An extended abstract [CRT24] of this article appears in the proceedings of the 31st ACM Conference on Computer and Communications Security (ACM CCS 2024). This is the full version.

On the Tight Security of the Double Ratchet

Daniel Collins[1,2], Doreen Riepel[3], Si An Oliver Tran[4]

[1] Purdue University
[2] Georgia Institute of Technology
[3] UC San Diego
[4] ETH Zurich
danielpatcollins@gmail.com, doreen.riepel@gmail.com, sitran@student.ethz.ch

Abstract. The Signal Protocol is a two-party secure messaging protocol used in applications such as Signal, WhatsApp, Google Messages and Facebook Messenger and is used by billions daily. It consists of two core components, one of which is the Double Ratchet protocol that has been the subject of a line of work that aims to understand and formalise exactly what security it provides. Existing models capture strong guarantees including resilience to state exposure in both forward security (protecting past secrets) and post-compromise security (restoring security), adaptive state corruptions, message injections and out-of-order message delivery. Due to this complexity, prior work has failed to provide security guarantees that do not degrade in the number of interactions, even in the single-session setting.

Given the ubiquity of the Double Ratchet in practice, we explore tight security bounds for the Double Ratchet in the multi-session setting. To this end, we revisit the modelling of Alwen, Coretti and Dodis (EUROCRYPT 2019) who decompose the protocol into modular, abstract components, notably continuous key agreement (CKA) and forward-secure AEAD (FS-AEAD). To enable a tight security proof, we propose a CKA security model that provides one-way security under key checking attacks. We show that multi-session security of the Double Ratchet can be tightly reduced to the multi-session security of CKA and FS-AEAD, capturing the same strong security guarantees as Alwen et al.

Our result improves upon the bounds of Alwen et al. in the random oracle model. Even so, we are unable to provide a completely tight proof for the Double Ratchet based on standard Diffie-Hellman assumptions, and we conjecture it is not possible. We thus go a step further and analyse CKA based on key encapsulation mechanisms (KEMs). In contrast to previous works, our new analysis allows for tight constructions based on the DDH and post-quantum assumptions.

1 Introduction

The Signal Protocol is the de-facto standard two-party end-to-end encrypted messaging protocol [EM19] used in applications like Signal [Mar16], WhatsApp [Wha23], Google Messages [Goo22] and Facebook Messenger [Met17]. It consists of two core components, namely (1) the Extended Triple Diffie-Hellman (X3DH) key exchange protocol to bootstrap conversations,[5] after which keys are continually updated during message exchange via (2) the Double Ratchet protocol. The Double Ratchet, the focus of this work, provides strong guarantees under state exposure, in particular forward security (protecting past secrets) and post-compromise security (restoring security given a passive adversary). Its security has been extensively studied in the literature, starting from the work of Cohn-Gordon et al. [CCD+20] that analyses its key schedule, followed by a line of work that model the Double Ratchet as a messaging protocol in both game-based [ACD19] and simulation-based models [BFG+22,CJSV22].

Secure Messaging as a Cryptographic Primitive. Alwen, Coretti and Dodis [ACD19], in the following denoted by ACD19, provide a formal definition of a secure messaging (SM) scheme that captures the security properties that the Double Ratchet provides. In ACD19, this is captured in a monolithic, game-based model. Here, the adversary is given significant power, including the power

[5] Recently, Signal deployed their transitional PQXDH key exchange protocol that additionally provides post-quantum confidentiality [KS23].

to adaptively corrupt protocol participants, inject and drop messages and manipulate parties' randomness. Security covers three main properties, namely correctness, privacy and authenticity, where the latter two guarantees are provided insofar as state exposure does not trivially violate them.

Due to the high complexity of the security model, ACD19 modularise a Double Ratchet-like protocol and provide abstractions and (simpler) security definitions for each building block. Firstly, a continuous key agreement (CKA) scheme, which captures the asymmetric ratchet of the Double Ratchet and generalises its continuous Diffie-Hellman key exchange, each iteration marking a different *epoch* in the Double Ratchet. Secondly, a forward-secure authenticated encryption scheme with associated data (FS-AEAD), capturing the symmetric ratchet associated with each epoch, which provides confidentiality, privacy and forward security. Finally, a two-input hash function (PRF-PRNG) which is fed the output of the CKA as it iterates each epoch and seeds the initialisation for each instance of FS-AEAD. Overall, this modularity makes the task of modifying the protocol easier, for example for post-quantum security.

Taking a closer look at the concrete security bounds of the ACD19 (and similar) proofs, we observe that the security reductions are non-tight. As an example, consider an adversary that breaks the Double Ratchet protocol with advantage $\epsilon_{\mathcal{A}}$, then the proof constructs an adversary that breaks security of the underlying CKA scheme with advantage $\epsilon_{\mathcal{B}} = \epsilon_{\mathcal{A}}/(q_e^2)$, where q_e is the number of epochs. This means security (in the single-session setting) already degrades by a factor q_e^2. When generically extending the proof to the multi-session setting via a hybrid argument, we get an additional loss that is linear in the number of sessions.

Why Multi-Session Security Matters. For traditional key exchange, multi-session security was considered already in the first formal security models [BR94,BR95], capturing that in practice these protocols are executed by billions of users. While we observe non-tightness there as well, it has motivated the study of tight security for real-world protocols like SIGMA and TLS 1.3 [DJ21,DG21,DDGJ22] to justify how these protocols are implemented and used in practice.

With secure messaging being a very active research field for the past few years, the focus has been on modelling and achieving very strong security rather than tight security proofs. However, just like TLS, we use the Signal protocol daily. To estimate the number of sessions in practice, consider that WhatsApp has almost 3 billion ($\approx 2^{31}$) monthly active users alone.[6] In addition to that, it is the case that in practice several instances of the Double Ratchet may be spun up in the lifetime of a conversation by the higher level session management module [CJN23]. The Double Ratchet is also used in the Sender Keys group messaging protocol used by WhatsApp and Signal to exchange and update keying material, where all pairs of users keep a Double Ratchet session alive to send updated keying material [BCG23], so users can often have hundreds or more active sessions from group chats alone. These applications indicate that the security loss is in fact significant and it suggests to adapt system parameters to take it into account. In Figure 1 we provide some estimates for the Double Ratchet protocols as well as variants we consider in this paper.

With the above in mind, we aim to address the following two questions, namely (1) whether the security loss in previous statements is merely a matter of proof techniques and (2) what are the requirements to actually achieve tight security? To investigate the first question, we briefly outline the overall approach of ACD19's security proof. (In fact, other game-based security proofs for SM schemes, such as [CZ24], follow a similar approach). The main difficulty is, as mentioned above, that security models are highly complex and previous proofs simplify these models by considering correctness, privacy and authenticity as separate goals. This way, they are able to show that strong SM security can be achieved by combining comparably weak security definitions for each of them. While this simplifies the analysis of each property, this approach comes at a cost, namely that independent of how the scheme is instantiated, one will always inherit the security loss, even if a tighter proof may exist.

1.1 Contributions

Our starting point is the security model of ACD19. First, we extend ACD19's definitions to capture the more realistic multi-session setting. Then, we turn to the building blocks of secure messaging

[6] https://www.statista.com/statistics/1306022/whatsapp-global-unique-users/

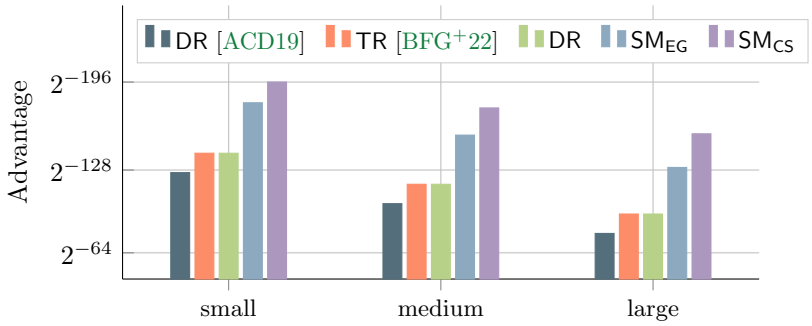

Fig. 1: Advantage bounds for Diffie-Hellman based messaging schemes for three different levels of (adversarial) resources when instantiated over a 256-bit elliptic curve. DR stands for Double Ratchet and TR for Triple Ratchet. $\mathsf{SM}_{\mathsf{EG}}$ and $\mathsf{SM}_{\mathsf{CS}}$ are messaging schemes based on ElGamal and Cramer-Shoup respectively. Bars with references refer to the bounds in the respective paper, whereas the others are from our analysis. A more detailed description (including bounds for a 384-bit curve) can be found in Section 6.

schemes, namely CKA and FS-AEAD, and revisit their security definitions. Our goal is to give security definitions which are strong enough yet as weak as possible such that their composition allows for a *tight* security proof. Finally, we (re)prove security of Signal's Double Ratchet scheme. Since, unfortunately, our security bounds are still non-tight (except in the generic group model), the reasons for which we discuss in the main body of the paper, we propose and analyse new schemes. Our results and a comparison to related works, mainly that of ACD19 and BFGMR22 [BFG+22], are given in Figure 2. We now detail our contributions, additionally providing insights about technical hurdles and an interpretation of our results.

New Definitions Enabling Tight Composition. The extension of the SM security model of ACD19 to the multi-session setting is natural and straightforward. We allow the adversary to run multiple sessions in parallel and for each of them adaptively corrupt protocol participants, control the message flows and manipulate parties' randomness. The main technical difficulty lies in defining security for the main building blocks such that tightness is preserved and we can still find efficient instantiations. Due to the adaptive nature of the model, this is however non-trivial. It is not initially known whether an adversary decides to corrupt a party's secret state or whether it will ask for a challenge. This gives rise to the so-called *commitment problem* [Nie02]. Namely, a security reduction must be able to provide consistent answers to all of the adversary's queries, or it will incur a tightness loss. In other words, it must *commit* to which secret values it knows and where it will embed the instance of a computational problem. In order to resolve the commitment problem, we draw from observations made in the context of tight security for the authenticated key exchange protocols. Most interesting for our goal are those works studying tightness of Diffie-Hellman key exchange [CCG+19] and generic constructions from key encapsulation mechanisms (KEMs) [JKRS21,PWZ23a].

In order to construct SM schemes, we first turn to the underlying continuous key agreement scheme. We define a one-way security game, where the adversary has to recover the CKA key. This definition, as opposed to the "standard" indistinguishability based security definition, allows us to avoid the commitment problem by additionally relying on the random oracle model.

Equipped with our CKA security notion, we then extend the FS-AEAD security model of ACD19 to the multi-instance setting. Using CKA and FS-AEAD, we then show tight security of ACD19's generic scheme in a multi-session setting.

Our Results for the Double Ratchet. The good news is that we can improve upon the ACD19 bound for Signal's Double Ratchet instantiation. ACD19 proves security with a loss of q_e^2 based on the Decisional Diffie-Hellman (DDH) assumption, where q_e is the number of epochs. In an n-session setting, this becomes a loss of nq_e^2. Our result "only" incurs a tightness loss of nq_e and is based on the strong Computational Diffie-Hellman (StCDH) assumption. A similar bound was

Scheme	Δ_{SM}	Comm. (asym)	Security Notion	Multi-User	Security Loss	Assumption (asym)	Model
DR [ACD19]	3	$\lvert\mathbb{G}\rvert$	SM	✗	$\mathrm{O}(nq_e^2)$	DDH	Standard
DR [BFG+22]	3	$\lvert\mathbb{G}\rvert$	$\mathcal{F}_{\mathsf{DR}}$	✗	$\mathrm{O}(nq_e)$	StSqCDH	ROM, ICM
TR [BFG+22]	2	$\lvert\mathbb{G}\rvert$	$\mathcal{F}_{\mathsf{TR}}$	✗	$\mathrm{O}(nq_e)$	StCDH	ROM, ICM
DR (this work)	3	$\lvert\mathbb{G}\rvert$	n-SM	✓	$\mathrm{O}(nq_e)$	StCDH	ROM
DR (this work)	3	$\lvert\mathbb{G}\rvert$	n-SM	✓	$\mathrm{O}(1)$	nq_e-A-StCDH-Corr	ROM
$\mathsf{SM}_{\mathsf{EG}}$ [ACD19]	2	$2\lvert\mathbb{G}\rvert$	SM	✗	$\mathrm{O}(nq_e^2)$	DDH	Standard
$\mathsf{SM}_{\mathsf{EG}}$ (this work)	2	$2\lvert\mathbb{G}\rvert$	n-SM	✓	$\mathrm{O}(q_C)$	StCDH	ROM
$\mathsf{SM}_{\mathsf{CS}}$ (this work)	2	$3\lvert\mathbb{G}\rvert$	n-SM	✓	$\mathrm{O}(1)$	DDH	ROM
$\mathsf{SM}_{\mathsf{KEM}}$ [ACD19]	2	$\lvert\mathsf{pk}\rvert+\lvert\mathsf{ct}\rvert$	SM	✗	$\mathrm{O}(nq_e^2)$	CPA	Standard
$\mathsf{SM}_{\mathsf{KEM}}$ (this work)	2	$\lvert\mathsf{pk}\rvert+\lvert\mathsf{ct}\rvert$	n-SM	✓	$\mathrm{O}(q_E)$	OW-PCA	ROM
$\mathsf{SM}_{\mathsf{KEM}}$ (this work)	2	$\lvert\mathsf{pk}\rvert+\lvert\mathsf{ct}\rvert$	n-SM	✓	$\mathrm{O}(1)$	q_E-OW-PCA-Corr	ROM
eSM [CZ24]	2	$\lvert\mathsf{pk}\rvert+\lvert\mathsf{vk}\rvert+$ $6\lvert\mathsf{ct}\rvert+2\lvert\sigma\rvert$	eSM	✗	$\mathrm{O}(nq_e^3q_m)$	CCA+SUF	Standard

Fig. 2: Comparison of our results with previous work. Δ_{SM} refers to how fast parties recover from state compromise. Column "Comm." lists the communication complexity of asymmetric primitives. SM is the security model of ACD19, where n-SM is its generalization to the multi-session setting. eSM is an extension of SM with more fine-grained security and $\mathcal{F}_{\mathsf{DR}}/\mathcal{F}_{\mathsf{TR}}$ refer to the ideal functionalities from [BFG+22]. Since previous proofs are all in the single-session setting, the column "Security Loss" adds an additional factor n (the number of sessions) to the original bounds. q_e is the number of epochs per session, $q_E \leq nq_e$ is the total number of epochs across all sessions, $q_C \leq q_E$ is the total number of corruptions, and q_m is the number of pre-keys in the eSM model. The security loss refers to asymmetric primitives only. For simplicity, we assume multi-user security for symmetric primitives.

already achieved by BFGMR22 who prove security in the UC model. However, their proof relies on the strong Square Diffie-Hellman assumption (StSqCDH), which is stronger than StCDH and only non-tightly implies StCDH (cf. [MW96]).

The question remains whether our bound is indeed optimal. Unfortunately, while our results are a strong indication towards that, there are technical hurdles towards a formal impossibility result. More specifically, it appears that current techniques, such as those from [BJLS16,CCG+19], do not allow to prove such a result. On a positive note, when restricting to generic attacks (i. e., those that only exploit the group structure and in particular not the representation), our new modular approach directly gives us information-theoretic lower bounds in that setting, using a recent result from [KPRR23].

New SM Constructions with Tight Security. In light of a potential impossibility result, we leverage our analysis to find alternative constructions that do achieve tight security. To this end, we look at generic CKA constructions from KEMs. We use multi-user one-way secure KEMs in the presence of adaptive corruptions and a key checking oracle. A similar definition was used by Pan, Wagner and Zeng [PWZ23a] to construct tightly-secure key exchange from lattice assumptions. Our definition is strictly weaker than theirs and potentially allows for more efficient constructions. As their key exchange, our final construction also relies on the random oracle model.

1.2 Related Work

Comparison with Related Models. The Double Ratchet was first fully examined formally by Cohn-Gordon et al. [CCD+20], who analyse its key schedule. As mentioned, ACD19 takes a game-based approach. [BFG+22] and [CJSV22] concurrently considered simulation-based security guarantees for the Double Ratchet; the model of [BFG+22] is generally stronger than that of ACD19, whereas [CJSV22] does not allow randomness manipulation and so at least in that regard is weaker. ACD19 describe and sketch, but do not prove secure, a protocol that provides additional fine-grained

security guarantees by additionally using public-key encryption and signatures even in the "symmetric" ratchet. Recently, Cremers and Zhao [CZ24] formalised and extended this protocol; their formalism extends that of ACD19 and additionally captures properties like deniability and the existence of long-term keys within messaging sessions. We discuss [BFG+22] and [CZ24] further in Sections 7.2 and 7.3, respectively. [DG19] and [DH23] study continuous key agreement (CKA), the former proposing an efficient code-based CKA and a CKA combiner, and the latter for active attack detection on the CKA layer. ASMesh [BRT23] is a mesh messaging protocol with a particular focus on anonymity, while maintaining confidentiality and strong post-compromise and forward secrecy. Their underlying CKA builds upon ACD19, while the final model is simulation-based. In [Ste24], Stebila analyzes Apple's iMessage PQ3 protocol in a multi-stage key exchange security model. The protocol adds a post-quantum secure key encapsulation mechanism to both the initial key exchange and the asymmetric ratchet. Both ASMesh and PQ3 have comparably loose bounds as the eSM scheme in Figure 2.

Additional Related Work. For two-party messaging more generally, a line of work initiated by Bellare et al. [BSJ+17] examines the theoretical security of a messaging protocol and provide trade-offs between performance and security, sometimes inherently relying on strong, HIBE-like primitives [JS18,PR18,BRV20] or being weaker and therefore more efficient than Double Ratchet. A comparable line of work under the umbrella of continuous group key agreement (CGKA), the core component in the recent IETF MLS messaging standard [ACDT21], that studies similar questions [ACDT20,ACJM20,BDG+22]. On multi-session security in messaging, [CHK21] considers the effects on security *between* instances, and [AAB+21] on improving performance. To the best of our knowledge, however, tightness has not been considered as a first-class goal in the literature on modern secure messaging, and many proofs are very non-tight, like that of [ACDT21] which incurs at least q^4 tightness loss in the number of queries q the adversary makes, even for a single group. In the context of traditional authenticated key exchange protocols, recent works aimed at giving tight(er) proofs for TLS 1.3 [DJ21,DG21], whereas other works focused on efficient constructions in the random oracle model [GJ18,CCG+19,JKRS21,PWZ23a] as well as in the standard model [BHJ+15,HLG21,HJK+21].

