People Who Live in Glass Houses Should not Throw Stones: Targeted Opening Message Franking Schemes

Long Chen and Qiang Tang

New Jersey Institute of Technology
{longchen,qiang}@njit.edu

Abstract. Message franking enables a receiver to report a potential abuse in a secure messaging system which employs an end to end encryption. Such mechanism is crucial for accountability and is already widely adopted in real world product such as Facebook messenger. Grubs et al [5] initiated a systematic study of such a new primitive, and Dodis et al [2] gave a more efficient construction.

We observe that in all existing message franking schemes, the receiver has to reveal the whole communication for a session in order to report one abuse. This is highly undesirable in many settings where revealing other non-abusive part of the communication leaks too much information; what is worse, a foxy adversary may intentionally mixing private information of the receiver with the abusive message so that the receiver will be reluctant to report. This essentially renders the abuse reporting mechanism ineffective.

To tackle this problem, we propose a new primitive called targeted opening compactly committing AEAD (TOCE for short). In a TOCE, the receiver can select arbitrary subset of bits from the plaintext to reveal during opening, while keep all the rest still secure as in an authenticated encryption. We gave a careful formulation and give a generic construction. The generic construction allowing a bit level opening may require a substantial number of passes of symmetric key ciphers when encrypting a large message such as a picture. We thus further set forth and give a more efficient non-black-box construction allowing a block-level (e.g., 256 bit) opening. We also propose a privacy-efficiency trade off if we can relax the security of non-opened messages to be one way secure (they are still semantically secure if no opening).

1 Introduction

End-to-end encryption enables users to securely communicate with each other, without worrying the message to be seen to any third party including the platform that hosts the secure messaging service. Multiple large scale secure messaging systems such as WhatsApp, Signal, and Facebook Messenger have already been deployed to serve more than a billion users across the globe. On the other hand, confidentiality also brings new challenges of other security goals.

Most notably, when one user spreads misinformation such as harassing messages, phishing links and/or any other improper contents, the recipients should be allowed to report the malicious behavior to the service provider, so that the sender could be penalized (e.g., blocked). On the same time, no dishonest reporter should be able to fabricate any fake message to frame an innocent sender. To address this pressing challenge, Facebook Messenger [3, 4] recently introduced the concept of *message franking* for such verifiable abuse reporting in encrypted systems. Also, a new cryptographic primitive called compactly committing authenticated encryption with associated data (ccAEAD) was proposed in recent cryptography literature [5, 2] to provide formal investigations.

A ccAEAD first is a standard authenticated encryption, but with the extra property that enables a receiver to open the plaintext if he chooses to. Two natural properties arise: a malicious sender could not deny the opening of an honest receiver, and a malicious receiver cannot arbitrarily open the message to frame an honest sender. Besides these two, in practice, it is preferable to add a short tag to enable these two properties whose size is independent of the message size (this property refers to compactness).

Intuitively, in a message franking system, some short tag served as a proof will be attached to the ciphertext so that a later "opening" of the ciphertext can be verified with this tag. Imagine a user Bob receives a ciphertext which is "stamped" by the server from a user Alice, (server signs or even stores the tag ensuring the ciphertext is indeed sent from Alice to Bob). If Bob decodes the ciphertext, validates the tag, and obtains an improper message, he will then reveal either the message or the secret key, so that the service provider can check the reported abuse. Existing methods including the one deployed in Facebook Messenger and the constructions in [5, 2] all follow this pattern.

The undesirability of "all or nothing" abuse reporting. However, revealing the whole piece of the plaintext transmitted during a session when reporting an abusive message in many cases is undesirable to the recipient, as some parts of the plaintext (or the existence of the conversation itself) could contain private information. Consider the following exemplary scenarios: a doctor or a pharmacist is communicating with one patient about the situation of some diseases the patient is suffering from or the medicine the patient is taking; two members of a cult group are discussing or debating on some issue regarding their special interests; a merchant on Ebay is explaining to a customer about the product (which could be for some special hobby and the customer would prefer to keep private) he is selling. In all those scenarios, the conversations contain some private information of the recipient, if improper messages such as harassing messages are generated during those conversations, the recipients may feel reluctant to reveal his name or the other personal information, thus not reporting the abuse.

What's worse, having this in mind, a malicious sender would intentionally insert some kind of personal information of the receiver when sending improper messages. Given current all-or-nothing type of opening in ccAEAD schemes, the attacker can simply concatenate some piece of receiver personal information with the abusive message and send over together via the secure messaging system. Doing this essentially renders abuse reporting in message franking as an ineffective deterrence towards resolving the misinformation problem in secure messaging. Such potential threat of existing message franking schemes calls for a formal study that whether the abuse reporting can be done in a way that the receiver can choose flexibly which part to reveal. In this paper, we are seeking to answer the following question:

Can we design a message franking scheme that enables the receiver to selectively report the abusing message, without revealing any information about other parts of the plaintext?

Our contributions. To defend against the above attack to message franking, we strengthen the ccAEAD notion to enable a targeted opening. In more details:

Targeted opening ccAEAD: modeling. It turns out that adding a targeted opening property influences all security properties. We formally define the targeted opening property for ccAEAD, such that a recipient could reveal any part of the message of his choice, while other parts remain as secure as in an authenticated encryption. This means for both confidentiality and ciphertext integrity, we need to allow the attacker an extra query about opening of a targeted part of the plaintext. Furthermore, the two binding properties also need to be revised accordingly. More specifically, it means that the sender must not be able to provide a valid ciphertext which can be decrypted successfully but can not be targeted opened correctly for some positions. Also the receiver can not maliciously open any targeted part of the ciphertext to a abusive message.

Targeted opening ccAEAD: a generic construction. Unfortunately, none of existing constructions of ccAEAD satisfies targeted opening, since their opening algorithm will reveal the whole information about the message. To follow the generic commit-then-encrypt methodology in [5] to construct a ccAEAD and also to support targeted opening, we propose a modular approach. We first introduce a cryptographic notion of targeted opening commitment scheme. The most attractive property for this primitive is that it allows one to open any part of the message while the rest remain hidden. Moreover, the verifier can confirm that the partial message is indeed extracted from the exact bit positions of the original message. Such a special instance of a more advanced notions like a functional commitment [7] or vector commitment [1] allows us to give a very simple construction that only utilizes collision resistant hash. We then give a generic construction from a targeted opening commitment and an AEAD and gave a detailed security analysis.

Targeted opening ccAEAD: more efficient constructions. Our generic construction leverages bitwise operations thus incurs large overhead for AEAD, this makes the scheme only applicable when encrypting short messages such as texts. To also consider the applicability in the setting of encrypting large messages such as pictures, and even videos, we set forth to consider more efficient constructions. We first consider a block-wise targeted opening. However, straightforward instantiation of our generic construction, yields an encryption algorithm that needs four passes of block cipher operations for a string of message. We observe that, if we reuse some intermediate results of the encryption part to the commitment part, we can construct a nonce based block wise targeted opening ccAEAD with three passes in the random oracle model. Furthermore, if we weaker the confidentiality definition and allow the unopened message part to be one-way secure instead of semantic secure, we can get a construction of toccAEAD with only two passes. Note that even for our latter weaker version construction, its security is still strictly stronger than previous ccAEAD constructions in [5, 2].

Scheme	Without Opening	Targeted Opening	Pass
CtE1/CtE2[5]	Semantic	No	2
CEP [5]	Semantic	No	2
HFC [2]	Semantic	No	1
CEP2 [6]	Semantic	No	2
CEP-AOP1/2 [6]	Semantic	Semantic	4
TOCE	Semantic	Semantic	4
bTOCE	Semantic	Semantic	3
wbTOCE	Semantic	One-Way	2

Table 1. Comparison for existing schemes. Without Opening and Targeted Opening denote the security level of the confidentiality without opening and targeted opening respectively. Pass denotes the complexity of encryption and decryption.

Related work and Comparisons. Grubs et al. [5] initiated a systematic formal study of message franking, and formalized a cryptographic scheme called compactly committing authenticated encryption with associated data (ccAEAD). The authors did a thorough examination of existing concrete AEAD schemes and generic constructions in use. Finally, they also provide a nonce based construction for ccAEAD with two passes, which is as efficient as our weaker confidentiality toccAEAD. Dodis et al. [2] demonstrated a concrete attack that the Facebook message franking scheme is actually insecure. They also gave an efficient construction of ccAEAD that only involves a one-pass of symmetric key ciphers. We stress that all constructions do not support a selective opening of only a part of the plaintext.

Concurrent work. Recently Leontiadis and Vaudeny [6] considered a similar security definition for the message franking scheme, named after opening privacy. This property allows only the abusive blocks are opened while the rest non-abusive blocks of the message remain private. While we further considered other properties under the targeted open capability. They also gave two constructions to show feasibility, thus their constructions at least need four passes block cipher operations, while our efficient construction reduces it to three and even fewer when taking the privacy-efficiency trade off.

2 Preliminary and Background

In this section, we explain several definitions of cryptographic primitives necessary as our preliminary.

2.1 Classical cryptographic primitives

Let $\{0,1\}^n$ denote the bit string with length n. Specifically, $\{0,1\}^*$ denote the bit string with arbitrary length.

Nonce-based pseudorandom generator. a nonce based pseudo random generator (PRG) $\mathcal{G}$ is a deterministic algorithm that takes as input a key K, a nonce N, and an output length l. It outputs a string of length l bits. The PRG advantage of an adverary $\mathcal{A}$ against $\mathcal{G}$ is defined by

$$\mathbf{Adv}_G^{prg}(\mathcal{A}) = \left| \Pr_{K \leftarrow \{0,1\}^k} \left[\mathcal{A}^{G(K,\cdot,\cdot)} = 1 \right] - \Pr \left[\mathcal{A}^{R(\cdot,\cdot)} = 1 \right] \right|$$

where R works as follows. On query N, l it checks if a previous query N,l' was submitted. If $l' < l$ it picks a new random string of length $l - l'$, appends it to the previous returned string for N, records it in a table indexed by N, and returns the concatenated random string. If no previous query exists, then it picks a random string of length l, records and then returns it. We call a PRG adversary $\mathcal{A}$ nonce-respecting if all its queries use a unique nonce N. One can build a nonce-based pseudorandom generator from a block cipher in CTR mode.

Commitment scheme. A commitment scheme with verification $\mathsf{CS} = (\mathsf{Com}, \mathsf{VerC})$ consist of two algorithms. Associated to any commitment scheme is an opening space $\mathcal{K}_f \subseteq \{0,1\}^*$, and a commitment space $\mathcal{C} \subseteq \{0,1\}^*$. The algorithm Com is randomized and takes as input a $M \in \{0,1\}^*$ and outputs a pair $(K, C) \in \mathcal{K}_f \times \mathcal{C}$ or an error symbol $\perp$. We assume that Com return $\perp$ with probability one if $M \notin \mathcal{M}$. The algorithm VerC is deterministic. It takes input a tuple $(K, C, M) \in \{0,1\}^*$ and outputs a bit. We assume that VerC returns 0 if its input $(K, C, M) \notin \mathcal{K}_f \times \mathcal{C} \times \mathcal{M}$.

