channel made from a secure committing AEAD scheme and a message authentication code.

**Outline.** In section 2 we introduce some notation for the manuscript. In section 3 we recap the reader the message franking protocol definitions and Facebook message franking protocol. We continue in section 4 with the syntactical definition of a message franking channel and in section 5 with the security properties thereof. Before presenting our concrete instantiation in section 7, we prove the security of the canonical composition of two unidirectional message franking channels to build a bidirectional one in section 6. Finally, we conclude our work in section 8.

## 2 Notation

A participant is referred to interchangeably as a client, a user or a party. We write $A \parallel B$ to denote the concatenation of $A$ with $B$ and $|A|$ to denote the length of $A$. We write $\Pr[G \Rightarrow x]$ to denote the probability sa game $G$ outputs $x$. If $\mathcal{X}$ is a set, then $X \leftarrow_{\$} \mathcal{X}$ means that $X$ is uniformly sampled from $\mathcal{X}$. If $G$ is a randomized algorithm, we write $x \leftarrow_{\$} G$ to denote the fact that $x$ takes the value output by $G$. If $G$ is deterministic, we write $x \leftarrow G$.
In the different games, we denote the initialization of an array $A$ by $A \leftarrow []$. At each position $i$, an array can be assigned a single value $x$ or a tuple of values $(x_1, \ldots, x_n)$. We denote these events by $A[i] \leftarrow x$ and $A[i] \leftarrow (x_1, \ldots, x_n)$, respectively. We write **abort** for "Stop the game and return 0 as a failure". If $G$ is a game that returns a value $n$, we denote by $\Pr[G \Rightarrow n]$ the probability that $G$ returns $n$.
Finally, when this is clear from the context, we denote by $*$ any value that could fit. For example, if $T \in \mathcal{X} \times \mathcal{Y}$ is a tuple of two values and $X \in \mathcal{X}$, we write $T == (X, *)$ to denote the event that there exists some $Y \in \mathcal{Y}$ such that $T$ is equal to $(X, Y)$.

## 3 Message franking protocols

### 3.1 Facebook's scheme

Facebook recently proposed an abuse-report mechanism (cf. Figure 1) for their messaging service, referred to as *message franking* [Fac16]. The idea is that a client sends a commitment for the message along with the encrypted message. The router (i.e. Facebook's server) then tags the commitment and forwards everything to the receiving party. Then, if the client wants to report the message as abusive, it sends to Facebook the message, the commitment, the opening key and the router tag. As such, Facebook can check that the message reported as abusive was indeed a message sent to the receiver by a specific sender.

Let $M$ be the message and $K$ the secret key shared between Alice and Bob. The sending party samples a one-time key $K_f$ (i.e. a nonce) from the key space and computes the HMAC $C_2$ of $M \parallel K_f$ using the one-time key $K_f$. One can observe that $C_2$ is a commitment on the message. Then, the message $M$ and the HMAC key $K_f$ are encrypted with the secret key $K$. Let $C_1$ be the resulting ciphertext. Then, both the ciphertext $C_1$ and the commitment $C_2$ are sent to Facebook. The latter computes a HMAC $T_R$ (i.e. a tag) on $C_2$ and some context value (e.g. sender/receiver, timestamp), with its secret key $K_R$. This will allow Facebook to verify the commitment $C_2$ in case of message report, without storing every commitment $C_2$.

If the receiver wants to report an abusive message $M$, it sends the opening key $K_f$, the message $M$, Facebook's tag $T_R$ and the context data to Facebook. Then, Facebook can reconstruct the commitment $C_2$ from the received data and can verify this commitment by computing its HMAC with $K_R$ and comparing with the tag $T_R$ sent by the receiver.

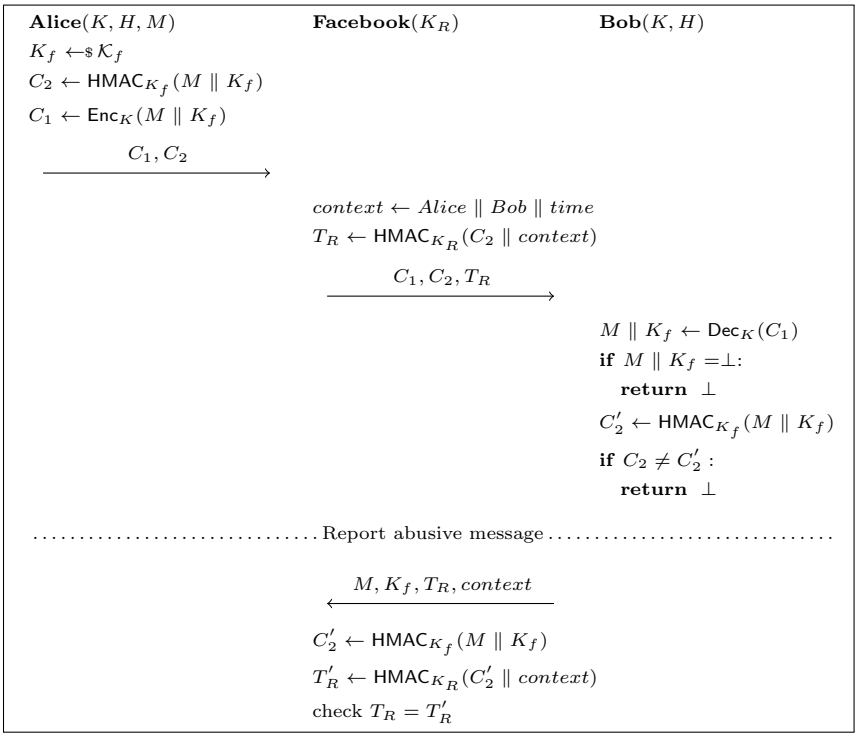


**Fig. 1.** Facebook message franking scheme where $\mathcal{K}_f$is the space for the one-time keys $K_f$.

### 3.2 Commiting AEAD

Grubbs et al. [GLR17] formalized the concept of message franking into the definition of *committing AEAD*. Roughly, the idea is to define a cryptographic primitive that creates a ciphertext and a commitment on the plaintext. Then, one can decrypt to retrieve the plaintext and an opening key, which is used to verify the commitement on the plaintext. We present here the definition of committing AEAD where all randomness is defined by a public nonce. Formally, a nonce-based committing AEAD is defined as follows:

**Definition 1 (Nonce-based Committing AEAD).** *A nonce-based committing AEAD scheme* $\mathsf{nCE} = (\mathsf{init}, \mathsf{enc}, \mathsf{dec}, \mathsf{vrf})$ *is a set of four algorithms. Associated to this scheme is a key space* $\mathcal{K}$*, a header space* $\mathcal{H}$*, a message space* $\mathcal{M}$*, a ciphertext space* $\mathcal{C}$*, an opening space* $\mathcal{K}_f$*, a franking tag space* $\mathcal{T}$ *and a nonce space* $\mathcal{N}$*. An error symbol* $\perp$ *is also required. The four algorithms are as follows:*

- $K \leftarrow\!\!\$\ \mathsf{init}$*: The initialization algorithm* $\mathsf{init}$ *outputs a random key* $K \in \mathcal{K}$*.*
- $(C_1, C_2) \leftarrow \mathsf{enc}_K(N, H, M)$*: The encryption algorithm* $\mathsf{enc}$ *takes a key* $K \in \mathcal{K}$*, a nonce* $N \in \mathcal{N}$*, a header* $H \in \mathcal{H}$ *and a message* $M \in \mathcal{M}$*, and it outputs a ciphertext* $C_1 \in \mathcal{C}$ *and a franking tag* $C_2 \in \mathcal{T}$*.*

- $(M, K_f) \leftarrow \mathsf{dec}_K(N, H, C_1, C_2)$*: The decryption algorithm* $\mathsf{dec}$ *takes a key* $K \in \mathcal{K}$*, a nonce* $N \in \mathcal{N}$*, a header* $H \in \mathcal{H}$*, a ciphertext* $C_1 \in \mathcal{C}$ *and a franking tag* $C_2 \in \mathcal{T}$*, and it outputs a message* $M \in \mathcal{M}$ *and an opening* $K_f \in \mathcal{K}_f$ *or an error symbol* $\perp$*.*
- $b \leftarrow \mathsf{vrf}(H, M, K_f, C_2)$*: The verification algorithm* $\mathsf{vrf}$ *takes a header* $H \in \mathcal{H}$*, a message* $M \in \mathcal{M}$*, an opening value* $K_f \in \mathcal{K}_f$ *and a franking tag* $C_2 \in \mathcal{T}$*, and it outputs a verification bit* $b$*, regarding the correctness of the reporting procedure.*

The first procedure $\mathsf{init}$ is randomized while the others are deterministic, since the randomness is defined by the nonce value $N$.

*Correctness.* For the correctness of the scheme, we require that for any $K \in \mathcal{K}$, $N \in \mathcal{N}$, $H \in \mathcal{H}$ and $M \in \mathcal{M}$ and $(C_1, C_2) = \mathsf{enc}_K(N, H, M)$

$$\Pr[\mathsf{dec}_K(N, H, C_1) = (M, K_f)] = 1$$

for some $K_f \in \mathcal{K}_f$.
Also, if we let $(C_1, C_2) = \mathsf{enc}_K(N, H, M)$ and $(M, K_f) = \mathsf{dec}_K(N, H, C_1, C_2)$ then we require that

$$\Pr[\mathsf{vrf}(H, M, K_f, C_2) = 1] = 1$$

for any $K \in \mathcal{K}$, $N \in \mathcal{N}$, $H \in \mathcal{H}$ and $M \in \mathcal{M}$.
Finally, we require the length of the ciphertexts $(C_1, C_2)$ to be deterministic given $H$ and $M$. In other words, there is a deterministic function $len(H, M)$ s.t. $(|C_1|, |C_2|) = len(H, M)$.

### 3.3 Security of committing AEAD

*Multi and single opening security.* In some cases, one would like to encrypt multiple ciphertexts with the same key and keep all ciphertexts secure even if an opening is known. This notion is called multi-opening security. For example, if the key used for encryption is the same as the opening key (i.e. $K = K_f$), the scheme would not be multi-opening secure since the knowledge of one opening $K_f$ would compromise all ciphertexts encrypted with the key $K$. On the other hand, if the secret key is meant to be used only once (e.g. Signal protocol), we can require only single-opening security. In this paper, we focus solely on multi-opening secure schemes.
*Confidentiality.* The multiple-opening confidentality of nonce-based committing AEAD is defined in MO-nREAL and MO-nRAND games (cf. 2). In these games an adversary $\mathcal{A}$ tries to differentiate between ciphertexts $(C_1, C_2)$ derived from a legit message and ciphertexts derived from random bitstrings. Observe that in both games, the adversary can query the decryption oracle $\mathcal{O}^{dec}$ only with ciphertexts that were output by the oracle $\mathcal{O}^{enc}$. This avoids simple wins, where the adversary submits the ciphertexts obtained from the $\mathcal{O}^{chal}$ oracle to the $\mathcal{O}^{dec}$ oracle. Also, to avoid trivial wins, we require the adversary playing these games to be nonce-respecting. This means that an adversary cannot query an

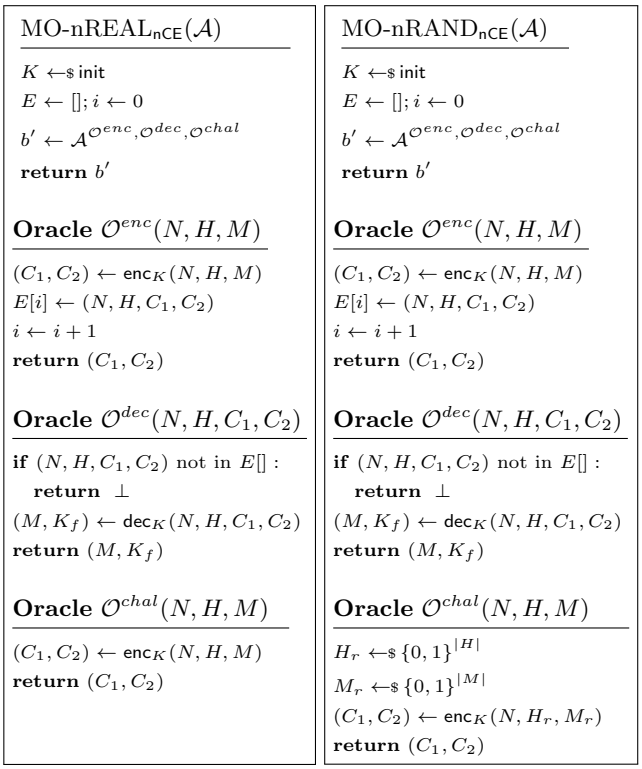


**Fig. 2.** Confidentality games for nonce-based committing AEAD

encryption oracle with the same nonce $N$ more than once (i.e. it cannot query $\mathcal{O}^{enc}$ or $\mathcal{O}^{chal}$ twice with the same $N$ nor can it query $\mathcal{O}^{enc}$ and $\mathcal{O}^{chal}$ with the same $N$).