2 Preliminaries

Notation. For integers n, $m > n$, we denote $[n, m] = \{n, \ldots, m\}$. For $[1, n]$ we simply write $[n]$. For s, t we denote $s \leftarrow t$ the assignment of t to s. Similarly, $s_1, \ldots, s_n \leftarrow t$ stands for the assignment of t to $s_1, \ldots, s_n$ and $(s_1, \ldots, s_n) \leftarrow (t_1, \ldots, t_n)$ denotes the element-wise assignment. Sometimes we set a variable t_i to $*$ when the assignment is syntactically required but the value is not important. Also we sometimes use **return** $s \leftarrow t$ to first assign t to s and then returning s. For an integer i, the notation i++ means $i \leftarrow i + 1$. For a finite set X, we write $x \leftarrow\!\!\$\ X$ as sampling a uniformly random element from X. We write $X \xleftarrow{+} x$ and $X \xleftarrow{-} x$ for $X \leftarrow X \cup \{x\}$ and $X \leftarrow X \backslash \{x\}$, respectively. $\perp$ and λ denote two types of "empty" value; λ is used for assignment exclusively and $\perp$, which can also be used in an assignment, is used as return value of algorithms. If A is a (probabilistic) algorithm, then $y \leftarrow\!\!\$\ A(x)$ denotes running A on x and assigning the output to y. Sometimes we will make the random coins explicit and write $y \leftarrow A(x; r)$. An adversary is a probabilistic algorithm and we write A^{O} to indicate that A has oracle access to O.

Let $\mathcal{D}$ be a dictionary. For ease of notation, we use an *array-notation*. It is important to note they are to be implemented by a data structure whose size grows (linearly) with the number of elements in the dictionary (unlike arrays). We write $\mathcal{D}[\cdot] \leftarrow \lambda$ to initialise $\mathcal{D}$ as empty dictionary. The operation $\mathcal{D}[i] \leftarrow v$ stores v/overwrites the value at entry i; if $v = \lambda$, then this amounts to deleting the stored element. The operation $\mathcal{D}[i]$ returns the value at entry i; the returned value can also be the empty value λ.

Game Conventions. Throughout the paper we use code-based games, where $\Pr[\mathsf{G} \Rightarrow 1]$ denotes the probability that the final output of game G is 1. We use special keywords that simplify oracle specifications (for the reader) and enhance readability. The instruction **req** [*condition*] checks if *condition* is satisfied and if this is not the case, then the oracle/algorithm exits and all actions

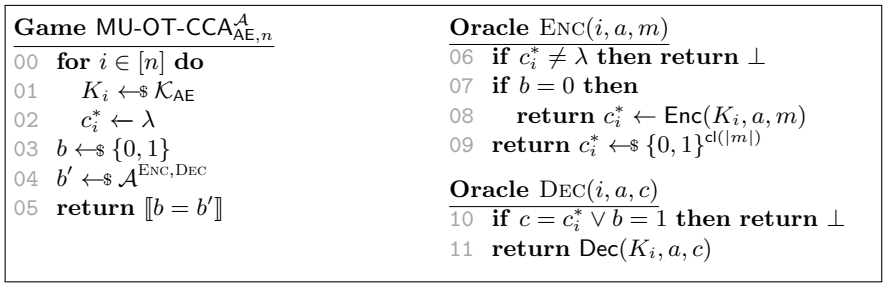

Fig. 3: Multi-instance game MU-OT-CCA for an authenticated encryption scheme AE.

performed by it are undone. If at some point during the game the flag $\mathbf{win_e}$ for event **e** is set to **true**, then the execution stops and attacker wins the game immediately. For a Boolean statement B, the notation $[\![\text{B}]\!]$ refers to a bit that is 1 if the statement is true and 0 otherwise.

Groups. Throughout the paper, let $\mathcal{G} = (\mathbb{G}, p, g)$ be the description of a cyclic group of prime order p with generator g.

Definition 1 (StCDH). *The strong computational Diffie-Hellman* (StCDH) *problem is defined relative to a group $\mathcal{G} = (\mathbb{G}, p, g)$. We define the advantage of an adversary $\mathcal{A}$ against* StCDH *as*

$$\mathbf{Adv}^{\mathsf{StCDH}}_{\mathcal{G}}(\mathcal{A}) \coloneqq \Pr[\mathcal{A}^{\text{DDH}}(\mathcal{G}, g^x, g^y) \Rightarrow g^{xy}] \ ,$$

where $x, y \leftarrow\$ \mathbb{Z}_p$ and $\text{DDH}(a, \cdot, \cdot)$ for $a \in \{x, y\}$ is an oracle that on input (Y, Z) returns a Boolean whether $Z = Y^a$.

Definition 2 (DDH). *The decisional Diffie-Hellman* (DDH) *problem is defined for a group $\mathcal{G} = (\mathbb{G}, p, g)$. We define the advantage of an adversary $\mathcal{A}$ against* DDH *as*

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{G}}(\mathcal{A}) \coloneqq \ |\Pr[\mathcal{A}(\mathcal{G}, g^x, g^y, g^{xy}) \Rightarrow 1] - \Pr[\mathcal{A}(\mathcal{G}, g^x, g^y, g^z) \Rightarrow 1]| \ ,$$

where $x, y, z \leftarrow\$ \mathbb{Z}_p$.

Authenticated Encryption. An AEAD scheme defines a key space $\mathcal{K}_{\mathsf{AE}}$ and a tuple of algorithms $\mathsf{AE} = (\mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

- Enc takes a key $K \in \mathcal{K}_{\mathsf{AE}}$, associated data a and a message m as input, and produces a ciphertext $c \leftarrow \mathsf{Enc}(K, a, m)$.
- Dec takes a key $K \in \mathcal{K}_{\mathsf{AE}}$, associated data a and a ciphertext c as input, and produces a plaintext $m \leftarrow \mathsf{Dec}(K, a, c)$ or a failure symbol $\perp$.

Note that we assume all randomness stems from the key K. We do not need additional randomness or nonces because we will only rely on one-time security (see below). We say an AEAD scheme is *correct* if for all K, a, m it holds that $\mathsf{Dec}(K, a, \mathsf{Enc}(K, a, m)) = m$.

Security. We consider one-time CCA security for AE in the multi-user setting. For this, we let $\mathsf{cl}(|m|)$ be the ciphertext length function. The challenge oracle in the security game takes as input a message (and associated data) and will either output the encryption of that message or a random ciphertext of the same length.

Definition 3. *Consider game* MU-OT-CCA *in Figure 3 for an AEAD scheme* AE*, positive integer n and an adversary $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in this game as*

$$\mathbf{Adv}^{n\text{-OT-CCA}}_{\mathsf{AE}}(\mathcal{A}) \coloneqq 2 \cdot \left|\Pr[\text{MU-OT-CCA}^{\mathcal{A}}_{\mathsf{AE},n} \Rightarrow 1] - 1/2\right| \ .$$

We will also provide an instantiation from KEMs and the DDH assumption, and so we define appropriate definitions for KEM below.

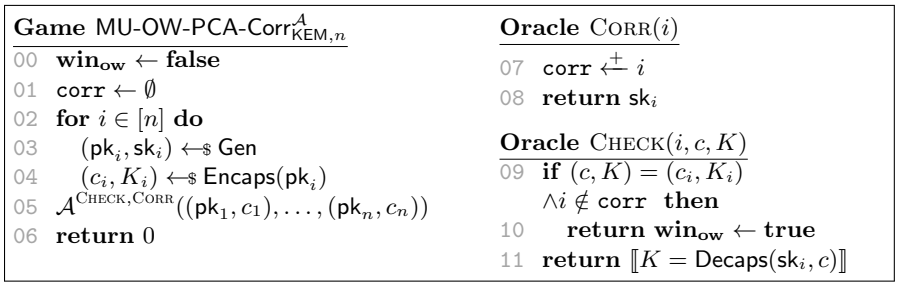

Fig. 4: Multi-instance game MU-OW-PCA-Corr for KEM.

Key Encapsulation Mechanisms (KEMs) A KEM scheme $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ specifies a key space $\mathcal{K}_{\mathsf{KEM}}$ and consists of the following three algorithms

- Gen outputs a pair of public key pk and secret key sk.
- $\mathsf{Encaps}(\mathsf{pk})$ takes as input a public key pk and returns a ciphertext c and a key $K \in \mathcal{K}_{\mathsf{KEM}}$, where c is an encapsulation of K.
- $\mathsf{Decaps}(\mathsf{sk}, c)$ takes as input a secret key sk and a ciphertext c and outputs a key $K \in \mathcal{K}_{\mathsf{KEM}}$ or a special failure symbol $\perp$.

Correctness. A KEM scheme KEM is (perfectly) correct if for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$, $(c, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, we have $\mathsf{Decaps}(\mathsf{sk}, c) = K$.

Security. In Figure 4 we define multi-user one-way security under key checking attacks and corruptions. In this game, the adversary gets as input n public keys and a challenge ciphertext for each of them. It may corrupt users adaptively and its task is to compute the KEM key for an uncorrupted user. It may also check whether a key belongs to a ciphertext using a Check oracle. This is captured in the following definition.

Definition 4. *Consider game* MU-OW-PCA-Corr *in Figure 4 for a KEM scheme* KEM*, positive integer n and an adversary $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in this game as*

$$\mathbf{Adv}^{n\text{-OW-PCA-Corr}}_{\mathsf{KEM}}(\mathcal{A}) := \Pr[\mathsf{MU\text{-}OW\text{-}PCA\text{-}Corr}^{\mathcal{A}}_{\mathsf{KEM},n} \Rightarrow 1] \ .$$

Further, we let OW-PCA *be the single-user variant of the game and the advantage be defined analogously.*

Remark 1. A stronger variant of this notion has been introduced in [PWZ23a], where there are multiple challenges for each public key and the adversary additionally has access to a decryption oracle. Thus, their notion tightly implies the above.

3 Multi-Instance Continuous Key Agreement

Continuous key agreement (CKA) is an abstraction that captures the asymmetric ratchet of the Double Ratchet, i.e., generalises its continuous Diffie-Hellman key exchange. In CKA, parties iteratively establish new keys in a ping-pong fashion, alternating between theirs role as senders and receivers, each notion. These keys are then used by the symmetric ratchet of the protocol, captured in the next section on FS-AEAD. CKA is expected to provide resilience to state exposure due intuitively to the injection of new randomness into the protocol session that an attacker does not have access to.

Syntax. A continuous key agreement scheme CKA defines an initialisation key space $\mathcal{K}^{\mathsf{init}}_{\mathsf{CKA}}$, a CKA key space $\mathcal{K}_{\mathsf{CKA}}$ and a quadruple of algorithms $\mathsf{CKA} = (\mathsf{CKA\text{-}Init\text{-}A}, \mathsf{CKA\text{-}Init\text{-}B}, \mathsf{CKA\text{-}S}, \mathsf{CKA\text{-}R})$, with the following syntax:

– CKA-Init-A (resp. CKA-Init-B) takes as input an initialisation key $k^{\mathsf{AB}} \in \mathcal{K}^{\mathsf{init}}_{\mathsf{CKA}}$, and produces an initial state γ^{A} (resp. γ^{B}).
– CKA-S takes a state γ as input, and produces a new state, a message, and a key $(\gamma', m, k) \leftarrow\!\!\$\ \mathsf{CKA\text{-}S}(\gamma)$.
– CKA-R takes a state γ and a message m as input, and produces a new state and a key $(\gamma', k) \leftarrow \mathsf{CKA\text{-}R}(\gamma, m)$.

Correctness. A CKA scheme CKA is (perfectly) correct[7] if for all $k^{\mathsf{AB}} \in \mathcal{K}^{\mathsf{init}}_{\mathsf{CKA}}$:

$$\Pr\left[\begin{array}{c|l} k_e = k'_e & (\gamma^{\mathsf{A}}_e, m_e, k_e) \leftarrow\!\!\$\ \mathsf{CKA\text{-}S}(\gamma^{\mathsf{A}}_{e-1}) \\ \wedge\ k_{e+1} = k'_{e+1} & (\gamma^{\mathsf{B}}_e, k'_e) \leftarrow \mathsf{CKA\text{-}R}(\gamma^{\mathsf{B}}_{e-1}, m_e) \\ & (\gamma^{\mathsf{B}}_{e+1}, m_{e+1}, k_{e+1}) \leftarrow\!\!\$\ \mathsf{CKA\text{-}S}(\gamma^{\mathsf{B}}_e) \\ & (\gamma^{\mathsf{A}}_{e+1}, k'_{e+1}) \leftarrow \mathsf{CKA\text{-}R}(\gamma^{\mathsf{A}}_e, m_{e+1}) \end{array}\right] = 1\ ,$$

where $\gamma^{\mathsf{A}}_0 \leftarrow\!\!\$\ \mathsf{CKA\text{-}Init\text{-}A}(k^{\mathsf{AB}}), \gamma^{\mathsf{B}}_0 \leftarrow\!\!\$\ \mathsf{CKA\text{-}Init\text{-}B}(k^{\mathsf{AB}})$ and we iterate over $e \in \{1, 3, \ldots\}$ using the assignment of the previous iteration in the next one.

3.1 Multi-Instance Security Game

The multi-instance CKA security game presented below, adapted from the single-instance CKA security notion from ACD19, is parameterised by Δ, the minimum number of epochs that need to pass after which the state cannot be used to trivially decrypt the challenge.

We give the security game in Figure 5. Similarly to ACD19, we require the attacker to be passive. However, we will use a one-way security definition. That is, conditioned on the transcript messages $m_1, m_2, \ldots$, a CKA scheme must ensure that the keys $k_1, k_2, \ldots$ cannot be computed by an attacker. This is in contrast to indistinguishability, which requires that keys look uniformly random. However, the adversary may control the random coins r used by the sender or leak the current state of a user in a given instance. Keys k_i generated under such conditions are not required to satisfy the above property. In addition to that, our game provides a key checking oracle, which allows the adversary to test whether a key belongs to a specified epoch.

Since we consider n instances which are initialised at the beginning of the game, each oracle takes an additional index i to specify the targeted instance.

Definition 5. *Consider game* MI-OW-CKA *in Figure 5 for a CKA scheme* CKA*, non-negative integer Δ, positive integer n and adversary $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in this game as the probability that flag* $\mathbf{win_{ow}}$ *is set to true, i. e.,*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA},\Delta}(\mathcal{A}) \coloneqq \Pr[\mathbf{win_{ow}}]\ .$$

Comparison to [ACD19]. The (single-instance) CKA security game of ACD19 requires the adversary to announce the challenge epoch using an oracle Init. This, together with conditions when state exposure is possible, allows the inference about possible other epochs in which the corruption oracles can be queried. While this simplifies their CKA proof (and, in fact, makes it tight), the tightness loss is deferred to the proof of the secure messaging scheme. Thus, using their main theorem, a tight bound cannot be achieved even in the single-session model and independent of the CKA construction. Our definition is weaker in the sense that it is a one-way notion, however, it is also stronger in that it considers multiple instances.

Further Comparison. The CKA security game presented in [BFG+22] is a more fine-grained version of that in ACD19, where the predicates differentiate between corruptions and adversarially chosen randomness. We discuss it in more detail in Section 7.2. They use this to prove security of the Double Ratchet (and their Triple Ratchet) protocol in the UC model. A similar approach is taken in [BRT23]. In terms of tight security, the overall proof strategies are however similar to that of ACD19 and the tightness loss occurs in the transformation from CKA to secure messaging. We thus leave a formal study of achieving tight security in the UC model for future work.

[7] Following ACD19, we will consider perfect correctness. Note that capturing non-perfect correctness can be defined by including a correctness condition in the security experiment (as we do for the secure messaging primitive (cf. Figure 12), or to quantify over all messages (modelling worst-case correctness).

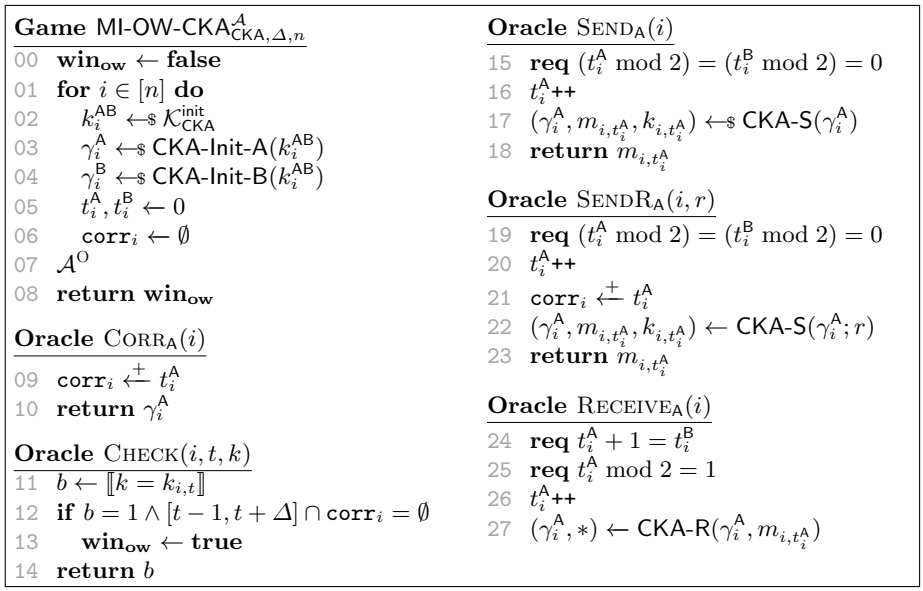

Fig. 5: Multi-instance checkable one-way security game MI-OW-CKA for a CKA scheme, where $\mathcal{A}$ has access to oracles $\mathrm{O} = \{\textsc{Send}_{\mathsf{P}}, \textsc{SendR}_{\mathsf{P}}, \textsc{Corr}_{\mathsf{P}}, \textsc{Receive}_{\mathsf{P}}, \textsc{Check}\}_{\mathsf{P}\in\{\mathsf{A},\mathsf{B}\}}$. The oracles for role B are similarly defined, with reversed roles and counter requirements.