- *Correctness.* A commitment is correct if for all $M \in \mathcal{M}$, $\Pr[\mathsf{VerC}(\mathsf{Com}(M), M) = 1]$ where the probability is over the coins used by Com
- *Binding.* A commitment is binding if no (computational or unbounded) adversary can output a tuple (K_c, M, K_c', M', C) where both (K_c, M, C) and (K_c', M', C) can pass the verification.
- *Hiding.* A commitment is hiding if a commitment is indistinguishable from a random bit string while the opening remaining secret.

2.2 ccAEAD and message franking

In this section, we will revisit the definition of ccAEAD in [5], and how the kidnapping attack works.

Definition of ccAEAD A committing AEAD scheme CE consists of four algorithms (KeyGen,Enc,Dec,Ver). Let us represent the key space as $\mathcal{K} \subseteq \{0,1\}^*$, the header space as $\mathcal{H} \subseteq \{0,1\}^*$, the message space as $\mathcal{M} \subseteq \{0,1\}^*$, the ciphertext space as $\mathcal{C} \subseteq \{0,1\}^*$, the opening space as $\mathcal{K}_f \subseteq \{0,1\}^*$, and the franking tag space $\mathcal{T} \subseteq \{0,1\}^*$.

- KeyGen: The randomized key generation algorithm KeyGen outputs a secret key $K \in \mathcal{K}$.
- Enc: The randomized algorithm Enc takes a triple $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ as input and outputs a pair $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$. Here C_1 is the ciphertext and C_2 is the franking tag.
- Dec: The deterministic algorithm Dec takes a tuple $(K, H, C_1, C_2) \in \mathcal{K} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T}$ as input and outputs a message, opening value pair $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$ or a distinguished error symbol $\perp$.
- Verify: The deterministic algorithm Ver takes a tuple $(H, M, K_f, C_2) \in \mathcal{H} \times \mathcal{M} \times \mathcal{K}_f \times \mathcal{T}$ as input and output a bit b. Specifically, we assume that Ver outputs 0 for $(H, M, K_f, C_2) \notin \mathcal{H} \times \mathcal{M} \times \mathcal{K}_f \times \mathcal{T}$.

Definition of nessage franking. We notice that in the verification algorithm, the input includes the entire message.

Alice	**Server**	**Bob**
$(C_1, C_2) \leftarrow \mathsf{Enc}(K, H, M)$		
$\xrightarrow{\quad (C_1, C_2) \quad}$		
	$md \leftarrow Alice \Vert Bob \Vert timestamp$	
	$s \leftarrow C_2 \Vert md$	
	$a \leftarrow \mathsf{HMAC}(K_{server}, s)$	
	$\xrightarrow{\quad C_1, C_2, a \quad}$	
		$(M, K_f) \leftarrow \mathsf{Dec}(K, H, C_1, C_2)$
	..................Report Abuse..................	
		$\xleftarrow{\quad (M, K_f, md, a) \quad}$
	$b \leftarrow \mathsf{Ver}(H, M, K_f, C_2)$	
	$a' \leftarrow \mathsf{HMAC}(K_{server}, C_2 \Vert md)$	
	Report $b \wedge (a = a')$	

Since in the report phase, Bob will provide the whole message to the server, otherwise the server can not proceed the correct verification.

Security definition of ccAEAD. For a secure CE scheme, we require that it can satisfies the following properties: confidentially, ciphertext integrity, sender binding and receiver binding.

Specifically, the confidentially can be defined as the difference between the probability of returning 1 in the game $\mathrm{REAL}^{\mathcal{A}}_{CE}$ and $\mathrm{RAND}^{\mathcal{A}}_{CE}$ in Figure 2.2 is negligible, while the integrity is defined as the probability of returning 1 in the game $\mathrm{RAND}^{\mathcal{A}}_{CE}$ is negligible.

$\mathrm{REAL}^{\mathcal{A}}_{CE}$ $K \leftarrow \mathsf{KeyGen}$ $b \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalReal}}$ Return b	$\mathrm{RAND}^{\mathcal{A}}_{CE}$ $K \leftarrow \mathsf{KeyGen}$ $b \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalRand}}$ Return b	$\mathrm{CTXT}^{\mathcal{A}}_{CE}$ $K \leftarrow \mathsf{KeyGen}$; $win \leftarrow 0$ $\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{ChalDec}}$ Return win
Oracle $\mathbf{Enc}(H, M)$ $(C_1, C_2) \leftarrow \mathsf{Enc}_K(H, M)$ $\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(H, C_1, C_2)\}$ Return (C_1, C_2)	Oracle $\mathbf{Dec}(H, C_1, C_2)$ If $(H, C_1, C_2) \notin \mathcal{Y}_1$ then Return $\perp$ $(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$	Oracle $\mathbf{Dec}^*(H, C_1, C_2)$ Return $\mathsf{Dec}_K(H, C_1, C_2)$
Oracle $\mathbf{ChalReal}(H, M)$ $(C_1, C_2) \leftarrow \mathsf{Enc}_K(H, M)$ Return (C_1, C_2)	Oracle $\mathbf{ChalRand}(H, M)$ $(C_1, C_2) \leftarrow \mathcal{C} \times \mathcal{T}$ Return (C_1, C_2)	Oracle $\mathbf{ChalDec}^*(H, C_1, C_2)$ If $(H, C_1, C_2) \in \mathcal{Y}$ Return $\perp$ $(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$ If $M \neq \perp$ then $win \leftarrow 1$ Return (M, K_f)

Fig. 1. Security Games for confidentially and integrity for ccAEAD

The CE scheme not only require confidentially and ciphertext binding, but also the sender binding security and the receiver binding security. Sender binding ensures the sender of a message is bound to the message it actually sent. This property can prevent the sender to generate a bogus commitment that does not give the receiver the ability to report the message. It is formal defined as that the probability of the game $\text{s-BIND}^{\mathcal{A}}_{CE}$ on the left column of returning ture of Figure 2 is negligible. The receiver binding is adopted from the traditional binding notions for the commitment. It formalizes the intuition that a malicious receiver should not be able to accuse a non-abusive sender of having said something malicious, which can be defined as the probability of the game $\text{r-BIND}^{\mathcal{A}}_{CE}$ return 1 on the right column of Figure 2 is negligible.

$$
\begin{array}{ll}
\underline{s-BIND^{\mathcal{A}}_{CE}} & \underline{r-BIND^{\mathcal{A}}_{CE}} \\
(K,H,C_1,C_2) \leftarrow\$\, \mathcal{A} & ((H,M,K_f),(H',M',K'_f),C_2) \leftarrow\$\, \mathcal{A} \\
(M',K_f) \leftarrow \mathsf{Dec}(K,H,C_1,C_2) & b \leftarrow \mathsf{Ver}(H,M,K_f,C_2) \\
\text{If } M' = \bot \text{ then Return } \mathsf{false} & b' \leftarrow \mathsf{Ver}(H',M',K'_f,C_2) \\
b \leftarrow \mathsf{Ver}(H,M',K_f,C_2) & \text{If } (H,M) = (H',M') \text{ then} \\
\text{If } b = 0 \text{ then} & \quad \text{Return } \mathsf{false} \\
\quad \text{Return } \mathsf{true} & \text{Return}(b = b' = 1) \\
\text{Return } \mathsf{false} &
\end{array}
$$

Fig. 2. The security games for the binding properties for ccAEAD.

3 Targeted Opening Compactly Committing AEAD

As briefly discussed in the introduction, a foxy attacker in a message franking scheme (or the underlying ccAEAD scheme) could leverage the fact that recipients may be reluctant to report an abusive message if his private information is contained in the session plaintext, when the opening requires the recipient to reveal the whole piece of the plaintext. The attacker could intentionally embed private information about the recipient to make abuse reporting functionality essentially nullified. For this reason, we initiate a systematic study about targeted opening property in a ccAEAD scheme. The targeted opening property allows a recipient to pick exclusively the abusive message from the plaintext, only revealing the abusive message to the server as evidence, while keep all other plaintext still confidential.

A targeted opening compactly committing AEAD scheme (tocc-AEAD for short, sometimes we also use TOCE) consists of five algorithms, i.e., $\mathsf{TOCE} = (\mathsf{KG},\mathsf{Enc},\mathsf{Dec},\mathsf{TOpen},\mathsf{TVer})$. Let us represent the key space as $\mathcal{K} \subseteq \{0,1\}^*$, the header space as $\mathcal{H} \subseteq \{0,1\}^*$, the message space as $\mathcal{M} \subseteq \{0,1\}^*$, the ciphertext space as $\mathcal{C} \subseteq \{0,1\}^*$, the opening space as $\mathcal{K}_f \subseteq \{0,1\}^*$, the targeted opening space as $\mathcal{S} \subseteq \{0,1\}^*$and the franking tag space $\mathcal{T} \subseteq \{0,1\}^*$.

Before we describe the syntax, we first define the position function φ as follows. If M is a bit string with length n, φ is a function that take as input the message M, picks the bits of M with the indices $\{i_1, i_2, \ldots, i_j\}$, (for each index chosen from $\{1, \ldots, n\}$) depending on φ's definition. Without loss of generality, we denote the identity function I as choosing all the positions from $\{1, \ldots, n\}$, i.e., $I(M) = M$. We define the space of all position functions as Φ.

- **Key generation**: The randomized algorithm KeyGen outputs a secret key $K \in \mathcal{K}$.
- **Encryption**: The randomized algorithm Enc takes a triple $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ as input and outputs a pair $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$. Here C_1 is the ciphertext that carries the payload and C_2 is the franking tag, and H is the associated data.

- **Decryption**: The deterministic algorithm Dec takes a tuple $(K, H, C_1, C_2) \in \mathcal{K} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T}$ as input and outputs a message and opening value pair $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$ or the error symbol $\perp$.
- **Targeted open**: The deterministic algorithm TOpen takes as input a tuple $(H, M, K_f, C_2, \varphi) \in \mathcal{H} \times \mathcal{M} \times \mathcal{K}_f \times \mathcal{T} \times \Phi$, and outputs the targeted value represented as $\varphi(M)$ and the corresponding targeted opening S.
- **Verification**: The deterministic algorithm TVer takes as input a tuple values of $(H, \varphi(M), S, \varphi, C_2)$ and output a bit b. Specifically, we assume that Ver outputs 0 if the targeted opening is not valid.

Compactness. Similarly to previous work [5, 2], we also require the tocc-AEAD scheme to be compact, which means the length of the tag part of the ciphertext is independent with the message length. This is important for the server to authenticate the tag and even store the short tag.