We define the nonce-based multiple opening real-or-random (MO-nROR) advantage of any adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{mo-nror}}_{\mathsf{nCE}}(\mathcal{A}) = |\Pr[\text{MO-nREAL}_{\mathsf{nCE}}(\mathcal{A}) \Rightarrow 1] - \Pr[\text{MO-nRAND}_{\mathsf{nCE}}(\mathcal{A}) \Rightarrow 1]|$$

*Integrity.* The multiple-opening integrity of nonce-based committing AEAD is defined with the MO-nCTXT game (cf. Figure 3). In this game, an adversary can query an encryption oracle $\mathcal{O}^{enc}$ and a decryption oracle $\mathcal{O}^{dec}$. Then, it sends a header, a nonce and ciphertexts $(N,H,C_1,C_2)$ to a challenge oracle $\mathcal{O}^{chal}$. If the decryption is successful and the input values were not previously output by $\mathcal{O}^{enc}$, the adversary wins. Informally, the adversary wins if it can forge a valid tuple $(N,H,C_1,C_2)$. We define the nonce-based multiple opening ciphertext integrity (MO-nCTXT) advantage of any adversary $\mathcal{A}$ playing the MO-nCTXT game as

$$\mathsf{Adv}^{\text{mo-nctxt}}_{\mathsf{nCE}}(\mathcal{A}) = \Pr[\text{MO-nCTXT}_{\mathsf{nCE}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

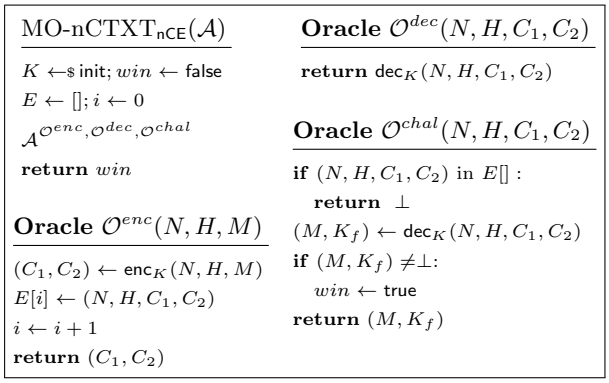


**Fig. 3.** Integrity game for nonce-based committing AEAD

*Binding security.* In addition to the standard security definitions (i.e. confidentiality and integrity), Grubbs et al. [GLR17] defined two binding security notions, namely *sender binding* and *receiver binding*.

The intuition behind sender binding is that a participant cannot send a message which cannot be reported later. The s-BIND game in Figure 4 defines this security notion. An adversary wins if it can find a ciphertext and a franking tag such that a receiving participant can decrypt but the verification of the franking tag with the obtained message fails. For any adversary $\mathcal{A}$ playing s-BIND and any committing AEAD $\mathsf{nCE}$, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{s-bind}}_{\mathsf{nCE}}(\mathcal{A}) = \Pr[\text{s-BIND}_{\mathsf{CE}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

The idea of receiver binding is that a malicious receiver should not be able to accuse another client of sending an abusive message that was never sent (i.e. it should not be able to report a message that was not sent). The game r-BIND in Figure 4 defines this security notion. An adversary wins if it can find two different messages such that their commitments are the same. For any adversary $\mathcal{A}$ playing the r-BIND game and any committing AEAD $\mathsf{nCE}$, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{r-bind}}_{\mathsf{nCE}}(\mathcal{A}) = \Pr[\text{r-BIND}_{\mathsf{CE}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

## 4 Cryptographic channel for message franking (MFC)

A cryptographic channel is a set of algorithms that allows several participants to exchange messages (i.e. send and receive) with confidentiality, message integrity, resistance to replay attacks, out-of-order delivery and message drops. For the reasons exposed above and the fact that a bidirectional channel is more generic than a unidirectional one, we are going to focus on a bidirectional channel where two participants (Alice and Bob) exchange messages. However, in the message

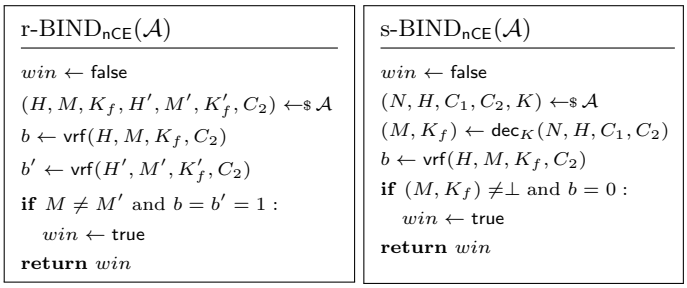


**Fig. 4.** Games for committing AEAD binding notions.

franking case, the participants do not only exchange messages but they also report them as abusive to a third entity, which we refer to as *router*. Therefore, we need to define a new model for a cryptographic channel, which we call a message franking channel (MFC). Informally, we raise the nonce-based committing AEAD concept to the channel level, in the context of message franking.

We define a message franking channel (MFC), using the syntax used by Marson et al. [MP17]:

**Definition 2 (Message Franking Channel).** *A message franking channel* $\mathsf{Ch} = (\mathsf{init}, \mathsf{snd}, \mathsf{tag}, \mathsf{rcv}, \mathsf{rprt})$ *is a five-tuple of algorithms. Associated to this channel is a key space* $\mathcal{K}$*, a nonce space* $\mathcal{N}$*, a header space* $\mathcal{H}$*, a message space* $\mathcal{M}$*, a ciphertext space* $\mathcal{C}$*, an opening space* $\mathcal{K}_f$*, a franking tag space* $\mathcal{T}$*, a router tag space* $\mathcal{T}_R$ *and a state space* $\mathcal{S}$*. The participants space is* $\mathcal{P} = \{A, B\}$ *(for Alice and Bob). We also require a special rejection symbol* $\perp \notin (\mathcal{K}_f \times \mathcal{M}) \cup \mathcal{S}$*. The five procedures are defined as follows:*

- $(st_A, st_B, st_R) \leftarrow\!\!\$\ \mathsf{init}$*: The initialization algorithm* $\mathsf{init}$ *samples a key* $K \in \mathcal{K}$ *and a key* $K_R \in \mathcal{K}$*, and it outputs initial states* $st_A, st_B, st_R \in \mathcal{S}$*.* $K$ *is the shared key resulting from a secure and authenticated key exchange protocol between both clients and* $K_R$ *is the secret key of the router.*
- $(st'_u, H, C_1, C_2) \leftarrow \mathsf{snd}(st_u, N, M)$*: The sending algorithm* $\mathsf{snd}$ *takes the sender's state* $st_u \in \mathcal{S}$*, a nonce* $N \in \mathcal{N}$ *and a message* $M \in \mathcal{M}$*, and it outputs an updated state* $st'_u \in \mathcal{S}$*, a header* $H \in \mathcal{H}$ *and a pair of ciphertext and franking tag* $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$*.*
- $(st'_R, T_R) \leftarrow \mathsf{tag}(st_R, id_s, H, C_2)$*: The router tagging algorithm* $\mathsf{tag}$ *takes the router's state* $st_R \in \mathcal{S}$*, the sender's identity* $id_s \in \mathcal{P}$*, a header* $H \in \mathcal{H}$ *and a franking tag* $C_2 \in \mathcal{T}$*, and it outputs an updated state* $st'_R \in \mathcal{S}$ *and a router tag* $T_R \in \mathcal{T}_R$*.*
- $(st'_u, M, K_f) \leftarrow \mathsf{rcv}(st_u, N, H, C_1, C_2)$*: The receiving algorithm* $\mathsf{rcv}$ *takes the receiver's state* $st_u \in \mathcal{S}$*, a nonce* $N \in \mathcal{N}$*, a header* $H \in \mathcal{H}$ *, a ciphertext* $C_1 \in \mathcal{C}$ *and a franking tag* $C_2 \in \mathcal{T}$*, and it outputs an updated state* $st'_u \in \mathcal{S} \cup \{\perp\}$*, and an opening value pair* $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$ *or* $\perp$*. We require* $st'_u = \perp$ *if* $(M, K_f) = \perp$*.*

– $(st'_R, b) \leftarrow \mathsf{rprt}(st_R, id_r, H, M, K_f, C_2, T_R)$ : *The router's verification algorithm* rprt *takes the router's state* $st_R \in \mathcal{S}$*, the reporter's identity* $id_r \in \mathcal{P}$*, a header* $H \in \mathcal{H}$*, a message* $M \in \mathcal{M}$*, an opening* $K_f \in \mathcal{K}_f$*, a franking tag* $C_2 \in \mathcal{T}$ *and a router tag* $T_R \in \mathcal{T}_R$*, and it outputs an updated router's state* $st'_R \in \mathcal{S} \cup \{\perp\}$ *and a verification bit* $b \in \{0, 1\}$*. We require* $st'_R = \perp$ *if* $b = 0$*.*

We assume the channel is stateful, i.e. the participants save their state between send/tag/receive/report calls. The rcv procedure can verify the commitment $C_2$ and outputs $\perp$ if the verification fails. Also, note that the rprt procedure does not depend on the nonce $N$ nor on the ciphertext $C_1$. In particular, this means that the router must be able to verify the validity of the router tag $T_R$ given only its state $st_R$, the header $H$ and the franking tag $C_2$. The channel uses $\perp$ to indicate an error. Once a state is marked as bogus (i.e. $st = \perp$), it cannot be used anymore in the invokation of the functions snd, rcv and rprt, since $\perp \notin S$. This corresponds to the reasonable behaviour of an application refusing to process any more data once an error has been detected. Error management in channels is a full topic in itself (e.g. [BDPS13]) and can lead to vulnerabilities (e.g. padding oracle attack [Vau02]). Here, we assume that an adversary does not learn anything from an error apart from the failure of the corresponding procedure. The randomness is uniquely determined by public nonces $N$. Nonces can be used to perform randomized encryption but also to generate a random opening key $K_f$, as in Facebook's scheme. The role of the nonce is determined by the underlying schemes used by the different algorithms. Finally, we give a visualization of the channel in Figure 5.

*Remarks.* The key $K_R$ is the router's secret key that can be used to generate the router tags $T_R$. In a real-life message franking protocol, all messages go through the router, where they get tagged. Otherwise, clients could bypass the tagging procedure and messages would not be reportable. Therefore, one could imagine that a unique procedure for sending and tagging would be sufficient. While making the MFC definition simpler, this would render the instantiation of a real MFC difficult. Indeed, snd is meant to be executed by a client while tag is run by the router. Thus, such a simplification would be impractical. However, in the oracles of adversarial models used in the following sections, snd and tag procedures will sometimes be considered as one operation, to model the fact that a message sent is always seen by the router.