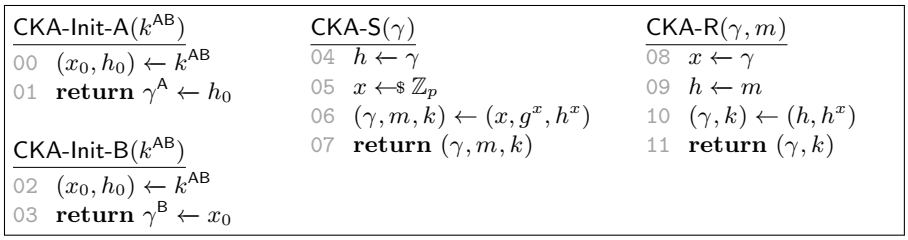

Fig. 6: CKA scheme $\mathsf{CKA}_{\mathsf{DH}}$. The initialisation key space is $\mathcal{K}^{\mathsf{init}}_{\mathsf{CKA}} = \{(x,h) \mid x \in \mathbb{Z}_p, h = g^x\}$.

3.2 Signal's CKA Scheme

We want to revisit the scheme analyzed in ACD19 which is based on the current implementation of Signal's Double Ratchet [Sig24] and the recommendation from Signal's technical documentation [Mar16].[8] The CKA scheme used in the actual protocol flow is based on the Diffie-Hellman key exchange instantiated over a cyclic group $\mathbb{G}$ of prime order p with generator $g \in \mathbb{G}$. We will denote the scheme by $\mathsf{CKA}_{\mathsf{DH}}$ and provide a description in Figure 6. The initial shared state $k = (x_0, h_0)$ consists of a random value $x_0 \leftarrow\!\!\$\ \mathbb{Z}_p$ and the corresponding group element $h_0 = g^{x_0}$. The party in role A who holds h runs the CKA-S algorithm: It picks a random element x_1, computes h^{x_1}, stores x_1 as its new state and sends $h_1 = g^{x_1}$. The party in role B runs the CKA-R algorithm: It gets h_1, computes $h_1^{x_0}$ and stores h_1 as its new state. Now this party will become the sender. It is easy to see that $\mathsf{CKA}_{\mathsf{DH}}$ satisfies correctness.

We will prove the security of $\mathsf{CKA}_{\mathsf{DH}}$ in the multi-instance setting. To simplify the analysis, we introduce a multi-instance variant of CDH that supports corruptions which we will introduce in the next paragraph. In particular, this assumption allows a tight reduction.

Multi-User CDH with Corruptions. The assumption that we use is a multi-user variant of the strong Diffie-Hellman assumption, denoted MU-A-StCDH-Corr, and is given in Figure 7, where

[8] As mentioned in [BFG+22], the specification suggests that the scheme should achieve a (slightly) stronger security notion but that this is not the case (refer Section 1.4 of [BFG+22]). The authors thus propose a protocol called the Triple Ratchet and stronger models (both for CKA and the ideal functionality for the protocol itself). We leave a study of tight bounds in their model for future work, but provide some intuition in Section 7.2.

A stands for "adjacent". A similar (but strictly stronger) assumption was given in [KPRR23] to analyze authenticated key exchange protocols. Ours mimics the CKA scheme in that the adversary has to compute the Diffie-Hellman secret for one pair of adjacent (consecutive) group elements. For concreteness, we only allow queries to the DDH oracle for such pairs, hence one index is enough. Further, we also allow the adversary to corrupt exponents which will be necessary to answer corruptions in the CKA experiment. We define our assumption and also show its relation to the standard StCDH assumption.

Definition 6. *Consider game* MU-A-StCDH-Corr *in Figure 7 for $\mathcal{G}$, integer n and adversary $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in this game as*

$$\mathbf{Adv}^{n\text{-A-StCDH-Corr}}_{\mathcal{G}}(\mathcal{A}) := \Pr[\text{MU-A-StCDH-Corr}^{\mathcal{A}}_{\mathcal{G},n} \Rightarrow 1] \ .$$

In order to give a non-tight bound for $\mathsf{CKA}_{\mathsf{DH}}$ from a standard (single-user) assumption, we first reduce the security of MU-A-StCDH-Corr to the more standard StCDH assumption. This relation is known in the literature for stronger multi-user variants and we only repeat it for completeness. Then we can directly apply it to Theorem 1 which is given below.

Lemma 1. *Let n be a positive integer. For any adversary $\mathcal{A}$ in game* MU-A-StCDH-Corr *for $\mathcal{G}$, there exists an adversary $\mathcal{B}$ against* StCDH *for $\mathcal{G}$ such that*

$$\mathbf{Adv}^{n\text{-A-StCDH-Corr}}_{\mathcal{G}}(\mathcal{A}) \leq n \cdot \mathbf{Adv}^{\mathsf{StCDH}}_{\mathcal{G}}(\mathcal{B}) \ ,$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof (Sketch). The proof goes via a simple guessing argument. Reduction $\mathcal{B}$ gets as input a challenge (g^x, g^y), chooses a random index $i \in [n]$ and sets h_{i-1}, h_i to be the challenge. It samples all remaining elements itself. DDH queries can then either be simulated locally or using $\mathcal{B}$'s own oracles. When $\mathcal{A}$ terminates with output (i^*, Z) and $i^* = i$, then $\mathcal{B}$ outputs Z. Otherwise, it aborts. Note that the probability that $\mathcal{B}$ guessed correctly is $1/n$. If $\mathcal{B}$ did not abort, it wins whenever $\mathcal{A}$ wins. □

Remark 2. One can show that this non-tight bound is optimal (for simple adversaries and black-box reductions) using the meta-reduction proof technique from [BJLS16].

Security of $\mathsf{CKA}_{\mathsf{DH}}$. We now establish the security of the $\mathsf{CKA}_{\mathsf{DH}}$ scheme based on our new assumption. Corollary 1 also states the corresponding non-tight bounds.

Theorem 1 (Security of $\mathsf{CKA}_{\mathsf{DH}}$). *Let n be the number of instances and q_e be an upper bound on the number of epochs. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for* $\mathsf{CKA}_{\mathsf{DH}}$ *and $\Delta = 1$ that makes q queries to* Check*, there exists an adversary $\mathcal{B}$ in game* MU-A-StCDH-Corr *for $\mathcal{G}$ such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{DH}},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{n'\text{-A-StCDH-Corr}}_{\mathcal{G}}(\mathcal{B}) \ ,$$

where $n' = n(q_e + 1)$ and $\mathcal{B}$ makes q queries to DDH*. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$.*

Proof. We construct a reduction $\mathcal{B}$ as follows: It gets as input nq_e group elements, which we denote by $\{m_{i,0}, \ldots, m_{i,q_e}\}_{i \in [n]}$, and has access to oracles DDH and Corr. It maintains a counter for each instance to keep track of how many epochs have passed. It will use the first q_e group elements for the first instance (initialising the state with $m_{1,0}$), the following q_e group elements for the second instance (initialising the state with $m_{2,0}$), and so on. Then, it simulates the output to $\mathcal{A}$'s queries as follows:

- Queries to $\textsc{Send}_{\mathsf{P}}(i)$ are simulated by outputting the next element $m_{i,t}$ from $\mathcal{B}$'s input as the CKA message. In this case, the CKA key as well as next state are not known to the reduction.
- Queries to $\textsc{SendR}_{\mathsf{P}}(i)$ can be simulated directly. Using the randomness, parsed as $x \in \mathbb{Z}_p$, provided by $\mathcal{A}$, it computes the next message g^x and stores the key $k_{i,t} = m^x_{i,t}$ and the new state x.
- Queries to $\textsc{Receive}_{\mathsf{P}}$ are simulated by only updating the state.

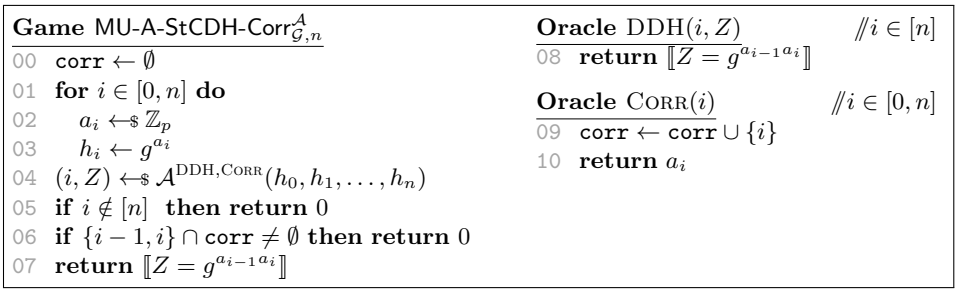

Fig. 7: Multi-user CDH game MU-A-StCDH-Corr for a cyclic group $\mathcal{G} = (\mathbb{G}, g, p)$.

- When CorrP is queried, $\mathcal{B}$ outputs the current state. If the state is unknown, it reveals the respective exponent by querying Corr.
- When Check is queried on (i,t,k), $\mathcal{B}$ checks whether it knows the corresponding CKA key $k_{i,t}$. In this case, it just does the comparison itself. If $\mathcal{B}$ does not know the CKA key, it queries DDH for the respective index i' (derived from i and t), $m_{i,t-1}$ and k. If the result is 1 and $\mathcal{A}$ has not previously corrupted either exponent, then $\mathcal{B}$ stops with solution (i',k).

The theorem follows by observing that $\mathcal{B}$ perfectly simulates the game and that whenever $\mathcal{A}$ wins, $\mathcal{B}$ also wins. □

Using the above lemma, we can also derive non-tight bounds for the CKA scheme.

Corollary 1. *Let n be the number of instances of CKA and q_e be an upper bound on the number of epochs. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for* $\mathsf{CKA}_{\mathsf{DH}}$ *and $\Delta = 1$, there exists an adversary $\mathcal{B}$ in game* MU-A-StCDH-Corr *for $\mathcal{G}$ such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{DH}},\Delta}(\mathcal{A}) \leq n(q_e+1) \cdot \mathbf{Adv}^{\mathsf{StCDH}}_{\mathcal{G}}(\mathcal{B}) \ ,$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

3.3 Interpretation of the Results

The resulting non-tight bound (cf. Corollary 1) improves the one given by [ACD19] for the overall messaging scheme by a factor of q_e and it matches the bound of [BFG+22] (refer also Figure 2). Unfortunately, our bound is still non-tight. In this section, we want to briefly discuss why a tight bound seems impossible, but that our analysis still enables valuable statements beyond those of previous work.

Towards an Impossibility Result. One might quickly come to the conclusion that proving tight security of $\mathsf{CKA}_{\mathsf{DH}}$ based on a standard assumption is impossible. In fact, the security game we use in the proof is very close to what has been studied in impossibility results such as [BJLS16]. Their result implies that the tightness loss in Lemma 1 is optimal (for simple adversaries and black-box reductions, and in the standard model). Intuitively, this is because the reduction (against single-instance StCDH) has to commit on which exponents it knows and where it embeds a challenge.

However, we also want to stress that this is not enough for a formal tightness impossibility result for $\mathsf{CKA}_{\mathsf{DH}}$. The assumption we chose as an intermediate step could simply be *stronger* than necessary. This is indeed the case because the n-A-StCDH-Corr challenger generates all group elements at the beginning of the game, while the CKA security game produces messages one-by-one. While we conjecture that the bound should still be optimal, the meta-reduction of [BJLS16] does not seem to be directly applicable to this adaptive setting, so we leave the exact study as an interesting open question.

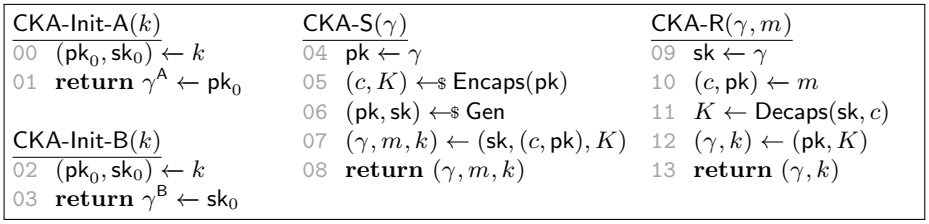

Fig. 8: CKA scheme $\mathsf{CKA}_{\mathsf{KEM}}$. The shared key space is the support of Gen.

Tight Bounds in the Generic Group Model. Since the security loss in previous work is introduced when proving security for the entire secure messaging scheme, using CKA as one building block, it is not possible to achieve tight security even if all building blocks have tight proofs to standard assumptions. Pushing the security loss as much into the building blocks as possible allows us to also take a different view on the security of the Double Ratchet, focusing on the CKA scheme only. Namely, we can easily provide information-theoretic lower bounds on the hardness of the n-A-StCDH-Corr assumption in the generic group model (GGM) [Sho97,Mau05]. One can show that, when restricting to generic attackers, the bounds of $\mathsf{CKA}_{\mathsf{DH}}$ match those of the discrete logarithm problem and are thus optimal for this class of adversaries. This follows from the analysis in [KPRR23, Corollary 1] which proves GGM bounds for a stronger variant of the assumption.

3.4 CKA from KEM

We want to investigate the concrete security of CKA instantiated from a key encapsulation mechanism (KEM). We recall the construction of [ACD19] in Figure 8. The initialisation key consists of a key pair, where the party in role A obtains the public key pk_0 and the party in role B obtains the secret key sk_0. A starts by computing an encapsulation c_0 of a key K_0 to public key pk_0. It also generates a new key pair $(\mathsf{pk}_1, \mathsf{sk}_1)$, stores sk_1 as the new state and sends (c_0, pk_1) to B. In order to compute the same CKA key K_0, B decapsulates c_0 using sk_0. It then stores pk_1 as its new state and becomes the sender.

We prove security of $\mathsf{CKA}_{\mathsf{KEM}}$ with a similar approach that we took for $\mathsf{CKA}_{\mathsf{DH}}$, namely we rely on a multi-user definition of the KEM. More specifically, we use multi-user one-way security with a key checking oracle and corruptions (MU-OW-PCA-Corr, cf. Definition 4). The scheme achieves security for $\Delta = 0$ because each component is only used to derive one CKA key. For this reason we can also slightly improve the number of instances that the assumption requires. Whereas for $\mathsf{CKA}_{\mathsf{DH}}$ we needed to bound the maximal number of epochs *per* instance, the theorem below relies on the *total* number of epochs $q_E \leq nq_e$, which is likely to be much smaller. We establish the following theorem.

Theorem 2 (Security of $\mathsf{CKA}_{\mathsf{KEM}}$). *Let n be the number of instances and let q_E be the total number of epochs across all instances. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for* $\mathsf{CKA}_{\mathsf{KEM}}$ *and $\Delta = 0$, there exists an adversary $\mathcal{B}$ in game* MU-OW-PCA-Corr *for* KEM *such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{KEM}},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{q_E\text{-OW-PCA-Corr}}_{\mathsf{KEM}}(\mathcal{B}) \ ,$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$. Further, if $\mathcal{A}$ issues q_{ck} queries to its checking oracle, then $\mathcal{B}$ issues at most q_{ck} queries to its own checking oracle.

Proof (Theorem 2). We construct a reduction $\mathcal{B}$ as follows: It gets as input q_E public keys pk_i and ciphertexts c_i and has access to oracles Check and Corr. It maintains a counter for each instance to keep track of how many epochs have passed, as well as a counter to record the number of total epochs. It will use the first n public keys to initialise all instances. Then, it simulates the output to $\mathcal{A}$'s queries as follows:

– Queries to $\textsc{Send}_{\mathsf{P}}$ are simulated by using the ciphertext for the public key in the current state. The reduction outputs the ciphertext along with the next unused public key from its input, according to the counter value, as the CKA message. The CKA state will be the KEM key which is not known to the reduction. The next state is also not known to the reduction since it is the corresponding secret key.

– Queries to $\textsc{SendR}_{\mathsf{P}}$ can be simulated directly using the randomness provided by $\mathcal{A}$. It is used to encapsulate a key as well as generating a new key pair. The new state is then the secret key of that key pair.
– Queries to $\textsc{Receive}_{\mathsf{P}}$ are simulated by only updating the state.
– When $\textsc{Corr}_{\mathsf{P}}$ is queried, $\mathcal{B}$ outputs the current state. If the state is unknown, it reveals the corresponding secret key by querying $\textsc{Corr}$.
– When $\textsc{Check}$ is queried on (i, t, I), $\mathcal{B}$ checks whether it knows the corresponding CKA key $I_{i,t}$ which is the KEM key for the index i' derived from i and t. If it knows the key, it just does the comparison itself. Otherwise, it will query $\textsc{Check}(i', c', I)$ for the respective ciphertext c'. If the result is 1 and $\mathcal{A}$ has not previously corrupted the secret key, then $\mathcal{A}$ as well as $\mathcal{B}$ have won their game.

The theorem follows by observing that $\mathcal{B}$ perfectly simulates the game and that whenever $\mathcal{A}$ wins, $\mathcal{B}$ also wins. □

Using a straightforward guessing argument as for Lemma 1, we can give the following corollary.

Corollary 2. *Let n be the number of instances and let q_E be the total number of epochs across all instances. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for* $\mathsf{CKA}_{\mathsf{KEM}}$ *and $\Delta = 0$, there exists an adversary $\mathcal{B}$ in game* OW-PCA *for* KEM *such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{KEM}},\Delta}(\mathcal{A}) \leq q_E \cdot \mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}) \ ,$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

This means we directly get a non-tight CKA scheme from any KEM scheme that achieves the weak notion of OW-PCA security. Note that such a KEM can be generically constructed from a CPA secure PKE using the (first part of the) modular Fujisaki-Okamoto transform [HHK17]. Looking ahead, this improves the bound from [ACD19] for the overall secure messaging scheme from nq_e^2 to $q_E \leq nq_e$ (cf. Corollary 6).

Counting Corruptions. We can hope for even better improvements if we move to a more fine-grained setting. In particular, we observe that the bound relies on the fact that technically all session states except one can be corrupted. This is highly unlikely to happen in practice. For this reason, [BRTZ24] studies a setting where the number of corruptions is viewed as an additional parameter. Here, denoting the total number of corruptions by q_C, we have $q_C \leq q_E$. Thus, when applying the ideas to our setting, we can further bring down the tightness loss to the number of corruptions.