With the revised syntax, the message franking protocol would now be modified correspondingly as follows.

Alice	**Server**	**Bob**
$(C_1, C_2) \leftarrow \mathsf{Enc}(K, H, M)$		
$\xrightarrow{(C_1, C_2)}$		
	$md \leftarrow Alice \Vert Bob \Vert time$	
	$s \leftarrow C_2 \Vert md$	
	$a \leftarrow \mathsf{HMAC}(K_{server}, s)$	
	$\xrightarrow{C_1, C_2, a}$	
		$(M, K_f) \leftarrow \mathsf{Dec}(K, H, C_1, C_2)$
	..................Report Abuse..................	
		$S \leftarrow \mathsf{TOpen}(H, M, K_f, C_2, \varphi)$
		$\xleftarrow{(H, \varphi(M), S, \varphi, md, a)}$
	$b \leftarrow \mathsf{TVer}(H, \varphi(M), S, \varphi, C_2)$	
	$a' \leftarrow \mathsf{HMAC}(K_{server}, C_2 \Vert md)$	
	Report $b \wedge (a = a')$	

Fig. 3. Message franking supporting targeted opening

Correctness. We say a TOCE scheme has decryption correctness if for all $(K, H, M) \in \mathcal{K} \times \mathcal{H} \times \mathcal{M}$ it holds that $\Pr[\mathsf{Dec}(K, H, C_1, C_2) = M] = 1$ where the probability is taken over the coins in the key generation Kg and the encryption Enc. We say a target commitment ccAEAD scheme has targeted open commitment correctness if for all $(H, M, S, \varphi) \in \mathcal{H} \times \mathcal{M} \times \mathcal{S} \times \Phi$ it holds that

$$\Pr[\mathsf{TVer}(H, \varphi(M), S, \varphi, C_2) = 1] = 1$$

where the probability is taken over the random variables in the following procedure:

1. $K \leftarrow_{\$} \mathsf{KeyGen}$;
2. $(C_1, C_2) \leftarrow_{\$} \mathsf{Enc}_K(H, M)$;
3. $(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$;
4. $S \leftarrow \mathsf{TOpen}(H, M, K_f, C_2, \varphi)$.

3.1 Security definitions

In this section, we will give a detailed characterization of the security notions in the new setting allowing targeted opening (which could be multiple times for the same plaintext). We note that a TOCE would still satisfy the confidentiality, ciphertext integrity, sender binding, and receiver binding, but all of them are influenced by the targeted opening functionality thus we need to adapt carefully.

Message confidentiality. In a TOCE scheme, we require that the messages bits that have not been opened, remain semantically secure. This requires no single bit of information will be leaked except the explicitly revealed part. In the TO-IND game defined in Fig 4 below, we further allow the adversary to have access to **TOpen** oracle. To avoid trivial impossibility, we require that for the two challenge messages, the opened part to be identical. It is straightforward that if the recipient does not open any bits, the standard IND-CPA security will apply.

Ciphertext integrity. We also require that any adversary without the secret key can not generate new valid ciphertexts from existing ciphertexts. Thus we define the TO-CTXT game in Fig 4.

Sender binding ensures the sender of a message is bound to the message it actually sent. So we define the s-BIND game in Fig 4. Note that a CE scheme can generically meet sender binding by running the verification algorithm during the decryption and return $\perp$ if the verification returns 0. [2]

Receiver binding is shown in the r-BIND game in Fig 4.

3.2 A generic construction of toccAEAD

Targeted opening commitment scheme. In order to construct the targeted opening compactly committing AEAD, we first introduce a primitive named *Targeted opening commitment scheme* (TOC). The TOC allows to commit an

$\underline{TO-IND^{\mathcal{A}}_{CE}}$
$K \leftarrow_{\$} \mathsf{KG}$
$state \leftarrow \mathcal{A}^{\mathbf{Enc,Dec,TOpen}}$
$\{H^*, (M_0, M_1)\} \leftarrow_{\$} \mathcal{A}(st)$
$(C_1^*, C_2^*) \leftarrow \mathsf{Enc}_K(H^*, M_b)$
  for $b \leftarrow_{\$} \{0,1\}$
$b \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc,Dec,TOpen}}(C_1^*, C_2^*, st)$
Return b

$\underline{TO-r-BIND^{\mathcal{A}}_{CE}}$
$((m, S), (m', S'), C_2, \varphi) \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc,Dec,TOpen}}$
$b \leftarrow \mathsf{TVer}(m, S, C_2, \varphi)$
$b' \leftarrow \mathsf{TVer}(m, S', C_2, \varphi)$
If $m = m'$ then
  Return false
Return$(b = b' = 1)$

$\underline{TO-CTXT^{\mathcal{A}}_{CE}}$
$K \leftarrow_{\$} Kg$
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc,Dec^*,TOpen,ChalDec}}$
Return win

$\underline{TO-s-BIND^{\mathcal{A}}_{CE}}$
$(K, H, C_1, C_2, \varphi) \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc,Dec,TOpen}}$
$(M', K_f) \leftarrow \mathsf{Dec}(K, H, C_1, C_2)$
If $M' = \perp$ then Return false
$S \leftarrow \mathsf{TOpen}(H, M', K_f, C_2, \varphi)$
$b \leftarrow \mathsf{TVer}(H, \varphi(M), S, C_2)$
If $b = 0$ then Return true
Else Return false

Oracle $\mathbf{Enc}(H, M)$
$(C_1, C_2) \leftarrow \mathsf{Enc}_K(H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(H, C_1, C_2)\}$
Return (C_1, C_2)

Oracle $\mathbf{Dec}(H, C_1, C_2)$
If $(H, C_1, C_2) \notin \mathcal{Y}_1$
  then Return $\perp$
If $(H, C_1, C_2) = (H^*, C_1^*, C_2^*)$
  then Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
Return (M, K_f)

Oracle $\mathbf{TOpen}(H, C_1, C_2, \varphi)$
If $(H^*, C_1^*, C_2^*) = (H, C_1, C_2)$ then
  If $\varphi(M_0) \neq \varphi(M_1)$ then
    Return $\perp$
If $(H, C_1, C_2) \notin \mathcal{Y}_1$ then
  Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
$(\varphi(M), S) \leftarrow \mathsf{TOpen}_K(M, K_f, \varphi)$
Return $(m = \varphi(M), S)$

Oracle $\mathbf{Dec}^*(H, C_1, C_2)$
Return $\mathsf{Dec}_K(H, C_1, C_2)$

Oracle $\mathbf{ChalDec}(H, C_1, C_2)$
If $(H, C_1, C_2) \in \mathcal{Y}_1$ then
  Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}_K(H, C_1, C_2)$
If $M \notin \perp$ then
  $\mathsf{win} \leftarrow \mathsf{true}$

Fig. 4. Security games

bit string with length ℓ in such a way that one can later open the commitment of certain segment with length at specific positions (or arbitrary non-consecutive substring), so it can open in a way that m_i is the i-th bit of the committed message. [1]. Specifically, TOC can be formalized as follows.

- $\mathsf{TOC.KeyGen}(1^\lambda)$: Given the security parameter λ, the key generation outputs some public parameters pp.
- $\mathsf{TOC.Com}(pp, M)$: On input a bit string M and the public parameters pp, the committing algorithm outputs a commitment string C and the commit key K.
- $\mathsf{TOC.TOpen}$ (pp, C, K, ϕ, M) : On input the public parameter pp, the commitment C, the commitment key K, the position function ϕ and the message M. This target opening algorithm is to produce a opening S which prove that $\varphi(M)$ consists of the certain bits according to the position function φ.
- $\mathsf{TOC.TVer}(pp, C, S, m, \varphi)$: The verification algorithm accepts (i.e., it outputs 1) only if S is a valid proof that C was created to a bit string M such that $\phi(M) = m$.

The basic property of a target opening commitments is that it meets a attractive security requirement named position binding. Informally, this says that it should be infeasible, for any polynomially bounded adversary having knowledge of pp, to come up with a commitment C and two different valid openings for the same position i. More formally:

Definition 1 (position binding). *A target open commitment satisfies position binding if for every PPT adversary $\mathcal{A}$ and position function φ the following probability (which is taken over all honestly generated parameters) is at most negligible in λ:*

$$\Pr[\mathit{TOC.TVer}(C, m, S, \varphi) = 1 \wedge \mathit{TOC.TVer}(C, m', S', \varphi) = 1 \wedge m \neq m' | (C, m, m', S, S', \varphi) \leftarrow_{\$} \mathcal{A}]$$

Target open commitments also require the targeted hiding property, i.e., the bits that have not been opened will remain semantically secure. Informally, a TOC is hiding if an adversary cannot distinguish whether a commitment was created to a bit string M or to M', even after seeing some openings at certain bit positions where the two sequences agree.

Definition 2 (targeted hiding). *A target open commitment satisfies targeted hiding if for every PPT adversary $\mathcal{A}$ and position function φ the following probability (which is taken over all honestly generated parameters) is at most negligible in λ:*

[1] We may view a TOC as a special case of the more general notion of vector commitment [**?**] or functional commitment [**?**]. Both of them rely on algebraic structure and public key operations. We formulate the notion of ToC simply for the sake of potential efficient constructions that are more suitable for secure messaging.

$\text{IND-CPA}^{\mathcal{A}}_{\mathsf{Enc}}$

$b \leftarrow_{\$} \{0,1\}$
$pp \leftarrow_{\$} \mathsf{KGen}(1^n)$
$(M_0, M_1) \leftarrow_{\$} \mathcal{A}^{O}(1^n, pp)$
$(C^*, K_f) \leftarrow_{\$} \mathsf{TOC.Com}(pp, M_b)$
$b' \leftarrow_{\$} \mathcal{A}^{\boldsymbol{TOpen}}(1^n, pp, C)$
return $b = b'$

Oracle $\boldsymbol{TOpen}(C, \varphi)$

if $C = C^*$
  if $\varphi(M_0) \neq \varphi(M_1)$
    return $\perp$
else $S \leftarrow \mathsf{TOC.\ TOpen}(pp, C, K_f, \varphi, M)$
return $(S, \varphi(M))$

A simple instantiation of TOC. We can give a very simply construction of TOC that only involves a collision resistant hash.