It is important for the security of the scheme, in particular for the receiver-binding property, that the router knows the identity of the sender and the identity of the receiver. This is why these identities $id_s$ and $id_r$ are passed as arguments in the tag and rprt procedures, respectively. Obviously, the router should be able to verify these identities in order for the whole protocol to be secure, otherwise one could tag messages on behalf of another user. However, throughout this paper we assume that the parties and the router have established secure keys and that the router can authenticate the sender and the reporter in the corresponding procedures. It is a fair assumption since messaging protocols usually encrypt and authenticate the communications between a client and the server. For example, TextSecure (Signal's ancestor) used to encrypt client-to-

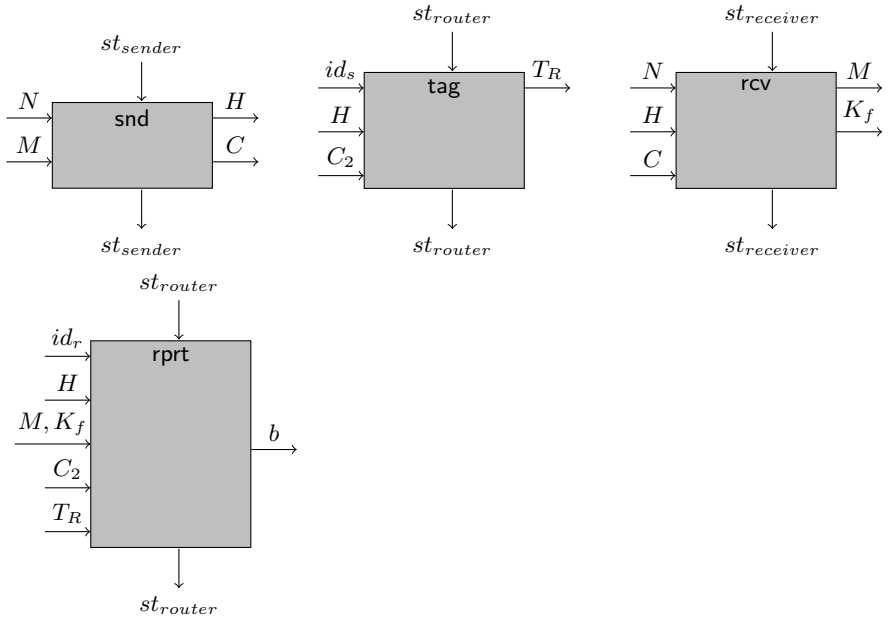


**Fig. 5.** Visualization of the message franking channel where $C$ stands for the pair $(C_1, C_2)$.

router communications with TLS and it used to authenticate the client with the phone number concatenated with some secret key [FMB+14]. Whatsapp encrypts communications to its servers with the Noise Protocol Framework and it stores the client's Curve25519 public key, allowing the router to authenticate the user during the Diffie-Hellman key exchange protocol [Per18,Wha17].

### 4.1 Correctness of the channel.

As in a standard bidirectional channel [MP17], we require a MFC to have certain properties. In short, we want messages sent by a client to be decryptable by the other participant without errors, assuming that the ciphertexts are not modified in the channel and that the order of the messages is preserved. Also, a message sent by a honest participant in the channel and correctly deciphered at the receiving end should be reportable, assuming that all messages sent on the channel are not modified (i.e. no active adversary).

Such requirements can be represented as a game, where only passive external adversaries (i.e. adversaries that can only see and relay messages) are allowed and where the participants are honest. The adversary can schedule snd/tag, rcv and rprt procedures and it wins if a message sent can not be decrypted or reported (i.e. rprt fails). This game is represented in Figure 6. We assume $u \in \{A, B\}$, $N \in \mathcal{N}$, $H \in \mathcal{H}$, $M \in \mathcal{M}$, $C_1 \in \mathcal{C}$, $C_2 \in \mathcal{T}$, $K_f \in \mathcal{K}_f$ and $T_R \in \mathcal{T}_R$. The

variables $s_A, s_B, r_A, r_B$ keep track of the number of messages sent and received by each participant. The variables $h_A$ and $h_B$ keep track of the state of each participant (whether it has received modified/out-of-order messages). Note that an adversary can be external and try to modify the messages on the channel or it can be a participant who communicates with the other benign participant and/or the router. If a participant $u$ receives out-of-order/modified data, (i.e. if the adversary actively attacks $u$), its variable $h_u$ is set to false. If $h_u = true$ we say that $u$ is clean. Observe that the adversary can win only if the participant $u$ used in the oracle query is clean. $M_{A,B}[]$, $M_{B,A}[]$, $C_{A,B}[]$ and $C_{B,A}[]$ record the messages and ciphertexts exchanged. The adversary has access to three oracles:

- $\mathcal{O}^{snd}(u, N, M)$: The $\mathcal{O}^{snd}$ oracle takes the client identity $u$, a message and a nonce, and it calls the $\mathsf{snd}$ and $\mathsf{tag}$ procedures on behalf of the participant $u$. Then, it records the resulting ciphertext and franking/router tags, if the client is still clean. This prevents the adversary from winning if the client $u$ had previously received out-of-order/modified messages.
- $\mathcal{O}^{rcv}(u, N, H, C_1, C_2)$: With the oracle $\mathcal{O}^{rcv}$, an adversary can make a user $u$ receive (i.e. decrypt). Now, if this participant is still clean and the ciphertext is delivered without modification (and in the right order) compared to the one sent (condition in line 4), then the message recovered should be the one sent. If this is not the case (condition in line 6), then the adversary wins. Otherwise, if the condition at line 4 is not respected, then the ciphertext/tags have been modified and the participant is flagged as not clean.
- $\mathcal{O}^{rprt}(u, n, H, M, K_f, C_2, T_R)$: The adversary can use also the $\mathcal{O}^{rprt}$ oracle to report a given message on behalf of a user $u$, by providing the message along with the tags. It also must specify the index $n$ of the message (we assume the adversary records the number of messages sent). As before, if the user is clean and the header, the franking tag, the router tag and the message were not modified compared to the one sent (conditions at lines 2-3), then the participant/adversary should be able to report the message. If this is not the case, the adversary wins. Otherwise, if the condition in line 3 is not respected (i.e. the message, ciphertext or tags have been tampered with), the participant is flagged as not clean.

Now, for any adversary $\mathcal{A}$ playing CORR and any channel $\mathsf{Ch}$, we denote the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{corr}}_{\mathsf{Ch}}(\mathcal{A}) = \Pr[\mathrm{CORR}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

**Definition 3 (Bidirectional Message Franking Channel Correctness).** *We say that a message franking bidirectional channel* $\mathsf{Ch}$ *is correct if for any adversary* $\mathcal{A}$ *playing the CORR game*

$$\mathsf{Adv}^{\mathrm{corr}}_{\mathsf{Ch}}(\mathcal{A}) = 0$$

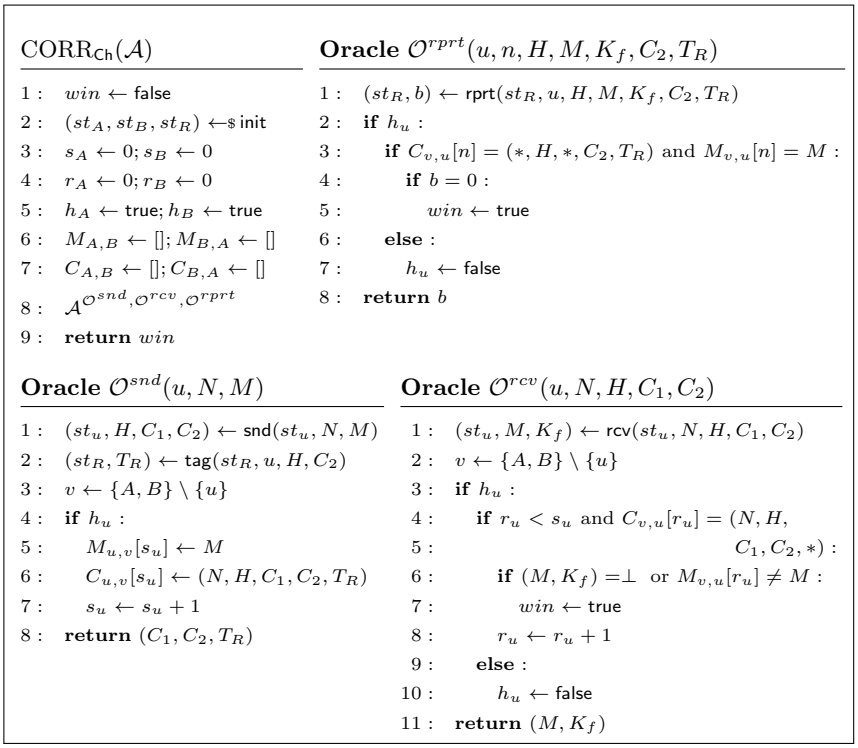


**Fig. 6.** Correctness game for a bidirectional message franking channel.

*Unidirectional correctness.* Finally, for a unidirectional channel (i.e. only Alice sends messages to Bob), the game of correctness is equivalent to the bidirectional game of correctness as we see in Figure 7. The main differences are that the participants are fixed in each oracle, the sender is always clean (since it cannot receive messages) and we need less variables (only one copy of $s$, $r$, $h$, $M$ and $C$).

## 5 Security for message franking channel

### 5.1 Confidentiality

We elevate the MO-nRAND confidentiality game for nonce-based committing AEAD schemes (see Section 3.3) to the bidirectional MFC case. Confidentiality means that no adversary is able to retrieve information about the messages exchanged by the participants. This notion concerns the exchange of messages and not the reporting phase. In addition, we note that once an abusive message is reported, it becomes public, in the sense that we do not specify how the message is sent to the router for reporting (e.g. it could be sent in clear).

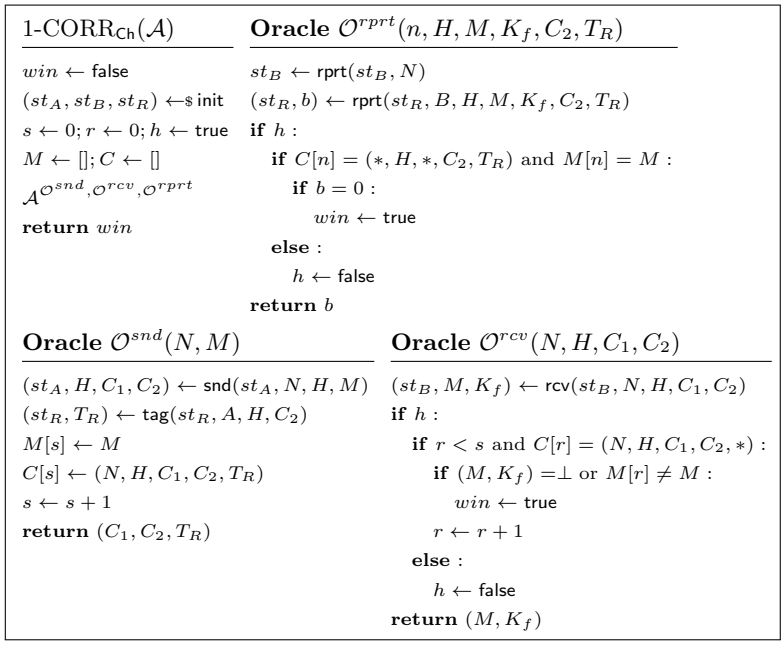


**Fig. 7.** Correctness game for a unidirectional message franking channel.

While in a standard channel the adversary is external to the participants, in the message franking case the adversary can also be the router. Therefore, in the following games we give the adversary access to the router's state $st_R$. This means the adversary does not need a tag and report oracle, as it is able to run these procedures on its own.

The MO-2nREAL and MO-2nRAND games of Figure 8 are adapted from the MO-nREAL and MO-nRAND games of Section 3.3. In both games, we assume the adversary is nonce-respecting (i.e. it cannot query $\mathcal{O}^{snd}$ or $\mathcal{O}^{chal}$ twice with the same nonce $N$.

The adversary wins if it can differentiate with non-negligible probability between the encryption of a message $M$ and the encryption of a random bitstring. The $h_u$ variables, as in the correctness games, let the adversary decrypt ciphertexts as long as it remains passive: the adversary only relays messages. In particular, this prevents the adversary from winning by decrypting the ciphertexts obtained from the $\mathcal{O}^{chal}$ oracle.

