A short version of this paper appears in the *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*, DOI: 10.1145/3460120.3484814.

The Security of ChaCha20-Poly1305 in the Multi-user Setting

Jean Paul Degabriele[1,2], Jérôme Govinden[2], Felix Günther[3], and Kenneth G. Paterson[3]

[1] Technology Innovation Institute Cryptography Research Center, Abu Dhabi, United Arab Emirates
`jeanpaul.degabriele@tii.ae`

[2] CNS, Technische Universität Darmstadt, Germany
`jerome.govinden@tu-darmstadt.de`

[3] Applied Cryptography Group, ETH Zurich, Switzerland
`mail@felixguenther.info`, `kenny.paterson@inf.ethz.ch`

Abstract. The ChaCha20-Poly1305 AEAD scheme is being increasingly widely deployed in practice. Practitioners need proven security bounds in order to set data limits and rekeying intervals for the scheme. But the formal security analysis of ChaCha20-Poly1305 currently lags behind that of AES-GCM. The only extant analysis (Procter, 2014) contains a flaw and is only for the single-user setting. We rectify this situation. We prove a multi-user security bound on the AEAD security of ChaCha20-Poly1305 and establish the tightness of each term in our bound through matching attacks. We show how our bound differs both qualitatively and quantitatively from the known bounds for AES-GCM, highlighting how subtle design choices lead to distinctive security properties. We translate our bound to the nonce-randomized setting employed in TLS 1.3 and elsewhere, and we additionally improve the corresponding security bounds for GCM. Finally, we provide a simple yet stronger variant of ChaCha20-Poly1305 that addresses the deficiencies highlighted by our analysis.

Keywords: ChaCha20-Poly1305 · Multi-user Security · GCM · Nonce Randomization · AEAD · TLS 1.3 · Tight Security

1 Introduction

ChaCha20-Poly1305 and Galois Counter Mode (GCM) are the two most popular AEAD schemes in use on the Internet today. The TLS 1.3 specification [Res18] mandates (`MUST`) support for AES128-GCM and strongly recommends (`SHOULD`) support for AES256-GCM and ChaCha20-Poly1305. In addition, ChaCha20-Poly1305 is the default AEAD scheme in OpenSSH, WireGuard, OTRv4, and the Bitcoin Lightning Network. GCM owes much of its popularity to its high parallelizability as well as native support for AES and carry-less multiplication on Intel and AMD CPUs, which allow it to run at record speeds. However, in the absence of such hardware support, ChaCha20-Poly1305 wins the race by a significant margin. Accordingly, ChaCha20-Poly1305 is generally the favored choice on mobile phone hardware and Apple's M1 processor. Moreover, ChaCha20-Poly1305 benefits from Intel's new AVX512 instruction set, which has further narrowed the performance gap between the two schemes on Intel hardware.[4] Its performance benefits derive from the minimalistic ARX design behind the ChaCha20 cipher, the amenability of Poly1305 to fast arithmetic, and the parallelizability in each of these components. Another reason for preferring ChaCha20-Poly1305 on generic hardware platforms is that it is easier to implement in constant time. As a consequence, it is less prone to timing side-channel attacks.

ChaCha20-Poly1305 combines the ChaCha20 stream cipher and the one-time MAC Poly1305 into a nonce-based AEAD scheme. Both were designed independently as separate components by Bernstein [Ber08, Ber05a], and Langley later adapted and combined the two into a nonce-based AEAD scheme and proposed its use in TLS [Lan13]. Despite its popularity, ChaCha20-Poly1305 has received very little formal security analysis. The only extant analysis is in a short, unpublished note by Procter [Pro14]. In contrast, GCM has been the subject of several formal security analyses [MV04, IOM12, NOMI15, BT16, LMP17, HTT18]. In particular, [IOM12] identified a flaw in GCM's original security proof in [MV04] and provided a corrected proof which was in turn improved upon in [NOMI15]. Later

[4] https://bench.cr.yp.to/results-stream.html#amd64-manny1024

works [BT16, LMP17, HTT18] have studied the security of GCM in the *multi-user setting*, where an adversary is given access to an encryption oracle and a decryption oracle for many distinct users and the adversary wins if it can break the security of any single user. Having access to encryptions under distinct keys of the same message and the same nonce can, for instance, facilitate key-recovery attacks on block ciphers [Bih02, BMS06]. Accordingly, the multi-user setting captures a very practical concern reflecting how a state-actor adversary can benefit from its ability to eavesdrop on TLS connections *en masse*. In fact, TLS 1.3 includes a nonce-randomization mechanism to mitigate against such multi-user attacks, and the rekeying frequencies for GCM in TLS, DTLS, and QUIC were based on its multi-user security bounds [Res18, RTM21, TT21]. In the specific cases of DTLS and QUIC, the multi-user security bounds are also used to determine the maximum number of failed decryptions that can be tolerated before a session is terminated, since decryption failures are not immediately fatal in these protocols. Moreover, in light of the tighter security bounds presented in [HTT18], the QUIC specification [TT21, Appendix B.1] now allows for larger limits than those prescribed in the TLS 1.3 specification.

Currently, for ChaCha20-Poly1305, the only multi-user security bound available to the IETF is that outlined by Luyks and Paterson in [LP17]. It leverages the single-user security bound from [Pro14] by applying a standard hybrid reduction technique. The resulting multi-user security bound is essentially the single-user bound, where each term is now multiplied by the number of users. However, this simple approach has three significant shortcomings. Firstly, as described in [LMP17], it generally results in loose bounds, leading to overly conservative security parameter estimates in practical protocols. Secondly, it fails to explicitly quantify the adversary's advantage in terms of its local computational resources. This is because in the standard-model security analysis in [Pro14] this aspect is concealed in a term capturing the PRF security of the ChaCha20 block function, which is not as easy to estimate concretely. Thirdly, the security proof described in Procter's note [Pro14] on which [LP17] relies is actually incorrect, as we explain in Appendix A.1. As such, the security of an important and increasingly widely deployed AEAD scheme currently rests on shaky foundations.

In this work, we remedy this state of affairs by presenting a new dedicated multi-user security analysis of ChaCha20-Poly1305, on par with that for GCM [HTT18]. A common misconception is that ChaCha20-Poly1305 is structurally equivalent to GCM but instantiated with different primitives. Accordingly, one might be tempted to think that it suffices to simply reuse GCM's security bound with ChaCha20-Poly1305's parameters. While the two constructions share some similarities, they have a number of important differences that preclude such an approach. In addition, their differences are further accentuated in the multi-user security model. For instance, while all multi-user security treatments of GCM are in the ideal-cipher model, ChaCha20-Poly1305 is better analyzed in the (unkeyed) random permutation model. We elaborate on this choice and the differences between GCM and ChaCha20-Poly1305 in Section 3. Below is a summary of our contributions.

1.1 Contributions

Single-user security. We start off by revisiting Procter's single-user security proof [Pro14]. We point out a flaw in the proof and provide a new proof under the same standard-model assumptions. Our proof retains the same security bound as that originally claimed by Procter. However, fixing the proof is not straightforward and required us to restructure and augment the sequence of games significantly. The single-user security analysis of ChaCha20-Poly1305 is located in Appendix A.

Multi-user security. Our main contribution is a tight multi-user security proof for ChaCha20-Poly1305. Through the security bounds we establish, we expose fundamental differences in its security traits compared to GCM [HTT18]. One case in point is that, while the multi-user security of GCM can be improved by rekeying more frequently, i.e., reducing the limit on the maximum amount of data that can be encrypted under a single key, this does not hold in the case of ChaCha20-Poly1305. This is due to the fact that our security bound does not depend on any per-user parameters—such as the number of queries per user or the amount of data per user. On an intuitive level, this distinction stems from the fact that under the hood ChaCha20 is based on a permutation and not a block cipher in counter mode. We provide attacks matching our bounds, showing that our observations are not just an artefact of weak bounds.

Another point of divergence is that for ChaCha20-Poly1305 the dominant term in the security bound depends on the number of verification queries made by the adversary and the tag length, whereas for GCM the dominant term depends on the total number of encrypted data blocks and the block length. Note that the number of verification queries represents the number of valid forgery attempts made by an adversary. In the context of secure communications protocols, this relates to the number of times that an adversary attempts to insert a new ciphertext into the secure channel.[5] Typical protocols running over reliable transport, like TLS [Res18], shut down upon the first verification error. Protocols running over unreliable transport, like DTLS [RTM21] or QUIC [TT21], however have to tolerate many invalid ciphertexts due to the network behavior [FGJ20], yet their number is still much smaller than the maximum number of messages that can be encrypted. Thus, our analysis shows that these two schemes require protocol designers to tune their security parameters rather differently depending on which scheme they use.

We note that our security proof borrows many ideas from [HTT18], but it also deviates from it substantially and even improves upon it in some respects.

Tightness. We describe attacks to prove the tightness of every term in our security bound. We adapt some of the attacks for GCM and introduce new ones. In particular, adapting the forgery attack on GCM to the case ChaCha20-Poly1305 requires substantially new techniques, due to the different finite field that they employ in their universal hash functions. We describe these attacks in Section 7.1.

Nonce randomization. Next, we turn our attention to the multi-user security of ChaCha20-Poly1305 when combined with the XN transform for nonce randomization that is currently employed in TLS 1.3. Similarly to [HTT18], we extend our multi-user security bound for plain ChaCha20-Poly1305 to this setting using a balls-into-bins argument. Interestingly, applying the balls-into-bins lemma from [HTT18] to our security bound results in a relatively weak security bound—on the order of 2^{-48}. We present an improved balls-into-bins lemma allowing us to translate our main bound to the XN transformed version of ChaCha20-Poly1305 without any degradation in the security bound. As a noteworthy side-effect, our improved bound for XN also eliminates the corresponding term in the bound of [HTT18] for the nonce-randomized version of GCM, enabling tighter advantage bounds than 2^{-48}. We also apply our improved balls-into-bins lemma to translate our main bound to the CN transformed version of ChaCha20-Poly1305. The CN transform, introduced in [HTT18], randomizes the nonce through concatenation with a random string and is used in TLS 1.2 and IPsec. We cover nonce randomization in Sections 7.2 and 7.3.

A stronger variant. As already observed, the dominant term in the security bound for ChaCha20-Poly1305 relates to the ciphertext integrity of the scheme, which in turn depends on the security of the Poly1305 MAC. Indeed, this term dominates all other terms by a substantial margin. Accordingly, our bound shows that the multi-user security of ChaCha20-Poly1305 would be strengthened significantly if Poly1305 were to be replaced with a more secure MAC. In Section 7.5 we explore this possibility by considering a simple variant of ChaCha20-Poly1305 which uses two independently-keyed Poly1305 MAC tags instead of one.

2 Preliminaries

Notation. The empty string is denoted by ε. We denote by $|x|$ the bit length of the bit string x, and by $|x|_n$ its size in n-bit blocks (and the empty string is assigned a block size of 1, i.e., $|x|_n = \max(1, \lceil |x|/n \rceil)$). We call strings whose bit length is a multiple of 8 *byte strings*. For $1 \le i < j \le |x|$, let $x[i{:}j]$ denote the substring of x spanning bit i to bit j inclusive, and $\mathsf{trunc}(x, n)$ denote the first n bits of x, i.e. $\mathsf{trunc}(x, n) = x[1{:}n]$. For any two bit strings x and y, their concatenation is denoted by $x \| y$. When $|x| = |y| = c \cdot n$ for some positive integer c, $x \overset{(n)}{+} y$ and $x \overset{(n)}{-} y$ denote the strings that result from individually adding and subtracting their n-bit subwords modulo 2^n when each word is interpreted as an unsigned integer.