Corollary 3 (Theorem 2 and [BRTZ24]). *Let n be the number of instances and let q_E be the total number of epochs across all instances. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for* $\mathsf{CKA}_{\mathsf{KEM}}$ *and $\Delta = 0$, there exists an adversary $\mathcal{B}$ in game q'_E-*OW-PCA *for* KEM *such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{KEM}},\Delta}(\mathcal{A}) \leq e \cdot (q_C + 1) \cdot \mathbf{Adv}^{q'_E\text{-OW-PCA}}_{\mathsf{KEM}}(\mathcal{B}) \ ,$$

where $e \approx 2.718$ is Euler's number, q_C is the total number of corruptions $\mathcal{A}$ makes and $q'_E = \lfloor (n-1)/(q_E+1) \rfloor$. Further, the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

The corollary follows from applying Theorem 5.3 in the full version of [BRTZ24] to Theorem 2.

We still rely on a multi-user assumption, but now *without* corruptions. Hence, this bound is of most interest for schemes where multi-user and single-user security are about the same. One prominent example is the ElGamal KEM, which was also considered in ACD19 and for which we gain substantial improvements in terms of tightness when the number of corruptions is low. We compare these improvements across different constructions in Section 6.

On the Non-Tightness. Unfortunately, the above bound is still non-tight. Thus, we may again ask whether there exists a proof showing that this bound is optimal for schemes with unique (or rerandomizable) keys (cf. [BJLS16]), e. g., the ElGamal KEM. To show this, we run into the same argument as for $\mathsf{CKA}_{\mathsf{DH}}$. Indeed, one would have to show the impossibility result for an interactive version of the assumption, where the adversary is provided by a public key and a ciphertext and then directly has to decide whether it wants to solve the challenge or whether it wants to receive the secret key. Then, moving to the next public key means essentially giving up on the previous challenge. For this experiment, the proof strategy of [BJLS16] does not seem to work.

Gen	Encaps(pk)	Decaps(sk, c)
00 $x_1, x_2 \leftarrow\!\!\$\ \mathbb{Z}_p$	04 $r \leftarrow\!\!\$\ \mathbb{Z}_p$	08 $(x_1, x_2) \leftarrow \mathsf{sk}$
01 $\mathsf{pk} \leftarrow g^{x_1} h^{x_2}$	05 $c \leftarrow (g^r, h^r)$	09 $(c_1, c_2) \leftarrow c$
02 $\mathsf{sk} \leftarrow (x_1, x_2)$	06 $K \leftarrow \mathsf{pk}^r$	10 $K \leftarrow c_1^{x_1} c_2^{x_2}$
03 **return** $(\mathsf{pk}, \mathsf{sk})$	07 **return** (c, K)	11 **return** K

Fig. 9: KEM scheme $\mathsf{KEM}_{\mathsf{CS}}$ for an efficient tightly-secure CKA scheme.

Tightly-Secure Construction from DDH. For the remainder of this section, we want to look at constructions from which we know that they are not covered by a potential impossibility result and give us tight security from standard assumptions. We construct a KEM scheme that achieves tight MU-OW-PCA-Corr security based on DDH and thus yields a tightly-secure CKA scheme. The scheme is based on the lite version of the Cramer-Shoup cryptosystem [CS98], hence the name $\mathsf{KEM}_{\mathsf{CS}}$ (and $\mathsf{CKA}_{\mathsf{CS}}$, respectively), and is given in Figure 9.[9]

Theorem 3. *Let $\mathsf{CKA}_{\mathsf{CS}}$ be the CKA scheme from Figure 8 instantiated with $\mathsf{KEM}_{\mathsf{CS}}$. Let n be the number of instances. For any adversary $\mathcal{A}$ in game* MI-OW-CKA *for $\mathsf{CKA}_{\mathsf{CS}}$ and $\Delta = 0$, there exists an adversary $\mathcal{B}$ against* DDH *such that*

$$\mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA}_{\mathsf{CS}},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{G}}(\mathcal{B}) + \frac{1}{p-1} + \frac{q_{\mathrm{ck}}}{p} \ ,$$

where q_{ck} is the number of queries that $\mathcal{A}$ issues to its checking oracle. Further, the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

We note that tight multi-user single-challenge security of Cramer-Shoup has already been studied in [BBM00]. For completeness, we repeat the proof and explain why answering corruption queries comes for free.

Proof. We use a multi-user variant of DDH (without corruptions), where the adversary is given a group element h and q_E tuples (g^{r_i}, h^{s_i}) for $r_i \leftarrow\!\!\$\ \mathbb{Z}_p$ and either $s_i = r_i$ or $s_i \leftarrow\!\!\$\ \mathbb{Z}_p$. We denote this assumption by q_E-DDH. By random self-reducibility, this assumption is tightly implied by the standard DDH assumption [EHK+13, Lemma 1] up to a term $1/(p-1)$. We now proceed with the actual proof of Theorem 2 using this assumption.

We construct an adversary $\mathcal{B}$ that simulates the MU-OW-PCA-Corr game for q_E users. It gets as input h and q_E tuples $(c_{1,i}, c_{2,i})$. It picks random values $x_{1,i}, x_{2,i} \leftarrow\!\!\$\ \mathbb{Z}_p$ as the secret keys and computes $\mathsf{pk}_i = g^{x_{1,i}} h^{x_{2,i}}$ for all $i \in \{1, \ldots, q_E\}$. It runs $\mathcal{A}$ on $((\mathsf{pk}_1, (c_{1,1}, c_{2,1})), \ldots, (\mathsf{pk}_{q_E}, (c_{1,q_E}, c_{2,q_E})))$ and simulates the oracles Check and Corr as follows. Since $\mathcal{B}$ knows all the secret keys, it can easily answer corruptions queries. If $\mathcal{A}$ queries Check for index $(i, (c_1, c_2), K)$, $\mathcal{B}$ will compute $K' = c_1^{x_{1,i}} c_2^{x_{2,i}}$ and compare the result to K. If the ciphertext was that user's challenge and $K' = K$, then $\mathcal{A}$ has won and $\mathcal{B}$ stops with output 0 ("real"). If at the end of the execution, $\mathcal{B}$ has not already stopped, it outputs 1 ("random").

The analysis of $\mathcal{B}$ follows the argument of Cramer-Shoup. If $\mathcal{B}$'s input consists of real DDH tuples, then it perfectly simulates game q_E-OW-PCA-Corr. Otherwise, if $\mathcal{B}$'s input consists of random tuples, then the key is information-theoretically hidden. If $\mathcal{A}$ nevertheless computes the correct key, which happens with q_{ck}/p, $\mathcal{B}$'s guess is not correct, which introduces the additional term into the bound. □

Tightly-Secure Construction from Lattices. In [PWZ23a], Pan, Wagner and Zeng use a similar (but stronger) security notion for KEMs to construct tightly-secure authenticated key exchange. Since their notion trivially implies ours, we get a tightly-secure CKA scheme based on the LWE assumption.

[9] The construction may easily be generalised to universal hash proof systems with a multi-fold subset membership problem, cf. e. g. [JKRS21].

4 Multi-Instance FS-AEAD

Forward-secure authenticated encryption with associated data (FS-AEAD) is an abstraction that captures the desired interface and properties of one instance of the symmetric ratchet of the Double Ratchet protocol. As its name suggests, it provides authentication and confidentiality (AEAD), as well as forward security under state compromise (FS) (and is thus stateful), and supports out-of-order message delivery. As in the Double Ratchet, we assume one party is the designated sender (A), and the other the receiver (B). In the following, we recall the definition from ACD19.

Syntax. An FS-AEAD scheme defines an initialisation key space $\mathcal{K}_{\mathsf{FS}}$ and a quadruple of algorithms $\mathsf{FS\text{-}AEAD} = (\mathsf{FS\text{-}Init\text{-}A}, \mathsf{FS\text{-}Init\text{-}B}, \mathsf{FS\text{-}Send}, \mathsf{FS\text{-}Rcv})$, where

- $\mathsf{FS\text{-}Init\text{-}A}$ (resp. $\mathsf{FS\text{-}Init\text{-}B}$) takes a key $k^{\mathsf{AB}} \in \mathcal{K}_{\mathsf{FS}}$ as input and produces an initial state v^{A} (resp. v^{B}).
- $\mathsf{FS\text{-}Send}$ takes a state v, associated data a and a message msg as input and produces a new state and a ciphertext $(v', c) \leftarrow \mathsf{FS\text{-}Send}(v, \mathsf{msg}, a)$
- $\mathsf{FS\text{-}Rcv}$ takes a state v, associated data a and a ciphertext c as input and produces a new state, an index and a message $(v', i, \mathsf{msg}) \leftarrow \mathsf{FS\text{-}Rcv}(v, c, a)$

We capture correctness and security (privacy and authenticity) for $\mathsf{FS\text{-}AEAD}$ schemes in the following definition. The pseudocode and a description for the game are given below.

Definition 7. *Consider game* $\mathsf{MI\text{-}FS}$ *in Figure 10 for* $\mathsf{FS\text{-}AEAD}$*, integer n and adversary $\mathcal{A}$. We define the advantage of $\mathcal{A}$ in this game as*

$$\mathbf{Adv}^{n\text{-}\mathsf{FS}}_{\mathsf{FS\text{-}AEAD}}(\mathcal{A}) \coloneqq \max\left\{2 \cdot \left|\Pr[\mathsf{MI\text{-}FS}^{\mathcal{A}}_{\mathsf{FS\text{-}AEAD},n} \Rightarrow 1] - 1/2\right|, \Pr[\mathbf{win_{auth}}],\ \Pr[\mathbf{win_{corr}}]\right\}.$$

The game provides authenticity and confidentiality guarantees for each instance of FS-AEAD and so corresponds to an IND-CCA-style AEAD definition in the absence of state exposure. In addition, under state exposure, messages contained in ciphertexts that cannot be trivially derived by the adversary due to correctness should remain secure, and the adversary should not be able to forge ciphertexts for the corresponding indices.

The adversary can win in three ways. Firstly, by violating correctness, i.e., if, given ciphertexts and associated data are faithfully transmitted, modulo out-of-order delivery, that for each message m sent by A sent with index i, B outputs a different $(m', i') \neq (m, i)$ when decrypting A's ciphertext. Secondly, if the adversary is able to successfully forge a ciphertext unless the adversary can trivially do so due to state exposure. Finally, by correctly guessing a bit uniformly sampled by the challenger (that is the same across all instances), given access to a challenge oracle $\textsc{Chall}_{\mathsf{A}}(\ell, a, m_o, m_1)$ that the adversary can query several times; as in the second case, the adversary cannot call $\textsc{Chall}_{\mathsf{A}}$ if he can trivially decrypt the output ciphertext. Note in the game that due to the fact that we consider a multi-instance setting, when $\textsc{Corr}_{\mathsf{B}}$ is queried for some ℓ, only challenge oracles in instance ℓ are then disabled (rather than all oracles as in ACD19).

Comparison to [BFG+22,ACD19]. Our definition is a multi-instance variant of ACD19. The single-instance security from BFGMR22 is formulated such that their secure messaging scheme achieves security in the UC model. For this reason they need a notion of "explainability" and their proof inherently relies on the (programmable) random oracle model (whereas we use the ROM below for tightness).

Instantiation. We use the instantiation proposed [ACD19], where we replace the PRF-PRNG with a random oracle G of appropriate domain and range. More specifically, we construct an FS-AEAD scheme $\mathsf{FS\text{-}AEAD}$ with key space $\mathcal{K}_{\mathsf{FS}}$ from an AEAD scheme $\mathsf{AE} = (\mathsf{Enc}, \mathsf{Dec})$ with key space $\mathcal{K}_{\mathsf{AE}}$ and a random oracle $\mathsf{G}\colon \mathcal{K}_{\mathsf{FS}} \to \mathcal{K}_{\mathsf{FS}} \times \mathcal{K}_{\mathsf{AE}}$. Following ACD19, the final secure messaging scheme will also use helper functions $\mathsf{FS\text{-}Stop}$ and $\mathsf{FS\text{-}Max}$ to terminate an individual instance, which we explain in more detail in Section 5. These are, however, not relevant on this level of abstraction. The scheme is described in Figure 11.

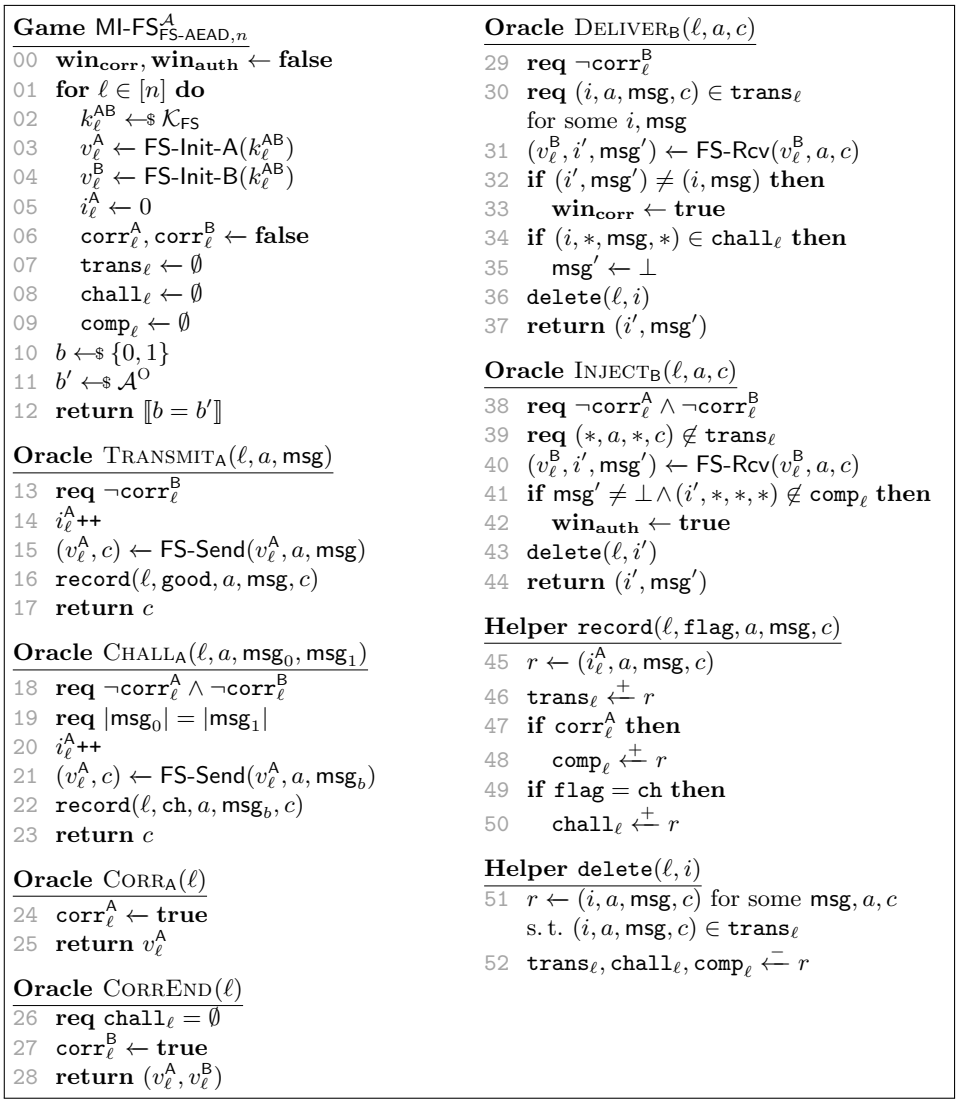

Fig. 10: Multi-instance game MI-FS for FS-AEAD, where $\mathcal{A}$ has access to oracles $\mathrm{O} = \{\textsc{Transmit}_{\mathsf{A}}, \textsc{Chall}_{\mathsf{A}}, \textsc{Corr}_{\mathsf{A}}, \textsc{CorrEnd}, \textsc{Deliver}_{\mathsf{B}}, \textsc{Inject}_{\mathsf{B}}\}$. Helper functions `record` and `delete` are for book-keeping so that winning conditions can be evaluated and trivial wins can be avoided.

Theorem 4. *Let n be the number of instances. For any adversary $\mathcal{A}$ in game* MI-FS *for* FS-AEAD *that issues at most q_M queries to* $\textsc{Transmit}_{\mathsf{A}}$ *and* $\textsc{Chall}_{\mathsf{A}}$ *combined across all instances and at most q_{RO} queries to random oracle* G*, there exists an adversary $\mathcal{B}$ in game* MU-OT-CCA *for* AE *such that*

$$\mathbf{Adv}^{n\text{-}\mathsf{FS}}_{\mathsf{FS\text{-}AEAD}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{q_M\text{-}\mathsf{OT\text{-}CCA}}_{\mathsf{AE}}(\mathcal{B}) + \frac{2(n + q_{\mathsf{RO}})^2}{|\mathcal{K}_{\mathsf{FS}}|} ,$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Intuitively, the additional statistical term captures that the FS-AEAD keys which are fed into the random oracle to obtain an AE key come from a sufficiently large key space to yield unpredictable and independent outputs.

We directly prove the above theorem in the random oracle model which allows us to get tight bounds "for free". We further discuss the necessity of idealized models for tight security and a possible alternative modularisation in Section 7.1.

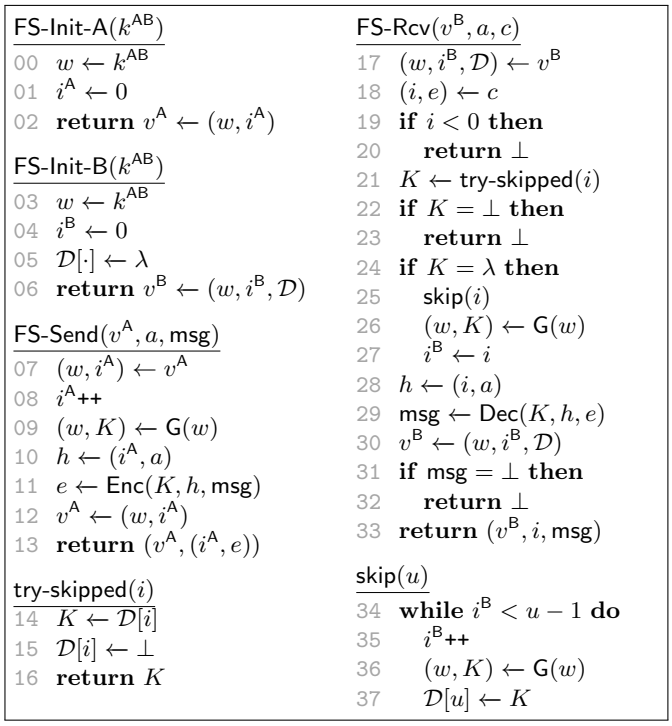

Fig. 11: FS-AEAD scheme FS-AEAD based on AE and G.