For a bidirectional $\mathsf{MFC}$ channel $\mathsf{Ch}$, we define the nonce-based multiple opening real-or-random (MO-2nRoR) advantage of any algorithm $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{mo-2nror}}_{\mathsf{Ch}}(\mathcal{A}) = |\Pr[\text{MO-2nREAL}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow 1] - \Pr[\text{MO-2nRAND}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow 1]|$$

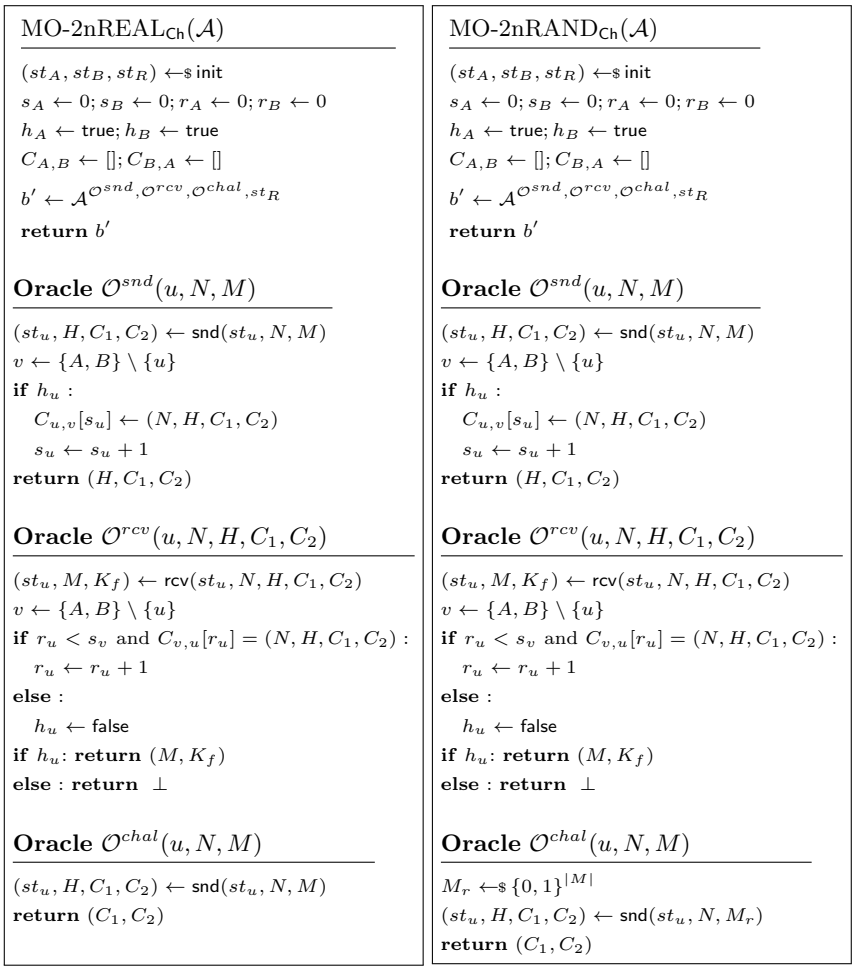


**Fig. 8.** Confidentality games for nonce-based bidirectional message franking channels.

### 5.2 Integrity

We adapt the MO-nCTXT integrity notions of committing AEAD to the bidirectional MFC. Ciphertext integrity means that a receiver can only receive and decrypt in-order and legitimate ciphertexts, which have been sent by another participant. As in the MO-2nROR confidentiality notion, the adversary can be the router and thus we give access to the state $st_R$.

Let MO-2nCTXT be the game in Figure 9. By observing the $\mathcal{O}^{rcv}$ oracle, one can see that the adversary wins if the rcv procedure is successful but the ciphertexts are not legitimate or they are received out-of-order. For a bidirectional MFC Ch, we define the nonce-based multiple opening integrity advantage

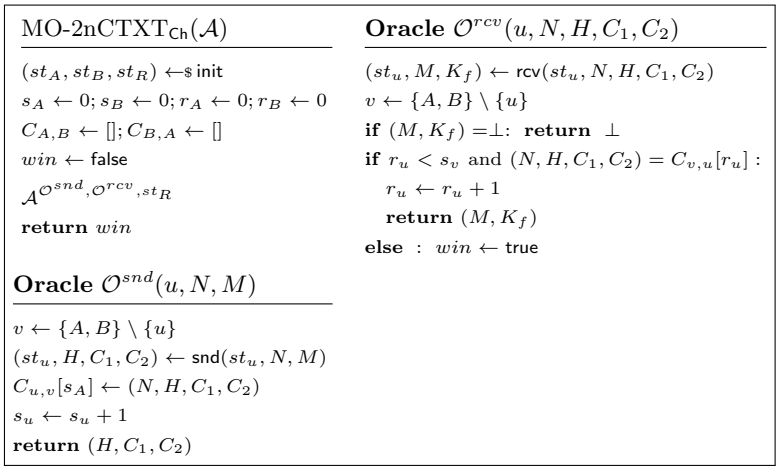


**Fig. 9.** Integrity game for nonce-based committing AEAD

of any algorithm $\mathcal{A}$ playing the MO-2nCTXT game as

$$\mathsf{Adv}^{\text{mo-2nctxt}}_{\mathsf{Ch}}(\mathcal{A}) = \Pr[\text{MO-2nCTXT}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

### 5.3 Binding security notions

In order to guarantee verifiable reporting of abusive messages, a MFC should adhere to sender and receiver binding notions as with message franking protocols [Fac16,GLR17]. One difference between the committing AEAD binding definitions and the ones defined here, is that in a channel the participants are stateful and their behavior may evolve over time, whereas such concepts do not exist in the security definitions of cryptographic primitives like committing AEAD. The threat in such security definitions is a malicious participant and not an external adversary. Therefore, in all the binding games of this section, we assume that the adversary can modify the states of the participants (Alice and Bob). In particular, this allows the adversary to control sthe encryption key $K$.

*Sender binding.* Sender binding for MFC guarantees that any message received without error by a client can be successfully reported to the router. This notion is defined with the game s-2BIND presented in Figure 10.

The adversary has access to three oracles in addition to the states of Alice and Bob. Here is a description of the s-2BIND game:

– s-2BIND$_{\mathsf{Ch}}(\mathcal{A})$: The $r_A, r_B, s_A, s_B$ variables keep track of the number of messages received and sent by each party. The arrays $S_{A,B}[], S_{B,A}[]$ store the franking tags $C_2$, the headers $H$ and the associated router tags $T_R$ sent from $A$ to $B$ and from $B$ to $A$, respectively. The arrays $R_{A,B}[]$ and $R_{B,A}[]$ store the tags and messages corresponding to ciphertexts when the rcv procedure outputs a valid pair $(M, K_f)$, after decryption.

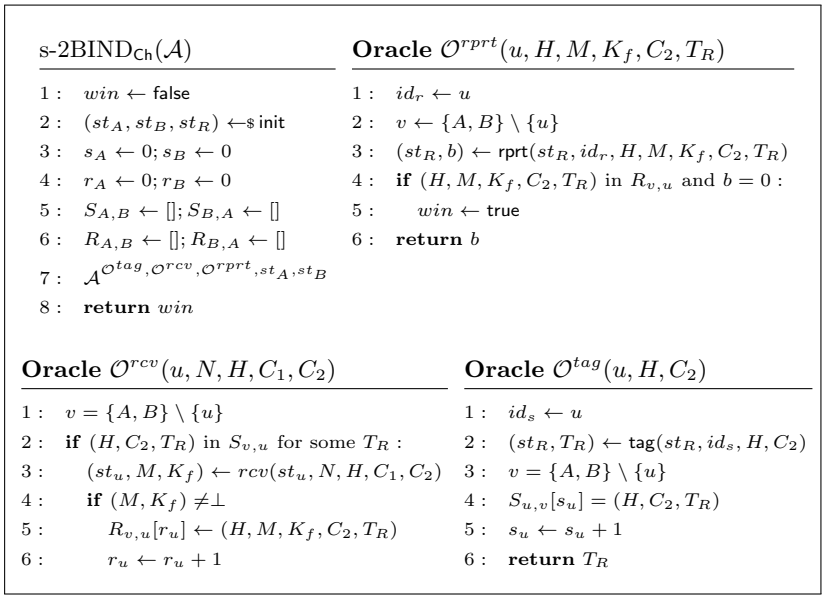


**Fig. 10.** Game for message franking channel sender-binding security definition.

- $\mathcal{O}^{tag}(u, H, C_2)$: This oracle is used to modify the router state $st_R$ and to obtain a router tag for a tuple $(H, C_2)$, with $u$ as the sender. The header $H$, the franking tag $C_2$ and the router tag obtained $T_R$ are stored in the corresponding array $S_{u,v}[]$ (line 4). This array is needed to ensure that a router tag $T_R$ really corresponds to a header,franking tag pair $(H, C_2)$. Then, the number of messages sent by the participant is incremented (line 5). Finally, the router tag is returned to the adversary (line 6).
- $\mathcal{O}^{rcv}(u, N, H, C_1, C_2)$: The $\mathcal{O}^{rcv}$ oracle first checks that $H$ and $C_2$ corresponds to a router tag $T_R$ (line 2) computed for a message sent by client $v$. If this is the case, the $\mathsf{rcv}$ procedure is called and it outputs a plaintext, opening key pair $(M, K_f)$. If this pair is valid (i.e. not equal to $\perp$), it is stored in the appropriate array $R_{v,u}$ alongside with the header and the tags ($C_2$, $T_R$) (line 5). This means that these values correspond to a ciphertext that decrypts to a valid message. Finally, the number of valid messages received by $u$ is incremented (line 6).
- $\mathcal{O}^{rprt}(u, H, M, K_f, C_2, T_R)$: The oracle $\mathcal{O}^{rprt}$ calls the $\mathsf{rprt}$ procedure on the the header, the message, the opening key and the tags, with $u$ as the reporter. This allows an adversary to update the router state $st_R$. Then, if the input values correspond to some ciphertext that outputs a valid message but the report procedure fails (line 4), the adversary wins.

The adversary can use the $\mathcal{O}^{rprt}$ oracle to report a message on behalf of a user $u$. If the message was actually sent and correctly decrypted by $u$ then it is in $R_{v,u}$ with the corresponding header and tags. Therefore, if the message is in $R_{v,u}$ but

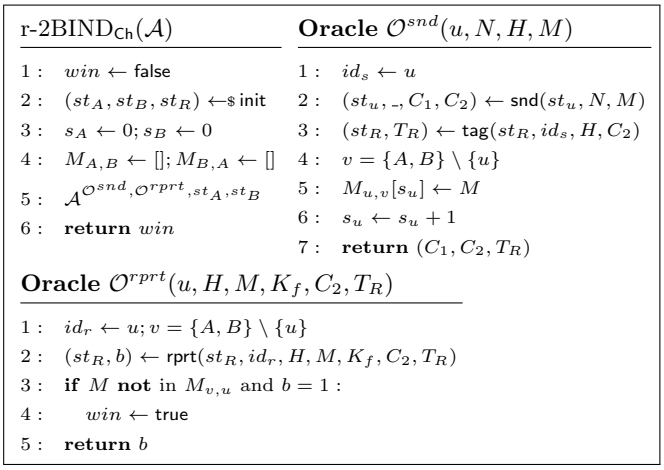


**Fig. 11.** Game for franking channel receiver-binding security definition.

the rprt procedure fails, the adversary wins. Indeed, we ask an adversary to win if the decryption is successful but the report procedure fails. We need to use a second pair of arrays $(S_{A,B}, S_{B,A})$ to store the router tag $T_R$ corresponding to a pair $(H, C_2)$. Otherwise, one could not store $T_R$ in the array $R_{v,u}$ at line 5 of $\mathcal{O}^{rcv}$. Indeed, if $T_R$ was not stored in $R_{v,u}$, the adversary could specify a random $T_R$ when reporting with legit values, making the verification fail. Finally, we note that despite the fact the adversary has access to both user states, in a real-life threat model the adversary would be only one of the participants (e.g. Alice). However, since both participants share everything, it does not matter if the adversary has access to one or both states. For any adversary $\mathcal{A}$ playing s-2BIND and any channel $\mathsf{Ch}$, the advantage of $\mathcal{A}$ is:

$$\mathsf{Adv}^{\mathrm{s-2bind}}_{\mathsf{Ch}}(\mathcal{A}) = \Pr[\text{s-2BIND}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