If S is a finite set then $|S|$ denotes its cardinality and $y \leftarrow\!\!\$\, S$ denotes the process of sampling uniformly at random an element from S and assigning it to y. If $\mathcal{A}$ is an algorithm or a procedure, $y \leftarrow \mathcal{A}(x_1, \dots)$ denotes running $\mathcal{A}$ on inputs $x_1, \dots$ and assigning the output to y. By convention, the running time of an adversary is the sum of its running time, the time to answer all of its oracle queries, and the size of its description.

Note that we specify some additional notation, that is specific to the ChaCha20-Poly1305 scheme, at the end of Section 3.

[5] Such a new ciphertext could be obtained by tampering with an existing one, or could result from reordering ciphertexts or creating entirely new ones from scratch.

Game-playing framework. We use the code-based game-playing framework of Bellare and Rogaway [BR06], where a game is a collection of procedures, an adversary is a single procedure and the interaction between a game and an adversary is implicit. An adversary $\mathcal{A}$ playing a game G is given as input the output of the initialize procedure of G, and has oracle access to the other procedures of G. We write $\mathcal{A}^G \Rightarrow y$ for the event that the adversary $\mathcal{A}$ outputs y when playing the game G and "$\mathcal{A}^G$ sets bad" for the event that the adversary $\mathcal{A}$ sets the boolean variable bad to true when playing game G. All variables, sets, and boolean variables are assumed to be initialized, to 0, $\emptyset$, and false, respectively. Unless stated otherwise, all array entries are initialized to $\perp$.

Pseudorandom functions. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function. Let $\mathrm{Func}(\mathcal{X}, \mathcal{Y})$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For any adversary $\mathcal{A}$, we define the PRF advantage of F as

$$\mathsf{Adv}_F^{\mathrm{PRF}}(\mathcal{A}) = \left| \Pr\left[\mathcal{A}^{G_F^{\text{Real-PRF}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G_F^{\text{Ideal-PRF}}} \Rightarrow 1\right] \right|,$$

where games $G_F^{\text{Real-PRF}}$ and $G_F^{\text{Ideal-PRF}}$ are defined in Fig. 1.

```
Game $G_F^{\text{Real-PRF}}$
procedure Initialize
  1: $K \leftarrow\!\$\ \mathcal{K}$
procedure F(X)
  1: return $F(K, X)$

Game $G_F^{\text{Ideal-PRF}}$
procedure Initialize
  1: $f \leftarrow\!\$\ \mathrm{Func}(\mathcal{X}, \mathcal{Y})$
procedure F(X)
  1: return $f(X)$
```

Fig. 1. Games defining the PRF advantage of a keyed function F.

AEAD syntax. A nonce-based authenticated encryption scheme with associated data $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a triple of algorithms:

- The key generation algorithm $\mathcal{K}$ takes no input and returns a secret key K. We overload $\mathcal{K}$ to also represent the key space associated to the key generation algorithm.
- The deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ takes as input a secret key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, associated data $AD \in \mathcal{AD}$, and a message $M \in \mathcal{M}$ and returns a ciphertext $C \in \mathcal{C}$. We require $\mathcal{E}$ to have constant expansion, i.e. for any $(K, N, AD, M) \in (\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M})$, the expansion $t = |\mathcal{E}(K, N, AD, M)| - |M|$ is constant.
- The deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ takes as input a secret key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, associated data $AD \in \mathcal{AD}$, and a ciphertext $C \in \mathcal{C}$ and returns either a message $M \in \mathcal{M}$ or the symbol $\perp$ to indicate an invalid ciphertext.

We refer to the associated sets $\mathcal{K}$, $\mathcal{N}$, $\mathcal{AD}$, $\mathcal{M}$, and $\mathcal{C}$ as the key space, the nonce space, the associated-data space, the message or plaintext space and the ciphertext space, respectively. We require every nonce-based AEAD to satisfy correctness, namely for all $(K, N, AD, M) \in (\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M})$, it must hold that if $C \leftarrow \mathcal{E}(K, N, AD, M)$ then $M \leftarrow \mathcal{D}(K, N, AD, C)$.

AEAD single-user security. We now give an all-in-one security definition for a nonce-based AEAD scheme, as introduced by Rogaway and Shrimpton [RS06]. This combines chosen-plaintext confidentiality and integrity of ciphertexts into a single notion.

We consider an adversary $\mathcal{A}$ having access to an encryption oracle $\mathsf{Enc}(N, AD, M)$ and a decryption oracle $\mathsf{Dec}(N, AD, C)$. Such an adversary is considered nonce-respecting if it never repeats a nonce N in a query to its encryption oracle. It is valid if it never asks a decryption query $\mathsf{Dec}(N, AD, C)$ with C being the output of a previous encryption query $\mathsf{Enc}(N, AD, M)$. Without loss of generality, any valid nonce-respecting adversary will also make no redundant queries.

Definition 2.1 (Single-user AE advantage). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme with expansion t, defined over $(\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C})$. For any valid nonce-respecting adversary $\mathcal{A}$, we define the AE advantage of Π in the single-user setting to be:*

$$\mathsf{Adv}_\Pi^{\mathrm{AE}}(\mathcal{A}) = \Pr\left[\mathcal{A}^{G_\Pi^{\text{Real-AE}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G_\Pi^{\text{Ideal-AE}}} \Rightarrow 1\right],$$

where games $G_\Pi^{\text{Real-AE}}$ and $G_\Pi^{\text{Ideal-AE}}$ are defined in Fig. 2.

```
Game $G_{\Pi}^{\text{Real-AE}}$
procedure Initialize
  1 : $K \leftarrow\!\$\ \mathcal{K}$
procedure Enc$(N, AD, M)$
  1 : $C \leftarrow \mathcal{E}(K, N, AD, M)$
  2 : return $C$
procedure Dec$(N, AD, C)$
  1 : $M \leftarrow \mathcal{D}(K, N, AD, C)$
  2 : return $M$
```

```
Game $G_{\Pi}^{\text{Ideal-AE}}$
procedure Enc$(N, AD, M)$
  1 : $C \leftarrow\!\$\ \{0,1\}^{|M|+t}$
  2 : return $C$
procedure Dec$(N, AD, C)$
  1 : return $\perp$
```

Fig. 2. Games used to define the single-user AE advantage.

AEAD multi-user security. We use the multi-user security definition for authenticated encryption put forward by Bellare and Tackmann [BT16] adapted to the ideal permutation model. Here, the encryption oracle $\text{Enc}(i, N, AD, M)$ and the verification oracle$\text{Vf}(i, N, AD, C)$ can be queired for distinct users (identified by the integer i). The adversary $\mathcal{A}$ is also given access to the ideal permutation π through the oracle $\text{Prim}(X)$ and its inverse $\text{Prim}^{-1}(Y)$.

An adversary is said to be nonce-respecting if it never repeats a pair (i, N) across encryption queries. Throughout, we require that every adversary be nonce-respecting and never asks a verification query $\text{Vf}(i, N, AD, C)$ with C being the output of a previous encryption query $\text{Enc}(i, N, AD, M)$ (since such queries permit trivial wins). Without loss of generality, we will assume that an adversary does not repeat any queries.

At points in our analysis, we will require that an adversary $\mathcal{A}$ be *d-repeating*, meaning that $\mathcal{A}$ does not repeat the same nonce with more than d users in its encryption queries. We say $\mathcal{A}$ is *strongly d-repeating* if it does not repeat the same nonce across all encryption and verification queries for more than d users. Note that if $\mathcal{A}$ is *strongly d-repeating*, then it is also *d-repeating.*

Definition 2.2 (Multi-user AE advantage). *Let $\Pi[\pi] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme constructed from an ideal permutation $\pi : \mathcal{X} \to \mathcal{X}$ with expansion t. We define the multi-user AE advantage of adversary $\mathcal{A}$ against $\Pi[\pi]$ as:*

$$\mathsf{Adv}_{\Pi[\pi]}^{\text{muAE}}(\mathcal{A}) = \Pr\left[\mathcal{A}^{G_{\Pi[\pi]}^{\text{Real-muAE}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G_{\Pi[\pi]}^{\text{Ideal-muAE}}} \Rightarrow 1\right],$$

where games $G_{\Pi[\pi]}^{\text{Real-muAE}}$ and $G_{\Pi[\pi]}^{\text{Ideal-muAE}}$ are defined in Figure 3.

```
Game $G_{\Pi[\pi]}^{\text{Real-muAE}}$
procedure Initialize
  1 : $K_1, K_2, \ldots \leftarrow\!\$\ \mathcal{K}$
procedure Enc$(i, N, AD, M)$
  1 : $C \leftarrow \mathcal{E}(K_i, N, AD, M)$, return $C$
procedure Vf$(i, N, AD, C)$
  1 : $M \leftarrow \mathcal{D}(K_i, N, AD, C)$, return $(M \neq \perp)$
procedure Prim$(X)$
  1 : return $\pi(X)$
procedure Prim$^{-1}(Y)$
  1 : return $\pi^{-1}(Y)$
```

```
Game $G_{\Pi[\pi]}^{\text{Ideal-muAE}}$
procedure Enc$(i, N, AD, M)$
  1 : $C \leftarrow\!\$\ \{0,1\}^{|M|+t}$, return $C$
procedure Vf$(i, N, AD, C)$
  1 : return false
procedure Prim$(X)$
  1 : return $\pi(X)$
procedure Prim$^{-1}(Y)$
  1 : return $\pi^{-1}(Y)$
```

Fig. 3. Games used to define the multi-user AE advantage.

Adversarial parameters. In the following, for an associated adversary $\mathcal{A}$, we will denote by q_e its maximum number of encryption queries and by σ_e an upper bound on its total number of encrypted blocks. We denote by q_v the maximum number of decryption/verification queries of $\mathcal{A}$ and by σ_v an upper bound on its total number of decrypted/verified blocks. Further, let ℓ_m denote the maximum size in t-bit blocks (including associated data) that $\mathcal{A}$ is allowed to query to its encryption and decryption/verification oracles. Finally, for the multi-user setting, p denotes the maximum number of ideal permutation queries and d denotes the maximum number of users queried with the same nonce (during encryption for a d-repeating adversary, or during encryption and verification for a strongly d-repeating adversary).

***Δ*-universal hash functions.** Let $\overline{H} : \mathcal{R} \times \mathcal{D} \to \{0,1\}^t$ be a family of keyed hash functions with key space $\mathcal{R}$, domain $\mathcal{D}$ and digest space $\{0,1\}^t$, for some positive integer t. We will consider hash function families over strings and string pairs. When $\mathcal{D} = \{0,1\}^*$, for any increasing function $\epsilon : \mathbb{N} \to \mathbb{R}^+$, we say that $\overline{H}$ is ϵ-almost $\varDelta$-universal if for all distinct $M, M' \in \{0,1\}^*$ and all $z \in \{0,1\}^t$, it holds that

$$\Pr_{r \leftarrow_{\$} \mathcal{R}}\left[\overline{H}_r(M) = \overline{H}_r(M') \overset{(t)}{+} z\right] \leq \frac{\epsilon\left(\max\left(|M|_t, |M'|_t\right)\right)}{2^t} .$$

Alternatively, when $\mathcal{D} = \{0,1\}^* \times \{0,1\}^*$, for any distinct $(AD, C), (AD', C') \in \{0,1\}^* \times \{0,1\}^*$ and all $z \in \{0,1\}^t$, we require that

$$\Pr_{r \leftarrow_{\$} \mathcal{R}}\left[H_r(AD, C) = H_r(AD', C') \overset{(t)}{+} z\right] \leq \frac{\epsilon\left(\max\left(|AD|_t + |C|_t, |AD'|_t + |C'|_t\right)\right)}{2^t}.$$

H-coefficient technique. The H-coefficient technique [Pat09, CS14] is a method for bounding the advantage of a computationally unbounded adversary $\mathcal{A}$, which wlog can be assumed to be deterministic, attempting to distinguish between a real and an ideal game. The technique focusses on the transcripts generated when $\mathcal{A}$ interacts with the oracles in these games, namely, the sequence of input-output pairs $\tau = ((x_1, y_1), (x_2, y_2), \ldots, (x_q, y_q))$. We use $\mathcal{T}_{\text{ideal}}$ to denote the random variable corresponding to the transcript generated by $\mathcal{A}$ in the ideal game. Then, $\mathsf{P}_{\text{ideal}}(\tau)$ and $\mathsf{P}_{\text{real}}(\tau)$ denote the probabilities that a given transcript τ is generated in the corresponding game when interacting with $\mathcal{A}$. A transcript τ is said to be attainable if there exists an adversary such that the probability of generating τ in the ideal game is strictly greater than 0.