- $\mathsf{TOC.KeyGen}(1^\lambda)$: Given the security parameter λ, the key generation outputs some public parameters pp.
- $\mathsf{TOC.Com}(pp, M)$: On input a bit string $M = m_1 m_2 \ldots m_\ell$ with length ℓ, and the public parameters pp, the committing algorithm first commits each bit running the bit commitment algorithm $bCom(m_i, pp)$ and outputs a commitment c_i and the opening r_i. In particular, $bCom$ can be instantiated simply with a collision resistant hash. Then apply a collision resistant hash H on the commitments of the message bit to build a Merkle tree using $c_1, \ldots, c_\ell$ as leaves, and the resulting root is denoted as C. The algorithm outputs commitment C and openings (or commitment key) $K := r_1, \ldots, r_\ell$.
- $\mathsf{TOC.TOpen}$ (pp, C, K, ϕ, M) : On input the public parameter pp, the commitment C, the commitment key K, the position function ϕ and the message M. This target opening algorithm first evaluates $\varphi(M)$, suppose $\varphi(M) := \{m_{i_j}\}$, where i_j is the index that the corresponding bit to be opened. For each i_j, the algorithm first reveals c_{i_j} and the corresponding Merkle proof π_{i_j}, and the corresponding openings r_{i_j}
- $\mathsf{TOC.TVer}(pp, C, S, m, \varphi)$: For each i_j, the verification algorithm first checks whether the Merkle proof π_{i_j} is correctly formed, and then checks whether $h(m_{i_j}, r_{i_j}) = c_{i_j}$. If all passes, the algorithm outputs 1, otherwise outputs 0.

Security sketch. The above construction is fairly simple, we only briefly explain the security here and defer a detailed proof to the full version. Regarding position binding, it is fairly easy to see. For an index i, supposed it is opened, the actual commitment c_i with its Merkle proof can be verified. Then following the property of the underlying bit commitment , the opening of c_i can be ensured by the binding property (or simply the collision resistance of the hash). Regarding targeted hiding, in the first step, the revealed contains only information about commitments of the unopened message bits, which are simulatable with nothing, thus the remaining bits are still semantically secure.

A generic construction of toccAEAD In this section, we will provide a generic construction of toccAEAD from a TOC and any AEAD. Let us represent the key space as $\mathcal{K}_{EC} \subseteq \Sigma^*$, the header space as $\mathcal{H}_{EC} \subseteq \Sigma^*$, the message space as $\mathcal{M}_{EC} \subseteq \Sigma^*$, the ciphertext space as $\mathcal{C}_{EC} \subseteq \Sigma^*$, the opening space as $\mathcal{K}_f \subseteq \Sigma^*$, the selective opening space as $\mathcal{S} \subseteq \Sigma^*$and the franking tag space $\mathcal{T} \subseteq \Sigma^*$.

- **Key generation** $\mathsf{TOCE.Kg}(1^\lambda)$: On input the security parameter λ, use the key generation algorithm for AEAD scheme $AEAD.Kg$ to generate the secret key K. Output K as the secret key for the AEAD scheme.
- **Encryption** $\mathsf{TOCE.Enc}(M)$: First, we commit the message M under the target opening commitment scheme and get the commitment C_2 and the commitment key K_f. Second, We encrypt the C_2, message M and the commitment key K_f and get $C_1 = Enc_k(C_2, M\|K_f)$. Finally, the algorithm outputs the ciphertext (C_1, C_2).
- **Decryption**$\mathsf{TOCE.Dec}$ Firstly, use the AEAD decryption Dec to recover M and K_f. If $M = \perp$ then return $\perp$. Second, use the target opening algorithm $S \leftarrow \mathsf{TOpen}(C_2, M, K_f, I)$ for the identity position function$I$. If $\mathsf{TVer}(S, K_f, M, I) = 0$, abort; Else,return (M, K_f)
- **Targeted open**: If choose to open according the position function φ. use the target opening algorithm $S \leftarrow \mathsf{TOpen}(C_2, M, K_f, I)$ for the identity position function I.
- **Verification**: If $\mathsf{TVer}(C_2, S, M, I) = 0$, output 0; otherwise output 1.

3.3 Security analysis

Theorem 1. *Let $CtE = TOCtE[TOC, SE]$. Let $\mathcal{A}$ be an $TO-IND_{CtE}$ adversary Then we give the adversary $\mathcal{B}$ against the TOC scheme and the adversary $\mathcal{C}$ against the AEAD scheme, which satisfy*

$$\mathsf{Adv}^{\text{toce}}_{\mathcal{A},\mathsf{PRF}}(n) \leq \mathsf{Adv}^{\text{aead}}_{\mathcal{C}}(n) + \mathsf{Adv}^{\text{toc}}_{\mathcal{B}}(n)$$

Proof. We start from the Game 1, which is the real game that the adversary faces. According to the confidentiality of the AEAD, we can replace the encryption result of the AEAD to the encryption of random message, then we get the Game 2.

Next, we will reduce the hiding property of the target opening commitment to Game 2. Precisely, given the adversary $\mathcal{A}$ to attack the Game 2, we can construct an adversary $\mathcal{B}$ to attack the hiding property of the TOC scheme. $\mathcal{B}$ works as follows. He first generates the secret key K for AEAD, provide public parameter for $\mathcal{A}$, then answer the oracles for $\mathcal{A}$. Note that $\mathcal{B}$ can easily answer the Enc oracle by generate the commitment by himself as C_1 and encrypt a random message by K as C_2. $\mathcal{B}$ answer the decryption oracle by search (H, M_1, M_2) in the list $\mathcal{Y}_1$ and find the corresponding (M, K_f). When $\mathcal{A}$ output the message pair (M_1, M_2), $\mathcal{B}$ provide same message pair to the challenger. When the challenger reply the commitment C^*, $\mathcal{B}$ output the challenge ciphertext (C_1^*, C_2^*) to *adv* where C_1^* is the AEAD encryption of H and a random message M and C_2^* is the commitment.

To answer the oracle **TOpen**, given (H, C_1, C_2, ϕ), if (H, C_1, C_2) is the challenge ciphertext, $\mathcal{B}$ first check whether $\varphi(M_0) \neq \varphi(M_1)$, then use the (C_2, φ) to ask the **TOC.TOpen** oracle and output $S, \varphi(M)$, otherwise return $\perp$. If (H, C_1, C_2) is not the challenge ciphertext, $\mathcal{B}$first check whether (H, C_1, C_2) is in $\mathcal{Y}_\infty$. If not, return $\perp$. Otherwise $\mathcal{B}$can find corresponding (M, K_f) and compute the **TOC.TOpen** by himself.

Finally, we get the result

$$\mathsf{Adv}^{\mathrm{toce}}_{\mathcal{A},\mathsf{PRF}}(n) \leq \mathsf{Adv}^{\mathrm{aead}}_{\mathcal{C}}(n) + \mathsf{Adv}^{\mathrm{toc}}_{\mathcal{B}}(n)$$

Theorem 2. *Let $CtE = TOCtE[TOC, SE]$. Let $\mathcal{A}$ be an $TO - CTXT_{CtE}$ adversary making at most q queries to its oracles. Then we give adversaries $\mathcal{B}$.*

Proof. We have the TO-CTXT game as follows. First of all, we can transform the TO-CTXT game to the game G_1. In game G_1, the only difference is that we replace the oracle $\mathbf{Dec}^*$ with the oracle $\mathbf{Dec}'$.

Game G_1

$K \leftarrow \mathsf{AEAD.Kg}$
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}$
return win

Oracle $\mathbf{Enc}(H, M)$
$(C_2, K_f) = \mathsf{TOC.Com}(H, M)$
$C_1 \leftarrow \mathsf{AEAD.Enc}_k(C_2, M \| K_f)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{H, C_1, C_2\}$
$D[H, C_1, C_2] \leftarrow (M, K_f)$
Return (H, C_1, C_2)

Oracle $\mathbf{Dec}^*(H, C_1, C_2)$
if $D[H, C_1, C_2] \neq \perp$ **then**
  return $D[H, C_1, C_2]$
else if $\perp \leftarrow \mathsf{AEAD.Dec}(H, C_1, C_2)$ **then**
    return $\perp$
  else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, C_1, C_2)$
    $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
    if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
    return $\perp$
    else $\mathsf{win} \leftarrow \mathsf{true}$
    return (M, K_f)

Oracle $\mathbf{TOpen}(H, C_1, C_2, \varphi)$
if $D(H, C_1, C_2) \neq \perp$
  return $(M, K_f) \leftarrow D(H, C_1, C_2)$
else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}_K(H, M_1, M_2)$
  $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
  if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$ **then**
    return $\perp$
  else $S \leftarrow \mathsf{TOpen}(C_2, M, K_f, \varphi)$
    $\mathsf{win} \leftarrow \mathsf{true}$
    return $(S, \varphi(M))$

Oracle $\mathbf{ChalDec}(H, C_1, C_2)$
if $(H, C_1, C_2) \in \mathcal{Y}_1$
  return $\perp$
if $\perp \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
  return $\perp$
else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
    $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
    if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
    return $\perp$
  if $M \neq \perp$
  $\mathsf{win} \leftarrow \mathsf{true}$
    else return (M, K_f)

The Confidentiality of AEAD

$\mathsf{Game}_1(n)$	$\mathsf{Game}_2(n)$
$K \leftarrow \mathsf{AEAD.Kg}$	$K \leftarrow \mathsf{AEAD.Kg}$
$(H, M_1, M_2, state) \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}$	$(H, M_1, M_2, state) \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}$
$(C_2^*, K_f) = \mathsf{TOC.Com}(H \| M_b)$	$(C_2^*, K_f) = \mathsf{TOC.Com}(H \| M_b)$
$C_1^* \leftarrow \mathsf{AEAD.Enc}_k(C_2^*, M_b \| K_f)$	$C_1^* \leftarrow \mathsf{AEAD.Enc}_k(C_2^*, \{0,1\}^{\lvert M_b \| K_f \rvert})$
$b' \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}(state, C_1^*, C_2^*)$	$b' \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}(state, C_1^*, C_2^*)$
Return $b' = b$	return $b' = b$
Oracle $\mathbf{Enc}(H, M)$	Oracle $\mathbf{Enc}(H, M)$
$(C_2, K_f) = \mathsf{TOC.Com}(H \| M)$	$(C_2, K_f) = \mathsf{TOC.Com}(H \| M)$
$C_1 \leftarrow \mathsf{AEAD.Enc}_k(C_2, \{0,1\}^{\lvert M \| K_f \rvert})$	$C_1 \leftarrow \mathsf{AEAD.Enc}_k(C_2, \{0,1\}^{\lvert M \| K_f \rvert})$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{H, C_1, C_2\}$	$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{H, C_1, C_2\}$
Return (H, C_1, C_2)	return (H, C_1, C_2)
Oracle $\mathbf{Dec}(H, C_1, C_2)$	Oracle $\mathbf{Dec}(H, C_1, C_2)$
if $(H, C_1, C_2) \notin \mathcal{Y}_1$	if $(H, C_1, C_2) \notin \mathcal{Y}_1$
then return $\perp$	then return $\perp$
$(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$	Search (H, M_1, M_2) in $\mathcal{Y}_1$ and find (M, K_f)
$S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$	$S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$	if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
then return $\perp$	then return $\perp$
Oracle $\mathbf{TOpen}(H, C_1, C_2, \varphi)$	Oracle $\mathbf{TOpen}(H, C_1, C_2, \varphi)$
if $(H^*, C_1^*, C_2^*) = (H, C_1, C_2)$ then	if $(H^*, C_1^*, C_2^*) = (H, C_1, C_2)$ then
if $\varphi(M_0) \neq \varphi(M_1)$	if $\varphi(M_0) \neq \varphi(M_1)$
then return $\perp$	then return $\perp$
if $(H, C_1, C_2) \notin \mathcal{Y}_1$	if $(H, C_1, C_2) \notin \mathcal{Y}_1$
then return $\perp$	then return $\perp$
$(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$	Search (H, C_1, C_2) in $\mathcal{Y}_1$ and find (M, K_f)
$S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$	$S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$	if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
then return $\perp$	then return $\perp$
$S \leftarrow \mathsf{TOpen}(C_2, M, K_f, \varphi)$	$S \leftarrow \mathsf{TOpen}(C_2, M, K_f, \varphi)$
return $S, \varphi(M)$	return $S, \varphi(M)$

decryption of previously encrypted value in oracle **Dec** and **TOpen** is done by the table lookup. If a ciphertext is submitted to oracle **Dec** that can be successfully decrypted but not present in the table, the flag win will be setted to true. Then we have that

$$\mathbf{Adv}_{TOCtE}^{TO-CTXT}(\mathcal{A}) \leq \Pr[G_1^{\mathcal{A}} \Rightarrow \mathsf{true}]$$