*Receiver binding.* We recall that receiver binding assures that a malicious participant cannot report an abusive message that was never sent. Receiver binding for message franking channels is defined with the game r-2BIND$_{\mathsf{Ch}}$ presented in Figure 11. In this game, the adversary represents two colluding participants that can schedule snd/tag/rprt operations for message exchanges and then one of the participants tries to report a message never sent to her/him. Since the adversary controls the users states, it would need only access to tag/rprt oracles since it can run the other algorithms by itself. However, the game should store the sent messages (i.e. the ones the adversary requested a router tag for) in order to compare them with the reported message in the $\mathcal{O}^{rprt}$ oracle. Therefore, we let the $\mathcal{O}^{snd}$ oracle run the snd procedure as well as the tag procedure. Here is a description of the game:

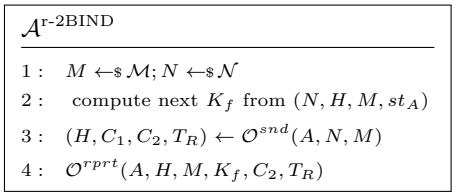


**Fig. 12.** Attack on r-BIND game.

- r-2BIND$_{\mathsf{Ch}}(\mathcal{A})$: The $s_A, s_B$ variables keep track of the number of messages sent by each party. They are used as indexes for the arrays $M_{A,B}[], M_{B,A}[]$ that store all messages sent by $A$ to $B$ and $B$ to $A$, respectively.
- $\mathcal{O}^{snd}(u, N, H, M)$: In the $\mathcal{O}^{snd}$ oracle, the message sent $M$ is recorded in the corresponding array (line 5) and the number of messages sent by the participant is incremented (line 6). The ciphertext, franking tag and router tag are returned to the adversary (line 7). In short, this oracle allows an adversary to obtain the router tag for any header/message tuple, while recording the message. We let the adversary pass the header $H$ as an argument since we want the adversary to be able to get a router tag for any header.
- $\mathcal{O}^{rprt}(u, H, M, K_f, C_2, T_R)$: The $\mathcal{O}^{rprt}$ takes a header $H$, the message to be reported $M$, an opening key $K_f$, a franking tag $C_2$, a router tag $T_R$ and the identity of the reporter $u$. At line 3, the condition checks that the submitted message was not sent to the reporting participant and that the message passes the $\mathsf{rprt}$ procedure. If this is the case, the adversary wins.

In order to respect this receiver binding notion, the router must be able to check the sender and receiver of a message. Indeed, if this is not the case, one can construct an adversary $\mathcal{A}^{\text{r-2BIND}}$ that always wins, as shown in Figure 12. Let Alice be the malicious participant. She picks a random message and a random nonce and sends the corresponding ciphertext to Bob using the sending oracle. Thus, the message is in the array $M_{A,B}$. Note that Alice can compute the opening key $K_f$ since she controls $st_A, N, H$ and $M$. Finally, she reports Bob as the sender of the message by calling the $\mathcal{O}^{rprt}$ oracle. Since the message is in $M_{A,B}$ but not in $M_{B,A}$, the first part of the condition in line 3 of $\mathcal{O}^{rprt}$ is fulfilled. Thus, since all values used to report are legitimate, the report procedure will succeed unless the router knows the message was actually sent by Alice to Bob.
One solution to this problem is to incorporate the receiver/sender identities in the router tag $T_R$. This is done in the Facebook protocol by putting the sender and receiver identities in some context data. The router tag $T_R$ becomes $T_R = \mathsf{HMAC}_{K_R}(C_2 \parallel \mathit{sender} \parallel \mathit{receiver})$ and the router can check whether the participant $A$ reporting an abusive message from participant $A$ is telling the truth or not and accept or reject accordingly. This shows the importance to correctly manage the identity of the participants.

For any adversary $\mathcal{A}$ playing r-2BIND and any channel $\mathsf{Ch}$, we write the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathrm{r-2bind}}_{\mathsf{Ch}}(\mathcal{A}) = \Pr[\text{r-2BIND}_{\mathsf{Ch}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

In other words, in addition to the control of participants' states, the adversary must be able to modify the state of the router, as in a real exchange of messages. *Unidirectional binding notions.* We present in Figure 13 the games r-1BIND and s-1BIND defining unidirectional receiver and sender binding notions, respectively. These games are similar to their bidirectional equivalent. However, in a unidirectional $\mathsf{MFC}$, if $A$ is the sender and $B$ is the receiver, then $\mathsf{snd}(st_u, *)$ fails if $u \neq A$ , $\mathsf{tag}(st_R, id_s, *)$ fails if $id_s \neq A$, $\mathsf{rprt}(st_R, id_r, *)$ fails if $id_r \neq B$ and $\mathsf{rcv}(st_u, *)$ fails if $u \neq B$. Therefore, the adversary can query $\mathcal{O}^{snd}, \mathcal{O}^{tag}$ only with $u = A$ and $\mathcal{O}^{rcv}, \mathcal{O}^{rprt}$ only with $u = B$. For simplicity we omit the argument $u$ in all oracles.

## 6 Binding notions in the canonic composition

A bidirectional standard cryptographic channel made of two secure unidirectional channels is not necessarily secure [MP17]. Therefore, it is of interest to study the binding security of such constructions adapted to the $\mathsf{MFC}$. More precisely, we want to analyze the relations between unidirectional and bidirectional receiver/sender binding, in a special channel called the *canonic composition* [MP17]. This channel captures the idea that real-life communication protocols are often designed to be the composition of two independent unidirectional secure channels.

### 6.1 Message franking canonic composition

The canonic construction is the straightforward composition of two unidirectional $\mathsf{MFC}$, when one party sends and the other receives and reports. Let $\mathsf{Ch} = (\mathsf{init}, \mathsf{snd}, \mathsf{tag}, \mathsf{rcv}, \mathsf{rprt})$ be a unidirectional $\mathsf{MFC}$ with a participants space $\mathcal{P}$, a key space $\mathcal{K}$, a nonce space $\mathcal{N}$, a header space $\mathcal{H}$, a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, an opening key space $\mathcal{K}_f$, a franking tag space $\mathcal{T}$, a router tag space $\mathcal{T}_R$ and a state space $\mathcal{S}$. The canonic composition will use one instance of $\mathsf{Ch}$ for the communication from Alice to Bob ($\rightarrow$) and another for the communication from Bob to Alice ($\leftarrow$).
Let $\mathsf{Ch}' = (\mathsf{init}', \mathsf{snd}', \mathsf{tag}', \mathsf{rcv}', \mathsf{rprt}')$ be the bidirectional $\mathsf{MFC}$ resulting from the canonical composition of $\mathsf{Ch}$ described in Figure 14. Associated to this channel $\mathsf{Ch}'$ is a participants space $\mathcal{P}' = \mathcal{P}$, a key space $\mathcal{K}' = \mathcal{K} \times \mathcal{K}$, a nonce space $\mathcal{N}' = \mathcal{N}$, a header space $\mathcal{H}' = \mathcal{H}$, a message space $\mathcal{M}' = \mathcal{M}$, a ciphertext space $\mathcal{C}' = \mathcal{C}$, an opening key space $\mathcal{K}'_f = \mathcal{K}_f$, a franking tag space $\mathcal{T}' = \mathcal{T}$, a router tag space $\mathcal{T}'_R = \mathcal{T}_R$ and a state space $\mathcal{S}' = \mathcal{S} \times \mathcal{S}$.

A participant's state consists of a state used to receive and another to send, initialized in the $\mathsf{init}'$ procedure. This procedure also creates a router state for

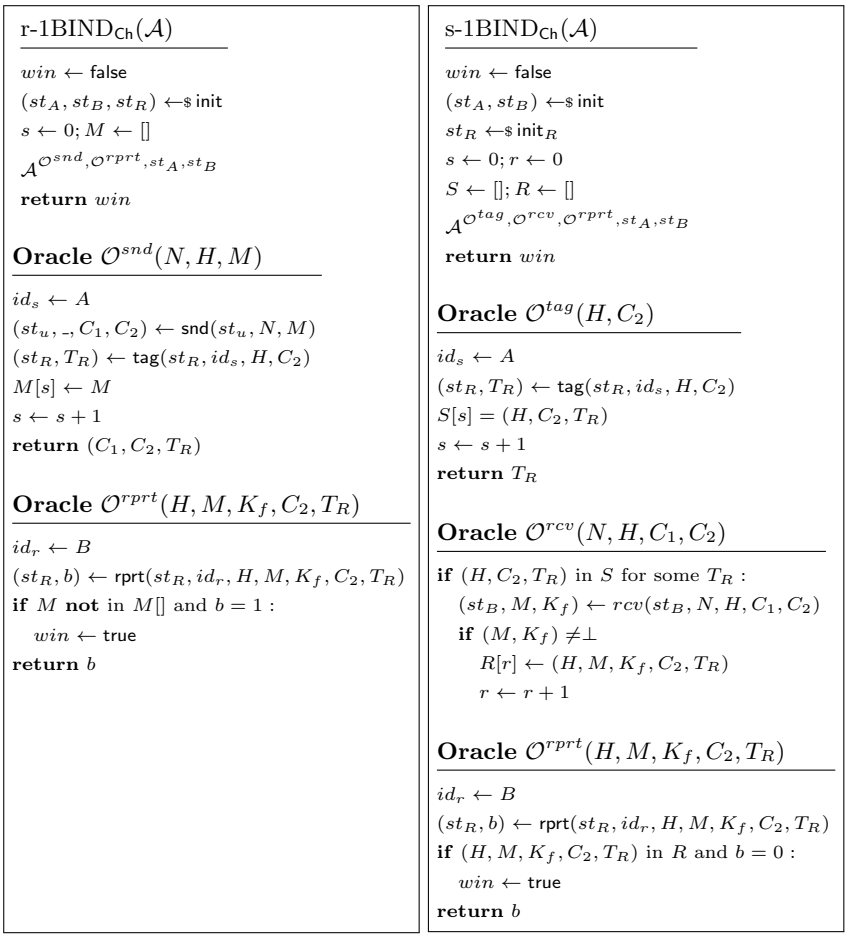


**Fig. 13.** Games for receiver and sender binding security definitions in the unidirectional MFC case.

the $\rightarrow$ channel and another for the $\leftarrow$ channel. Then, when a party wants to send a message, $\mathsf{snd}'$ extracts the sending state and uses it as the state in the $\mathsf{snd}$ call. The sender and reporter's identities $id_s$ and $id_r$ are used in $\mathsf{tag}'$ and $\mathsf{rprt}'$ to determine the direction of the communication (i.e. $\rightarrow$ or $\leftarrow$). This is necessary for choosing the right router state $st_R$, which is itself needed to invoke $\mathsf{tag}$ and $\mathsf{rprt}$. When a client receives a message, the receiving state is extracted and $\mathsf{rcv}$ is called. If an error occurs during this procedure, the state is set to $\perp$.