The H-coefficient technique relies on identifying a suitable partition of attainable transcripts, applying the following theorem, and then calculating ϵ_1 and ϵ_2 (for a proof, see [CS14]):

Theorem 2.1 (H-coefficient Technique). *Let $\mathcal{A}$ be a computationally unbounded adversary trying to distinguish between a real game G^{Real} and an ideal game G^{Ideal}. Let $\mathbb{T} = \mathbb{T}_{\text{good}} \sqcup \mathbb{T}_{\text{bad}}$ be a partition of the set of attainable transcripts.*
If there exist $\epsilon_1, \epsilon_2 \geq 0$ such that

$$\forall \tau \in \mathbb{T}_{\text{good}}, \frac{\mathsf{P}_{\text{real}}(\tau)}{\mathsf{P}_{\text{ideal}}(\tau)} \geq 1 - \epsilon_1 \qquad \textit{and} \qquad \Pr[\mathcal{T}_{\text{ideal}} \in \mathbb{T}_{\text{bad}}] \leq \epsilon_2,$$

then

$$\left|\Pr\left[\mathcal{A}^{G^{\text{Real}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G^{\text{Ideal}}} \Rightarrow 1\right]\right| \leq \epsilon_1 + \epsilon_2.$$

3 The ChaCha20-Poly1305 Scheme

In this section, we provide a brief overview of ChaCha20-Poly1305, as defined in RFC 8439 [NL18], and lay some of the groundwork for our security analysis.

The AEAD composition. Pseudocode for ChaCha20-Poly1305 is shown in Figure 5, consisting of the encryption algorithm $\mathcal{E}$, the decryption algorithm $\mathcal{D}$, and their subcomponents: the ChaCha20 stream cipher, the one-time MAC Poly1305_Mac, and the MAC's key-generation algorithm Poly1305_Key_Gen. In turn, these subcomponents are based on the ChaCha20 block function CC_block and the $\varDelta$-universal hash function family H over string pairs.

The encryption algorithm $\mathcal{E}$ is represented in Figure 4. It takes as input a 256-bit secret key K, a 96-bit nonce N, a variable-length byte string of associated data AD, and a variable-length byte-string

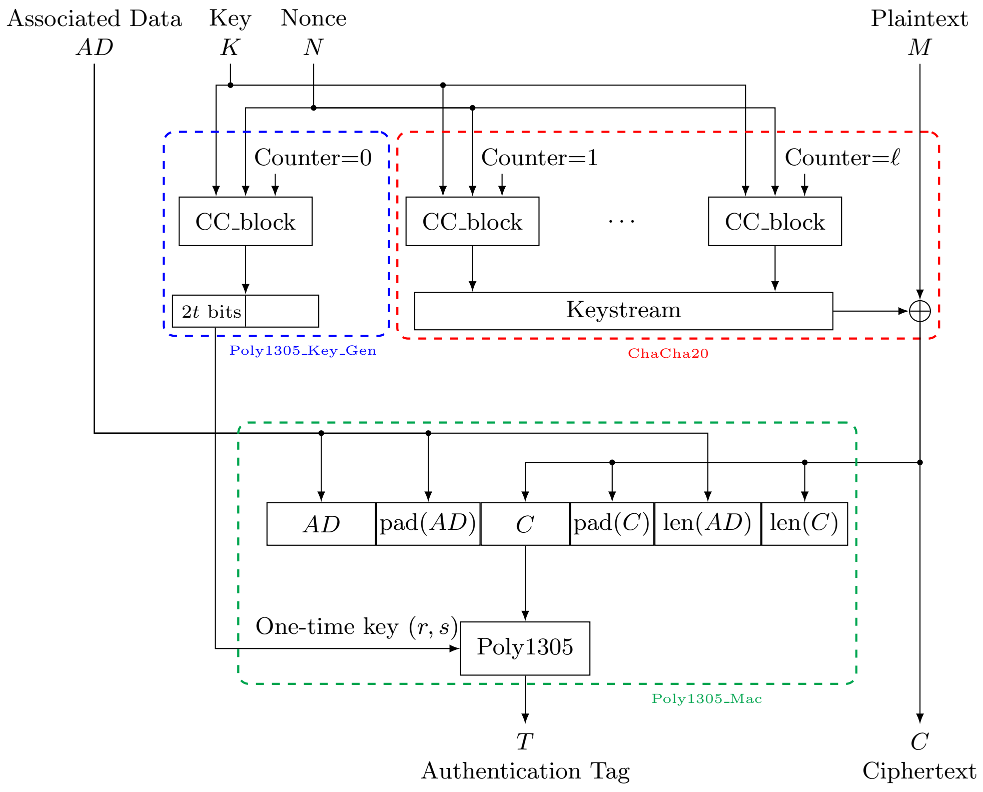

Fig. 4. The encryption procedure in ChaCha20-Poly1305.

message M. It returns a ciphertext C consisting of the ChaCha20 encryption of M, and a 128-bit tag T, computed over AD and C using Poly1305_Mac with a one-time key (r, s) consisting of two 128-bit strings. The one-time key (r, s) is derived anew in each encryption by running the ChaCha20 block function in Poly1305_Key_Gen on K, N, and the counter value zero—which is reserved solely for this purpose. The decryption algorithm $\mathcal{D}$ proceeds analogously: it first derives the one-time key, it recomputes the MAC tag and checks whether it matches that supplied in the ciphertext. If so, it proceeds to decrypt the rest of the ciphertext using ChaCha20 and returns the decrypted message. Otherwise it returns $\perp$, indicating error.

Chacha20. Designed by Bernstein, the ChaCha20 stream cipher [Ber08] is a refinement of the Salsa stream cipher [Ber05b]. It uses a 256-bit secret key K and a 96-bit nonce N to encrypt (or decrypt) an arbitrary-length message M (or ciphertext C). As with any stream cipher, it generates a pseudorandom keystream that is XORed to the message. The keystream is generated in blocks of 512 bits through the ChaCha20 block function CC_block. The CC_block function is keyed with K and is evaluated over an input composed of the 96-bit nonce N and a 32-bit block counter j. This way, it is employed as a pseudorandom function, but under the hood it really consists of a 512-bit permutation π in a Davies–Meyer-type configuration. More specifically, the key, counter, and nonce are concatenated and prepended with a constant to form the input to the ChaCha20 permutation and then added once again to the permutation's output using modulo 2^{32} addition on word-by-word basis, i.e., $\text{CC\_block}(K, N, j) = \pi(Z\|K\|j\|N) \overset{(32)}{+} (Z\|K\|j\|N)$, where Z is a 128-bit constant.

Poly1305. The Poly1305 algorithm [Ber05a] is a one-time MAC, also designed by Bernstein. It takes a key consisting of two 128-bit strings (r, s); the tag corresponding to a message M is computed as $\overline{H}_r(M) \overset{(t)}{+} s$. Its security relies on the $\bar{\epsilon}$-almost Δ-universality of the hash function $\overline{H}$, as shown by Bernstein [Ber05a]. The hash function $\overline{H}$ is outlined in Definition 3.1 and its security is stated in Theorem 3.1, reproduced here from [Ber05a] for completeness.

Definition 3.1 (The Hash Function $\overline{H}$ in Poly1305). *Let t be a positive integer multiple of 8, let $\mathsf{p} \geq 2^{t+1}$ be a prime, let r be a t-bit string and let M be any byte string. Parse M as $M = M_1\| \cdots \|M_\ell$*

```
procedure $\mathcal{E}(K, N, AD, M)$
1 : $r\|s \leftarrow$ Poly1305_Key_Gen$(K, N)$
2 : $C \leftarrow$ ChaCha20$(K, N, M)$
3 : $T \leftarrow$ Poly1305_Mac$((r, s), AD, C)$
4 : return $C\|T$

procedure ChaCha20$(K, N, M)$
1 : $M_1\|\cdots\|M_\ell \leftarrow M$
2 : for $j = 1$ to $\ell - 1$ do
3 :     $C_j \leftarrow M_j \oplus$ CC_block$(K, N, j)$
4 : $C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}($CC_block$(K, N, \ell), |M_\ell|)$
5 : return $C_1\|\cdots\|C_\ell$

procedure $\mathcal{D}(K, N, AD, C\|T)$
1 : $r\|s \leftarrow$ Poly1305_Key_Gen$(K, N)$
2 : $T' \leftarrow$ Poly1305_Mac$((r, s), AD, C)$
3 : if $T \neq T'$ then return $\perp$
4 : return ChaCha20$(K, N, C)$

procedure Poly1305_Key_Gen$(K, N)$
1 : return $\mathsf{trunc}($CC_block$(K, N, 0), 2t)$

procedure Poly1305_Mac$((r, s), AD, C)$
1 : return $H_r(AD, C) \overset{(t)}{+} s$
```

Fig. 5. Pseudocode description of ChaCha20-Poly1305.

where $|M_j| = t$ for $j < \ell$ and $0 < |M_\ell| \leq t$. Then, $\overline{H}_r(M)$ is given as the t-bit string representation of

$$(c_1 x^\ell + c_2 x^{\ell-1} + \cdots + c_\ell x^1 \mod \mathsf{p}) \mod 2^t,$$

where c_j is the integer representation of the $(t+1)$-bit string $M_j\|1$ and x is the integer representation of r after fixing 22 of its bit positions to zero (a process referred to as "clamping").

We refer the reader to [Ber05a] for further details on the bit clamping in r and the conversion between integers and strings. Note that for increased generality in our security analysis, we consider a general tag size t, but for ChaCha20-Poly1305 this should be understood to be equal to 128 bits and $\mathsf{p} = 2^{130} - 5$. Indeed these values give rise to the function $\bar{\epsilon}(\ell) = 2^{25} \cdot \ell$ in the next theorem.

Theorem 3.1 ($\overline{H}$ is AΔU). *Let $\bar{\epsilon}(\ell) = 2^{25} \cdot \ell$, then for any t-bit string s and any pair of distinct byte strings (M, M'), it holds that:*

$$\Pr_{r \leftarrow \$ \{0,1\}^t}\left[\overline{H}_r(M) = \overline{H}_r(M') \overset{(t)}{+} s\right] \leq \frac{\bar{\epsilon}(\max(|M|_t, |M'|_t))}{2^t}.$$

Carefully note here, that this theorem only holds for messages being bytes strings and not for bit strings.

The one-time MAC (Poly1305_Mac) that is used in ChaCha20-Poly1305 (Figure 5) extends the original Poly1305 algorithm to authenticate a pair of strings instead of one. This is accomplished by augmenting the hash function with an appropriate encoding mapping the string pair to a single string. The encoding demarcates the boundary between the two strings. Importantly, this encoding needs to be injective. Definition 3.2 describes how H is constructed from $\overline{H}$. In Theorem 3.2 we show that if $\overline{H}$ is a $\bar{\epsilon}$-almost Δ-universal for single strings, then H is also ϵ-almost Δ-universal over string pairs, where $\epsilon(\ell) = \bar{\epsilon}(\ell + 1)$.