Proof (Theorem 4). Let $\mathcal{A}$ be an adversary in game MI-FS for FS-AEAD. We prove this theorem by describing a sequence of games G_0-G_3 and proceed by game hopping. We let S_i be the event that the final output in game G_i is 1.

Game G_0. This is the MI-FS security game with respect to $\mathcal{A}$, where G is modeled as a random oracle. Thus,

$$\mathbf{Adv}^{n\text{-FS}}_{\text{FS-AEAD}}(\mathcal{A}) = |\Pr[S_0] - 1/2| \ .$$

Game G_1. This game is identical to game G_0, except the winning condition in $\textsc{Deliver}_\mathsf{B}$ is removed. By perfect correctness of FS-AEAD, we have $\Pr[S_0] = \Pr[S_1]$.

Game G_2. This game makes two changes. First, we abort when there is a collision between any of the initial inputs w_ℓ and any of the output values w of the random oracle G. Second, we also abort when w is sampled as an output of the random oracle such that w was previously queried by the adversary, i.e., the adversary predicts an output. Birthday collision bound yields

$$|\Pr[S_1] - \Pr[S_2]| \leq \frac{2(n + q_{\mathsf{RO}})^2}{|\mathcal{K}_{\mathsf{FS}}|} \ .$$

Game G_3. This game is identical to game G_3, except that all AEAD ciphertexts created in $\textsc{Chall}_\mathsf{A}$ are replaced by random ciphertexts from $\{0,1\}^{\mathsf{cl}(|m|)}$. For queries to $\textsc{Deliver}_\mathsf{B}$, the game simply returns the message that was provided to $\textsc{Transmit}_\mathsf{A}$. Further, the winning condition in $\textsc{Inject}_\mathsf{B}$ is removed and injections for which the state was not compromised are always rejected. We construct adversaries $\mathcal{B}_{b'}$ for bit b' against MU-OT-CCA of AE such that for some $\mathcal{B}$ we have

$$|\Pr[S_2] - \Pr[S_3]| \leq 2 \cdot \mathbf{Adv}^{q_M\text{-OT-CCA}}_{\mathsf{AE}}(\mathcal{B}) \ ,$$

where q_M is the total number of messages. $\mathcal{B}_{b'}$ simulates G as in G_2, and in particular simulates the initialisation algorithms by uniformly sampling values $w \in \mathcal{K}_{\mathsf{FS}}$. $\mathcal{B}_{b'}$ simulates queries to $\textsc{Transmit}_\mathsf{A}(\ell, a,$ msg$)$ by simulating FS-Send locally (in particular simulating the query to G, which may have been previously queried if B's state was previously exposed). $\mathcal{B}_{b'}$ simulates queries to $\textsc{Chall}_\mathsf{A}(\ell, a,$ msg$_0,$ msg$_1)$ by calling $\textsc{Enc}(\ell, a,$ msg$_{b'})$ and simulating the corresponding call to G

with unknown output K (note that if the corresponding input w is ever queried to G, then the game aborts, so we don't need to simulate beyond this). Oracle $\textsc{Deliver}_{\mathsf{B}}$ can then be simulated using $\mathcal{B}_{b'}$'s records. For queries to $\textsc{Inject}_{\mathsf{B}}$ where $\mathcal{A}$ has not compromised the state, $\mathcal{B}_{b'}$ simulates by calling $\textsc{Dec}$. $\mathcal{B}_{b'}$ simulates corruption queries by outputting the relevant simulated values.

Note then that $\mathcal{B}_{b'}$ simulates G_2 perfectly given the n-$\mathsf{OT\text{-}CCA}$ challenge bit is 0 and the G_2 bit is b', and G_3 perfectly if the n-$\mathsf{OT\text{-}CCA}$ challenge bit is 1, since the adversary gets randomly chosen ciphertexts and also the $\textsc{Dec}$ oracle always outputs $\perp$. The claim then follows by application of the triangle inequality.

Finally, observe that $\Pr[S_3] = 1/2$. Thus, the theorem follows from collecting the probabilities. □

5 Multi-Session Secure Messaging

This section presents the formal syntax of secure messaging schemes and their security, based on the definitions from ACD19.

Syntax. A secure messaging scheme SM specifies an initialisation key space $\mathcal{K}_{\mathsf{SM}}$ and and the four algorithms $\mathsf{SM} = (\mathsf{SM\text{-}Init\text{-}A}, \mathsf{SM\text{-}Init\text{-}B}, \mathsf{SM\text{-}Send}, \mathsf{SM\text{-}Rcv})$, where

- $\mathsf{SM\text{-}Init\text{-}A}$ (resp. $\mathsf{SM\text{-}Init\text{-}B}$) takes a (shared) key $k^{\mathsf{AB}} \in \mathcal{K}_{\mathsf{SM}}$ as input and outputs an initial state s^{A} (resp. s^{B}).
- $\mathsf{SM\text{-}Send}$ takes a state s and a message msg as input, and outputs a new state and a ciphertext $(s', c) \leftarrow\!\$\ \mathsf{SM\text{-}Send}(s, \mathsf{msg})$.
- $\mathsf{SM\text{-}Rcv}$ takes a state s, and a ciphertext c as input, and outputs a new state, an epoch number, an index, and a message $(s', t, i, \mathsf{msg}) \leftarrow \mathsf{SM\text{-}Rcv}(s, c)$.

Let A be the (first) sending user and B be the (first) receiving user. Formally, an SM scheme consists of an initialisation algorithm, that takes as input a shared key, which sets up the state for A to communicate with B securely and bidirectionally, a sending and a receiving algorithm, both that keep state across invocations. In order to determine the order of messages the sending user sent, the receiving algorithm outputs an epoch number and an index, defined below, as well to determine the transmission order of the messages.

Our goal is to extend the SM security model of ACD19 to the multi-session setting. Before that, we recall several properties of SM schemes, as given by ACD19.

Epochs. An SM scheme proceeds in so-called epochs, which roughly correspond to the "back-and-forth" between two parties A and B. By convention, odd epoch numbers t are associated with A sending and B receiving, and the other way around for even epochs.

Note, however, that SM schemes are completely asynchronous, and, hence, epochs overlap to a certain extent. W.l.o.g, consider two epoch counters t^{A} and t^{B} for A and B, respectively, satisfying the following properties:

- The two counters are never more than one epoch apart, i. e., $|t^{\mathsf{A}} - t^{\mathsf{B}}| \leq 1$ at all times.
- When A receives an epoch-t message from B for $t = t^{\mathsf{A}} + 1$, it sets $t^{\mathsf{A}} \leftarrow t$ (even). The next time A sends a message, t^{A} is incremented again (to an odd value).
- Similarly, when B receives an epoch-t message from A for $t = t^{\mathsf{B}} + 1$, it sets $t^{\mathsf{B}} \leftarrow t$ (odd). The next time B sends a message, t^{B} is incremented again (to an even value).

Message Index. Within an epoch, messages are identified by a simple counter. To capture the property of immediate decryption, i. e., out-of-order message delivery and support for dropped messages, the receive algorithm of an SM scheme is required to output the correct epoch number and index *immediately* upon reception of a ciphertext, even when messages arrive out of order.

Corruptions and their Consequences. Since SM schemes are required to be forward-secure and to recover from state compromise, any SM security game must allow the attacker to learn the state of either party at any given time. Moreover, to capture authenticity and privacy, the attacker should be given the power to inject malicious ciphertexts and to call a (say) left-or-right challenge oracle, respectively. These requirements, however, interfere as follows:

- When either party is in a compromised state, the attacker cannot invoke the challenge oracle since this would allow him to trivially distinguish.
- When either party is in a compromised state, the attacker can trivially forge ciphertexts and must therefore be barred from calling the inject oracle.
- When the receiver of messages in transmission is compromised, these messages lose all security, i. e., the attacker learns their content and can replace them by a valid forgery. Consequently, while any challenge ciphertext is in transmission, the recipient may not be corrupted. Similarly, an SM scheme must be able to deal with forgeries of compromised messages (once the parties have healed).

Natural SM Scheme. Similarly to Definition 7 of ACD19, we define natural SM schemes and assume for simplicity that SM schemes in this work are natural SM schemes.

Definition 8. *An SM scheme* $\mathsf{SM} = (\mathsf{SM\text{-}Init\text{-}A}, \mathsf{SM\text{-}Init\text{-}B}, \mathsf{SM\text{-}Send}, \mathsf{SM\text{-}Rcv})$ *is* natural *if the following criteria are satisfied:*

A) Whenever $\mathsf{SM\text{-}Rcv}(s, c)$*, for some ciphertext* c*, outputs* $\mathsf{msg} = \bot$*, the state remains unaltered.*
B) Any given ciphertext c *corresponds to an epoch* t *and an index* i*, i. e., the values* t *and* i *output by* $\mathsf{SM\text{-}Rcv}(s, c)$ *are an (efficiently computable) function of* c*.*
C) $\mathsf{SM\text{-}Rcv}(s, c)$*, for some ciphertext* c*, never accepts two messages corresponding to the same pair* (t, i)*.*
D) A party always rejects ciphertexts corresponding to an epoch in which it does not act as receiver.
E) If a party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$ *accepts a ciphertext corresponding to an epoch* t*, then* $t^{\mathsf{P}} \geq t - 1$*.*

Remark 3. Property C) may also be viewed as an integrity property, but, following ACD19, we include this property in the definition of a natural scheme. While it is implied by forward security, it can be guaranteed unconditionally by storing received epochs or indices.

5.1 Multi-Session Security Game

We now extend the model of ACD19 to the multi-session setting. The game is initialised by creating n pairs of sender and receiver states. As in the CKA game, each oracle now takes as an additional input an index ℓ that refers to which instance is used for sending/receiving.

Note the multi-session secure-messaging security game presented in Figure 12 is parameterised by Δ. Occasionally, we will refer to a particular record (or a set thereof) by specifying only some elements of it, e. g., the expression $(\mathsf{P}, *, *, *, *, *) \notin \mathtt{chall}_\ell$ is abbreviated with $\mathsf{P} \notin \mathtt{chall}_\ell$, and $\mathtt{trans}_\ell(\mathsf{B})$ denotes the set of all tuples in $\mathtt{trans}_\ell$ with first entry B.

Definition 9. *Consider game* MS-SM *in Figure 12 for an SM scheme* SM*, non-negative integer* Δ*, positive integer* n *and adversary* $\mathcal{A}$*. We define the advantage of* $\mathcal{A}$ *in this game as*

$$\mathbf{Adv}^{n\text{-}\mathsf{SM}}_{\mathsf{SM},\Delta}(\mathcal{A}) := \max\left\{2 \cdot \left|\Pr[\mathsf{MS\text{-}SM}^{\mathcal{A}}_{\mathsf{SM},\Delta,n} \Rightarrow 1] - 1/2\right|, \Pr[\mathbf{win_{auth}}],\ \Pr[\mathbf{win_{corr}}]\right\}.$$

5.2 Instantiation

We recall the Signal-based Secure Messaging scheme from ACD19, replacing the PRF-PRNG by a random oracle as done in BFGMR22. The SM scheme is given in Figure 13. It is constructed from an FS-AEAD scheme FS-AEAD with initialisation key space $\mathcal{K}_{\mathsf{FS}}$, a CKA scheme CKA with initialisation key space $\mathcal{K}^{\mathsf{init}}_{\mathsf{CKA}}$, CKA key space $\mathcal{K}_{\mathsf{CKA}}$ and randomness space $\mathcal{R}_{\mathsf{CKA}}$ and a random oracle $\mathsf{H}\colon \Sigma_{\mathsf{root}} \times (\mathcal{K}_{\mathsf{CKA}} \cup \{\varepsilon\}) \to \Sigma_{\mathsf{root}} \times \mathcal{K}_{\mathsf{FS}}$, where Σ_{root} is the root key space and ε is the empty string.

Game $\mathsf{MS\text{-}SM}^{\mathcal{A}}_{\mathsf{SM},\Delta,n}$

00 $\mathbf{win}_{\mathbf{corr}}, \mathbf{win}_{\mathbf{auth}} \leftarrow \mathbf{false}$
01 **for** $\ell \in [n]$ **do**
02 $\quad k_\ell^{\mathsf{AB}} \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{SM}}$
03 $\quad s_\ell^{\mathsf{A}} \leftarrow \mathsf{SM\text{-}Init\text{-}A}(k_\ell^{\mathsf{AB}})$
04 $\quad s_\ell^{\mathsf{B}} \leftarrow \mathsf{SM\text{-}Init\text{-}B}(k_\ell^{\mathsf{AB}})$
05 $\quad t_\ell^{\mathsf{A}}, t_\ell^{\mathsf{B}} \leftarrow 0$
06 $\quad i_\ell^{\mathsf{A}}, i_\ell^{\mathsf{B}} \leftarrow 0$
07 $\quad t_\ell^{\mathsf{L}} \leftarrow -\infty$
08 $\quad \mathtt{trans}_\ell, \mathtt{chall}_\ell, \mathtt{comp}_\ell \leftarrow \emptyset$
09 $b \leftarrow\!\!\$\ \{0,1\}$
10 $b' \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{O}}$
11 **return** $[\![b = b']\!]$

Oracle $\textsc{Transmit}_{\mathsf{A}}(\ell, \mathsf{msg}, r)$

12 $(r, \mathtt{flag}) \leftarrow \mathtt{sam\text{-}if\text{-}nec}(r)$
13 $\mathtt{ep\text{-}mgmt}(\ell, \mathsf{A}, \mathtt{flag})$
14 $i_\ell^{\mathsf{A}}\mathtt{++}$
15 $(s_\ell^{\mathsf{A}}, c) \leftarrow \mathsf{SM\text{-}Send}(s_\ell^{\mathsf{A}}, \mathsf{msg}; r)$
16 $\mathtt{record}(\ell, \mathsf{A}, \mathtt{norm}, \mathsf{msg}, c)$
17 **return** c

Oracle $\textsc{Chall}_{\mathsf{A}}(\ell, \mathsf{msg}_0, \mathsf{msg}_1, r)$

18 $(r, \mathtt{flag}) \leftarrow \mathtt{sam\text{-}if\text{-}nec}(r)$
19 $\mathtt{ep\text{-}mgmt}(\ell, \mathsf{A}, \mathtt{flag})$
20 **req** $\mathtt{safe\text{-}ch}_\ell^{\mathsf{A}} \wedge |\mathsf{msg}_0| = |\mathsf{msg}_1|$
21 $i_\ell^{\mathsf{A}}\mathtt{++}$
22 $(s_\ell^{\mathsf{A}}, c) \leftarrow \mathsf{SM\text{-}Send}(s_\ell^{\mathsf{A}}, \mathsf{msg}_b; r)$
23 $\mathtt{record}(\ell, \mathsf{A}, \mathtt{ch}, \mathsf{msg}_b, c)$
24 **return** c

Oracle $\textsc{Deliver}_{\mathsf{A}}(\ell, c)$

25 **req** $(\mathsf{B}, t, i, \mathsf{msg}, c) \in \mathtt{trans}_\ell$ for some t, i, msg
26 $(s_\ell^{\mathsf{A}}, t', i', \mathsf{msg}') \leftarrow \mathsf{SM\text{-}Rcv}(s_\ell^{\mathsf{A}}, c)$
27 **if** $(t', i', \mathsf{msg}') \neq (t, i, \mathsf{msg})$ **then**
28 $\quad \mathbf{win}_{\mathbf{corr}} \leftarrow \mathbf{true}$
29 **if** $(t, i, \mathsf{msg}) \in \mathtt{chall}_\ell$ **then**
30 $\quad \mathsf{msg}' \leftarrow \bot$
31 $t_\ell^{\mathsf{A}} \leftarrow \max(t_\ell^{\mathsf{A}}, t)$
32 $\mathtt{delete}(\ell, t, i)$
33 **return** (t, i, msg')

Oracle $\textsc{Inject}_{\mathsf{A}}(\ell, c)$

34 **req** $(\mathsf{B}, c) \notin \mathtt{trans}_\ell \wedge \mathtt{safe\text{-}inj}_\ell$
35 $(s_\ell^{\mathsf{A}}, t', i', \mathsf{msg}') \leftarrow \mathsf{SM\text{-}Rcv}(s_\ell^{\mathsf{A}}, c)$
36 **if** $\mathsf{msg}' \neq \bot \wedge (\mathsf{B}, t', i') \notin \mathtt{comp}_\ell$ **then**
37 $\quad \mathbf{win}_{\mathbf{auth}} \leftarrow \mathbf{true}$
38 $t_\ell^{\mathsf{A}} \leftarrow \max(t_\ell^{\mathsf{A}}, t')$
39 $\mathtt{delete}(\ell, t', i')$
40 **return** (t', i', msg')

Oracle $\textsc{Corr}_{\mathsf{A}}(\ell)$

41 **req** $\mathsf{B} \notin \mathtt{chall}_\ell$
42 $\mathtt{comp}_\ell \xleftarrow{+} \mathtt{trans}_\ell(\mathsf{B})$
43 $t_\ell^{\mathsf{L}} \leftarrow \max(t_\ell^{\mathsf{A}}, t_\ell^{\mathsf{B}})$
44 **return** s_ℓ^{A}

Helper $\mathtt{delete}(\ell, t, i)$

45 $r \leftarrow (\mathsf{P}, t, i, \mathsf{msg}, c)$ for $\mathsf{P}, \mathsf{msg}, c$ s.t. $r \in \mathtt{trans}_\ell$
46 $\mathtt{trans}_\ell, \mathtt{chall}_\ell, \mathtt{comp}_\ell \xleftarrow{-} r$