Note that for win to be set with a query (H, C_1, C_2), it must be that no previous encryption query (H, M) for some M returned (C_1, C_2). Let the winning query be on the values (H^*, C_1^*, C_2^*). We partition the probability of setting win into two cases, either (C_1^*, C_2^*) is distinct from all encryption outputs, or (C_1^*, C_2^*) is one of the encryption outputs and H^* is not the header for the encryption query that returned C_1^*, C_2^*. Let win_H be the event that $\mathcal{A}$ wins with a query where H^* is a different header, and win_C be the event that $\mathcal{A}$ wins with a query where (C_1^*, C_2^*) is distinct. Then

$$\Pr[G_0^{\mathcal{A}} \Rightarrow \mathsf{true} \leq \Pr[\mathsf{win}_H] + \Pr[\mathsf{win}_C]$$

We'll first bound $\Pr[\mathsf{win}_C]$. In this case we will construct an adversary $\mathcal{B}$ in the CTXT game of AEAD. This adversary simulates G_0 for $\mathcal{A}$, as follows. When $\mathcal{A}$ queries (H, M) to **Enc**, $\mathcal{B}$ first generates a targeted opening commitment and opening K_f, C_2. Then, $\mathcal{B}$ queries $\mathsf{enc}(C_2, M \| K_f)$. It stores the result in a table, then outputs C_1, C_2 to $\mathcal{A}$. It simulates **Dec***, **TOpen** and **ChalDec** queries that are outputs of previous **Enc** queries by consulting its table and outputting either the proper value (for **Dec** and **TOpen**) or $\perp$ (for **ChalDec**). When $\mathcal{A}$queries **Dec***, **TOpen** and **ChalDec** with a value not in the table, $\mathcal{B}$ submits (C_2, C_1) as a forgery to its decryption oracle, and use the returned results in the further calculations. Our $\mathcal{B}$perfectly simulates G_0 for $\mathcal{A}$. Since $\mathcal{A}$'s query must be a successful forgery and $\mathcal{B}$ will break CTXT of AEAD in this reduction with probability at least $\Pr[\mathsf{win}_C]$, i.e., $\Pr[\mathsf{win}_C] \leq \mathbf{Adv}_{AEAD}^{CTXT}(\mathcal{B})$.

To bound $\Pr[\mathsf{win}_H]$ and complete the proof we can build another reduction using a $tBIND_{\mathsf{TOC}}$ adversary $\mathcal{C}$. The adversary $\mathcal{C}$ simulates $\mathcal{A}$'s view of G_0 as $\mathcal{B}$did, except $\mathcal{C}$generates a random encryption key and computes $\mathsf{AEAD.Enc}$ and $\mathsf{AEAD.Dec}$ internally. When $\mathcal{A}$ makes a query (H, C_1, C_2) to **Dec***, **TOpen** or **ChalDec** where H is not the header input to the encryption query that output C_1, C_2, $\mathcal{C}$fetches from its stored values the message M and opening K_f corresponding to C_2, as well as H_0, the header part of the encryption query that produced C_1, C_2. In its game $\mathcal{C}$outputs $((H, M, K_f), (H_0, M, K_f), C_2)$. The environment of G_0 is perfectly simulated by $\mathcal{C}$. Since in this case the winning query must cause $TOC.Ver$ output 1. In this case, $\mathcal{A}$has broken binding property of the commitment.

TO-CTXT

$K \leftarrow$ AEAD.Kg
win $\leftarrow$ false
$\mathcal{A}^{\mathbf{Enc,Dec,SOpen}}$
return win

Oracle **Enc**(H, M)
$(C_2, K_f) = \mathsf{TOC.Com}(H \| M)$
$C_1 \leftarrow \mathsf{AEAD.Enc}_k(C_2, M \| K_f)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{H, C_1, C_2\}$
Return (H, C_1, C_2)

Oracle **Dec**$^*(H, C_1, C_2)$
if $\perp \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
  return $\perp$
else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
    $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
    if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
      return $\perp$
    else return (M, K_f)

Oracle **TOpen**(H, C_1, C_2, φ)
if $\perp \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$ **then**
  return $\perp$
else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
  $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
  if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$ **then**
    return $\perp$
    $S \leftarrow \mathsf{TOpen}(C_2, M, K_f, \varphi)$
return $(S, \varphi(M))$

Oracle **ChalDec**$^*(H, C_1, C_2)$
if $(H, C_1, C_2) \in \mathcal{Y}_1$
  return $\perp$
if $\perp \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
  return $\perp$
else $(M, K_f) \leftarrow \mathsf{AEAD.Dec}(H, M_1, M_2)$
    $S \leftarrow \mathsf{TOC.TOpen}(C_2, M, K_f, I)$
    if $\mathsf{TOC.TVer}(S, K_f, M, I) = 0$
      return $\perp$
if $M \neq \perp$
  win $\leftarrow$ true
    else return (M, K_f)

3.4 An efficient instantiation

In this section, we will provide an efficient instantiation for our targeted open commitment.

- $\mathsf{TOC.Com}(M)$: Given the message M as the input, parse M into a sequence of bits $m_1, \dots, m_l$. Then use the pseudo-random generator $\mathcal{G}$ with seed sd to generate a sequence of $r_1, \dots, r_l \in \{0,1\}^\lambda$, and compute $h_i = \mathcal{H}(r_i, m_i)$ for $i = 1, \dots, l$. Then use the Merkle tree to hash all h_i together and get the final commitment C. The corresponding opening is $K_f = sd$.
- $\mathsf{TOC.TOpen}(M, sd, \phi)$: Given the seed sd, we can generate the random number $r_1, \dots, r_l$. Then we can easily compute each h_i from r_i and m_i . Suppose that the position function φ denotes to targeted open the bits $m_{i_1}, m_{i_2}, \dots, m_{i_n}$ while conceal the rest bits $m_{j_1}, \dots, m_{j_m}$, so the targeted opening S should be the values h_j for $j \in \{j_1, \dots, j_m\}$ and (r_i, m_i) for $i \in \{i_1, \dots, i_n\}$.
- $\mathsf{TOC.TVer}(C, S, \varphi(M))$: Given the targeted opening S as h_j for $j \in \{j_1, \dots, j_m\}$ and (r_i, m_i) for $i \in \{i_1, \dots, i_n\}$, the verifier compute $h_i = \mathcal{H}(r_i, m_i)$ and gather all h_i for $i = 1, \dots, l$. Then verifier checks whether $\mathcal{H}(h_1, \dots, h_l) = C$.

The targeted hiding property can easily obtain if we model the hash function as the random oracle. Also, the targeted binding property can easily obtain from the collision resistance of the function $\mathcal{H}$.

4 More Efficient TOCE

Our generic construction above can achieve targeted opening in an ideal case, i.e., the receiver can choose arbitrary bit to open, thus requires cryptographic operations such as committing, to be called on each bit of the message. Such kind of selective capability is unnecessarily strong. For example, when the abusive message is an improper picture, the revealed message would not need the precision to bit. Even the abusive message is simply some texts, a meaningful sentence is also composed of multiple consecutive characters which are at least hundreds of bits. Moreover, such a method incurs large overhead when the message size is large, for instance, when one user is sending a picture, or a short video via secure messaging, applying the above construction may require a large number of hash operations.

In this section, we seek for more efficient constructions of toccAEAD with a slightly weaker targeted open capability which is still useful in many settings. In particular, as all AEAD schemes apply some ciphers on message blocks with size λ bits (λ could be 256 for example), we will restrict the targeted opening at the block level: plaintext M is now divided into $m_1, \ldots, m_\ell$, each m_i is with length λ. During the opening phase, the receiver will reveal the message blocks according to the indices, i.e., $\{m_j\} = \varphi(M, \lambda)$, now the selection function φ takes an extra input of message block size λ, and the indices are chosen by the recipient from $\{1, \ldots, \ell\}$, and $\ell = |M|/\lambda$.

There are two reasons to explore in depth such a block-wise targeted opening: (1) the recipient would still be able to choose some of the blocks to reveal to report abusive messages. If a block of 256 bits (just 32 English characters) already contains substantial amount of personal information, there won't be much room for abusive messages; even if the recipient chooses not to reveal this block, the missing tiny piece of information in this hidden block would not influence the abuse reporting much. To put it another way, in a revealed block of 256 bits chosen by the recipient, the leaked information excluding the abusive message would be insignificant to him. (2) trivial application of the generic construction to message blocks still has an large overhead, thus more specially designed constructions are needed.

To be more precise about the overhead, one efficiency metric we consider is the *pass* defined in [2], which characterizes the ratio of the number of calls of symmetric key cryptographic building block such as a cipher (or hash) needed for a ccAEAD over the number of calls for a regular AEAD scheme. In particular, in [2], the authors gave an elegant construction of ccAEAD that only requires one pass! The intuition is to chain the ciphertext together so that the binding properties are generated along the way of encryption.

Inefficiency of toccAEAD constructions. The toccAEAD of the previous generic construction can be bit-wise targeted opening. Essentially, it can also be trivially extended to blockwise targeted open. However, since the pseudo random generator need at least one path to generate enough randomness, the TOC scheme need at least two pass to compute the commitment and the AEAD scheme need at least one pass to encrypt, the generic construction of toccAEAD need at least four pass to compute encryption. Obviously, it is far from one desires for practical use, and we need to design more efficient specific constructions for block wise targeted opening.