Definition 3.2 (The hash function H in Poly1305_Mac**).** *Let r be a t-bit string and $\overline{H}$ be the hash function used in* Poly1305*. Then, the hash of any pair of byte strings (AD, C) is given by*

$$H_r(AD, C) = \overline{H}_r(AD\|\mathrm{pad}(AD)\|C\|\mathrm{pad}(C)\|\mathrm{len}(AD)\|\mathrm{len}(C)),$$

where $\mathrm{pad}(X)$ *returns the minimum number of zero bytes such that the bit length of $X\|\mathrm{pad}(X)$ is multiple of t, and $\mathrm{len}(X)$ is the $\frac{t}{2}$-bit representation of the byte length of X.*

Theorem 3.2 (H is AΔU). *Let $\epsilon(\ell) = 2^{25} \cdot (\ell + 1)$ and s be a t-bit string, then for any two distinct byte-string pairs (AD, C) and (AD', C') it holds that:*

$$\Pr_{r \leftarrow \$ \{0,1\}^t}\left[H_r(AD, C) = H_r(AD', C') \overset{(t)}{+} s\right] \leq \frac{\epsilon(\max(|AD|_t + |C|_t, |AD'|_t + |C'|_t))}{2^t}.$$

Again, it should be noted here that the theorem only holds for pairs of byte strings as input to H_r and not for bit strings.

Proof. Let

$$M = AD \| \mathrm{pad}(AD) \| C \| \mathrm{pad}(C) \| \mathrm{len}(AD) \| \mathrm{len}(C),$$
$$M' = AD' \| \mathrm{pad}(AD') \| C' \| \mathrm{pad}(C') \| \mathrm{len}(AD') \| \mathrm{len}(C').$$

If $M = M'$, then $\mathrm{len}(C) = \mathrm{len}(C')$ and $\mathrm{len}(AD) = \mathrm{len}(AD')$. Thus $C = C'$ and $AD = AD'$, which is not possible as $(AD, C) \neq (AD', C')$. Therefore $M \neq M'$. Using Theorem 3.1,

$$\begin{aligned}\Pr_{r \leftarrow \$ \{0,1\}^t}\left[\overline{H}_r(M) = \overline{H}_r(M') \overset{(t)}{+} s\right] &\leq \frac{2^{25} \cdot \max\left(|M|_t, |M'|_t\right)}{2^t} \\ &= \frac{2^{25} \cdot \max\left(|AD|_t + |C|_t + 1, |AD'|_t + |C'|_t + 1\right)}{2^t} \\ &= \frac{2^{25} \cdot \left(\max\left(|AD|_t + |C|_t, |AD'|_t + |C'|_t\right) + 1\right)}{2^t}.\end{aligned}$$

□

Notation specific to ChaCha20-Poly1305. In the sections that follow, we will consider a generalized form of the ChaCha20-Poly1305 construction parameterized by n, k, t, μ, ϵ, and $|Z|$. The first four parameters denote the size in bits of the ChaCha20 permutation, the ChaCha20 key K, the Poly1305_Mac tag and hash key r, and the nonce N, respectively. The last two parameters denote the security function of the almost Δ-universal hash function H in Poly1305_Mac and the bit length of the ChaCha20 constant Z. Note, from the ChaCha20-Poly1305 construction, that $n - (|Z| + k + \mu)$ is equal to the bit length of the ChaCha20 counter and $n \geq 2t$. Following RFC8439, and additionally applying Theorem 3.2, the actual values of these parameters are:

$$n = 512, \quad k = 256, \quad t = 128, \quad \mu = 96, \quad \epsilon(\ell) = 2^{25} \cdot (\ell + 1), \quad \text{and} \quad |Z| = 128.$$

For any n-bit string w we use $[w]^{K+}$ to denote the substring of w corresponding to the key portion when w is parsed as an input to the ChaCha20 block function, i.e., $w[|Z| + 1{:}|Z| + k]$. Similarly, we use $[w]^{K\text{-}}$ to denote the complementary portion of the string, i.e., $w[1{:}|Z|] \,\|\, w[|Z| + k + 1{:}n]$. In addition, $[w]^r$ and $[w]^s$ denote respectively the substrings corresponding to the hash key r, i.e., $w[1{:}t]$, and the blinding value s, i.e., $w[t + 1{:}2t]$, when w is parsed as a ChaCha20 block function output.

4 The Single-user Security of ChaCha20-Poly1305

The single-user security of ChaCha20-Poly1305 can of course be derived from its multi-user security theorem (Theorem 6.1) as a special case. A security proof in the single-user setting was already provided by Procter in [Pro14]. However, Procter's proof is in the standard model and thus relies on a different security assumption on ChaCha20 than our multi-user security proof. Unfortunately his proof is incorrect, but we remedy this in Theorem 4.1 where we recover the same security bound under the same assumptions. Our new security proof together with a note describing the issue in [Pro14] can be found in Appendix A.

Theorem 4.1 (Single-user security of ChaCha20-Poly1305**).** *Let n, k, t, ϵ be the parameters of the* ChaCha20-Poly1305 *AEAD scheme. Let $\mathcal{A}$ be a valid nonce-respecting adversary making at most q_v decryption queries, and let ℓ_m be an upper bound on the total size (in t-bit blocks) of every query that it makes. Then there exists a PRF adversary $\mathcal{A}_{prf}$ against the Chacha20 block function* CC_block *such that:*

$$\mathsf{Adv}^{\mathrm{AE}}_{\text{ChaCha20-Poly1305}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{PRF}}_{\text{CC\_block}}(\mathcal{A}_{prf}) + \frac{q_v \cdot \epsilon(\ell_m)}{2^t},$$

where $\mathcal{A}_{prf}$ makes the same number of queries as the number of encryption and decryption queries plus the total number of blocks queried by $\mathcal{A}$.

Note that for small values of ℓ_m this bound is tight, as evidenced by our forgery attack from Proposition 7.1.

The proof of our single-user theorem can be adapted without much modification to the multi-user setting in the standard model, by assuming that the Chacha20 block function CC_block is a secure *multi-user* PRF. The bound obtained would be similar to that in Theorem 4.1, where the PRF advantage would become the multi-user PRF advantage against CC_block and in the second term q_v would then represent the total number of decryption queries across all users.

5 Generalized Balls-Into-Bins

Before delving into the multi-user security of ChaCha20-Poly1305, we present an improved balls-into-bins result. We make extensive use of this result in the proof of our main theorem and in extending the security bound to the nonce-randomization setting. The balls-into-bins problem considers an experiment whereby m balls are thrown into n bins at random. In our case, we will consider a distribution that deviates from the uniform distribution, and we will be concerned with bounding the maximum load, i.e., the maximum number of balls landing in any bin. Maximum load results are known in the asymptotic setting [RS98] and in the concrete setting for uniform and slightly biased distributions [BHT18]. In Theorem 5.1, we give a more general result in the concrete setting that is valid for any biased ball distribution and any number of balls (Case 4). Indeed, through it, we are also able to improve significantly over prior bounds for GCM in the nonce-randomization setting.

Theorem 5.1 (Biased balls-into-bins). *Consider an experiment where at most Q balls are thrown into a set of bins, where each throw may depend on the outcome of the prior ones. Let $D \in (0,1]$ be an upper bound on the probability that, when conditioned on prior throws, a ball lands into any bin. Then, for any $m \in \mathbb{N}^*$, $\widetilde{m} > 0$ and $\lambda > 1$ satisfying one of the following conditions:*

1. $m = \left\lceil \frac{\log_\lambda(D^{-1}) + \widetilde{m}}{\log_\lambda((QD)^{-1})} \right\rceil$ *and* $Q \leq \frac{1}{D}$,
2. $m = \lceil \log_\lambda(D^{-1}) + \widetilde{m} \rceil$ *and* $Q \leq \frac{\log_\lambda(D^{-1}) + \widetilde{m}}{D\lambda \mathrm{e}}$,
3. $m = \lceil QD\lambda\mathrm{e} \rceil$ *and* $Q \geq \frac{\log_\lambda(D^{-1}) + \widetilde{m}}{D\lambda \mathrm{e}}$,
4. $m = \left\lceil \frac{\max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1}) + \widetilde{m})}{\max(1, \log_\lambda((QD)^{-1}))} \right\rceil$,

the probability that the heaviest bin contains m balls or more is at most $\lambda^{-\widetilde{m}}$.

The proof can be found in Appendix B. Compared to prior results, the first advantage of this theorem is its generality, as it allows more freedom in selecting parameters. Case 4 yields the best bound, giving the smallest maximum load m for a fixed maximum probability bound $\lambda^{-\widetilde{m}}$, and has no restrictions on the maximum number of balls Q. Moreover, Cases 1, 2 and 3 can be derived as subcases of Case 4.

In comparison to the biased balls-into-bins lemmas of Bose, Hoang, and Tessaro, Case 1 improves over [BHT18, Lemma 10] by introducing a trade-off parameter $\widetilde{m}$ between the maximum load and the probability bound instead of a fixed probability bound. This is essentially what allows us to lift the restrictive term of 2^{-48} in the bound for nonce-randomized GCM (cf. Sections 7.2 and 7.4).

In combination, Cases 2 and 3 are roughly equivalent to [BHT18, Lemma 11]. While the latter allows for an unrestricted number of balls, it yields a looser bound than [BHT18, Lemma 10] when the number of balls is small. Thus, when applying the lemmas of [BHT18], one has to choose between a good bound over a restricted range of Q, or a suboptimal bound extending over a larger range of Q. Case 4 improves over both lemmas in [BHT18] by combining their advantages in a single expression while additionally retaining the improvement from Case 1. We employ this improved bound in our multi-user security proof through Lemma C.1, when bounding the probabilities of bad transcripts in Appendix C.2.

6 The Multi-user Security of ChaCha20-Poly1305

The following theorem bounds the multi-user security of ChaCha20-Poly1305 in the ideal permutation model, i.e., when the ChaCha20 permutation is assumed to be a random permutation. This allows us to capture the local computation of the adversary expressed as the number of offline queries that it makes to the ChaCha20 permutation.

Theorem 6.1 (Multi-user security of ChaCha20-Poly1305**).** *Let* ChaCha20-Poly1305$[\pi]$ *be the AEAD scheme described in Figure 5 having parameters n, k, t, ϵ with its underlying permutation π modelled as a random permutation. Let $\mathcal{A}$ be a d-repeating adversary making at most p ideal permutation queries, q_e encryption queries totaling at most σ_e encrypted blocks, and q_v verification queries. Further, let ℓ_m denote the maximum size in t-bit blocks (including associated data) that it is allowed to query to its encryption and verification oracles. Then*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \leq \frac{q_v(\epsilon(\ell_m)+3)}{2^t} + \frac{d(p+q_e)}{2^k} + \frac{2p\cdot(n-k)}{2^k} + \frac{2q_v\cdot(n-k+4t)}{2^k} + \frac{(\sigma_e+q_e)^2}{2^{n+1}} + \frac{1}{2^{2t-2}} + \frac{1}{2^{n-k-2}}.$$

In the above we further require that $n - k \leq 2^{k-2}$, $\sigma_e \leq \frac{n-k}{6} \cdot 2^{n-k}$, $q_v \leq 2^{n-2}$, $d \leq \frac{2t}{3} \cdot 2^{2t}$, *and* $p \leq \min\left(\frac{2t-1}{6} \cdot 2^{2t}, \frac{n-k-1}{6} \cdot 2^{n-k}\right)$.

We note that the bound we obtain here is actually stronger than the simplified version presented in Theorem 6.1 for better exposition. In particular, the third and fourth terms in the bound in Theorem 6.1 are more accurately given by

$$\frac{2p \cdot \overline{(n-k)}^{\sigma_e}}{2^k} + \frac{2q_v \cdot \left(\overline{(n-k)}^{p} + \overline{2t}^{p} + \overline{2t}^{d}\right)}{2^k},$$

with the shorthand notation $\overline{\;i\;}^{j} = \left\lceil \frac{i}{\max(1, i - \log_2(2j))} \right\rceil$ and $\overline{\;i\;}^{0} = 0$, for any $i, j \in \mathbb{N}$. Note that always $\overline{\;i\;}^{j} \leq i$, yielding the terms in Theorem 6.1, and if $j \leq 2^{i/2-1}$ then $\overline{\;i\;}^{j} \leq 2$, yielding the terms in the following corollary.