Helper $\mathtt{ep\text{-}mgmt}(\ell, \mathsf{P}, \mathtt{flag})$

47 **if** $(t_\ell^{\mathsf{P}}$ even $\wedge\ \mathsf{P} = \mathsf{A}) \vee (t_\ell^{\mathsf{P}}$ odd $\wedge\ \mathsf{P} = \mathsf{B})$ **then**
48 $\quad$ **if** $\mathtt{flag} = \mathtt{bad} \wedge \neg\mathtt{safe\text{-}ch}_\ell^{\mathsf{P}}$ **then**
49 $\quad\quad t_\ell^{\mathsf{L}} \leftarrow t_\ell^{\mathsf{P}} + 1$
50 $\quad t_\ell^{\mathsf{P}}\mathtt{++}$
51 $\quad i_\ell^{\mathsf{P}} \leftarrow 0$

Helper $\mathtt{sam\text{-}if\text{-}nec}(r)$

52 $\mathtt{flag} \leftarrow \mathtt{bad}$
53 **if** $r = \bot$ **then**
54 $\quad r \leftarrow\!\!\$\ \mathcal{R}$
55 $\quad \mathtt{flag} \leftarrow \mathtt{good}$
56 **return** $(r, \mathtt{flag})$

Helper $\mathtt{record}(\ell, \mathsf{P}, \mathtt{flag}, m, c)$

57 $r \leftarrow (\mathsf{P}, t_\ell^{\mathsf{P}}, i_\ell^{\mathsf{P}}, \mathsf{msg}, c)$
58 $\mathtt{trans}_\ell \xleftarrow{+} r$
59 **if** $\neg\mathtt{safe\text{-}ch}_\ell^{\mathsf{P}}$ **then**
60 $\quad \mathtt{comp}_\ell \xleftarrow{+} r$
61 **if** $\mathtt{flag} = \mathtt{ch}$
62 $\quad \mathtt{chall}_\ell \xleftarrow{+} r$

Predicate $\mathtt{safe\text{-}ch}_\ell^{\mathsf{P}}$

63 **return** $[\![(t_\ell^{\mathsf{P}} \geq t_\ell^{\mathsf{L}} + \Delta)]\!]$

Predicate $\mathtt{safe\text{-}inj}_\ell$

64 **return** $[\![\min(t_\ell^{\mathsf{A}}, t_\ell^{\mathsf{B}}) \geq t_\ell^{\mathsf{L}} + \Delta]\!]$

Fig. 12: Multi-session game MS-SM for scheme SM, where $\mathcal{A}$ has access to oracles $\mathrm{O} = \{\textsc{Transmit}_{\mathsf{P}}, \textsc{Chall}_{\mathsf{P}}, \textsc{Corr}_{\mathsf{P}}, \textsc{Deliver}_{\mathsf{P}}, \textsc{Inject}_{\mathsf{P}}\}_{\mathsf{P}\in\{\mathsf{A},\mathsf{B}\}}$. The oracles for the receiving parties in role B are defined similarly.

State. The SM scheme keeps internal states s for a session between two parties A and B. W.l.o.g. A, that initialises communication with B, invokes SM-Init-A on input k to initialise state s, which will be updated by SM-Send and SM-Rcv. It consists of:

- an ID field $\mathsf{id} \in \{\mathsf{A}, \mathsf{B}\}$,
- the state σ_{root},
- the states $v[0], v[1], v[2], \ldots$ of the FS-AEAD instances,
- the state γ of the CKA scheme,
- the current CKA message $m_{\mathtt{cur}}$,

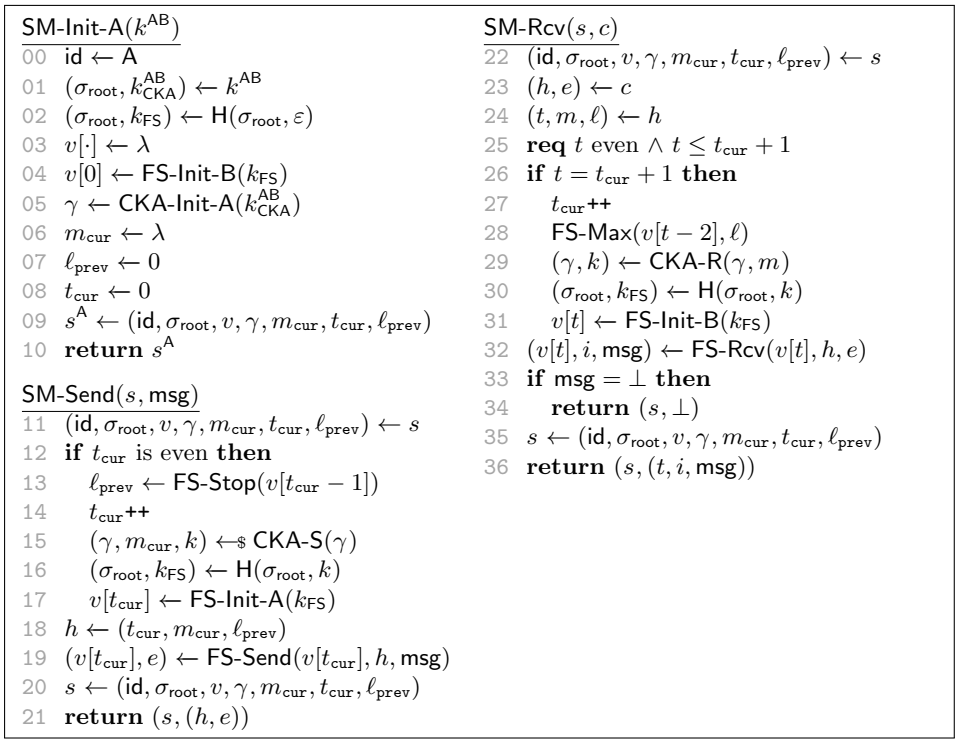

Fig. 13: Secure-messaging scheme SM based on FS-AEAD, CKA and random oracle H. The initialization algorithm SM-Init-B is defined analogously, with id set to B and initializing $v[0]$ via FS-Init-A. For simplicity, we specify SM-Send and SM-Rcv for A only. For B, "even" will be changed to "odd".

- the epoch counter t_{cur}, and
- the number of messages ℓ_{prev} sent in the previous epoch.

Helper functions for FS-AEAD As in ACD19, we define the following two functions for memory management:

- FS-Stop takes a state $v[i]$ as input and outputs an integer ℓ_{prev} that indicates how many messages have been sent using that FS-AEAD instance $v[i]$. It then "erases" that instance from memory;
- FS-Max takes a state $v[i]$ and an integer ℓ, "remembers" internally such that the instance corresponding to $v[i]$ is erased from memory as soon as ℓ messages have been received.

In the SM scheme, FS-Stop is called when a party initiates a new epoch, meaning that the previous FS-AEAD instance it used for sending is no longer needed. The integer ℓ_{prev} lets the receiver know whether it received all messages from that instance. That is, using FS-Max it records that number and can decide whether the old instance can be deleted or how many messages it still expects.[10]

Security Proof. Unlike ACD19, that performs security reduction for the individual properties, namely correctness, privacy and authenticity, thus incurring a quadratic loss in the number of epochs (in the single-session setting), we will perform reductions directly to the building blocks.

Theorem 5. *Let n be the number of sessions and let q_E be the total number of epochs across all sessions. Let* FS-AEAD *and* CKA *be perfectly correct. Let $\Delta = \Delta_{CKA} + 2$. For any adversary $\mathcal{A}$ in game* MS-SM *for* SM *with root key space Σ_{root} (as shown in Figure 13), there exists an adversary*

[10] Note that these functions can be generically defined for any FS-AEAD scheme and also that an adversary can infer the values from the protocol messages which is why these functions are not explicit in the FS-AEAD security game.

$\mathcal{B}$ *in game* MI-OW-CKA *for* CKA *and an adversary* $\mathcal{C}$ *in game* MI-FS *for* FS-AEAD *such that*

$$\mathbf{Adv}^{n\text{-SM}}_{\mathsf{SM},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA},\Delta_{\mathsf{CKA}}}(\mathcal{B}) + \mathbf{Adv}^{q_E\text{-FS}}_{\mathsf{FS\text{-}AEAD}}(\mathcal{C}) + \frac{(q_{\mathsf{RO}} + n + q_E)^2 + q_{\mathsf{RO}} q_E}{|\Sigma_{\mathsf{root}}|} \ ,$$

where q_{RO} *is the number of queries to random oracle* H *and* $\mathcal{B}$ *makes* q_{RO} *queries to oracle* CHECK. *Further, the running times of* $\mathcal{B}$ *and* $\mathcal{C}$ *are about that of* $\mathcal{A}$.

Proof. Let $\mathcal{A}$ be an adversary in game MS-SM for secure messaging scheme SM (Figure 13). We prove this theorem by describing a sequence of games G_0-G_3 and proceed by game hopping. We let S_i be the event that the final output in game G_i is 1.

GAME G_0. This is the MS-SM security game with respect to $\mathcal{A}$ and n sessions, where H is modeled as a random oracle. Thus,

$$\mathbf{Adv}^{n\text{-SM}}_{\mathsf{SM},\Delta}(\mathcal{A}) = |\Pr[S_0] - 1/2| \ .$$

GAME G_1. This game is identical to game G_0, except the winning condition in the DELIVER$_\mathsf{P}$ oracles for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$, i.e., the lines "$\mathbf{win}_{\mathbf{corr}} \leftarrow \mathbf{true}$", are removed. By the correctness of CKA and FS-AEAD, and construction of SM, we have $\Pr[S_1] = \Pr[S_0]$.

GAME G_2. This game is identical to game G_1, except we disallow any collision on the random oracle H output σ_{root} and the initial σ_{root} values sampled for each session. It follows by a standard birthday argument that:

$$\Pr[S_2] - \Pr[S_1] \leq \frac{(q_{\mathsf{RO}} + n + q_E)^2}{|\Sigma_{\mathsf{root}}|}$$

since the adversary issues q_{RO} queries and the challenger issues $n + q_E$ queries.

GAME G_3. This game is identical to game G_2, except that for those keys $k^{\mathsf{AB}}_{\mathsf{CKA}}$ output by CKA-S and CKA-R that the adversary is not able to trivially derive, the game does not query the random oracle directly, but directly samples random values (which it records for book-keeping and to keep sender and receiver states consistent). In the following, we will also implicitly assume that for such keys, the random oracle has not been queried on $(\sigma_{\mathsf{root}}, *)$ before. (Otherwise, programming the random oracle would fail.) Since we only replace keys, where the state has not been corrupted, σ_{root} is independent of the adversaries view and such a query can have happened only with probability at most $q_{\mathsf{RO}} q_E / |\Sigma_{\mathsf{root}}|$. We then construct an adversary $\mathcal{B}$ against MI-OW-CKA security of CKA such that

$$|\Pr[S_2] - \Pr[S_3]| \leq \mathbf{Adv}^{n\text{-OW-CKA}}_{\mathsf{CKA},\Delta_{\mathsf{CKA}}}(\mathcal{B}) + \frac{q_{\mathsf{RO}} q_E}{|\Sigma_{\mathsf{root}}|} \ .$$

Recall $\mathcal{B}$ has access to oracles SEND$_\mathsf{P}(i)$, SENDR$_\mathsf{P}(i, r)$, CORR$_\mathsf{P}(i)$, RECEIVE$_\mathsf{P}(i)$ and for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$ as well as CHECK(i, t, k), plus has access to random oracle H for which queries are of the form $(\sigma_{\mathsf{root}}, k_{\mathsf{FS}}) \leftarrow \mathsf{H}(\sigma_{\mathsf{root}}, k)$. $\mathcal{B}$ simulates as follows; we assume if not specified that for a call with input ℓ that we are simulating only for that session, and that calls that fail due to **req** conditions being false are handled directly:

- For each ℓ, $\mathcal{B}$ simulates lines (00)-(09) of Figure 12 (which are the same in G_2 and G_3) locally, except that $\mathcal{B}$ does not simulate $k^{\mathsf{AB}}_{\mathsf{CKA}}$, and simulates each call to CKA-Init-A and CKA-Init-B by marking the CKA states γ as blank for now. In particular, $\mathcal{B}$ samples the same bit $b^* \leftarrow\!\!\$\ \{0, 1\}$ for all ℓ to simulate with hereafter. $\mathcal{B}$ then initialises a map $keys[\cdot] \leftarrow \lambda$ (used across all sessions).
- TRANSMIT$_\mathsf{P}(\ell, \mathsf{msg}, r)$: For A, if t_{cur} is even (resp. odd for B), $\mathcal{B}$ simulates as follows. If $r = \bot$, $\mathcal{B}$ calls SEND$_\mathsf{P}(\ell)$, which outputs message m. $\mathcal{B}$ stores $keys[\sigma_{\mathsf{root}}] \leftarrow (\ell, t_{\mathsf{cur}})$, then simulates $(\sigma_{\mathsf{root}}, k_{\mathsf{FS}}) \leftarrow \mathsf{H}(\sigma_{\mathsf{root}}, k)$ by marking k as blank (observe σ_{root} was previously simulated either in a SM-Send/ SM-Rcv call or at initialisation). $\mathcal{B}$ otherwise simulates locally. If $r \neq \bot$, $\mathcal{B}$ instead simulates with SENDR$_\mathsf{P}(\ell, r)$. Otherwise (t_{cur} odd/even for A/B), $\mathcal{B}$ simulates locally.
- CHALL$_\mathsf{P}(\ell, \mathsf{msg}_0, \mathsf{msg}_1, r)$: $\mathcal{B}$ simulates locally as in TRANSMIT$_\mathsf{P}(\ell, \mathsf{msg}_{b^*}, r)$ modulo differences in the oracles in terms of local variables.
- CORR$_\mathsf{P}(\ell)$: Consider variable t_{cur} associated with P in session ℓ. $\mathcal{B}$ calls its own oracle $\gamma \leftarrow$ CORR$_\mathsf{P}(\ell)$ and otherwise simulates locally. Before returning the state to $\mathcal{A}$ (and hereafter implicitly), $\mathcal{B}$ derives here any keys k that can be trivially derived from knowledge of γ, and for such calls, programs $\mathsf{H}(\sigma_{\mathsf{root}}, k)$ for the corresponding σ_{root}.

- $\textsc{Deliver}_{\mathsf{P}}(\ell, c)$: $\mathcal{B}$ simulates locally except that it queries $\textsc{Receive}_{\mathsf{P}}(\ell)$ to simulate CKA-R if c is associated with an epoch that P has not previously received a message for.
- $\textsc{Inject}_{\mathsf{P}}(\ell, c)$: Note $c = (h, e)$, where $h = (t, m, \ell)$. If P has previously received a ciphertext w.r.t. epoch t, $\mathcal{B}$ simulates locally, since $\mathcal{B}$ has simulated $v[t]$ locally. Otherwise, this is P's first epoch t ciphertext (assume $\mathsf{P} = \mathsf{B}$; the other case is identical):
  - If m was output by a previous call to $\textsc{Send}_{\mathsf{A}}(\ell)$ or $\textsc{SendR}_{\mathsf{A}}(\ell, r)$ in epoch t, then $\mathcal{B}$ calls $\textsc{Receive}_{\mathsf{B}}(\ell)$ and otherwise simulates locally (in future calls to $\textsc{Inject}_{\mathsf{B}}(\ell, c')$ or $\textsc{Deliver}_{\mathsf{B}}(\ell, c')$ for the same m, do not re-call $\textsc{Receive}_{\mathsf{B}}(\ell)$).
  - Otherwise, m was not previously output by $\textsc{Send}_{\mathsf{A}}(\ell)$ or $\textsc{SendR}_{\mathsf{A}}(\ell, r)$ in epoch t. In this case, note that we assumed that $\mathcal{A}$ has not previously called $\mathsf{H}(\sigma_{\mathsf{root}}, k')$ for any k'. Thus, $\mathcal{B}$ simulates $(\sigma_{\mathsf{root}}, k_{\mathsf{FS}}) \leftarrow \mathsf{H}(\sigma_{\mathsf{root}}, k)$ simply by marking k as blank and simulates the rest of the query; if the SM-Rcv call fails, roll back the changes made to H.
- $\mathsf{H}(\sigma_{\mathsf{root}}, k)$: If $keys[\sigma_{\mathsf{root}}] = \bot$, $\mathcal{B}$ simulates by lazy sampling when needed. Else, let $(i, t) = keys[\sigma_{\mathsf{root}}]$. $\mathcal{B}$ then calls $\textsc{Check}(i, t, k)$: the simulation ends if this query is successful if the key was honestly generated. Note by definition of G_3 that no collision is possible on output σ_{root}; consequently, each value in $keys[\sigma_{\mathsf{root}}]$ corresponds to a unique CKA message m. If Check returns **false** or the input came from a trivial injection, $\mathcal{B}$ lazily samples the output of H as needed. Note that in total $\mathcal{B}$ issues at most q_{RO} queries to Check.

This concludes the description of $\mathcal{B}$.

Finally, we bound the last game by constructing an adversary $\mathcal{C}$ for the MI-FS game such that $|\Pr[S_3] - 1/2| \leq \mathbf{Adv}^{q_E\text{-}\mathsf{FS}}_{\mathsf{FS\text{-}AEAD}}(\mathcal{C})$, where q_E is the total number of epochs across all instances.