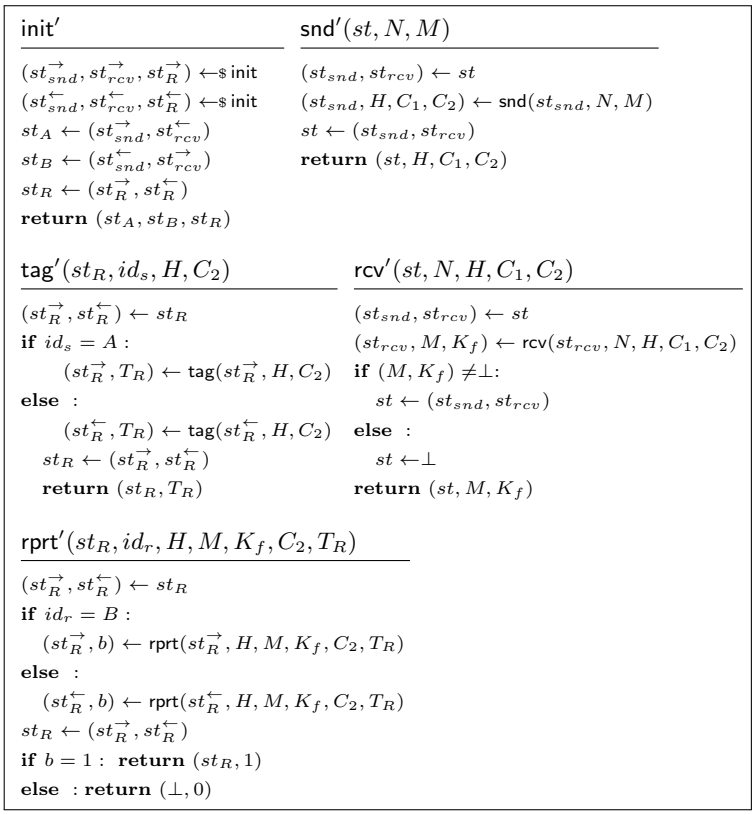


**Fig. 14.** Canonic composition of two unidirectional MFC.

### 6.2 Security analysis

Let $\mathsf{Ch}$ be a unidirectional $\mathsf{MFC}$ and $\mathsf{Ch}'$ be the bidirectional $\mathsf{MFC}$ resulting from the canonic composition of two channels $\mathsf{Ch}$. We study the relations between the bidirectional binding notions r-2BIND, s-2BIND and their equivalent in the unidirectional case r-1BIND, s-1BIND. We prove the two following results:

$$\text{r-1BIND} + \text{r-1BIND} \iff \text{r-2BIND} \tag{4}$$

$$\text{s-1BIND} + \text{s-1BIND} \iff \text{s-2BIND} \tag{5}$$

The $\Leftarrow$ direction is trivial in both cases, as if one is able to break the binding security of one of the two channels composing $\mathsf{Ch}'$, then $\mathsf{Ch}'$ will not be binding either. Therefore, we state only the $\implies$ relations.

**Theorem 1 (Receiver binding).** *Let* $\mathsf{Ch}$ *be a receiver binding unidirectional* $\mathsf{MFC}$ *(r-1BIND). Then, its canonic composition* $\mathsf{Ch}'$ *is receiver binding (r-2BIND). More precisely, for any adversary* $\mathcal{A}$ *playing r-2BIND against* $\mathsf{Ch}'$*, there exists a r-1BIND adversary* $\mathcal{B}$ *against* $\mathsf{Ch}$ *s.t.*

$$\mathsf{Adv}^{r\text{-}2bind}_{\mathsf{Ch}'}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{r\text{-}1bind}_{\mathsf{Ch}}(\mathcal{B})$$

**Theorem 2 (Sender binding).** *Let* $\mathsf{Ch}$ *be a sender binding unidirectional* $\mathsf{MFC}$ *(r-1BIND). Then, its canonic composition* $\mathsf{Ch}'$ *is sender binding (s-2BIND). More precisely, for any adversary* $\mathcal{A}$ *playing s-2BIND against* $\mathsf{Ch}'$*, there exists a s-1BIND adversary* $\mathcal{B}$ *against* $\mathsf{Ch}$ *s.t.*

$$\mathsf{Adv}^{s\text{-}2bind}_{\mathsf{Ch}'}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{s\text{-}1bind}_{\mathsf{Ch}}(\mathcal{B})$$

Due to space contraints proofs for Theorem 1 and 2 are deffered in the appendix section.

### 6.3 Generic Composition

Both Theorem 1 and Theorem 2 hold in the case of the canonic composition but not for any composition. For example, one can easily design a bidirectional $\mathsf{MFC}$ which is not r-2BIND from two r-1BIND unidirectional $\mathsf{MFC}$. Consider the canonic composition of Figure 14 but we modify it such that the communication keys and the router keys are the same in both directions (i.e. $K^{\rightarrow} = K^{\leftarrow}$ and $K_R^{\rightarrow} = K_R^{\leftarrow}$). That can be achieved by making only one call to $\mathsf{init}$ in $\mathsf{init}'$ and making each state equal in both directions. These modifications do not change the r-1BIND security of the channel. In particular, if the unidirectional $\mathsf{MFC}$ used in the composition is r-1BIND, then the composition is still r-1BIND in each direction.

Now consider the adversary $\mathcal{A}^{\text{r-2BIND}}$ of Figure 12 presented in Section 5.3. It is easy to see that this adversary will win with probability 1 against the

new composition. Indeed, it is interesting to observe that the original canonic composition is r-2BIND secure because the router keys $K_R^{\rightarrow}$ and $K_R^{\leftarrow}$ are likely to be different. Therefore, they prevent $\mathcal{A}^{\text{r-2BIND}}$ from winning since the router key from the $\rightarrow$ channel is used to generate the router tag $T_R$ but the adversary tries to report a message in the $\leftarrow$ channel. In fact, the router keys act as the identifiers of the sender/receiver in the router tag. Hence, if the router keys are the same for both directions (as in the modified composition), the resulting bidirectional MFC will not be r-2BIND secure.

## 7 MFC Instantiation

Now that the security properties of a MFC have been defined, we wish to instantiate a practical MFC that fulfills these properties. Our construction is based on two cryptographic primitives, namely a committing AEAD and a message authentication code (MAC) scheme. We show how to combine these primitives with some message counters to obtain a practical bidirectional MFC.

### 7.1 Message Authentication Code (MAC)

We recall the definition of MAC:

**Definition 4.** *A message authentication code (MAC) scheme is a tuple of three algorithms* $\mathsf{MAC} = (\mathsf{init}_{\mathsf{mac}}, \mathsf{tag}_{\mathsf{mac}}, \mathsf{vrfy}_{\mathsf{mac}})$*. Associated to the scheme is a key space* $\mathcal{K}$*, a message space* $\mathcal{M}$ *and a tag space* $\mathcal{T}$*. The three procedures of a MAC scheme operate as follows:*

- $K \leftarrow_{\$} \mathsf{init}_{\mathsf{mac}}$*: The initialization procedure samples a new key* $K \leftarrow_{\$} \mathcal{K}$ *and returns it.*
- $T \leftarrow \mathsf{tag}_{\mathsf{mac}}(K, M)$*: The tagging procedure* $\mathsf{tag}_{\mathsf{mac}}$ *takes a key* $K \in \mathcal{K}$ *and a message* $M \in \mathcal{M}$*, and it outputs a tag* $T \in \mathcal{T}$*.*
- $b \in \mathsf{vrfy}_{\mathsf{mac}}(K, M, T)$*: The verification algorithm* $\mathsf{vrfy}_{\mathsf{mac}}$ *takes a* $K \in \mathcal{K}$*, a message* $M \in \mathcal{M}$ *and a tag* $T \in \mathcal{T}$*, and it outputs a result bit* $b \in \{0, 1\}$*. It is required that* $\mathsf{vrfy}_{\mathsf{mac}}(K, M, T)$ *outputs* 1 *if* $T = \mathsf{tag}_{\mathsf{mac}}(K, M)$ *and* 0 *otherwise.*

The usual security property for a MAC scheme MAC is unforgeability under chosen-message attack (UF-CMA). It is defined by the game UF-CMA presented in Figure 15. In this game, the adversary can query tags for any message and verify any tag given a message. The adversary wins if the verification is successful in $\mathcal{O}^{vrfy}$ but the message was not queried to the tag oracle $\mathcal{O}^{tag}$. For a given MAC scheme MAC, we define the UF-CMA advantage of any adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\text{uf-cma}}_{\mathsf{MAC}}(\mathcal{A}) = \Pr[\text{UF-CMA}_{\mathsf{MAC}}(\mathcal{A}) \Rightarrow \mathsf{true}]$$

| UF-CMA$_{\mathsf{MAC}}(\mathcal{A})$ | **Oracle** $\mathcal{O}^{tag}(M)$ | **Oracle** $\mathcal{O}^{vrfy}(M,T)$ |
|---|---|---|
| $win \leftarrow$ **false** ; $K \leftarrow\!\!\$\ \mathcal{K}$ | $S[i] \leftarrow M$ | $b \leftarrow \mathsf{vrfy}_{\mathsf{mac}}(K,M,T)$ |
| $i \leftarrow 0; S \leftarrow []$ | $i \leftarrow i+1$ | **if** $b = 1$ and $M \notin S[]$ |
| $\mathcal{A}^{\mathcal{O}^{tag},\mathcal{O}^{vrfy}}$ | $T \leftarrow \mathsf{tag}_{\mathsf{mac}}(K,M)$ | $win \leftarrow$ **true** |
| **return** $win$ | **return** $T$ | **return** $b$ |

**Fig. 15.** Unforgeability under chosen-message attack game.

### 7.2 Construction

Let $\mathsf{nCE} = (\mathsf{init}_{\mathsf{nCE}}, \mathsf{enc}, \mathsf{dec}, \mathsf{vrfy}_{\mathsf{nCE}})$ be a secure nonce-based committing AEAD and $\mathsf{MAC} = (\mathsf{init}_{\mathsf{mac}}, \mathsf{tag}_{\mathsf{mac}}, \mathsf{vrfy}_{\mathsf{mac}})$ be a secure MAC scheme. Our $\mathsf{MFC}$ construction is given in Figure 16. We refer to it as $\mathsf{MF}$. The channel operates as follows:

- $(st_A, st_B, st_R) \leftarrow\!\!\$\ \mathsf{init}$: The initialization procedure samples the keys and creates the states. Alice and Bob states are made of their identity, the secret key and the send and receive counters. The router's state is made of the router key.
- $(st, H, C_1, C_2) \leftarrow \mathsf{snd}(st, N, M)$: The sending procedure computes the header $H$ as the identity concatenated with the number of sent messages. Then, the ciphertexts are computed, the sent counter is incremented and the values are returned.
- $(st_R, T_R) \leftarrow \mathsf{tag}(st_R, id_s, H, C_2)$: The router tag is computed as a MAC on the sender/receiver identities, the header and the franking tag, with the router key $K_R$.
- $(st, M, K_f) \leftarrow \mathsf{rcv}(st, N, H, C_1, C_2)$: The sender identity and the message sequence number are extracted from the header and the ciphertext is decrypted. If this fails or if the number of received messages is not equal to the message sequence number, an error is returned. Otherwise, the number of received messages is incremented and the new state, the plaintext and the opening key are returned.
- $(st_R, b) \leftarrow \mathsf{rprt}(st_R, id_r, H, M, K_f, C_2, T_R)$: The reporter's identity $u$ and the alleged sender's identity $v$ are extracted. Then, the router's tag $T_R$ is verified with the identities $v$ and $u$ along with the header $H$ and the franking tag $C_2$. Finally, the message is verified to be legit using the franking tag and the opening key $K_f$. If everything is successful, the procedure returns 1, as a success.

### 7.3 Security analysis

*Confidentiality.* The ciphertexts sent by $\mathsf{MF}$ are output by the nonce-base committing AEAD $\mathsf{nCE}$. Then, if an adversary can differentiate between real and random ciphertexts output by $\mathsf{MF}$, then it can differentiate between real and random ciphertexts output by $\mathsf{nCE}$. Therefore, if $\mathsf{nCE}$ is MO-nRoR secure, then

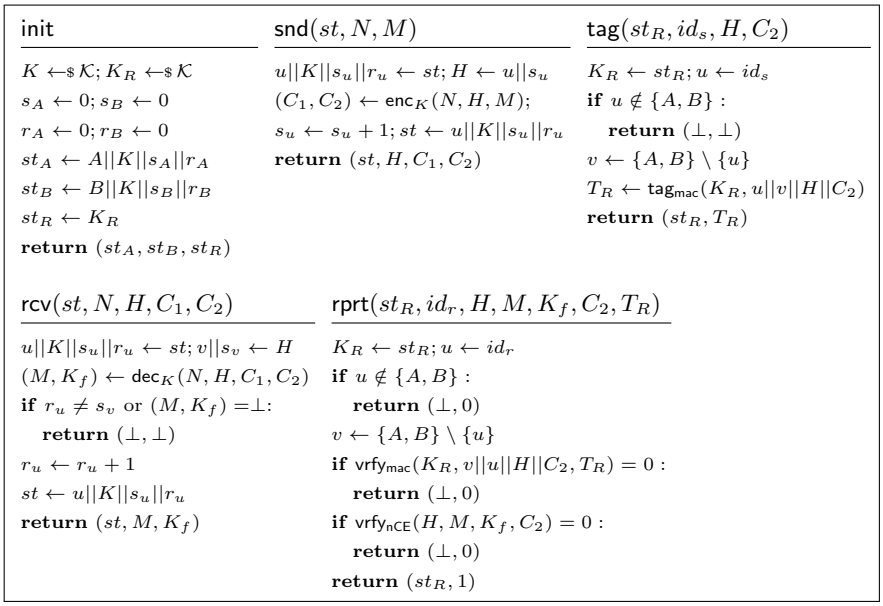


**Fig. 16.** Instantiation of a real MFC.

MF is MO-2nRoR secure. We state this formally in Theorem 3, skipping the proof that simply follows from the observation given above.