4.1 Block-wise targeted open ccAEAD definitions

Now let us define the block-wise toccAEAD. The syntax is essentially the same as regular toccAEAD, with the only exception that each message bit now becomes a message block with length λ.

Targeted opening, compactness, and using few passes seem to be antagonistic to each other. The cascaded construction in [2] achieved both compactness and using only one pass; however, the messages are all chained together, verifying m_i requires knowledge of m_{-1}, thus inherently difficult to enable targeted opening. On the other hand, processing each data block separately to enable targeted opening, then different randomness seems required for each message block. Generating those randomness already somehow requires one pass of crypto calls. Together with the encryption itself, and the commitment, this already causes three passes. (If we need further compress all the commitments for compactness, requires one more pass such as the trivial instantiation of our generic construction). Those attempts motivate us to consider a potential weaker notion, and seek for a non-black box construction to reduce the number of passes needed.

Nonce-based scheme. The above generic construction of targeted opening committing AEAD is randomized. However, cryptographers have advocated that modern AEAD schemes should be designed as nonce-based instead. Thus the internal randomness during the encryption should be replaced with an input nonce, and the security should hold as long as the nonce never repeats throughout the course of encrypting messages with a particular key.

Formally, a nonce-based block-wise targeted opening committing AEAD is a following tuple of algorithms (KeyGen,Enc,Dec,TOpen,TVer), which is similar to the previous definition. In addition to the other sets, we associate to any nbTOCE scheme a nonce space $\mathcal{N} \in \{0,1\}^*$. We also define the block-wise position function φ as follows. If M is a bit string with length n^2, φ_t is a function that divides M into $l = n/t$ blocks $m_1, \ldots, m_l$ of size t, then picks a subset of the blocks m_i with the indices $i_1, i_2, \ldots, i_j$ depending on φ_t's definition. The space of all the block-wise position functions is Φ_t.

[2] For simplicity, we assume that l can be divided by t, otherwise we can pad M with bits zeros.

- **Key generation**: The randomized algorithm KeyGen outputs a secret key $K \in \mathcal{K}$.
- **Encryption**: The deterministic algorithm Enc takes a triple $(K, N, H, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{M}$ as input and outputs a pair $(C_1, C_2) \in \mathcal{C} \times \mathcal{T}$. Here C_1 is the ciphertext that carries the payload and C_2 is the franking tag.
- **Decryption**: The deterministic algorithm Dec takes a tuple $(K, N, H, C_1, C_2) \in \mathcal{K} \times \mathcal{N} \times \mathcal{H} \times \mathcal{C} \times \mathcal{T}$ as input and outputs a message and opening value pair $(M, K_f) \in \mathcal{M} \times \mathcal{K}_f$.
- **Targeted open**: The deterministic algorithm TOpen takes as input a tuple $(H, M, K_f, C_2, \varphi_t) \in \mathcal{H} \times \mathcal{M} \times \mathcal{K}_f \times \mathcal{T} \times \Phi_t$, the targeted value represented as $\varphi_t(M)$, and the corresponding opening S.
- **Verification**: The deterministic algorithm TVer takes as input a tuple values of $(H, \varphi_t(M), S, C_2, , \varphi_t) \in \mathcal{H} \times \varphi_t(\mathcal{M}) \times \mathcal{S} \times \mathcal{T} \times \Phi_t$ and output a bit b. Specifically, we assume that TVer outputs 0 if the targeted opening is not valid.

Blockwise toccAEAD security definitions. We weaken the confidentiality after opening part of the message blocks of a toccAEAD, which enables us to search for a more efficient construction that uses fewer passes. There are multiple ways of weakening on confidentiality of the remaining message blocks. The first one requires that a message block that has not been opened, will remain semantically secure, if the message block is unpredictable, i.e., generated from a distribution that has sufficient entropy. The second one requires that a message block that has not been opened, will remain one way secure, i.e., adversary who sees the opening of some other message blocks, cannot recover the remaining unopened ones. Clearly, the first definition is strictly stronger, so we adopt the first weakened definition. We emphasize here that all messages satisfy the standard semantic security if no message blocks are revealed by the receiver.

Formally, we define the security games for the nonce based block wise targeted opening ccAEAD in Figure 5. Note that the adversary never repeat the same N across a pair of encryption queries, and the challenge nonce N^* also will not be queried for the **Enc** oracle. To achieve more efficient construction, we also provide a more weaker notion of confidentiality as follows:

4.2 Block wise toccAEAD construction

We now proceed to describe our nonce based block-wise targeted open ccAEAD construction. Let $\mathcal{G}$ is a nonce-based pseudo-random generator. $\mathcal{H}$ is a collision resistant hash function which can be modelized as a random oracle. Integer t denotes the block size of the message (e.g., 256 for popular ciphers). So the scheme is as follows.

- $\mathsf{bTOCE.KeyGen}(1^\lambda)$: Generate a seed sd for the pseudo random generator $\mathcal{G}$. The secret key K is sd.
- $\mathsf{bTOCE.Enc}(N, H, K, M)$: Given the nonce N, the secret key $K = sd$ and the message $M \in \{0, 1\}^{lt}$, do the following steps:

$\underline{TO - nIND^{\mathcal{A}}_{nTOCE}}$
$K \leftarrow_{\$} \mathsf{KeyGen}$
$st_1 \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}$
$\{N^*, H^*, (M_0, M_1), st_2\} \leftarrow_{\$} \mathcal{A}(st_1)$
$b \leftarrow_{\$} \{0, 1\}$
$(C_1^*, C_2^*) \leftarrow \mathsf{Enc}(K, N^*, H^*, M_b)$
$b \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}(C_1^*, C_2^*, st_2)$
Return b

$\underline{TO - r - nBIND^{\mathcal{A}}_{nTOCE}}$
$((m, S), (m', S'), C_2, \varphi) \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}$
$b \leftarrow \mathsf{TVer}(m, S, C_2, \varphi)$
$b' \leftarrow \mathsf{TVer}(m, S', C_2, \varphi)$
If $m = m'$ then
  Return false
Return$(b = b' = 1)$

$\underline{TO - nCTXT^{\mathcal{A}}_{CE}}$
$K \leftarrow_{\$} \mathsf{KeyGen}$
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec}^*,\mathbf{TOpen},\mathbf{ChalDec}}$
Return win

$\underline{TO - s - nBIND^{\mathcal{A}}_{CE}}$
$(K, H, C_1, C_2, \varphi) \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}$
$(M', K_f) \leftarrow \mathsf{Dec}(K, N, H, C_1, C_2)$
If $M' = \perp$ then Return false
$S \leftarrow \mathsf{TOpen}(H, M', K_f, C_2, \varphi)$
$b \leftarrow \mathsf{TVer}(H, \varphi(M'), S, C_2)$
If $b = 0$ then Return true
Else Return false

Oracle $\mathbf{Enc}(N, H, M)$
$(C_1, C_2) \leftarrow \mathsf{Enc}(K, N, H, M)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{(N, H, C_1, C_2)\}$
Return (C_1, C_2)

Oracle $\mathbf{Dec}(N, H, C_1, C_2)$
If $(N, H, C_1, C_2) \notin \mathcal{Y}_1$
  then Return $\perp$
If $(N, H, C_1, C_2) = (N^*, H^*, C_1^*, C_2^*)$
  then Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}(K, N, H, C_1, C_2)$
Return (M, K_f)

Oracle $\mathbf{TOpen}(N, H, C_1, C_2, \varphi)$
If $(H^*, C_1^*, C_2^*) = (H, C_1, C_2)$ then
  If $\varphi(M_0) \neq \varphi(M_1)$ then
    Return $\perp$
If $(N, H, C_1, C_2) \notin \mathcal{Y}_1$ then
  Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}(K, N, H, C_1, C_2)$
$(\varphi(M), S) \leftarrow \mathsf{TOpen}(H, M, K_f, C_2, \varphi)$
Return $(m = \varphi(M), S)$

Oracle $\mathbf{Dec}^*(N, H, C_1, C_2)$
Return $\mathsf{Dec}(K, N, H, C_1, C_2)$

Oracle $\mathbf{ChalDec}(N, H, C_1, C_2)$
If $(N, H, C_1, C_2) \in \mathcal{Y}_1$ then
  Return $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}(K, N, H, C_1, C_2)$
If $M \notin \perp$ then
  $\mathsf{win} \leftarrow \mathsf{true}$
 Return (M, K_f)

Fig. 5. The security games for the nonce based block wise targeted opening ccAEAD.

1. Use the pseudorandom generator $\mathcal{G}$ with the seed sd and the nonce N to generate bits strings R with the size of lt, i.e.,$R = (r_1, \ldots, r_\ell) \leftarrow \mathcal{G}(sd, N, lt)$ where each $r_i \in \{0,1\}^t$.
2. Divide each M into ℓ blocks $m_1, \ldots, m_\ell$, and every block has t bits. Then use one time pad to encrypt each message m_i, i.e., $C_1^i = r_i \oplus m_i$, for $i = 1, \ldots, \ell$;
3. Hash each r_i together with m_i and get $h_i = \mathcal{H}(r_i, m_i)$;
4. Compute the final tag $C_2 = \mathcal{H}(H, h_1, \ldots, h_l)$.

The finally output ciphertext is $C_1 = \{C_1^i\}_{i=1}^l$ and the tag C_2.

– $\mathsf{bTOCE.Dec}(K, N, H, (C_1, C_2))$: Firstly, use seed $K = sd$ to recover $R = (r_1, \ldots, r_\ell)$. Then one can get $m_i = C_1^i \oplus r_i$ and $h_i = \mathcal{H}(m_i, r_i)$ for $i = 1, \ldots, l$. If $C_2 = \mathsf{H}(H, h_1, \ldots, h_l)$, output the message $M = \{m_1, \ldots, m_l\}$ and the opening $K_f = R = \{r_1, \ldots, r_l\}$, otherwise output $\perp$.
– $\mathsf{bTOCE.TOpen}(H, M, R, \varphi_t)$: If the position function φ_t denotes to open the blocks with index $i_1, \ldots, i_j$, one just compute each $h_i = \mathcal{H}(r_i, m_i)$ and output the targeted opening $S = \big\{\{h_i\}_{i \notin \{i_1,\ldots,i_j\}}, \{r_i\}_{i \in \{i_1,\ldots,i_j\}}\big\}$ and the opened messages $\varphi_t(M) = \{m_i\}_{i \in \{i_1,\ldots,i_j\}}$.
– $\mathsf{bTOCE.TVer}(H, \varphi_t(M), S, C_2, \varphi_t)$: If the position function φ_t denotes to open the blocks with index $i_1, \ldots, i_j$, one parse the targeted opening S as $\{h_i\}_{i \notin \{i_1,\ldots,i_j\}}$ and $\{r_i\}_{i \in \{i_1,\ldots,i_j\}}$ and $\varphi_t(M)$ as $\{m_i\}_{i \in \{i_1,\ldots,i_j\}}$. Then compute $h_i = \mathcal{H}(r_i, m_i)$ for $i \in \{i_1, \ldots, i_j\}$ and check whether $C_2 = \mathcal{H}(H, h_1, \ldots, h_l)$. If the check is passed, output 1 otherwise output 0.