Since those keys k_{FS} that the adversary cannot trivially compute are now uniform, we can now simulate via MI-FS. To this end, let $\mathcal{A}$ be an adversary against the SM scheme in game G_3. We construct adversary $\mathcal{C}$ as follows:

- It does not pick a random bit b at the beginning of the game.
- When the $\textsc{Corr}_{\mathsf{A}}$ oracle is queried, then $\mathcal{C}$ makes all necessary corruption queries. $\mathcal{C}$ integrates the obtained FS-AEAD state into its current SM state, replacing the simulated FS-AEAD state, and returns the full SM state to $\mathcal{A}$.
- When the $\textsc{Transmit}_{\mathsf{A}}$ oracle is queried, $\mathcal{C}$ makes the corresponding query to its own $\textsc{Transmit}_{\mathsf{A}}$ oracle. The returned value from this query will be used to simulate `record` and returned to $\mathcal{A}$.
- When $\textsc{Chall}_{\mathsf{A}}$ is queried, $\mathcal{C}$ makes the corresponding query to its own challenge oracle. The returned value from this query will be used to simulate `record` and returned to $\mathcal{A}$.
- When the $\textsc{Deliver}_{\mathsf{P}}$ oracle is queried, $\mathcal{C}$ makes the corresponding query to its own $\textsc{Deliver}_{\mathsf{B}}$ oracle. The returned value from this query will be used to simulate G_4 function `delete` before being returned to $\mathcal{A}$.
- When the $\textsc{Inject}_{\mathsf{P}}$ oracle is queried, $\mathcal{C}$ makes the corresponding query to its own $\textsc{Inject}_{\mathsf{B}}$ oracle. If at any point in time $\mathcal{A}$ issues a query such that $\mathbf{win}_{\mathbf{auth}}$ is set to true, then the same is true for $\mathcal{C}$'s oracle $\textsc{Inject}_{\mathsf{B}}$. Otherwise, the returned value from this query will be used to simulate `delete` and returned to $\mathcal{A}$.

Oracles for role B can be simulated analogously. Note that $\mathcal{C}$ requires at most q_E instances. The theorem now follows by collecting the bounds. □

6 Putting Everything Together

We now want to give an overview and interpretation of the results established in this work. We will start with deriving the final bounds following from the previous sections.

Final Bounds. The following corollary states the multi-session security of the Double Ratchet protocol by combining Theorems 4 and 5 and Corollary 1.

Corollary 4 (Security of DR). *Let n be the number of sessions. Let q_e be an upper bound on the number of epochs per session and q_M be the total number of messages sent across all sessions. Let $|\mathcal{K}_{\mathsf{FS}}| = |\Sigma_{\mathsf{root}}| = 2^{\kappa}$ and $\Delta = 3$. Let H and G be random oracles. Let DR be the Double*

Ratchet scheme (i. e., the SM scheme from Figure 13 instantiated with $\mathsf{CKA}_{\mathsf{DH}}$ *from Figure 6 and the FS-AEAD scheme from Figure 11). For any adversary* $\mathcal{A}$ *in game* MS-SM *for* DR *that issues* q_{RO} *random oracle queries, there exists an adversary* $\mathcal{B}$ *against* StCDH *and an adversary* $\mathcal{C}$ *against* AE *such that*

$$\mathbf{Adv}^{n\text{-}\mathsf{SM}}_{\mathsf{DR},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{n(q_e+1)\text{-A-StCDH-Corr}}_{\mathcal{G}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{q_M\text{-OT-CCA}}_{\mathsf{AE}}(\mathcal{C}) + \frac{4(n+q_E+q_{\mathsf{RO}})^2}{2^\kappa} \ ,$$

where $\mathcal{B}$ *makes at most* q_{RO} *queries to* DDH*. Further, the running times of* $\mathcal{B}$ *and* $\mathcal{C}$ *are about that of* $\mathcal{A}$*. Further,*

$$\mathbf{Adv}^{n(q_e+1)\text{-A-StCDH-Corr}}_{\mathcal{G}}(\mathcal{B}) \leq n(q_e+1) \cdot \mathbf{Adv}^{\mathsf{StCDH}}_{\mathcal{G}}(\mathcal{B}') \ .$$

The first bound is useful to argue about security of the Double Ratchet in the generic group model (GGM), where this bound is actually tight (cf. [KPRR23]). The second bound (via Lemma 1) is with respect to a more standard assumption and improves upon that of ACD19 which (after applying a hybrid argument over the number of sessions) has a loss of nq_e^2 with respect to the DDH assumption. It matches that of BFGMR22 who, however, prove security based on the strong square Diffie-Hellman assumption which implies StCDH only with a non-tight (cube-root) loss [MW96].

Another advantage of our analysis is that the security loss with respect to symmetric-key primitives is expressed with respect to the multi-user variant of the AE scheme, which has been studied for example for AES-GCM [BT16,BHT18,HTT18], and only "counts" the number of messages actually sent, whereas previous proof strategies need to assume an upper bound on the number of messages per epoch. An interesting open question remains whether this bound is indeed optimal.

Our CKA scheme $\mathsf{CKA}_{\mathsf{CS}}$ gives rise to the first tightly-secure messaging scheme. We summarise its security in the following corollary which follows from Theorems 3 to 5. The bounds for the symmetric primitive are the same as in the previous statement.

Corollary 5 (Security of $\mathsf{SM}_{\mathsf{CS}}$). *Let* n *be the number of sessions. Let* q_M *be the total number of messages sent across all sessions. Let* $|\mathcal{K}_{\mathsf{FS}}| = |\Sigma_{\mathsf{root}}| = 2^\kappa$ *and* $\Delta = 2$*. Let* H *and* G *be random oracles. Let* $\mathsf{SM}_{\mathsf{CS}}$ *be the SM scheme from Figure 13 instantiated with* $\mathsf{KEM}_{\mathsf{CS}}$ *from Figure 9 and the FS-AEAD scheme from Figure 11. For any adversary* $\mathcal{A}$ *in game* MS-SM *for* $\mathsf{SM}_{\mathsf{CS}}$ *that issues* q_{RO} *random oracle queries, there exists an adversary* $\mathcal{B}$ *against* DDH *and an adversary* $\mathcal{C}$ *against* AE *such that*

$$\mathbf{Adv}^{n\text{-}\mathsf{SM}}_{\mathsf{SM}_{\mathsf{CS}},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathcal{G}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{q_M\text{-OT-CCA}}_{\mathsf{AE}}(\mathcal{C}) + \frac{4(n+q_E+q_{\mathsf{RO}})^2}{2^\kappa} + \frac{q_{\mathsf{RO}}+1}{p-1} \ ,$$

and the running times of $\mathcal{B}$ *and* $\mathcal{C}$ *are about that of* $\mathcal{A}$*.*

The scheme relies on the DDH assumption and the underlying CKA uses a key encapsulation mechanism, which allows faster healing after compromise (therefore $\Delta = 2$). However, it is also less efficient than the DR scheme. We will further elaborate on this below.

Our generic result for secure messaging schemes from KEMs follows from Theorems 2, 4 and 5 and Corollary 2. Here, we give two bounds.

Corollary 6 (Security of $\mathsf{SM}_{\mathsf{KEM}}$). *Let* n *be the number of sessions. Let* q_E*,* q_M *be the total number of epochs and messages across all sessions, respectively. Let* $|\mathcal{K}_{\mathsf{FS}}| = |\Sigma_{\mathsf{root}}| = 2^\kappa$ *and* $\Delta = 2$*. Let* H *and* G *be random oracles. Let* $\mathsf{SM}_{\mathsf{KEM}}$ *be the SM scheme from Figure 13 instantiated with* $\mathsf{CKA}_{\mathsf{KEM}}$ *from Figure 8 and the FS-AEAD scheme from Figure 11. For any adversary* $\mathcal{A}$ *in game* MS-SM *for* $\mathsf{SM}_{\mathsf{KEM}}$ *that issues at most* q_{RO} *random oracle queries, there exists an adversary* $\mathcal{B}$ *against* KEM *and an adversary* $\mathcal{C}$ *against* AE *such that*

$$\mathbf{Adv}^{n\text{-}\mathsf{SM}}_{\mathsf{SM}_{\mathsf{KEM}},\Delta}(\mathcal{A}) \leq \mathbf{Adv}^{q_E\text{-OW-PCA-Corr}}_{\mathsf{KEM}}(\mathcal{B}) + 2 \cdot \mathbf{Adv}^{q_M\text{-OT-CCA}}_{\mathsf{AE}}(\mathcal{C}) + \frac{4(n+q_E+q_{\mathsf{RO}})^2}{2^\kappa} \ ,$$

where the running times of $\mathcal{B}$ *and* $\mathcal{C}$ *are about that of* $\mathcal{A}$*. Note that in general*

$$\mathbf{Adv}^{q_E\text{-OW-PCA-Corr}}_{\mathsf{KEM}}(\mathcal{B}) \leq \min\left\{ q_E \cdot \mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}'),\ e \cdot (q_C+1) \cdot \mathbf{Adv}^{q'_E\text{-OW-PCA}}_{\mathsf{KEM}}(\mathcal{B}'') \right\} \ ,$$

where $e \approx 2.718$ *is Euler's number,* q_C *is the total number of corruptions* $\mathcal{A}$ *makes and* $q'_E = \lfloor (n-1)/(q_E+1) \rfloor$ *Further, the running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Resources						Advantage				
p	t	n_u	n_s	q_e	q_C	DR [ACD19]	TR [BFG+22]	DR	$\mathsf{SM}_{\mathsf{EG}}$	$\mathsf{SM}_{\mathsf{CS}}$
2^{256}	2^{60}	2^{30}	2^{10}	2^{15}	2^{15}	2^{-126}	2^{-141}	2^{-141}	2^{-180}	2^{-196}
2^{256}	2^{80}	2^{32}	2^{12}	2^{15}	2^{20}	2^{-102}	2^{-117}	2^{-117}	2^{-155}	2^{-176}
2^{256}	2^{100}	2^{32}	2^{15}	2^{15}	2^{25}	2^{-79}	2^{-94}	2^{-94}	2^{-130}	2^{-156}
2^{384}	2^{60}	2^{30}	2^{10}	2^{15}	2^{15}	2^{-254}	2^{-269}	2^{-269}	2^{-308}	2^{-324}
2^{384}	2^{80}	2^{32}	2^{12}	2^{15}	2^{20}	2^{-230}	2^{-245}	2^{-245}	2^{-283}	2^{-304}
2^{384}	2^{100}	2^{32}	2^{15}	2^{15}	2^{25}	2^{-207}	2^{-222}	2^{-222}	2^{-258}	2^{-284}

(a) Advantage bounds for different sizes of the group p, the adversary's running time t, number of users n_u, maximum number of sessions per user n_s (i.e., $n = n_u \cdot n_s$), number of epochs per session q_e, and number of *total* corruptions q_C (across sessions). We compute $Loss \cdot \mathbf{Adv}/t$ and use $\mathbf{Adv} = t^2/p$ for (single-instance) DH problems.

Instantiation		Cost		
Scheme	Curve	Exp	Time	Size
DR/TR	P-256	3	0.12 ms	32 bytes
$\mathsf{SM}_{\mathsf{EG}}$	P-256	4	0.16 ms	64 bytes
$\mathsf{SM}_{\mathsf{CS}}$	P-256	7	0.28 ms	96 bytes
DR/TR	P-384	3	0.27 ms	48 bytes
$\mathsf{SM}_{\mathsf{EG}}$	P-384	4	0.36 ms	96 bytes
$\mathsf{SM}_{\mathsf{CS}}$	P-384	7	0.63 ms	144 bytes

(b) Number of exponentiations (Exp), computation complexity (Time) and communication complexity (Size), focusing on public-key operations and messages (i.e., the underlying CKA scheme).

Table 1: Comparison of secure messaging schemes when instantiated over NIST curves P-256 and P-384: the Double Ratchet (DR), the Triple Ratchet, and secure messaging based on ElGamal ($\mathsf{SM}_{\mathsf{EG}}$) and Cramer-Shoup ($\mathsf{SM}_{\mathsf{CS}}$). In the upper table, text highlighted in green/orange/red means that the target security level (128 bit for P-256 and 192 bit security for P-384) is achieved/almost achieved/not achieved. In the lower table, the colors indicate whether such an instantiation is theoretically sound in a medium-scale scenario.

The first bound is tight with respect to a multi-user one-way secure KEM under key checking attacks and corruptions. Our result thus allows to construct tightly-secure messaging schemes from other than group-based assumptions. In particular, the notion is achieved by the lattice-based KEM from Pan, Wagner and Zeng [PWZ23a] which relies on the Learning with Errors (LWE) assumption. Unfortunately, the security notion is still quite strong, therefore we give a simpler (non-tight) bound as well, which only assumes either OW-PCA security of the KEM (first term) or multi-user security *without* corruptions (second term). The latter statement follows from [BRTZ24] and the bound (which depends on the number of corruptions) improves the fewer corruptions we have. Note that ElGamal tightly achieves MU-OW-PCA security assuming StCDH, which yields an overall tightness loss of q_C. We leave it open to determine how to apply the framework of [BRTZ24] to the MU-A-StCDH-Corr assumption – due to game's structure of only allowing challenges w.r.t. adjacent keys, one cannot directly parse it as a multi-user game in the framework. Further, we note that both single-user and multi-user notions for KEMs have also been analyzed in the (quantum) random oracle model [HHK17,DHK+21].

6.1 Evaluation

Advantage Bounds. In Table 1a, we compute the advantage for different settings to see how the bounds change when the adversary's running time and the number of sessions increase. We compare the schemes covered by the corollaries above with previous work: the Double Ratchet DR (both the original $\mathsf{ACD19}$ bound and the improved bound), the Triple Ratchet [BFG+22] (discussed further in Section 7.2), the KEM-based scheme instantiated with ElGamal $\mathsf{SM}_{\mathsf{EG}}$ and the tightly secure KEM-based scheme $\mathsf{SM}_{\mathsf{CS}}$.

For increased precision, we split the number of sessions into the number of actual users n_u and the number of sessions per user n_s. We note that this not only includes multiple conversations a user has, but also captures groups and duplicate sessions due to session handling. Our choice of n_u is reasonable in light of the fact that the Double Ratchet is used by almost 3 billion unique monthly active WhatsApp users alone.[2] Our choice of n_s assumes users have contact with (via groups or two-party chats) some hundreds of users; very conservatively, one could even justify choosing $n_u \approx n_s$, although we do not do this. In our comparison, we expect that the number of epochs per session stays around the same due to session handling and expiration. We also consider the improved bounds of the ElGamal-based scheme that depends on the total number of corruptions q_C, for which we assume that a large number of users may be corrupted (but still less than n_u).

We consider three settings: small-scale, medium-scale and large-scale (cf. also Figure 1), where resources increase from small- to large-scale.[11] We can see that for the original bound for DR from [ACD19], 128 bits of security is not achieved in any of these settings when using a group of size 2^{256} (e.g., NIST curve P-256), while it is met in at least the small-scale setting with our improved analysis. It is also worth noting that when considering security in the generic group model, the same bound as for $\mathsf{SM}_{\mathsf{CS}}$ can be achieved, which meets the target in all settings. Further, when using a larger curve (such as P-384), all schemes meet their target security level (192 bits).

Computational and Communication Complexity. In the following, we also want to compare the efficiency of the different schemes when taking the tightness loss described above into account. We will mainly focus on the public-key primitive, namely the underlying CKA scheme, since the loss affects the efficiency of those more than for symmetric-key primitives. We evaluate the schemes in terms of ciphertext size and computation speed.

We evaluate the computation complexity by counting exponentiations. To compare, we use the numbers from running the command `openssl speed ecdh` on a MacBook Air 2022 (Apple M2, 16 GB of RAM and macOS Sonoma 14.5). In our evaluation, an exponentiation takes 0.04ms on P-256 and 0.09ms on P-384. We provide numbers for instantiations on both curves P-256 and P-384, but highlight that a theoretically sound instantiation may require the latter curve. All numbers are shown in Table 1b.

The $\mathsf{CKA}_{\mathsf{DH}}$ scheme used in the Double Ratchet and its variant used in Triple Ratchet are the most efficient schemes when counting the number of exponentiations and group elements. However, in large-scale settings, the bounds suggest that they should be instantiated using a larger curve, which makes computation and communication more expensive (e.g., 0.27ms vs. 0.12ms).

The ElGamal-based scheme $\mathsf{CKA}_{\mathsf{EG}}$ requires one exponentiation and one group element more, whereas the Cramer-Shoup based scheme $\mathsf{CKA}_{\mathsf{CS}}$ requires 7 exponentiations and 3 group elements in total. Although $\mathsf{CKA}_{\mathsf{CS}}$ requires more communication, it is comparable with $\mathsf{CKA}_{\mathsf{DH}}$ in running time, since due to its tight proof, it can be soundly instantiated on the smaller curve P-256.

For practical applications, a lower communication cost might nonetheless be desirable. Therefore, the trade-off between tightness and communication/computation complexity requires careful assessment. For example, a planetary-scale service like WhatsApp may consider a modest communication cost increase to be too expensive. Our analysis further supports such a choice when restricting to generic attacks (i.e., when assuming the GGM).

[11] While 2^{100} in computation time may seem large, it is reasonable for large-scale adversaries. For comparison, Bitcoin miners in 2023 alone executed more than 2^{93} SHA-256 hash calls, cf. https://www.blockchain.com/explorer/charts/hash-rate.

7 Discussion

In this section, we first examine the use of the random oracle model in our results, and then explore to what extent tight security can be shown for variants and extensions of the Double Ratchet protocol, namely the Triple Ratchet [BFG+22], the Extended Secure Messaging (eSM) scheme [CZ24] and Signal's sealed sender feature [jlu18].

7.1 On Idealized Models

Like previous work on the tight security of real-world protocols like TLS and SIGMA [DG21,DJ21,DDGJ22], our analysis relies on the random oracle model. To give concrete bounds for post-quantum secure constructions, our analysis is therefore limited, considering that the bounds of ACD19 hold in the standard model. We discuss below the challenges in adapting the ACD19 proof to match our bounds without relying on the ROM, as well as challenges to prove (tight) security in the quantum random oracle model (QROM).

The Security Loss in ACD19. Our proof strategy relies on extracting the solution to a computationally hard problem (i.e., the security of the CKA) from random oracle queries, rather than relying on an indistinguishability notion. The original security proof of ACD19 is in the standard model and incurs a security loss of q_e^2 in the single-session setting. We believe that a tighter proof for ACD19's SM notion seems difficult without different assumptions or to any standard indistinguishability notion. Namely, the reduction cannot commit to embedding a CKA challenge in the SM game without knowing if a compromise will occur, therefore additionally guessing the "last corruption epoch" before each challenge solves this, and we do not see how to avoid this without the ROM.

Recall in Section 4 that we model the hash function responsible for the symmetric key schedule inside of the FS-AEAD as a random oracle G. Instead of doing this, one could define a multi-instance PRG assumption with corruptions that captures this key schedule directly and plausibly obtain a tight reduction. Since in the symmetric ratchet the output of the PRG for a given index has to be used as a PRG secret for the next index, it is unclear whether such a primitive can be instantiated tightly without e.g. a related-key assumption, similar to the ad-hoc pseudorandomness property from [YV20, Definition 5]. An advantage of this approach would be better modularity which may allow for alternative instantiations or at least to defer the random oracle to such a primitive.