Corollary 6.1 (Multi-user Security of ChaCha20-Poly1305**).** *Let* ChaCha20-Poly1305$[\pi]$ *be the AEAD scheme described in Figure 5 having parameters* n, k, t, ϵ *with its underlying permutation* π *modelled as a random permutation. Let* $\mathcal{A}$ *be a* d*-repeating adversary making at most* p *ideal permutation queries,* q_e *encryption queries totaling at most* σ_e *encrypted blocks, and* q_v *verification queries. Further, let* ℓ_m *denote the maximum size in* t*-bit blocks (including associated data) that it is allowed to query to its encryption and verification oracles. Then*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \leq \frac{q_v(\epsilon(\ell_m) + 3)}{2^t} + \frac{d(p + q_e)}{2^k} + \frac{4p + 12q_v}{2^k} + \frac{(\sigma_e + q_e)^2}{2^{n+1}} + \frac{1}{2^{2t-2}} + \frac{1}{2^{n-k-2}}.$$

In the above we further require that: $2 \leq t$, $3 \leq n - k \leq 2^{k-2}$, $\sigma_e \leq 2^{\frac{n-k}{2}-1}$, $q_v \leq 2^{n-2}$, $d \leq 2^{t-1}$ *and* $p \leq \min\left(2^{t-1}, 2^{\frac{n-k}{2}-1}\right)$.

Corollary 6.1 is of practical interest, as it gives a simpler and better bound than Theorem 6.1 and for real-world parameters (i.e., $n = 512$, $k = 256$, $t = 128$), the further restricted range of queries compared to the main theorem ($\sigma_e, p, d \leq 2^{127}$) is not a limitation in practice. Moreover, for practical parameters of ChaCha20-Poly1305 and GCM, the range of queries allowed by Corollary 6.1 is similar to the one from [HTT18, Theorem 3.1], so practical comparison should be done between the two of them. The improved bound we obtain in Corollary 6.1 is made possible by our improved balls-into-bins theorem. Another corollary could have been given with a wider range of queries than Theorem 6.1, using again our generalized balls-into-bins theorem to obtain a variant of Lemma C.1 without any upper bound on Q, but the bound obtained is more complex and not of practical interest.

6.1 Proof Overview

The proof of Theorem 6.1 is based on the H-coefficient technique (Theorem 2.1) which we apply to the augmented games described in Figures 6 and 7. These games modify the multi-user AEAD games shown in Figure 3 to give the adversary more information, thereby capturing an even stronger notion of security, but which nevertheless facilitates our proof. Specifically, the adversary is given access to an additional oracle Reveal, that it *must* query exactly once as its last query, right before returning its output—which in turn triggers the Finalize procedure.

In the real world, the Reveal oracle returns the outputs V_j corresponding to all the internal calls made by the encryption oracle to the ChaCha20 block function and all the user keys K_i. In the ideal world, on the other hand, the Reveal oracle returns randomly distributed strings V_j and keys K_i. Note that through $\{V_j\}$ and $\{K_i\}$, the adversary implicitly also learns the direct outputs of the ideal permutation in the real world, as it can easily reconstruct them. Since the augmented games are strictly stronger than the original ones, we can bound any adversary's multi-user advantage by bounding its distinguishing advantage with respect to the augmented games.

In the rest of this section we gradually set up all the components needed to apply the H-coefficient technique. We start in Section 6.2 by specifying the format of transcripts with respect to the augmented games. In Section 6.3 we define six sets of bad transcripts. In Section 6.4 we bound the H-coefficient over good transcripts, and in Section 6.5 we bound the probability of each individual set of bad transcripts. Then plugging the two bounds into Theorem 2.1 yields the desired result.

```
                                        Game $G^{\text{augReal-muAE}}_{\Pi[\pi]}$
procedure INITIALIZE
  1 :  $K_1, K_2, \cdots \leftarrow\!\!\$ \{0,1\}^k, \eta_e \leftarrow 0$
procedure ENC$(i, N, AD, M)$
  1 :  $C \| T \leftarrow \mathcal{E}(K_i, N, AD, M)$
  2 :  $\eta_e \leftarrow \eta_e + 1$
  3 :  $\overline{V}[\eta_e] \leftarrow (i, N, AD, M, C, T)$
  4 :  return $C \| T$
procedure VF$(i, N, AD, C)$
  1 :  $M \leftarrow \mathcal{D}(K_i, N, AD, C)$
  2 :  return $(M \neq \bot)$
procedure PRIM$(X)$
  1 :  return $\pi(X)$
procedure PRIM$^{-1}(Y)$
  1 :  return $\pi^{-1}(Y)$
procedure REVEAL     // last query before FINALIZE
  1 :  for $\eta = 1$ to $\eta_e$ do
  2 :      $(i, N, AD, M, C, T) \leftarrow \overline{V}[\eta]$
  3 :      $M_1 \| \cdots \| M_\ell \leftarrow M$
  4 :      for $j = 0$ to $\ell$ do
  5 :          $V_j \leftarrow \pi(Z \| K_i \| j \| N) \overset{(32)}{+} (Z \| K_i \| j \| N)$
  6 :      $V^{(\eta)} \leftarrow V_0 \| \cdots \| V_\ell$
  7 :  return $V^{(1)}, \ldots, V^{(\eta_e)}, K_1, K_2, \ldots$
procedure FINALIZE$(b)$
  1 :  return $b$
```

Fig. 6. Real Augmented Game.

```
                                        Game $G^{\text{augIdeal-muAE}}_{\Pi[\pi]}$
procedure INITIALIZE
  1 :  $\eta_e \leftarrow 0$
procedure ENC$(i, N, AD, M)$
  1 :  $C \| T \leftarrow\!\!\$ \{0,1\}^{|M|+t}$
  2 :  $\eta_e \leftarrow \eta_e + 1$
  3 :  $\overline{V}[\eta_e] \leftarrow (i, N, AD, M, C, T)$
  4 :  return $C \| T$
procedure VF$(i, N, AD, C)$
  1 :  return false
procedure PRIM$(X)$
  1 :  return $\pi(X)$
procedure PRIM$^{-1}(Y)$
  1 :  return $\pi^{-1}(Y)$
procedure REVEAL     // last query before FINALIZE
  1 :  for $\eta = 1$ to $\eta_e$ do
  2 :      $(i, N, AD, M, C, T) \leftarrow \overline{V}[\eta]$
  3 :      $r \leftarrow\!\!\$ \{0,1\}^t, W \leftarrow\!\!\$ \{0,1\}^{n-2t}$
  4 :      $V_0 \leftarrow r \| (T \overset{(t)}{-} H_r(AD, C)) \| W$
  5 :      $M_1 \| \cdots \| M_\ell \leftarrow M, \quad C_1 \| \cdots \| C_\ell \leftarrow C$
  6 :      for $j = 1$ to $\ell - 1$ do
  7 :          $V_j \leftarrow M_j \oplus C_j$
  8 :      $W' \leftarrow\!\!\$ \{0,1\}^{n-|M_\ell|}$
  9 :      $V_\ell \leftarrow (M_\ell \oplus C_\ell) \| W'$
 10 :      $V^{(\eta)} \leftarrow V_0 \| \cdots \| V_\ell$
 11 :  $K_1, K_2, \cdots \leftarrow\!\!\$ \{0,1\}^k$
 12 :  return $V^{(1)}, \ldots, V^{(\eta_e)}, K_1, K_2, \ldots$
procedure FINALIZE$(b)$
  1 :  return $b$
```

Fig. 7. Ideal Augmented Game.

6.2 Transcripts and Multi-sets

In the H-coefficient technique, we only need to consider *attainable* transcripts, i.e., ones that have a probability strictly greater than 0 of occurring in the ideal world. Note that here we define transcripts slightly differently, as we include additional information beyond the input-output pairs corresponding to the adversary's queries. We define them this way to facilitate the classification of good and bad transcripts and other aspects in the proof.

Transcripts. A transcript τ of an adversary interacting with an augmented game consists of the following entries:

- **Revealed key entries:** (key, i, K_i).
  Keys are returned as part of the output to the Reveal query. In the real world, these entries correspond to the actual user keys, whereas in the ideal world they correspond to values sampled independently of the rest of the transcript. In the real augmented game these are generated during initialization, whereas in the ideal augmented game they are not sampled until Reveal is queried.
- **Ideal permutation entries:** $(\mathsf{prim}, x, y, +)$ and $(\mathsf{prim}, x, y, -)$.
  An entry $(\mathsf{prim}, x, y, +)$ corresponds to a query $\textsc{Prim}(x)$ to the ideal permutation oracle with answer y, and an entry $(\mathsf{prim}, x, y, -)$ corresponds to a query $\textsc{Prim}^{-1}(y)$ to the inverse of the ideal permutation oracle with answer x.
- **Encryption entries:** $(\mathsf{enc}, i, N, AD, M, C\|T, V^{(\eta)})$.
  These entries contain the values specified in each encryption query $\textsc{Enc}(i, N, AD, M)$ together with the corresponding response $C\|T$. They additionally include the associated list $V^{(\eta)}$ of internal ChaCha20 block function calls made by the encryption algorithm in that encryption query. In particular, the $V^{(\eta)}$ values contain the key material used in Poly1305_Mac. Note, however, that while in the transcript, for convenience, we include $V^{(\eta)}$ in the encryption entries they are *not* actually returned to the adversary by the encryption oracle. In the augmented games these values are only revealed to the adversary at the end in the Reveal query. In the ideal world, $V^{(\eta)}$ is instead generated at random, as described in Figure 7, so that all good transcripts (defined below) have a probability strictly greater than 0 of occurring in the real world. This, in turn, ensures that the H-coefficient is not zero. Intuitively, including $V^{(\eta)}$ in the transcript allows us to better define the set of bad transcripts for our proof and thereby get a better H-coefficient ratio.
- **Verification entries:** $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false})$.
  Entries of this type correspond to verification queries $\textsc{Vf}(i, N, AD, C\|T)$ which return false as an answer. In the H-coefficient technique, we only need to be concerned with *attainable* transcripts, and in the ideal world, verification queries always return false as an answer.

Multi-sets. The H-coefficient technique requires us to bound from below the ratio of the real-world and ideal-world probabilities, for any good transcript, to a value close to one. We accomplish this via a counting argument. In addition to the probability of the user keys being sampled, the probability of a transcript can be reduced to counting the number of distinct ideal permutation calls and random blocks generated (for encryption queries in the ideal world). If their sum is close in either world, we obtain a good H-coefficient ratio that is close to one. To facilitate our counting argument we introduce the following three multi-sets (sets where elements can repeat) and calculate their cardinality:

$$S_1(\tau) = \{(x, y) \mid (\mathsf{prim}, x, y, \cdot) \in \tau\}.$$

The set $S_1(\tau)$ contains those input-output pairs of the ideal permutation π that were induced through the ideal permutation queries to Prim or $\textsc{Prim}^{-1}$.

$$S_2(\tau) = \{(Z\|K_i\|0\|N, V_0 \overset{(32)}{-} (Z\|K_i\|0\|N)), \ldots, (Z\|K_i\|\ell\|N, V_\ell \overset{(32)}{-} (Z\|K_i\|\ell\|N)) \mid \\ (\mathsf{enc}, i, N, AD, M, C\|T, V_0\| \cdots \|V_\ell) \in \tau\}.$$

The set $S_2(\tau)$ contains input-output pairs of the ideal permutation π that were induced through encryption queries to Enc. In the ideal world, the outputs are the random blocks generated, in the real world they are the π outputs.

$$S_3(\tau) = \{(Z\|K_i\|0\|N) \mid ((\mathsf{vf}, i, N, AD, C\|T, \mathsf{false}) \in \tau) \wedge ((Z\|K_i\|0\|N, \cdot) \notin S_1(\tau) \cup S_2(\tau))\}.$$

The set $S_3(\tau)$ contains the inputs to the ideal permutation π called during verification queries to $\textsc{Vf}$ in the real world, if they are not also called (or obtained) during a primitive or encryption query.