Comparison. The main advantage of bTOCE is efficiency. Note that bTOCE only need tree pass, while the generic construction need at least four passes.

4.3 Security Analysis

Next we will provide a security analysis for our bTOCE scheme.

Confidentiality. The confidentiality of the scheme can be seen from the following theorem.

Theorem 3 (Confidentiality). *Let $\mathcal{G}$ is a nonce-based pseudo-random generator. $\mathcal{H}$ is a collision resistant hash function which can be modelized as a random oracle. Let $\mathcal{A}$ be the TO-nIND adversary, and $\mathcal{B}$ be the adversary against the pseudo random generator $\mathcal{G}$.*

$$\boldsymbol{Adv}^{\mathcal{A}}_{\textit{TO-nIND}} \leq \mathsf{negl}(n) + \boldsymbol{Adv}^{\mathcal{B}}_{\mathcal{G}}$$

Proof. Let $G_0 = \text{TO-nIND}^{\mathcal{A}}_{bTOCE}$. Firstly, we replace the pseudo random string R with truly random string and obtain game G_1 in Figure 6. Hence we have

$$\mathbf{Adv}^{\mathcal{A}}_{\text{TO-nIND}} \leq \mathbf{Adv}^{\mathcal{A}}_{G_1} + \mathbf{Adv}^{\mathcal{B}}_{\mathcal{G}}$$

where $\mathcal{B}$ is the adversary to attack the PRG $\mathcal{G}$.

Secondly, we replace each h_i in game G_1 with newly generated random string according to the random oracle model and get Game G_2. Hence we have

$$\mathbf{Adv}^{\mathcal{A}}_{G_1} \le \mathbf{Adv}^{\mathcal{A}}_{G_2}.$$

Since C_1 and C_2 in Game G_2 are both independent of the message M_b, we have

$$\mathbf{Adv}^{\mathcal{A}}_{G_2} \le \mathsf{negl}(n)$$

Game G_1

$K \leftarrow_\$ \mathsf{KeyGen}$
$st_1 \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}(1^\lambda)$
$\{N^*, H^*, (M_0, M_1), st_2\} \leftarrow_\$ \mathcal{A}(st_1)$
$b \leftarrow_\$ \{0,1\}$
$(H^*, C_1^*, C_2^*) \leftarrow \mathsf{Enc}_K(N^*, H^*, M_b)$
$b' \leftarrow_\$ \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}}(C_1^*, C_2^*, st_2)$
return $b = b'$

Oracle $\mathbf{Enc}(sd, N, H, M)$
$R = (r_1, \ldots, r_l) \leftarrow_\$ \{0,1\}^{lt}$
for i from 1 to n
  $C_1^i = m_i \oplus r_i$
  $h_i = \mathcal{H}(r_i, m_i)$
$C_1 = \{C_1^i\}_{i=1}^n$
$C_2 = \mathcal{H}(H, h_1, \ldots, h_l)$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{N, H, C_1, C_2\}$
$D[N, H, C_1, C_2] \leftarrow (M, K_f)$
Return (N, H, C_1, C_2)

Oracle $\mathbf{Dec}(sd, N, H, C_1, C_2)$
if $D[N, H, C_1, C_2] \neq \perp$ **then**
  return $D[N, H, C_1, C_2]$
else return $\perp$

Oracle $\mathbf{TOpen}(sd, N, H, C_1, C_2, \varphi)$
if $D[N, H, C_1, C_2] = \perp$ **then**
return $\perp$
else $(M, R) \leftarrow D[N, H, C_1, C_2]$
parse M as $(m_1, \ldots, m_l)$
parse R as $(r_1, \ldots, r_l)$
φ corresponds to positions $(i_1, \ldots, i_j)$
for j from 1 to s
  $h_{i_j} = \mathcal{H}(r_{i_j}, m_{i_j})$
$S = \left(\{h_i\}_{i \in \{l\}/\{i_1,\ldots,i_j\}}, \{r_i\}_{i \in \{i_1,\ldots,i_j\}}\right)$
return $(S, \varphi(M))$

Fig. 6. Game G_1 for confidentiality proof

□

Integrity. Next we will show the integrity of our bTOCE scheme.

Lemma 1. *Let $\mathcal{H}$ be a collision resistant hash function. $M = (m_1, \ldots, m_l) \in \{0,1\}^{l\lambda}$ and $R = (r_1, \ldots, r_l) \in \{0,1\}^{l\lambda}$. So the function*

$$\mathcal{F}(H, M, R) = \mathcal{H}(H, \mathcal{H}(r_1, m_1), \ldots, \mathcal{H}(r_l, m_l)) \tag{1}$$

is a MAC scheme with respect to the key R and the message (H, M). Specifically, $\mathcal{F}$ is collision resistance and multi-user unforgeable under chosen-message attack.

Theorem 4 (Integrity). *Let $\mathcal{A}$ be the TO-nCTXT adversary, $\mathcal{B}$ be the adversary against the collision resistance of $\mathcal{F}$ defined in (1), $\mathcal{C}$ be the multi-user unforgeability under chosen-message attack (MU-UF-CMA) for $\mathcal{F}$.*

$$\boldsymbol{Adv}^{bTOCE}_{TO\text{-}nCTXT}(\mathcal{A}) \leq \boldsymbol{Adv}^{CR}_{\mathcal{F}}(\mathcal{B}) + \boldsymbol{Adv}^{MU-UF-CMA}_{\mathcal{F}}(\mathcal{C})$$

Proof. Let $G_0 = \text{TO-nCTXT}^{\mathcal{A}}_{bTOCE}$. We modify game G_0 to obtain game G_1. The differences are that:

1. queries to $\mathbf{Dec}^*$ on tuples (N, H, C_1, C_2) for which there was a previous query to $\mathbf{Enc}(N, H, M)$ that returned C_1,C_2 simply reply with (M) without bothering to do decryption;
2. we set win to true if any other query to **Dec** successfully decrypts.

The first difference is without loss, since the $\mathbf{Dec}^*$ in G_1 would have anyway returned (M, S). The second difference only increases the adversary's probability of success. Thus

$$\Pr[G_0^{\mathcal{A}}) \Rightarrow 1] \leq \Pr[G_1^{\mathcal{A}} \Rightarrow 1].$$

We now bound $\mathcal{A}$'s probability of success in G_1 by its ability to forge against the MAC $\mathcal{F}$ in the equation (1). The $MU-UF-CMA$ adversary $\mathcal{C}$ can simulate the environment $\mathcal{A}$by using the **Tag** and **Ver** oracles to perform tagging and verification via $\mathcal{F}$.

We need to show that anytime win would have been set in G_2 for ciphertext (N^*, H^*, C_1^*, C_2^*), the corresponding query to $\mathbf{Ver}(N^*, (H^*, M^*), C_2^*)$ is a successful forgery for the MAC scheme $\mathcal{F}$,i.e., C_2^* is not generated from the oracle **Tag**. To proof this claim, we need to consider two scenarios. The first scenario is that C_2^* did not exist in previous answers of **Enc** queries, so obviously the C_2^* can not be the answer of the **Tag** oracle and $(N^*, (H^*, M^*), C_2^*)$ is a successful MAC forgery. The second scenario is that C_2^* is included in previous answers of **Enc** which corresponding input is (N, H, M). Let the return from the corresponding **Enc** query on inputs (N, H, M) be the pair (N, H, C_1, C_2^*). Let the ciphertext (N^*, H^*, C_1^*, C_2^*) is decrypted to the message M^*. According to the collision resistance of $\mathcal{F}$, $(N, H, M) = (N^*, H^*, M^*)$ with the probability $1 - Adv^{cs}_{\mathcal{F}}$. Then since the encryption algorithm is deterministic, $C_1 = C_1^*$ due to the same input (N^*, H^*, M^*) of the **Enc** query. This is contradict to our assumption which states (N^*, H^*, C_1^*, C_2^*) is not generate from **Enc** oracle, because in the second scenario the $\mathcal{A}$have queried the **Enc** oracle with the point $(N, H, M) = (N^*, H^*, M^*)$. Hence we have

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}^{CR}_{\mathcal{F}}(\mathcal{B}) + \mathbf{Adv}^{MU-UF-CMA}_{\mathcal{F}}(\mathcal{C})$$

□

Receiver Binding. We have the following theorem.

Theorem 5. *Let $\mathcal{A}$ be the TO-nCTXT adversary. There is an adversary against the collision resistance of the hash function $\mathcal{H}$.*

$$\boldsymbol{Adv}^{bTOCE}_{TO\text{-}nCTXT}(\mathcal{A}) \leq l \cdot \boldsymbol{Adv}^{CR}_{\mathcal{H}}(\mathcal{B})$$

$C^{\mathbf{Tag},\mathbf{Ver}}$

$sd \leftarrow \{0,1\}^{\lambda}$
$\mathsf{win} \leftarrow \mathsf{false}$
$\mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{SOpen}}$
return win

Oracle $\mathbf{Enc}_K(N, H, M)$
$R = (r_1, \ldots, r_l) \leftarrow\!\!\$\ \mathcal{G}(sd, N, l \cdot t)$
for i from 1 to n
  $C_1^i = m_i \oplus r_i$
$C_1 = \{C_1^i\}_{i=1}^n$
$C_2 = \mathbf{Tag}(N, (H, M))$
$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{H, C_1, C_2\}$
$D[N, H, C_1, C_2] \leftarrow (M, R)$
Return (N, H, C_1, C_2)

Oracle $\mathbf{Dec}^*(N, H, C_1, C_2)$
if $D[N, H, C_1, C_2] \neq \perp$ **then**
  return $D[N, H, C_1, C_2]$
$R = (r_1, \ldots, r_l) \leftarrow \mathcal{G}(sd, N, l \cdot t)$
for i from 1 to n
  $m_i = C_i^i \oplus r_i$
if $0 \leftarrow\!\!\$\ \mathbf{Ver}(N, C_2, (H, M))$
  return $\perp$
else $\mathsf{win} \leftarrow \mathsf{true}$
return $M = (m_1, \ldots, m_l)$ and R

Oracle $\mathbf{TOpen}(sd, N, H, C_1, C_2, \varphi_t)$
if $D[N, H, C_1, C_2] \neq \perp$ **then**
  $(M, R) \leftarrow D[N, H, C_1, C_2]$
$R = (r_1, \ldots, r_l) \leftarrow \mathcal{G}(sd, N, l \cdot t)$
φ_t corresponds to positions $(i_1, \ldots, i_s)$
for j from 1 to s
  $h_{i_j} = \mathcal{H}(r_{i_j}, m_{i_j})$
$S = \left(\{h_i\}_{i \in \{l\}/\{i_1,\ldots,i_j\}}, \{r_i\}_{i \in \{i_1,\ldots,i_j\}}\right)$
return $(S, \varphi(M))$