**Theorem 3 (MF confidentiality).** *Let* MF *be the* MFC *given in Figure 16, based on a secure committing AEAD scheme* nCE*. Then, for any adversary* $\mathcal{A}$ *playing the MO-2nRoR game there exists an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}_{\mathsf{MF}}^{mo\text{-}2nror}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{nCE}}^{mo\text{-}nror}(\mathcal{B})$$

*Integrity.* The integrity property of MF follows from the use of send and receive counters and from the integrity of nCE. Formally, the following theorem holds:

**Theorem 4.** *Let* MF *be the* MFC *given in Figure 16, based on a committing AEAD scheme* nCE*. Then, for any adversary* $\mathcal{A}$ *playing the MO-2nCTXT game there exists an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}_{\mathsf{MF}}^{mo\text{-}2nctxt}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{nCE}}^{mo\text{-}nctxt}(\mathcal{B})$$

*Proof.* We give here a sketch of the proof. We consider the MO-2nCTXT game of Figure 9. For $\mathcal{A}$ to win either $s_v \leq r_u$ or $(N, H, C_1, C_2) \neq C_{v,u}[r_u]$ in $\mathcal{O}^{rcv}$. Bboth cannot happen since $C_{v,u}[r_u]$ is not defined if $r_u \geq s_v$.

Let $cond^1$ be the event that the adversary wins with the first condition evaluating to true. It means that MF decrypted successfully but more messages were received by $u$ than sent by $v$. However, MF keeps track of the number of messages sent and received by putting the number of sent messages in the header.

Therefore, $cond^1$ happens only if the $\mathsf{rcv}$ procedure of Figure 16 is invoked with $H' = v||s'_v$, where $s'_v$ is greater or equal than the actual sent messages, and the decryption was successful. Therefore, we know that the adversary found $(N', H', C'_1, C'_2)$ that decrypts successfully, with $H'$ different from all headers previously sent.

Now, let $cond^2$ be the event that the adversary wins and the second condition is satisfied, which implies that $cond^1$ is false. As for the first condition, one can deduce by inspection that if $cond^2$ is true, then the adversary found a tuple $(N', H', C'_1, C'_2)$ that was not output by the $\mathcal{O}^{snd}$ oracle before. Hence, these two arguments mean that one can construct an adversary $\mathcal{B}$ playing the MO-nCTXT game with $\mathsf{nCE}$, which wins with a probability at least $\Pr[\mathcal{A} \text{ wins }] = \Pr[\{cond^1\} \cup \{cond^2\}]$.

*Sender binding.* The sender binding property follows directly from the sender binding property of $\mathsf{nCE}$. Formally, we state the following theorem:

**Theorem 5.** *Let* $\mathsf{MF}$ *be the* $\mathsf{MFC}$ *given in Figure 16, based on a committing AEAD scheme* $\mathsf{nCE}$*. Then, for any adversary* $\mathcal{A}$ *playing the S-2BIND game there exists an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{s\text{-}2bind}_{\mathsf{MF}}(\mathcal{A}) \leq \mathsf{Adv}^{s\text{-}bind}_{\mathsf{nCE}}(\mathcal{B})$$

*Proof.* In the s-2BIND game (Figure 10), the adversary wins if the values $(H, M, K_f, C_2, T_R)$ were successfully received by user $u$ (i.e. it is in the $R_{v,u}$ array) but the $\mathsf{rprt}$ procedure fails. In the $\mathsf{MF}$ construction, $\mathsf{rprt}$ can fail for two reasons:

1. The first reason is that the verification of the router tag $T_R$ failed. Now, in the s-2BIND game, if the adversary wins by querying $\mathcal{O}^{rprt}$ as client $u$, then the tuple $(H, C_2, T_R)$ is in the $R_{v,u}$ array. This means that this tuple was previously in the $S_{v,u}$ array and therefore, $T_R$ was output by $\mathsf{tag}(\mathsf{st_R}, \mathsf{v}, \mathsf{H}, \mathsf{C_2})$, with $v = \{A, B\} \setminus \{u\}$. Thus, $T_R = \mathsf{tag_{mac}}(K_R, v||u||H||C_2)$ and the router tag verification algorithm outputs the result of $\mathsf{vrfy_{mac}}(K_R, v||u||H||C_2, T_R)$, which can only be a success if we assume the MAC scheme to be correct. Hence, if the adversary wins, that contradicts the failure of the router tag verification procedure.
2. The second reason is that the franking tag verification $\mathsf{vrfy_{nCE}}(H, M, K_f, C_2)$ fails. Since the input values are in $R_{v,u}$ (assuming $\mathcal{A}$ queried $\mathcal{O}^{rprt}$ as user $u$), it means that $\mathsf{rcv}$ was successful, implying that $\mathsf{dec}_K(N, H, C_1, C_2)$ was successful. This means that $\mathcal{A}$ found a tuple $(N, H, C_1, C_2, K)$ such that the decryption is successful but the verification fails.

Hence, given an adversary $\mathcal{A}$ who plays s-2BIND, it is trivial to construct an adversary $\mathcal{B}$ that wins the s-BIND game with at least the same probability as $\mathcal{A}$. □

*Receiver binding.* The receiver binding property of our construction is a consequence of the receiver binding property of the committing AEAD scheme and of the security against forgery of the MAC scheme used. We state the following theorem:

**Theorem 6.** *Let* MF *be the* MFC *given in Figure 16, based on a committing AEAD scheme* nCE *and a MAC scheme* MAC*. Then, for any adversary $\mathcal{A}$ playing the S-2BIND game there exists an adversary $\mathcal{B}$ and an adversary $\mathcal{C}$ such that*

$$\mathsf{Adv}^{r\text{-}2bind}_{\mathsf{MF}}(\mathcal{A}) \leq \mathsf{Adv}^{r\text{-}bind}_{\mathsf{nCE}}(\mathcal{B}) + \mathsf{Adv}^{uf\text{-}cma}_{\mathsf{MAC}}(\mathcal{C})$$

*Proof.* We proceed with the game hopping technique. Let the game $G^0$ in Figure 17 be the r-2BIND game played by adversary $\mathcal{A}$ against MF, except that the rprt procedure has been made explicit in $\mathcal{O}^{rprt}$ and that the router tags $T_R$ output by the $\mathcal{O}^{snd}$ oracle are stored in an array $T[]$ along with the corresponding $(u, v, H, C_2)$ tuple.

Now, consider the modified game $G^1$ (modifications boxed in Figure 17), which aborts if the router tag verification is successful but $T_R$ was not obtained from a tag query with $(v, u, H, C_2)$ in the sending oracle $\mathcal{O}^{snd}$. Let this condition be $forge$. Then, we have $|\Pr[G^0 \Rightarrow 1] - \Pr[G^1 \Rightarrow 1]| \leq \Pr[forge]$. By inspection, we see that if $forge$ happens, it means that the adversary found a message $m' = v||u||H||C_2$ and a corresponding tag $T_R$, without having queried $\mathsf{tag}_{\mathsf{mac}}(K_R, m')$. Hence, given $\mathcal{A}$ one can construct a straightforward UF-CMA adversary $\mathcal{C}$ against MAC that achieves an advantage of $\Pr[forge]$.

Let $G^2$ be the game $G^1$ but $\mathcal{O}^{rprt}$ is modified such that it aborts if everything is successful, but the message $M$ is not in $M_{v,u}$ (modifications in grey in Figure 17). In other words, we modify $G^1$ such that the adversary never wins. If this condition is fulfilled in line 11 of $\mathcal{O}^{rprt}$, we know that $(v, u, H, C_2, T_R)$ was obtained from a query to $\mathcal{O}^{snd}$ with some message $M'$ because line 7 of $\mathcal{O}^{rprt}$ was not executed. Therefore, $M'$ is in $M_{v,u}$ and thus $M \neq M'$. If this condition is satisfied, it means that the adversary found a tuple $(H, M, M', K_f, K'_f, C_2)$ with $M \neq M'$ such that $\mathsf{vrf}_{\mathsf{nCE}}(H, M, K_f, C_2) = \mathsf{vrf}_{\mathsf{nCE}}(H, M', K'_f, C_2) = 1$. Hence, given the adversary $\mathcal{A}$, one can construct a straightforward r-BIND adversary $\mathcal{B}$ against nCE such that $|\Pr[G^1 \Rightarrow 1] - \Pr[G^2 \Rightarrow 1]| \leq \mathsf{Adv}^{\text{r-bind}}_{\mathsf{nCE}}(\mathcal{B})$. Finally, we have $\Pr[G^2 \Rightarrow 1] = 0$, which concludes the theorem. □

## 8 Conclusion

In this paper, we introduced a *message franking channel* (MFC) which apart from message confidentiality/integrity, resistance to message drops, out of order delivery and replay attacks guarantees sender and receiver binding: Namely, the sender cannot send an abusive message, which cannot be reported to a third party and the receiver cannot report a fake message. Even if all of the definitions were presented in the unidirectional and bidirectional case, we focused mainly on bidirectional security definitions. We presented two results on binding

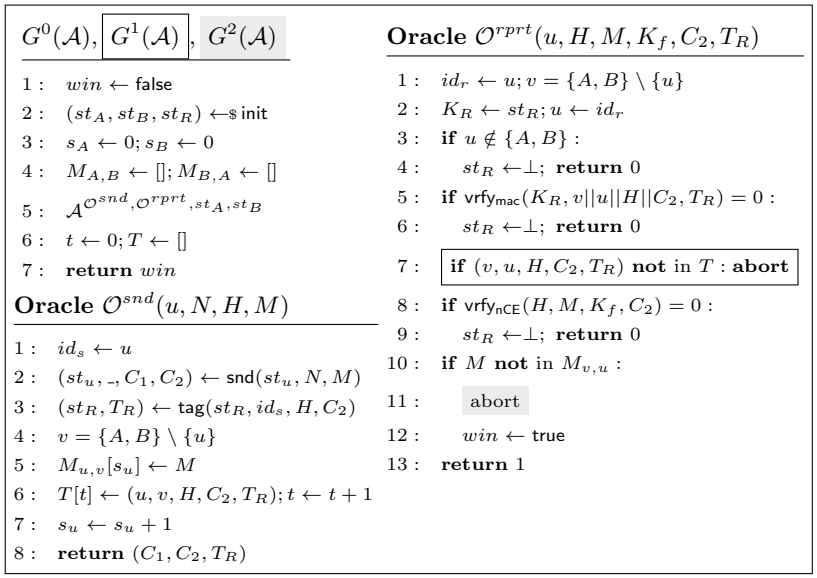


**Fig. 17.** Games for proof of Theorem 6.

properties in the canonic composition of two unidirectional MFC. In particular, we proved that two unidirectional receiver binding MFC are sufficient to create a bidirectional receiver binding MFC. In addition, we stressed that these results do not necessarily hold in general but only in the canonic composition. Finally, we gave an instantiation of a bidirectional MFC given a secure nonce based committing AEAD [GLR17] and a message authentication code.

## References

BDPS13. Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In *International Workshop on Fast Software Encryption*, pages 367–390. Springer, 2013.