6.3 Bad Transcripts

Our overarching methodology for defining bad transcripts (i.e., the set $\mathbb{T}_{\mathrm{bad}}$) is to rule out transcripts that: 1) have a different multi-set cardinality in the real world compared to the ideal world or 2) have zero probability of occurring in the real world. This way we ensure that the H-coefficient is close to one. Towards the former, we will ensure that each entry in the first two multi-sets corresponds to a unique and independent call to the ideal permutation π, or a unique and independently generated random block. In the second case, even if the transcripts do not result in repeated multi-set entries they may still be impossible in the real world. Thus a transcript is in $\mathbb{T}_{\mathrm{bad}}$ if it satisfies one of the following:

Case 1 $(x_1, y_1) \in S_1(\tau)$ and $(x_2, y_2) \in S_2(\tau)$ where $x_1 = x_2$. In this case, in the real world, two calls are made to the ideal permutation on the same input, through one ideal permutation query and one encryption query. This case also encompasses the case where $x_1 = x_2$ and $y_1 \neq y_2$, which is impossible in the real world. From this case, we can define the following simplified bad transcript description (which is the only possibility of this case happening):

Bad_1**:** There are two entries $(\mathsf{prim}, x, y, \cdot)$ and $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\|\cdots\|V_\ell)$ such that $x \in \{Z\|K\|0\|N, \ldots, Z\|K\|\ell\|N\}$ and $K_i = K$.

Case 2 $(x_1, y_1) \in S_1(\tau)$ and $(x_2, y_2) \in S_2(\tau)$ where $y_1 = y_2$. In this case, in the real world, two calls with the same output are made to the ideal permutation, through one ideal permutation query and one encryption query. This case also encompasses the case where $y_1 = y_2$ and $x_1 \neq x_2$, which is impossible in the real world. From this case, we can define the following simplified bad transcript description:

Bad_2**:** There are two entries $(\mathsf{prim}, x, y, \cdot)$ and $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\|\cdots\|V_\ell)$ such that $y \in \{V_0 \overset{(32)}{-} (Z\|K_i\|0\|N), \ldots, V_\ell \overset{(32)}{-} (Z\|K_i\|\ell\|N)\}$.

Case 3 $(x_1, y_1), (x_2, y_2) \in S_2(\tau)$ where $x_1 = x_2$. In this case, in the real world, two calls are made to the ideal permutation on the same input, through one or two encryption queries. This case also encompasses the case where $x_1 = x_2$ and $y_1 \neq y_2$, which is impossible in the real world. From this case, we can define the following simplified bad transcript description:

Bad_3**:** There are two entries $(\mathsf{enc}, i, N, AD, M, C\|T, V)$ and $(\mathsf{enc}, i', N', AD', M', C'\|T', V')$ with $N = N'$ (block counter values do not overlap across different nonces), $i \neq i'$ (because nonce reuse is not allowed for the same user) and $K_i = K_{i'}$.

Case 4 $(x_1, y_1), (x_2, y_2) \in S_2(\tau)$ where $y_1 = y_2$. In this case, in the real world, two calls with the same output are made to the ideal permutation, through one or two encryption queries. The case where $y_1 = y_2$ and $x_1 = x_2$ being already considered in the previous point, we can restrict this case to $y_1 = y_2$ and $x_1 \neq x_2$, which in fact is impossible in the real world. From this case, we can define the following simplified bad transcript description:

Bad_4**:** There are two entries $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\|\cdots\|V_\ell)$ and $(\mathsf{enc}, i', N', AD', M', C'\|T', V'_0\|\cdots\|V'_{\ell'})$ such that $(K_i, j, N) \neq (K_{i'}, j', N')$ and $V_j \overset{(32)}{-} (Z\|K_i\|j\|N) = V'_{j'} \overset{(32)}{-} (Z\|K_{i'}\|j'\|N')$ for $0 \leq j \leq \ell$ and $0 \leq j' \leq \ell'$.

Case 5 $(x, y) \in S_1(\tau)$ and $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false}) \in \tau$ where $x = (Z\|K_i\|0\|N)$ and $\exists r \in \{0,1\}^t$, $W \in \{0,1\}^{n-2t}$ such that $y \overset{(32)}{+} x = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$. This case corresponds to an impossible transcript in the real world. As in the real world $\pi(x) = y$ and from this case we obtain $\pi(Z\|K_i\|0\|N) \overset{(32)}{+} (Z\|K_i\|0\|N) = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$. Therefore, $\mathsf{trunc}(\mathrm{CC\_block}(K_i, N, 0), 2t) = r\|(T \overset{(t)}{-} H_r(AD, C))$. Thus $(N, AD, C\|T)$ is a valid nonce/AD/ciphertext triplet under key i, and in the real world, the verification query considered would have returned true. From this case, we can define the following simplified bad transcript description:

Bad_5**:** There are two entries $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false})$ and $(\mathsf{prim}, x, y, \cdot)$ such that $x = (Z\|K\|0\|N)$, $K_i = K$ and $\exists r \in \{0,1\}^t, W \in \{0,1\}^{n-2t}$ such that $y \overset{(32)}{+} x = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$.

Case 6 $(x, y) \in S_2(\tau)$ and $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false}) \in \tau$ where $x = (Z\|K_i\|0\|N)$ and $\exists r \in \{0,1\}^t$, $W \in \{0,1\}^{n-2t}$ such that $y \overset{(32)}{+} x = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$. This case corresponds to an impossible transcript in the real world, for a similar reason as the previous case, i.e., in the real world, the verification query considered would have returned true. From this case, we can define the following simplified bad transcript description:

Bad_6**:** There are two entries $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false})$ and $(\mathsf{enc}, i', N, AD', M', C'\|T', V'_0\| \cdots \|V'_\ell)$ such that $K_{i'} = K_i$ and $\exists r \in \{0,1\}^t, W \in \{0,1\}^{n-2t}$ such that $V'_0 = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$.

For $j \in \{1, \ldots, 6\}$, let Bad_j be the set of attainable transcripts satisfying the j-th simplified bad transcript description defined above. Then $\mathbb{T}_{\text{bad}} = \bigcup_{j=1}^{6} \mathsf{Bad}_j$. Note that Bad_1, Bad_2, and Bad_3 contain attainable transcripts calling more than one time the ideal permutation on the same input-output during primitive and encryption queries, and Bad_4, Bad_5, and Bad_6 contain attainable transcripts impossible in the real world.

6.4 Good Transcript Ratio (H-Coefficient)

An attainable transcript that is not in $\mathbb{T}_{\text{bad}}$ is called good, and the set of all good transcripts is denoted by $\mathbb{T}_{\text{good}}$. In the H-coefficient technique, we need to bound the probability ratio of a good transcript being generated in the real world with respect to that of it being generated in the ideal world.

Anomalous transcripts that result in a weak H-coefficient or that make it hard to evaluate have been weeded out as bad transcripts in the previous section. Roughly, these were transcripts that resulted in different multi-set cardinalities across the two worlds and transcripts with a real-world probability of zero. Consequently, we can now easily bound the H-Coefficient (of good transcripts) through a simple counting argument which yields the bound specified in Proposition 6.1. Its proof can be found in Appendix C.1.

Proposition 6.1 (Probability of good transcripts). *For any good transcript* $\tau \in \mathbb{T}_{\text{good}}$ *it holds that*

$$\frac{\mathsf{P}_{\text{real}}(\tau)}{\mathsf{P}_{\text{ideal}}(\tau)} \geq 1 - \frac{2q_v}{2^t}.$$

6.5 Bad Transcript Probabilities

We now bound the probability that $\mathcal{T}_{\text{ideal}} \in \mathbb{T}_{\text{bad}}$, i.e., the probability that a transcript generated in the ideal world is bad.

The probabilities of events $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_j$, for $j \in \{1, \ldots, 6\}$ are bounded in Lemmas C.2–C.7, the proofs of which can be found in Appendix C.2. The corresponding bounds are reproduced below in Equations (1)–(6). Proposition 6.2 is then obtained by a direct application of the union bound and substituting terms through Equations (1)–(6). To quickly describe the arguments used when bounding the different sets of bad transcripts, we bound the probability that $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_4$ by the probability of a collision through the randomly generated V blocks during REVEAL. We bound the probabilities of $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_j$, for $j \in \{1, 2, 3, 5, 6\}$ through a counting argument of specific pairs of entries, after which we apply a union bound over all of these possible pairs for a specific key sampling event. The description of the pairs of entries and the key sampling events are specific to each set Bad_j. We notably use our improved balls-into-bins theorem to count the pairs of entries in a transcript for Bad_2, Bad_5 and Bad_6. In addition, we also use the ϵ-almost Δ-universal property of the function H to bound the probability of $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_6$.

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_1] \leq \frac{pd}{2^k}. \tag{1}$$

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2] \leq \frac{p \cdot 2\overline{(n-k)}^{\sigma_e}}{2^k} + \frac{1}{2^{n-k}}. \tag{2}$$

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_3] \leq \frac{q_e(d-1)}{2^k}. \tag{3}$$

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_4] \leq \frac{(\sigma_e + q_e)^2}{2^{n+1}}. \tag{4}$$

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5] \leq \frac{q_v \cdot 2\left(\overline{(n-k)}^p + \overline{2t}^p\right)}{2^k} + \frac{1}{2^{n-k-1}} + \frac{1}{2^{2t-1}}. \tag{5}$$

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_6] \leq \frac{q_v}{2^t} + \frac{q_v \cdot \epsilon(\ell_m)}{2^t} + \frac{q_v \cdot 2 \cdot \overline{2t}^d}{2^k} + \frac{1}{2^{2t}}. \tag{6}$$

Proposition 6.2 (Probability of bad transcripts). *Let $\mathcal{T}_{\text{ideal}}$ be the random variable representing the transcript generated by $\mathcal{A}$ in the ideal game. It then holds that:*

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathbb{T}_{\text{bad}}] \leq \frac{q_v \cdot (\epsilon(\ell_m) + 1)}{2^t} + \frac{d(p + q_e)}{2^k} + \frac{2 \cdot p\overline{(n-k)}^{\sigma_e}}{2^k} + \frac{q_v \cdot 2\left(\overline{(n-k)}^p + \overline{2t}^p + \overline{2t}^d\right)}{2^k} + \frac{(\sigma_e + q_e)^2}{2^{n+1}} + \frac{1}{2^{2t-2}} + \frac{1}{2^{n-k-2}},$$

again using the shorthand notation $\overline{i}^j = \left\lceil \frac{i}{\max(1, i - \log_2(2j))} \right\rceil$ and $\overline{i}^0 = 0$, for any $i, j \in \mathbb{N}$.

Theorem 6.1 is obtained by combining Propositions 6.1 and 6.2 with Theorem 2.1, where Proposition 6.1 yields ϵ_1 and Proposition 6.2 yields ϵ_2.

7 Implications of the Main Result

In this section, we discuss some properties and implications of our multi-user security theorem from Section 6. Succinctly, we prove the tightness of the bound and extend the security result to the nonce-randomized setting. Then we explain how our bounds improve over previous results and how they compare and contrast with the security profile of GCM. We further discuss the security limits of ChaCha20-Poly1305 induced by the dominant term in our bound and propose a potential variant scheme that overcomes these limitations.