Challenges in the QROM. Thanks to advanced technical tools introduced in recent years (e.g., [JZC+18,SXY18,AHU19]), security proofs in the quantum random oracle model (QROM) have become tighter and simpler to write in many settings. Therefore, it is plausible that our composition theorem can be lifted to the QROM using such techniques. However, a careful analysis will be needed to determine the exact bounds and security assumptions required on the underlying building blocks. In particular, adaptive state compromise might hinder us from giving an efficient and completely tight construction from standard post-quantum assumptions at all. This is because no such KEM that supports adaptive corruptions is known to date.[12] Related works that study the tight security of authenticated key exchange (AKE) in the QROM [PWZ23b,PRZ24] "accept" a security loss linear in the number of users, while at least avoiding any square-root and session-dependent loss. Translating the AKE setting (where users can be adaptively corrupted) to secure messaging (where each epoch's state can be revealed), we therefore generally expect at least a security loss in the total number of epochs, similar to the current bounds for the Double Ratchet, where the final bound will also depend on the concrete security of the KEM.

7.2 The Triple Ratchet

We first describe the Triple Ratchet scheme from BFGMR22 [BFG+22]. We believe that our proof strategy can be extended to this setting, as we explain below. Since the main technical difference

[12] A KEM and signature scheme from LWE with almost tight security in the standard model were given in [HLWG23]. These are, however, "not quite practical at the moment" [HLWG23].

between the Double Ratchet and the Triple Ratchet [BFG+22] lies in the CKA, the discussion below focuses on the tightness of the CKA^+ scheme rather than the full SM scheme.

The difference between the Triple Ratchet and the Double Ratchet is that CKA states are computed differently. More specifically, in the Triple Ratchet, when the sender retrieves h_{i-1} from its state and picks exponent x_i to compute the CKA key $k = h^{x_i}$, it does not store x_i for the next epoch, but instead derives $x'_i \coloneqq x_i \cdot \mathsf{H}'(k)$, where H' is a random oracle. The receiver gets g^{x_i} and sets his Diffie-Hellman share in the state to $h_i = g^{x_i \cdot \mathsf{H}'(k)}$.

One benefit of the scheme is that it achieves a slightly stronger notion. For this, the authors of BFGMR22 strengthen the CKA notion (which they call CKA^+) and the SM notion (which is formulated in the UC framework) to capture the above observation. They further prove security of the Triple Ratchet under the StCDH assumption and in the ROM and Ideal Cipher Model, the latter being required to explain ciphertexts after state exposure [Nie02] for full UC security.

The security notions for CKA and CKA^+ in BFGMR22 are closer to ours than ACD19 in that they are already one-way notions with a checking oracle, albeit in the single session setting, which combined with our use of the ROM allows us to achieve tight security from a multi-session CKA notion. Then for CKA^+, their bounds w.r.t. StCDH are the same as ours for the Double Ratchet in the single-session setting (whereas their bounds for CKA rely on the stronger StSqCDH assumption). Nonetheless, a better reduction to StCDH does not seem possible. This is because the strengthening of the model from CKA to CKA^+ only differs in what is allowed *after* a challenge, but the commitment problem arises when embedding the Diffie-Hellman challenge *before* the adversary's challenge in the SM game. More concretely, note that in the above we can write h_{i-1} as

$$h_{i-1} = g^{x'_{i-1}} \qquad \text{and} \qquad x'_{i-1} = x_{i-1} \cdot \mathsf{H}'(k')\,,$$

where $k' = g^{x'_{i-2}x_{i-1}}$ is used to derive the receiver's previous state x'_{i-1}, and x'_{i-2} is the sender's state before h_{i-1}. Further, the CKA key for epoch i is

$$k = g^{x'_{i-1}x_i} = g^{x_{i-1}x_i \cdot \mathsf{H}'(k')}\,.$$

Now assume the adversary wants to send a challenge that uses an AEAD key derived from k. The only requirement is that the adversary has not revealed x'_{i-1}. Note that it cannot explicitly reveal x_i because it is never stored. (The only way to reveal it is by computing k and revealing x'_i.) However, the adversary is allowed to reveal x'_{i-2}, from which it can compute $\mathsf{H}'(k')$. This way, the CKA key k is uniquely determined by $g^{x_{i-1}}$ and g^{x_i}.

The above gives an intuition why a tighter proof for the Triple Ratchet from standard assumptions seems as hard as one for the Double Ratchet; we leave a formal study for future work. At the same time, a reduction using the MU-A-StCDH-Corr problem seems possible (even in the stronger model of BFGMR22). The idea would be to use the reduction's input as the states $(g^{x_0}, g^{x'_1}, g^{x'_2}, \ldots)$ and send CKA messages $m_i = g^{x'_i/r_i}$ for random $r_i \leftarrow\!\!\$\ \mathbb{Z}_p$. Then the output of $\mathsf{H}'(g^{x'_{i-1}x_i}) = \mathsf{H}'(g^{x'_{i-1}x'_i/r_i})$ can be programmed to be r_i if all previous states have been revealed and the Diffie-Hellman secret can be computed by the adversary. However, to efficiently recognize a solution to the MU-A-StCDH-Corr game (without requiring to check all i'), it might be necessary to additionally include $g^{x'_{i-1}}$ or g^{x_i} into the hash.

7.3 Extensions with Public-Key Cryptography

Extended Secure Messaging. A Public-Key Secure Messaging (PKSM) scheme was already suggested by ACD19, combining CKA, FS-AEAD and PRF-PRNG with a KEM and signature scheme. However, they do not give a formal security proof. Recently, Cremers and Zhao [CZ24] define a similar primitive which they call extended secure messaging (eSM). This primitive additionally has long-term keys and pre-keys, offering strong resilience against fine-grained secret compromise. Instead of using the abstract building blocks from ACD19, they construct an eSM scheme directly from KEMs and signatures, as well as standard symmetric primitives. The latter consist of a one-time CCA secure AEAD and five key derivation functions with different properties: apart from standard PRFs and PRGs, they rely on a dual PRF [Bel06] and a triple PRF, where at least one input acts as a key. Their proof incurs a security loss of $nq_e^3q_m$, where n and q_e are defined as before and q_m is the number of pre-keys (cf. also Figure 2).

An interesting question is whether tighter bounds are possible in the (Q)ROM. In the ROM, it seems plausible that multi-user OW-PC(V)A security with corruptions for KEMs is sufficient. (Applying a KDF that is modelled a random oracle allows one to relax security from IND-CCA to OW-PCVA security.) At the same time, multi-user unforgeability with corruptions for signatures will be required since a state corruption will leak the signing key. This notion is typically used to give tight security bounds for AKE, e.g., [GJ18,DG21,DJ21]. It therefore remains to investigate the exact trade-off between tightness and performance of the instantiated scheme, as well as bounds in the QROM.

Sealed Sender. In practice, Signal supports *sealed sender* [jlu18], which aims at (but does not fully succeed at [MKA+21]) hiding the identity of a sender from Signal's servers. This has been abstracted away in previous (and our) cryptographic analysis of the Double Ratchet. By default, this feature is enabled between mutual contacts.

Essentially, every time Alice wants to message Bob, Alice encrypts her Double Ratchet ciphertext with a key derived from a value g^{ab}, where g^a is an ephemeral secret sampled every message by Alice, and g^b is a long-term identity key of Bob (the other details are less relevant for a tightness analysis). This in fact provides additional security – for example, unless the adversary exposes Bob's identity key, it cannot decrypt messages sent to Bob, and is reminiscent of the approach taken by Cremers and Zhao.

It is likely that tight security can be shown in a 'reasonable' model capturing this functionality, plausibly under the k-A-StCDH-Corr assumption where the adversary makes q send oracles queries and $k = O(q)$. We leave a formal investigation to future work.

8 Conclusion

Due to the high relevance for practical applications, we expect an increasing interest in improving the concrete security of secure messaging protocols in the future. Our analysis provides the first step in this direction and provides concrete security bounds for the Double Ratchet scheme and its KEM-based variant suggested by Alwen, Coretti and Dodis [ACD19]. While our bounds for the Double Ratchet are not tight under standard assumptions, our analysis extracts the "core" of this family of schemes by giving appropriate multi-user assumptions. Interesting future directions include proving the optimality of these bounds and extending the analysis to protocols like the Triple Ratchet and the group setting. Further, Apple recently released the hybrid PQ3 messaging protocol [DH23] based on the Double Ratchet, and it is expected that other services will eventually transition to a hybrid approach using KEMs now that CRYSTALS-Kyber [BDK+18] has been selected for standardisation by the NIST.[13] Therefore, an analysis focusing on optimal tightness in the QROM and for schemes with stronger or more fine-grained security guarantees will be interesting in that regard.

Acknowledgments

We thank the anonymous reviewers and Rune Fiedler for their valuable feedback. Part of this work was done during Oliver Tran's semester project at EPFL. Daniel Collins was supported in part by AnalytiXIN and by Sunday Group, Inc., and completed most of this work while working at EPFL. Doreen Riepel was supported in part by Mihir Bellare's KACST grant.

References

AAB+21. Joël Alwen, Benedikt Auerbach, Mirza Ahad Baig, Miguel Cueto Noval, Karen Klein, Guillermo Pascual-Perez, Krzysztof Pietrzak, and Michael Walter. Grafting key trees: Efficient key management for overlapping groups. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 222–253. Springer, Cham, November 2021.

[13] https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization

ACD19. Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Cham, May 2019.

ACDT20. Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 248–277. Springer, Cham, August 2020.

ACDT21. Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Modular design of secure group messaging protocols and the security of MLS. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1463–1483. ACM Press, November 2021.

ACJM20. Joël Alwen, Sandro Coretti, Daniel Jost, and Marta Mularczyk. Continuous group key agreement with active security. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 261–290. Springer, Cham, November 2020.

AHU19. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Cham, August 2019.

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Berlin, Heidelberg, May 2000.

BCG23. David Balbás, Daniel Collins, and Phillip Gajland. WhatsUpp with sender keys? Analysis, improvements and security proofs. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 307–341. Springer, Singapore, December 2023.

BDG+22. Alexander Bienstock, Yevgeniy Dodis, Sanjam Garg, Garrison Grogan, Mohammad Hajiabadi, and Paul Rösler. On the worst-case inefficiency of CGKA. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part II*, volume 13748 of *LNCS*, pages 213–243. Springer, Cham, November 2022.

BDK+18. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *EuroS&P*. IEEE, 2018.

Bel06. Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Berlin, Heidelberg, August 2006.

BFG+22. Alexander Bienstock, Jaiden Fairoze, Sanjam Garg, Pratyay Mukherjee, and Srinivasan Raghuraman. A more complete analysis of the Signal double ratchet algorithm. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 784–813. Springer, Cham, August 2022.

BHJ+15. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 629–658. Springer, Berlin, Heidelberg, March 2015.

BHT18. Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, Cham, April / May 2018.

BJLS16. Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 273–304. Springer, Berlin, Heidelberg, May 2016.

BR94. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Berlin, Heidelberg, August 1994.

BR95. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th ACM STOC*, pages 57–66. ACM Press, May / June 1995.

BRT23. Alexander Bienstock, Paul Rösler, and Yi Tang. ASMesh: Anonymous and secure messaging in mesh networks using stronger, anonymous double ratchet. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1–15. ACM Press, November 2023.

BRTZ24. Mihir Bellare, Doreen Riepel, Stefano Tessaro, and Yizhao Zhang. Count corruptions, not users: Improved tightness for signatures, encryption and authenticated key exchange. In *ASIACRYPT 2024*, LNCS, December 2024.

BRV20. Fatih Balli, Paul Rösler, and Serge Vaudenay. Determining the core primitive for optimally secure ratcheting. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 621–650. Springer, Cham, December 2020.

BSJ+17. Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 619–650. Springer, Cham, August 2017.

BT16. Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, Berlin, Heidelberg, August 2016.

CCD+20. Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, October 2020.

CCG+19. Katriel Cohn-Gordon, Cas Cremers, Kristian Gjøsteen, Håkon Jacobsen, and Tibor Jager. Highly efficient key exchange protocols with optimal tightness. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 767–797. Springer, Cham, August 2019.

CHK21. Cas Cremers, Britta Hale, and Konrad Kohbrok. The complexities of healing in secure group messaging: Why cross-group effects matter. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 1847–1864. USENIX Association, August 2021.

CJN23. Cas Cremers, Charlie Jacomme, and Aurora Naska. Formal analysis of session-handling in secure messaging: Lifting security from sessions to conversations. In *Usenix Security*, 2023.

CJSV22. Ran Canetti, Palak Jain, Marika Swanberg, and Mayank Varia. Universally composable end-to-end secure messaging. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 3–33. Springer, Cham, August 2022.

CRT24. Daniel Collins, Doreen Riepel, and Si An Oliver Tran. On the Tight Security of the Double Ratchet. In *ACM CCS 2024*. ACM Press, 2024.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Berlin, Heidelberg, August 1998.

CZ24. Cas Cremers and Mang Zhao. Secure messaging with strong compromise resilience, temporal privacy, and immediate decryption. In *S&P*. IEEE, 2024.

DDGJ22. Hannah Davis, Denis Diemert, Felix Günther, and Tibor Jager. On the concrete security of TLS 1.3 PSK mode. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 876–906. Springer, Cham, May / June 2022.

DG19. Nir Drucker and Shay Gueron. Continuous key agreement with reduced bandwidth. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 33–46. Springer, 2019.

DG21. Hannah Davis and Felix Günther. Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21International Conference on Applied Cryptography and Network Security, Part II*, volume 12727 of *LNCS*, pages 448–479. Springer, Cham, June 2021.

DH23. Benjamin Dowling and Britta Hale. Authenticated continuous key agreement: Active mitm detection and prevention. Cryptology ePrint Archive, Paper 2023/228, 2023. https://eprint.iacr.org/2023/228.

DHK+21. Julien Duman, Kathrin Hövelmanns, Eike Kiltz, Vadim Lyubashevsky, and Gregor Seiler. Faster lattice-based KEMs via a generic fujisaki-okamoto transform using prefix hashing. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2722–2737. ACM Press, November 2021.

DJ21. Denis Diemert and Tibor Jager. On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. *Journal of Cryptology*, 34(3):30, July 2021.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013.

EM19. Ksenia Ermoshina and Francesca Musiani. "standardising by running code": the signal protocol and de facto standardisation in end-to-end encrypted messaging. *Internet Histories*, 3(3-4):343–363, 2019.

GJ18. Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 95–125. Springer, Cham, August 2018.

Goo22. Google. Messages end-to-end encryption - overview: Technical paper. version 1.2. https://www.gstatic.com/messages/papers/messages_e2ee.pdf, February 2022.

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 341–371. Springer, Cham, November 2017.

HJK+21. Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan, Doreen Riepel, and Sven Schäge. Authenticated key exchange and signatures with tight security in the standard model. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 670–700, Virtual Event, August 2021. Springer, Cham.

HLG21. Shuai Han, Shengli Liu, and Dawu Gu. Key encapsulation mechanism with tight enhanced security in the multi-user setting: Impossibility result and optimal tightness. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 483–513. Springer, Cham, December 2021.

HLWG23. Shuai Han, Shengli Liu, Zhedong Wang, and Dawu Gu. Almost tight multi-user security under adaptive corruptions from LWE in the standard model. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 682–715. Springer, Cham, August 2023.

HTT18. Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1429–1440. ACM Press, October 2018.

JKRS21. Tibor Jager, Eike Kiltz, Doreen Riepel, and Sven Schäge. Tightly-secure authenticated key exchange, revisited. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 117–146. Springer, Cham, October 2021.

jlu18. jlund. Technology preview: Sealed sender for signal, 2018.

JS18. Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 33–62. Springer, Cham, August 2018.

JZC+18. Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 96–125. Springer, Cham, August 2018.

KPRR23. Eike Kiltz, Jiaxin Pan, Doreen Riepel, and Magnus Ringerud. Multi-user CDH problems and the concrete security of NAXOS and HMQV. In Mike Rosulek, editor, *CT-RSA 2023*, volume 13871 of *LNCS*, pages 645–671. Springer, Cham, April 2023.

KS23. Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol, 2023. https://signal.org/docs/specifications/pqxdh/pqxdh.pdf.

Mar16. Moxie Marlinspike. The double ratchet algorithm. https://signal.org/docs/specifications/doubleratchet/, November 2016.

Mau05. Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005.

Met17. Meta. Messenger secret conversations: Technical whitepaper. version 2.0. https://about.fb.com/wp-content/uploads/2016/07/messenger-secret-conversations-technical-whitepaper.pdf, May 2017.

MKA+21. Ian Martiny, Gabriel Kaptchuk, Adam J Aviv, Daniel S Roche, and Eric Wustrow. Improving signal's sealed sender. In *NDSS*, 2021.

MW96. Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer, Berlin, Heidelberg, August 1996.

Nie02. Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, Berlin, Heidelberg, August 2002.

PR18. Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 3–32. Springer, Cham, August 2018.

PRZ24. Jiaxin Pan, Doreen Riepel, and Runzhi Zeng. Key exchange with tight (full) forward secrecy via key confirmation. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 59–89. Springer, Cham, May 2024.

PWZ23a. Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Lattice-based authenticated key exchange with tight security. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 616–647. Springer, Cham, August 2023.

PWZ23b. Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Tighter security for generic authenticated key exchange in the QROM. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part IV*, volume 14441 of *LNCS*, pages 401–433. Springer, Singapore, December 2023.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

Sig24. Signal repository, 2024. https://github.com/signalapp.

Ste24. Douglas Stebila. Security analysis of the imessage pq3 protocol. Cryptology ePrint Archive, Paper 2024/357, 2024. https://eprint.iacr.org/2024/357.

SXY18. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 520–551. Springer, Cham, April / May 2018.

Wha23. WhatsApp. Whatsapp encryption overview: Technical white paper. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, 2023.

YV20. Hailun Yan and Serge Vaudenay. Symmetric asynchronous ratcheted communication with associated data. In Kazumaro Aoki and Akira Kanaoka, editors, *IWSEC 20*, volume 12231 of *LNCS*, pages 184–204. Springer, Cham, September 2020.