BH17. Colin Boyd and Britta Hale. Secure channels and termination: The last word on tls. Cryptology ePrint Archive, Report 2017/784, 2017. https://eprint.iacr.org/2017/784.

BHMS15. Colin Boyd, Britta Hale, Stig Frode Mjolsnes, and Douglas Stebila. From stateless to stateful: Generic authentication and authenticated encryption constructions with application to tls. Cryptology ePrint Archive, Report 2015/1150, 2015. https://eprint.iacr.org/2015/1150.

BKN02. Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in ssh: provably fixing the ssh binary packet protocol. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 1–11. ACM, 2002.

DGRW18. Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA*, 2018.

Fac16. Facebook. Messenger secret conversations, 2016. https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf.

FMB+14. Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Joerg Schwenk, and Thorsten Holz. How secure is textsecure? Cryptology ePrint Archive, Report 2014/904, 2014. https://eprint.iacr.org/2014/904.

GLR17. Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. Cryptology ePrint Archive, Report 2017/664, 2017. https://eprint.iacr.org/2017/664.

GM17. Felix Günther and Sogol Mazaheri. A formal treatment of multi-key channels. Cryptology ePrint Archive, Report 2017/501, 2017. https://eprint.iacr.org/2017/501.

KPB03. Tadayoshi Kohno, Adriana Palacio, and John Black. Building secure cryptographic transforms, or how to encrypt and mac. Cryptology ePrint Archive, Report 2003/177, 2003. https://eprint.iacr.org/2003/177.

Mar17. Giorgia Azzurra Marson. *Real-World Aspects of Secure Channels: Fragmentation, Causality, and Forward Security*. PhD thesis, Technische Universität Darmstadt, 2017.

MP17. Giorgia Azzurra Marson and Bertram Poettering. Security notions for bidirectional channels. Cryptology ePrint Archive, Report 2017/161, 2017. https://eprint.iacr.org/2017/161.

Per18. Trevor Perrin. The noise protocol framework. http://noiseprotocol.org/noise.html, 2018. Accessed: 2018-09-03.

PR18. Bertram Poettering and Paul Rösler. Ratcheted key exchange, revisited. Cryptology ePrint Archive, Report 2018/296, 2018. https://eprint.iacr.org/2018/296.

Vau02. Serge Vaudenay. Security flaws induced by cbc padding — applications to ssl, ipsec, wtls... In Lars R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, pages 534–545, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Wha17. Whatsapp. Whatsapp encryption overview, 2017. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

# A Supplementary Material

## A.1 Proofs

**Theorem 1 (Receiver binding).** *Let* $\mathsf{Ch}$ *be a receiver binding unidirectional* $\mathsf{MFC}$ *(r-1BIND). Then, its canonic composition* $\mathsf{Ch}'$ *is receiver binding (r-2BIND). More precisely, for any adversary* $\mathcal{A}$ *playing r-2BIND against* $\mathsf{Ch}'$*, there exists a r-1BIND adversary* $\mathcal{B}$ *against* $\mathsf{Ch}$ *s.t.*

$$\mathsf{Adv}^{r\text{-}2bind}_{\mathsf{Ch}'}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}^{r\text{-}1bind}_{\mathsf{Ch}}(\mathcal{B})$$

*Proof.* In order to prove this statement, we follow the same method used to prove Theorem 2 in [MP17]. Let $win^{\rightarrow}$ be the event that $\mathcal{A}$ wins the r-2BIND game by querying $\mathcal{O}^{rprt}(u, *)$ with $u = B$. In other words, $win^{\rightarrow}$ happens when line 4 of $\mathcal{O}^{rprt}$ in Figure 11 is executed and $u = B$. Similarly, $win^{\leftarrow}$ is the event that line 4 of $\mathcal{O}^{rprt}$ in Figure 11 is executed and $u = A$.
Now, we consider a sequence of games. Let $G^0$ be the r-2BIND game played by adversary $\mathcal{A}$ against the channel $\mathsf{Ch}'$. Let $G^1$ be the same game as $G^0$ but the game returns $\mathsf{false}$ when $win^{\rightarrow}$ happens. Let $G^2$ be the same as $G^1$ but the game returns $\mathsf{false}$ when $win^{\leftarrow}$ happens. We deduce the following results:

$$\begin{aligned} |\Pr[G^0 \Rightarrow \mathsf{true}] - \Pr[G^1 \Rightarrow \mathsf{true}]| &\leq \Pr[win^{\rightarrow}] \\ |\Pr[G^1 \Rightarrow \mathsf{true}] - \Pr[G^2 \Rightarrow \mathsf{true}]| &\leq \Pr[win^{\leftarrow}] \\ \Pr[G^2 \Rightarrow \mathsf{true}] &= 0 \end{aligned}$$

Also,

$$\begin{aligned} \mathsf{Adv}^{\text{r-2bind}}_{\mathsf{Ch}'}(\mathcal{A}) &= \Pr[G^0 \Rightarrow \mathsf{true}] \\ &\leq |\Pr[G^0 \Rightarrow \mathsf{true}] - \Pr[G^1 \Rightarrow \mathsf{true}]| \\ &+ |\Pr[G^2 \Rightarrow \mathsf{true}] - \Pr[G^1 \Rightarrow \mathsf{true}]| \\ &+ \Pr[G^2 \Rightarrow \mathsf{true}] \end{aligned}$$

Hence,

$$\mathsf{Adv}^{\text{r-2bind}}_{\mathsf{Ch}'}(\mathcal{A}) \leq \Pr[win^{\rightarrow}] + \Pr[win^{\leftarrow}] \tag{6}$$

Now, given the adversary $\mathcal{A}$, we construct a r-1BIND adversary $\mathcal{B}^{\rightarrow}$ (respectively $\mathcal{B}^{\leftarrow}$) against channel $\mathsf{Ch}$ that wins with probability $\Pr[win^{\rightarrow}]$ (respectively $\Pr[win^{\leftarrow}]$). Consider the adversary $\mathcal{B}^{\rightarrow}$ playing the r-1BIND game with channel $\mathsf{Ch}$ in Figure 18. It has access to the $\mathcal{O}^{snd}$ and $\mathcal{O}^{rprt}$ oracles and the states $st^{\rightarrow}_{snd}, st^{\rightarrow}_{rcv}$ (i.e. $st_A$ and $st_B$ in the r-1BIND game in Figure 13). The adversary $\mathcal{B}^{\rightarrow}$ creates its own instance of $\mathsf{Ch}$ by sampling the necessary states randomly (line 1-4). Then, $\mathcal{B}^{\rightarrow}$ runs $\mathcal{A}$. The latter calls $\mathcal{O}^{snd}_{\mathcal{A}}$ and $\mathcal{O}^{rprt}_{\mathcal{A}}$ oracles. In both of these oracles, $\mathcal{A}$'s queries in the $\rightarrow$ direction (Alice sends/Bob reports) are relayed to $B^{\rightarrow}$'s own oracles (line 3 in $\mathcal{O}^{snd}_{\mathcal{A}}$ and $\mathcal{O}^{rprt}_{\mathcal{A}}$). $\mathcal{A}$'s queries in the other direction are answered by $B^{\rightarrow}$ running its own channel (lines 5-8 in $\mathcal{O}^{snd}_{\mathcal{A}}$ and

line 5 in $\mathcal{O}_{\mathcal{A}}^{rprt}$). Overall, $B^{\rightarrow}$ creates a canonic composition of its own channel for the $\leftarrow$ direction and of the channel created in the r-1BIND game for the $\rightarrow$ direction. $\mathcal{A}$ runs exactly as in the $G^0$ game, except when it wins. Now, consider $\mathcal{A}$ winning in the $G^0$ game with a $\rightarrow$ report query (i.e. $win^{\rightarrow}$ event). Then, in the r-1BIND game, the winning query is forwarded to $\mathcal{O}^{rprt}$ by $\mathcal{B}^{\rightarrow}$ and line 5 of $\mathcal{O}^{rprt}$ in the r-1BIND game of Figure 13 is executed, making $\mathcal{B}^{\rightarrow}$ win. $\mathcal{B}^{\rightarrow}$ wins because the array $M$ in the r-1BIND game played by $\mathcal{B}^{\rightarrow}$ is modified exactly as the array $M_{A,B}$ in the game $G^0$ played by $\mathcal{A}$. Thus, if $win^{\rightarrow}$ happens, then the condition allowing $\mathcal{B}^{\rightarrow}$ to win in r-1BIND (line 4 of $\mathcal{O}^{rprt}$ in Figure 13) evaluates to true. Therefore, we obtain

$$\mathsf{Adv}_{\mathsf{Ch}}^{\text{r-1bind}}(\mathcal{B}^{\rightarrow}) \geq \Pr[win^{\rightarrow}]$$

Similarly, one can construct an adversary $\mathcal{B}^{\leftarrow}$ by inverting the directions accordingly in Figure 18, obtaining

$$\mathsf{Adv}_{\mathsf{Ch}}^{\text{r-1bind}}(\mathcal{B}^{\leftarrow}) \geq \Pr[win^{\leftarrow}]$$

Finally, we construct an adversary $\mathcal{B}$ that chooses uniformly at random to run $\mathcal{B}^{\rightarrow}$ or $\mathcal{B}^{\leftarrow}$, achieving a total advantage of

$$\mathsf{Adv}_{\mathsf{Ch}}^{\text{r-1bind}}(\mathcal{B}) \geq \frac{1}{2}\Pr[win^{\rightarrow}] + \frac{1}{2}\Pr[win^{\leftarrow}]$$

Hence, by Eq. (6), we conclude that

$$\mathsf{Adv}_{\mathsf{Ch}'}^{\text{r-2bind}}(\mathcal{A}) \leq \Pr[win^{\rightarrow}] + \Pr[win^{\leftarrow}] \leq 2 \cdot \mathsf{Adv}_{\mathsf{Ch}}^{\text{r-1bind}}(\mathcal{B})$$

□

**Theorem 2 (Sender binding).** *Let* $\mathsf{Ch}$ *be a sender binding unidirectional* MFC *(r-1BIND). Then, its canonic composition* $\mathsf{Ch}'$ *is sender binding (s-2BIND). More precisely, for any adversary* $\mathcal{A}$ *playing s-2BIND against* $\mathsf{Ch}'$*, there exists a s-1BIND adversary* $\mathcal{B}$ *against* $\mathsf{Ch}$ *s.t.*

$$\mathsf{Adv}_{\mathsf{Ch}'}^{s\text{-}2bind}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{Ch}}^{s\text{-}1bind}(\mathcal{B})$$

*Proof.* The proof is similar to the one of Theorem 1.

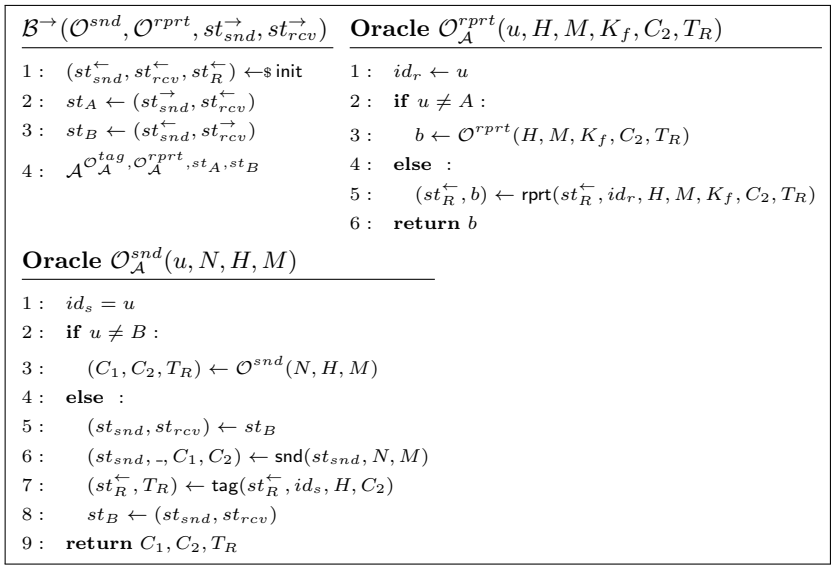


**Fig. 18.** Reduction for the proof of Theorem 1.