7.1 Tightness of the Bound

We establish the tightness of the bound in Theorem 6.1 by providing several attacks matching the theorem bounds via the Propositions 7.1–7.6 below. To the best of our knowledge, the attacks in Propositions 7.1 and 7.6 are new in this context and do not seem to be covered elsewhere in the literature. In the following, we summarize the attack ideas; the details for all attacks establishing the stated propositions are given in Appendix D.

Note that all our attacks are valid against the original multi-user game and not only against the augmented version used in the proof. Moreover, for all our attacks except the one in Proposition 7.6, we only need the underlying ChaCha20 permutation π to be a permutation over $\{0,1\}^n$ and not necessarily an ideal permutation.

Forgery attack. The main idea of the first attack is to create a forgery by trying to retrieve the hash key of Poly1305_Mac. Similarly, as the forgery attack against GCM [ABBT15, PC15], the attacker tests a set of ℓ_m possible hash keys through each verification query; if the correct hash key is in this set, the forgery attempt will be valid, distinguishing the real from the ideal game.

The attack is described in the multi-user setting but is also valid in the single-user setting as it queries only one user. The main difference with forgery attacks on GCM is that in Poly1305_Mac, 128-bit message/ciphertext blocks are encoded (padded with 1) before being transformed into coefficients of a polynomial modulo $2^{130} - 5$, preventing the attacker from accessing the full range of coefficients. We use here the technique from [Kry21] as a workaround.

Proposition 7.1. *Let t be the tag size of* Poly1305_Mac *and H its associated ϵ-almost Δ-universal hash function. Let $\ell_m \geq 5$ be the maximal number of t-bit input blocks in an encryption or verification query to the* ChaCha20-Poly1305 *AEAD scheme. There exists an adversary $\mathcal{A}$ making one encryption query and q_v verification queries such that:*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \geq \frac{q_v \cdot 2^{21} \cdot (\ell_m - 5)}{2^t}.$$

Key-recovery attack. The main idea of the two following attacks is to use offline computations (via the permutation oracle in our model) to try to recover one of the user keys in the real game.

For the first attack, the attacker makes d encryption queries on the same input (N, M, AD) across d different users. It then makes p permutation queries $\textsc{Prim}(Z \| K'_{i'} \| 1 \| N)$ for guessed keys $K'_{i'}$, to construct corresponding ciphertexts for M, checking if any ciphertexts match to one of the d encryption queries. In the ideal game, as ciphertexts are randomly generated strings, the probability of this event is smaller than in the real game, allowing the attacker to distinguish between the two games.

Proposition 7.2. *Let k be the key length of the* ChaCha20-Poly1305 *AEAD scheme. There exists a d-repeating adversary $\mathcal{A}$, that makes p permutation queries and $d \leq \frac{2^k}{p}$ encryption queries such that:*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \geq \frac{pd}{2^{k+2}}.$$

The previous attack can be adapted to verification queries instead of encryption queries, yielding the following variant.

Proposition 7.3. *Let k be the key length of the* ChaCha20-Poly1305 *AEAD scheme. There exists an adversary $\mathcal{A}$, that makes one permutation query and $q_v \leq 2^k$ verification queries such that:*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \geq \frac{q_v}{2^{k+1}}.$$

Key-collision attack. The main idea of the following attack is to detect when two users sample the same key in the real game.

For the attack, the adversary makes encryption queries on the same input across different users. If two of the received ciphertexts match, it means that the two corresponding users have the same key in the real game. In the ideal game, ciphertexts are randomly generated strings. Thus, if the ciphertexts are long enough, the probability that they match is smaller in the ideal game than in the real game, allowing the attacker to distinguish between the two games. We note that, although stated here as an attack against ChaCha20-Poly1305, it extends to a generic attack against any deterministic AE scheme with positive ciphertext expansion in the multi-user setting, similar to the key collision attack of [BHT18], but for d-repeating adversaries.

Proposition 7.4. *Let k be the key length of the* ChaCha20-Poly1305 *AEAD scheme. There exists a d-repeating adversary $\mathcal{A}$, that makes $2 \leq q_e \leq \frac{2^{k+1}}{d-1}$ encryption queries such that:*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \geq \frac{q_e(d-1)}{2^{k+3}}.$$

Block-collision attack. Finally, the two last attacks distinguish the output of the ChaCha20 block function from the output of a random function due to block collisions. By construction, the ChaCha20 block function outputs can have collisions for different inputs, but collisions cannot occur between the outputs of the ChaCha20 permutation. The first attack is a direct application of the idea that, for a single user, we can detect collisions by canceling the feed forward and looking at the difference between two values $C_j \oplus M_j$. In the second attack, we exploit the encryption queries of multiple users, looking only at the parts of $C_j \oplus M_j$ that do not correspond to the key location. This gives us access to a truncated part of the ChaCha20 permutation and reduces to the problem of distinguishing a truncated permutation from a random function.

Proposition 7.5. *Let n be the block length of the* ChaCha20-Poly1305 *AEAD scheme. There exists an adversary $\mathcal{A}$ that encrypt at most B blocks per user for a total number of $\sigma_e \leq \frac{2^{n+1}}{B-1}$ encrypted blocks across all users such that:*

$$\mathsf{Adv}^{\mathrm{muAE}}_{\mathrm{ChaCha20\text{-}Poly1305}[\pi]}(\mathcal{A}) \geq \frac{\sigma_e(B-1)}{2^{n+2}}.$$

```
procedure $\mathcal{K}^*()$
1 : $K \leftarrow\!\!\$\; \mathcal{K}$
2 : $J \leftarrow\!\!\$\; \{0,1\}^\mu$
3 : return $K \| J$
```

```
procedure $\mathcal{E}^*(K \| J, N, AD, M)$
1 : $N^* \leftarrow N \oplus J$
2 : $C \leftarrow \mathcal{E}(K, N^*, AD, M)$
3 : return $C$
```

```
procedure $\mathcal{D}^*(K \| J, N, AD, C)$
1 : $N^* \leftarrow N \oplus J$
2 : $M \leftarrow \mathcal{D}(K, N^*, AD, C)$
3 : return $M$
```

Fig. 8. The XN transform of an AEAD scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ into a nonce-randomized AEAD scheme $\Pi^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$.

Proposition 7.6. *Let n, k be the block and key length of the* ChaCha20-Poly1305$[\pi]$ *AEAD scheme, where the underlying* ChaCha20 *permutation π is modeled as an ideal random permutation. There exists an adversary $\mathcal{A}$ encrypting a total number of $\sigma_e \leq \min\left(2^{\frac{n-k}{2}}, 2^{n-k-1}\right)$ blocks such that:*

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) \geq \frac{\sigma_e(\sigma_e - 1)}{2^{n+2}}.$$

Matching the bound. The presented attacks closely match the most significant and some further terms in the bound of Theorem 6.1:

Proposition 7.1: first term, $\approx \frac{q_v \cdot \epsilon(\ell_m)}{2^t}$.
Proposition 7.2: first half of second term, $\approx \frac{pd}{2^k}$, and third term $\approx \frac{p}{2^k}$, up to a factor of $2(n-k)$.
Proposition 7.3: fourth term, $\approx \frac{q_v}{2^k}$, up to a factor of $2(n-k+4t)$.
Proposition 7.4: second half of second term, $\approx \frac{q_e d}{2^k}$.
Proposition 7.5: fifth term, $\approx \frac{\sigma_e^2}{2^n}$, up to a factor of $\frac{\sigma_e}{B}$, when $\sigma_e \leq \frac{2^{n+1}}{B-1}$. Note that $\frac{\sigma_e}{B} \leq 1$ in the single-user case.
Proposition 7.6: fifth term, $\approx \frac{\sigma_e^2}{2^n}$, up to a factor of 4, when $\sigma_e \leq \min\left(2^{\frac{n-k}{2}}, 2^{n-k-1}\right)$.

7.2 Nonce Randomization: The XN Transform

The record protocols of TLS 1.3 [Res18], DTLS 1.3 [RTM21] and QUIC [TT21] use a nonce-randomization technique to counter large-scale multi-user attacks. Analyzing this technique for the GCM mode, Bellare and Tackmann [BT16] provided the first multi-user treatment. Hoang, Tessaro, and Thiruvengadam [HTT18] captured the nonce-randomization mechanism as a generic transform XN, which we recall in Figure 8. The XN transform turns an AEAD scheme Π into a nonce-randomized scheme Π^*, and [HTT18] showed that a generic adversary against the multi-user security of Π^* can be reduced to the security of Π against a strongly d-repeating adversary via an adaptive balls-into-bins argument.

We reuse our Lemma B.1 to obtain a generalization of [HTT18, Lemma 4.1 and Theorem 4.2]; their version emerges from our Theorem 7.1, when setting the new parameter δ we introduce to $\delta = 1/2$.

Theorem 7.1 (Multi-user security of the XN transform). *Let $\Pi[\pi] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme with nonce length μ defined on top an ideal permutation π, and let $\Pi^*[\pi] = \mathsf{XN}(\Pi[\pi])$ for the* XN *transform defined in Figure 8. Let $\mathcal{A}$ be a nonce-respecting adversary against Π^* making at most q_e* Enc *queries and q_v* Vf *queries. Then, for some fixed $\delta > 0$, we can construct an adversary $\mathcal{B}$ of the same concrete query complexity as $\mathcal{A}$ which is (strongly) d-repeating for $d = \left\lceil \frac{(\delta+1)\cdot\mu}{\max(1, \mu - \log_2(q))} \right\rceil - 1$, if $q \leq 2^\mu \cdot \frac{(\delta+1)\cdot\mu}{6}$, where $q = q_e$ for d-repeating and $q = q_e + q_v$ for strongly d-repeating, such that*

$$\mathsf{Adv}^{\text{muAE}}_{\Pi^*[\pi]}(\mathcal{A}) \leq \mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B}) + \frac{1}{2^{\delta\mu}}.$$

The new parameter δ. The improvement over [HTT18, Theorem 4.2] comes from the introduction of a new, tweakable parameter δ in the security bound. The added benefit is that when the theorem is applied to an AEAD scheme with known advantage $\mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B})$, one can now minimize the bound by tuning δ based on the other parameters. If the advantage term $\mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B})$ can be expressed as a function of δ, one may be able to find the optimal value directly via calculus. Alternatively, if this is not possible, one could use numerical techniques to obtain a near-optimal value that is good enough in practice. One such approach to choose δ is as follows. Note that d increases as δ increases, and in turn $\mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B})$ may increase as d increases. Then the optimal value is attained when $\mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B})$ and $\frac{1}{2^{\delta\mu}}$ are roughly equal, and so a suitable choice for δ is $-\frac{\log_2(\mathsf{Adv}^{\mathrm{muAE}}_{\Pi[\pi]}(\mathcal{B}))}{\mu}$. Note that d is less or equal to $(\delta+1)\cdot\mu$ and the upper bound on the number of queries is bigger than 2^{μ}, improving over the nonce randomization theorems in [HTT18] even when $\delta = 1/2$. In Section 7.4 below we will see that choosing $\delta = 2$ allows to ensure that the term $2^{-\delta\mu}$ added through nonce randomization becomes non-dominant.

Proof. The proof follows the same strategy as for [HTT18, Theorem 4.2], but then applies our generalized bound from Lemma B.1. The d-repeating reduction $\mathcal{B}$ samples nonce randomizer values J_i for each user i, used to derive the effective nonce $N^* = N \oplus J$ in encryption and verification queries forwarded to its multi-user AE game. It keeps a counter for any used nonce value N^* in encryption queries, counting the number of users for which $\mathcal{B}$ queried this effective nonce to its encryption oracles. When any such counter reaches $d+1$, $\mathcal{B}$ sets a bad flag bad, stops immediately, and outputs 1 (ensuring that $\mathcal{B}$ is d-repeating[6]); otherwise, $\mathcal{B}$ relays the bit output by $\mathcal{A}$.