Oracle $\mathbf{ChalDec}(sd, N, H, C_1, C_2)$
if $(N, H, C_1, C_2) \in \mathcal{Y}_1$
  return $\perp$
$R = (r_1, \ldots, r_l) \leftarrow \mathcal{G}(sd, N, lt)$
for i from 1 to n
  $m_i = C_i^i \oplus r_i$
if $0 \leftarrow \mathbf{Ver}(N, C_2, (H, M))$
  return $\perp$
else $\mathsf{win} \leftarrow \mathsf{true}$
  return (M, R)

Game G_1

$K \leftarrow \mathsf{KeyGen}$

$\mathsf{win} \leftarrow \mathsf{false}$

$\mathcal{A}^{\textbf{Enc},\textbf{Dec},\textbf{ChalDec},\textbf{TOpen}}$

return win

Oracle $\textbf{Enc}(sd, N, H, M)$

$R = (r_1, \ldots, r_l) \leftarrow_\$ \mathcal{G}(sd, N, l\lambda)$

for i from 1 to n

  $C_1^i = m_i \oplus r_i$

  $h_i = \mathcal{H}(r_i, m_i)$

$C_1 = \{C_1^i\}_{i=1}^n$

$C_2 = \mathcal{H}(H, h_1, \ldots, h_l)$

$\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{N, H, C_1, C_2\}$

$D[N, H, C_1, C_2] \leftarrow (M, R)$

Return (H, C_1, C_2)

Oracle $\textbf{Dec}^*(sd, N, H, C_1, C_2)$

if $D[N, H, C_1, C_2] \neq \perp$ **then**

  return $D[N, H, C_1, C_2]$

else $R = (r_1, \ldots, r_l) \leftarrow \mathcal{G}(sd, N, l\lambda)$

  for i from 1 to n

    $m_i = C_i^i \oplus r_i$

    $h_i = \mathcal{H}(r_i, m_i)$

    if $C_2 \neq \mathcal{H}(H, h_1, \ldots, h_l)$

      return $\perp$

    else $win \leftarrow true$

    return $M = (m_1, \ldots, m_l)$ and R

Oracle $\textbf{TOpen}(sd, N, H, C_1, C_2, \varphi)$

if $D[N, H, C_1, C_2] \neq \perp$ **then**

  $(M, R) \leftarrow D[N, H, C_1, C_2]$

else $R = (r_1, \ldots, r_l) \leftarrow \mathcal{G}(sd, N, l\lambda)$

  for i from 1 to n

    $m_i = C_i^i \oplus r_i$

    $h_i = \mathcal{H}(r_i, m_i)$

  if $C_2 \neq \mathcal{H}(H, h_1, \ldots, h_l)$

    return $\perp$

  φ corresponds to positions $(i_1, \ldots, i_s)$

  else $\mathsf{win} \leftarrow \mathsf{true}$

  $S = \big(\{h_i\}_{i \in \{l\}/\{i_1,\ldots,i_j\}}, \{r_i\}_{i \in \{i_1,\ldots,i_j\}}\big)$

  return $(S, \varphi(M))$

Oracle $\textbf{ChalDec}(sd, H, C_1, C_2)$

if $(H, C_1, C_2) \in \mathcal{Y}_1$

  return $\perp$

$r_1, \ldots, r_l \leftarrow \mathcal{G}(sd, N, l\lambda)$

for i from 1 to n

  $m_i = C_i^i \oplus r_i$

  $h_i = \mathcal{H}(r_i, m_i)$

if $C_2 \neq \mathcal{H}(H, h_1, \ldots, h_l)$

  return $\perp$

else $\mathsf{win} \leftarrow \mathsf{true}$

  return (M, sd)

Fig. 7. Game G_1 for integrity proof

4.4 Construction with weaker confidentiality after opening

Definition 3 (Weak confidentiality after opening). *We define two security games for weaker confidentiality as Figure 8, where the oracle **Enc** and **Dec** are as same as that in the previous games in Figure 5. Formally, We say a nonce based block wise targeted opening ccAEAD scheme satisfies weaker confidentiality if it satisfies:*

- *the standard IND-CPA security without opening, i.e.,*

$$\Pr[\text{IND-CPA}^{\mathcal{A}}_{nTOCE} \Rightarrow 1] \leq \mathsf{negl}(\lambda);$$

- *the targeted opening one-way security, i.e.,*

$$\Pr[\textit{One-Way}^{\mathcal{A}}_{nTOCE} \Rightarrow 1] \leq \mathsf{negl}(\lambda).$$

$\text{IND-CPA}^{\mathcal{A}}_{nTOCE}$

$K \leftarrow_{\$} \mathsf{KeyGen}$
$st_1 \leftarrow \mathcal{A}^{\mathbf{Enc},\mathbf{Dec}}(1^\lambda)$
$\{N^*, H^*, (M_0, M_1), st_2\} \leftarrow_{\$} \mathcal{A}(st_1)$
$b \leftarrow_{\$} \{0,1\}$
$(H^*, C_1^*, C_2^*) \leftarrow \mathsf{Enc}_K(N^*, H^*, M_b)$
$b' \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc},\mathbf{Dec}}(C_1^*, C_2^*, st_2)$
return $b = b'$

$\text{One-Way}^{\mathcal{A}}_{nTOCE}$

$K \leftarrow_{\$} \mathsf{KeyGen}$
$(N^*, H^*, M^*) \leftarrow_{\$} \mathcal{N} \times \mathcal{H} \times \mathcal{M}$
$(N^*, H^*, C_1^*, C_2^*) \leftarrow \mathsf{Enc}(K, N^*, H^*, M^*)$
$M' \leftarrow_{\$} \mathcal{A}^{\mathbf{Enc},\mathbf{Dec},\mathbf{TOpen}^*}(N^*, H^*, C_1^*, C_2^*)$
return $M = M'$

Oracle $\mathbf{TOpen}^*(N, H, C_1, C_2, \varphi)$
if $(N, H, M) = (N^*, H^*, M^*)$
  if $\varphi = I$ **return** $\perp$
if $(N, H, C_1, C_2) \in \mathcal{Y}_1$ **return** $\perp$
$(M, K_f) \leftarrow \mathsf{Dec}(K, N, H, C_1, C_2)$
$(\varphi(M), S) \leftarrow \mathsf{TOpen}(H, M, K_f, C_2, \varphi)$
return $(m = \varphi(M), S)$

Fig. 8. The security games for weaker confidentiality

Next we will introduce a more efficient construction but with weaker confidentiality. Similarly, $\mathcal{H}$ is a hash function and $\mathcal{G}$ is a pseudo random generator.

- $\mathsf{bTOCE.KeyGen}(1^\lambda)$: Generate a seed sd for the pseudo random generator $\mathcal{G}$. The secret key K is sd.
- $\mathsf{bTOCE.Enc}(N, H, K, M)$: Given the nonce N, the secret key $K = sd$ and the message $M \in \{0,1\}^{lt}$, do the following steps:

1. Use the pseudorandom generator $\mathcal{G}$ with the seed sd and the nonce N to generate bits strings R with the size of lt, i.e.,$R = (r_0, r_1, \ldots, r_\ell) \leftarrow \mathcal{G}(sd, N, (l+1)t)$ where each $r_i \in \{0,1\}^t$.
2. Divide each M into ℓ blocks $m_1, \ldots, m_\ell$, and every block has t bits. Then use one time pad to encrypt each message m_i, i.e., $C_1^i = r_i \oplus m_i$, for $i = 1, \ldots, \ell$;
3. Hash each r_i together with m_i and get $h_i = \mathcal{H}(m_i)$;
4. Compute the final tag $C_2 = \mathcal{H}(H, h_1, \ldots, h_l, r_0)$.

The finally output ciphertext is $C_1 = \{C_1^i\}_{i=1}^l$ and the tag C_2.

– $\mathsf{bTOCE.Dec}(K, N, H, (C_1, C_2))$: Firstly, use seed $K = sd$ to recover $R = (r_1, \ldots, r_\ell)$. Then one can get $m_i = C_1^i \oplus r_i$ and $h_i = \mathcal{H}(m_i)$ for $i = 1, \ldots, l$. If $C_2 = \mathcal{H}(H, h_1, \ldots, h_l, r_0)$, output the message $M = \{m_1, \ldots, m_l\}$ and the opening $K_f = R = \{r_1, \ldots, r_l\}$, otherwise output $\perp$.
– $\mathsf{bTOCE.TOpen}(H, M, R, \varphi_t)$: If the position function φ_t denotes to open the blocks with index $i_1, \ldots, i_j$, one just compute each $h_i = \mathcal{H}(m_i)$ and output the targeted opening $S = \left\{\{h_i\}_{i \notin \{i_1, \ldots, i_j\}}, r_0\right\}$ and the opened messages $\varphi_t(M) = \{m_i\}_{i \in \{i_1, \ldots, i_j\}}$.
– $\mathsf{bTOCE.TVer}(H, \varphi_t(M), S, C_2, \varphi_t)$: If the position function φ_t denotes to open the blocks with index $i_1, \ldots, i_j$, one parses the targeted opening S as $\{h_i\}_{i \notin \{i_1, \ldots, i_j\}}$ and r_0, and $\varphi_t(M)$ as $\{m_i\}_{i \in \{i_1, \ldots, i_j\}}$. Then compute $h_i = \mathcal{H}(m_i)$ for $i \in \{i_1, \ldots, i_j\}$ and check whether $C_2 = \mathcal{H}(H, h_1, \ldots, h_l, r_0)$. If the check is passed, output 1 otherwise output 0.

Security Analysis. One can easily prove that the wbTOCE scheme satisfies the weak confidentially in Definition 3 in the random oracle model.

References

1. Dario Catalano and Dario Fiore. Vector commitments and their applications. In *Public-Key Cryptography–PKC 2013*, pages 55–72. Springer, 2013.
2. Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In *Annual International Cryptology Conference*, pages 155–186. Springer, 2018.
3. Facebook. Facebook messenger app, 2016. https://www.messenger.com/.
4. Facebook. Messenger secret conversations technical whitepaper, 2016. https://fbnewsroomus.files.wordpress.com/2016/07/secret conversations whitepaper-1.pdf.
5. Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In *Annual International Cryptology Conference*, pages 66–97. Springer, 2017.
6. Iraklis Leontiadis and Serge Vaudenay. Private message franking with after opening privacy. Cryptology ePrint Archive, Report 2018/938, 2018. https://eprint.iacr.org/2018/938.
7. Beno2̂t Libert, Somindu Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, 2016.