In the real world,

$$\Pr\left[\mathcal{B}^{G^{\mathrm{Real\text{-}muAE}}_{\Pi[\pi]}} \Rightarrow 1\right] \geq \Pr\left[\mathcal{A}^{G^{\mathrm{Real\text{-}muAE}}_{\Pi^*[\pi]}} \Rightarrow 1\right],$$

as $\mathcal{B}$ either outputs 1 or repeats $\mathcal{A}$'s output. Furthermore, $\mathcal{B}$ simulates the ideal world for $\mathcal{A}$ perfectly until the bad flag is set, by the identical-until-bad lemma [BR96],

$$\Pr\left[\mathcal{B}^{G^{\mathrm{Ideal\text{-}muAE}}_{\Pi[\pi]}} \Rightarrow 1\right] \leq \Pr\left[\mathcal{A}^{G^{\mathrm{Ideal\text{-}muAE}}_{\Pi^*[\pi]}} \Rightarrow 1\right] + \Pr\left[\mathcal{B} \text{ sets } \mathsf{bad} \text{ in } G^{\mathrm{Ideal\text{-}muAE}}_{\Pi[\pi]}\right].$$

Overall, this yields

$$\mathsf{Adv}^{\mathrm{muAE}}_{\Pi[\pi]}(\mathcal{B}) \geq \mathsf{Adv}^{\mathrm{muAE}}_{\Pi^*[\pi]}(\mathcal{A}) - \Pr\left[\mathcal{B} \text{ sets } \mathsf{bad} \text{ in } G^{\mathrm{Ideal\text{-}muAE}}_{\Pi[\pi]}\right],$$

so it remains to bound the probability of $\mathcal{B}$ setting bad in the ideal world game.

In $G^{\mathrm{Ideal\text{-}muAE}}_{\Pi[\pi]}$, the oracles answer independently of the queried effective nonces and $\mathcal{B}$'s randomizer values J_i. The bad flag is set when across all encryption (and, for the strongly d-repeating case, also verification) queries, an effective nonce derived by $\mathcal{B}$ is used across $d+1$ users. We can view each effective nonce N^* queried to a user i, as throwing a ball i into one of 2^{μ} bins, where the bin represents the nonce queried. In total, we throw at most q balls, where $q = q_e$ for the d-repeating case and $q = q_e + q_v$ for the strongly d-repeating case. With this perspective, if the flag bad is set, then there exist a set of $d+1$ balls corresponding to $d+1$ distinct user in the same bin. Thus we can bound the probability that bad is set by the probability that there exist a set of $d+1$ balls corresponding to $d+1$ distinct user in the same bin.

For distinct users, the ball throws are independent and uniformly random distributed through the randomizer values J_i. The probability for any set of $d+1$ balls of distinct users to hit the same bin is hence $2^{-\mu d}$. Throwing up to q balls means there are at most $\binom{q}{d+1}$ sets of $d+1$ balls. So, by the union bound,

$$\Pr\left[\mathcal{B} \text{ sets } \mathsf{bad} \text{ in } G^{\mathrm{Ideal\text{-}muAE}}_{\Pi[\pi]}\right] \leq \binom{q}{d+1} \cdot 2^{-\mu d}.$$

We can now apply Case 4 of Lemma B.1 with $m = d+1$, $Q = q$, $D = 2^{-\mu}$, $\widetilde{m} = \delta\mu$, and $\lambda = 2$, to upper-bound the bad-event probability that any bin contains $m = d+1$ or more balls at $\lambda^{-\widetilde{m}} = 2^{-\delta\mu}$ as claimed. For $q \leq 2^{\mu} \cdot \frac{(\delta+1)\cdot\mu}{6}$, we obtain

$$d = m - 1 = \left\lceil \frac{(\delta+1)\cdot\mu}{\max(1, \mu - \log_2(q))} \right\rceil - 1. \qquad \square$$

We can combine Theorem 7.1 with our multi-user AE security result for ChaCha20-Poly1305 in Theorem 6.1 to obtain the following bound for the nonce-randomized usage of ChaCha20-Poly1305 (where $n = 512$, $k = 256$, $t = 128$, and $\mu = 96$), against an adversary that is *not* necessarily d-repeating. We emphasize that to obtain this result, we only need to use Theorem 7.1 with a reduction adversary $\mathcal{B}$ that is d-repeating and not strongly d-repeating, as this assumption suffices to apply Theorem 6.1.

Theorem 7.2 (Multi-user security of nonce-randomized ChaCha20-Poly1305**).** *Let* ChaCha20-Poly1305[π] *be the AEAD scheme described in Figure 5 having parameters n, k, t, ϵ, μ and its underlying*

[6] For the *strongly* d-repeating case, $\mathcal{B}$ counts the number of users using a particular effective nonce N^* across both its encryption *and* verification queries. Aborting when a counter reaches $d+1$ then ensures $\mathcal{B}$ is strongly d-repeating.

```
procedure $\mathcal{K}^*()$
  1 : $K \leftarrow\!\!\$\ \mathcal{K}$
  2 : $J \leftarrow\!\!\$\ \{0,1\}^{\mu'}$
  3 : return $K \| J$

procedure $\mathcal{E}^*(K\|J, N, AD, M)$
  1 : $N^* \leftarrow J\|N$
  2 : $C \leftarrow \mathcal{E}(K, N^*, AD, M)$
  3 : return $C$

procedure $\mathcal{D}^*(K\|J, N, AD, C)$
  1 : $N^* \leftarrow J\|N$
  2 : $M \leftarrow \mathcal{D}(K, N^*, AD, C)$
  3 : return $M$
```

Fig. 9. The CN transform of an AEAD scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ into another AEAD scheme $\Pi^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$.

permutation π modelled as a random permutation. Let $\mathcal{A}$ be an adversary against the multi-user AE security of $\mathsf{XN}(\text{ChaCha20-Poly1305}[\pi])$ *making at most p ideal permutation queries, q_e encryption queries totaling at most σ_e encrypted blocks, and q_v verification queries. Further, let ℓ_m denote the maximum size in t-bit blocks (including associated data) that $\mathcal{A}$ is allowed to query to its encryption and verification oracles. Then*

$$\mathsf{Adv}^{\text{muAE}}_{\mathsf{XN}(\text{ChaCha20-Poly1305}[\pi])}(\mathcal{A}) \le \frac{q_v(\epsilon(\ell_m)+3)}{2^t} + \frac{d(p+q_e)}{2^k} + \frac{2p\cdot(n-k)}{2^k} + \frac{2q_v\cdot(n-k+4t)}{2^k} + \frac{(\sigma_e+q_e)^2}{2^{n+1}} + \frac{1}{2^{2t-2}} + \frac{1}{2^{n-k-2}} + \frac{1}{2^{\delta\mu}},$$

where $d = \left\lceil \frac{(\delta+1)\cdot\mu}{\max(1,\mu-\log_2(q_e))} \right\rceil - 1$, for any $0 < \delta \le \frac{t}{\mu}\cdot 2^{2t-1} - 1$.
In the above we further require that $n-k \le 2^{k-2}$, $\sigma_e \le \frac{n-k}{6}\cdot 2^{n-k}$, $q_e \le 2^{\mu}\cdot\frac{(\delta+1)\cdot\mu}{6}$, $q_v \le 2^{n-2}$, and $p \le \min\left(\frac{2t-1}{6}\cdot 2^{2t}, \frac{n-k-1}{6}\cdot 2^{n-k}\right)$.

Note that as long as $\delta \le \frac{t}{\mu}\cdot 2^{2t-1} - 1$, the restriction on d from Theorem 6.1 is satisfied. In Section 7.4 below, we discuss and interpret the above bound for nonce-randomized ChaCha20-Poly1305. In particular, we will see how to pick δ such that the term $\frac{1}{2^{\delta\mu}}$ induced through nonce randomization does not dominate the overall bound.

7.3 Nonce Randomization: The CN Transform

In TLS 1.2 [RD08] and IPsec [VM05, Nir15] a different nonce-randomization technique than the XN previously described is used. Again, [HTT18] captured the mechanism through a generic transform CN, which we recall in Figure 9. In the CN transform, the effective nonce N^* derived as the concatenation of an implicit nonce J of length μ' bits, chosen at random, and an explicit nonce N of length $\mu - \mu'$ bits. Practical values used in TLS 1.2 and IPsec are $\mu = 96$ and $\mu' = 32$. Revisiting the formal treatment by [HTT18], we reuse our Lemma B.1 to give an equivalent of Theorem 7.1 for the CN transform.

Theorem 7.3 (Multi-user security of the CN transform). *Let $\Pi[\pi] = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a nonce-based AEAD scheme with nonce length μ defined on top an ideal permutation π, and let $\Pi^*[\pi] = \mathsf{CN}(\Pi[\pi])$ be the scheme with nonce length $\mu - \mu'$ for the CN transform defined in Figure 9. Let $\mathcal{A}$ be a nonce-respecting adversary against Π^* making at most q_e ENC queries and q_v VF queries. Then, for some fixed $\delta > 0$, we can construct an adversary $\mathcal{B}$ of the same concrete query complexity as $\mathcal{A}$ which is (strongly) d-repeating for $d = \left\lceil \frac{\max\left(6q\cdot 2^{-\mu'}, (\delta+1)\cdot\mu'\right)}{\max(1,\mu'-\log_2(q))} \right\rceil - 1$, where $q = q_e$ for d-repeating and $q = q_e + q_v$ for strongly d-repeating, such that*

$$\mathsf{Adv}^{\text{muAE}}_{\Pi^*[\pi]}(\mathcal{A}) \le \mathsf{Adv}^{\text{muAE}}_{\Pi[\pi]}(\mathcal{B}) + \frac{1}{2^{\delta\mu'}}.$$

By setting $\delta = 7$ in Theorem 7.3 above, we obtain an equivalent bound to that in Theorem 5.2 of [HTT18]. Note that, compared to [HTT18], in our theorem the tradeoff parameter δ can be chosen to adapt the additional term $\frac{1}{2^{\delta\mu'}}$ to the other terms in the bound. E.g., when choosing $\delta = 4$, the added term $\frac{1}{2^{\delta\mu'}}$ stays non-dominant while slightly improving the value of d.

Proof. The proof is similar to that of Theorem 7.1 (and [HTT18, Theorem 5.2]), but uses Lemma B.1 with different parameters. Therefore we only describe in the following how it differs from the proof of Theorem 7.1:

– The effective μ-bit nonce N^* is no longer built as $N \oplus J$ but as $J \| N$ with a shorter nonce randomizer J of length μ' bits.
– We apply Case 4 of Lemma B.1 with parameters

$$m = d + 1, \quad Q = q, \quad D = 2^{-\mu'}, \quad \widetilde{m} = \delta\mu', \quad \text{and} \quad \lambda = 2. \qquad \square$$

To obtain a multi-user bound for nonce-randomized ChaCha20-Poly1305 as used in TLS 1.2 and IPsec, one simply combines Theorem 7.3 with our multi-user bound for ChaCha20-Poly1305 in Theorem 6.1, adding a $\frac{1}{2^{\delta\mu'}}$ term to the latter similar to Theorem 7.2 for the XN transform.

7.4 Interpreting the Bounds

The dominant term. A closer look at Theorems 6.1 and 7.1 tells us that the most significant term in our bounds for practical scenarios is likely to be $\frac{q_v \cdot \epsilon(\ell_m)}{2^t}$. This is mainly due to the fact that for current ChaCha20-Poly1305 parameters, the block size is large ($n = 512$) and the key size $k = 256$ of ChaCha20 is twice as big as the tag size $t = 128$ of Poly1305, where the latter is effectively further reduced by the factor 2^{25} due to the clamping of 22 bits in the hash key r (cf. Definition 3.1 and Theorem 3.2). Hence, for the ChaCha20-Poly1305 parameters, this term in practical scenarios dominates the second-most significant term: $\frac{q_v \cdot \epsilon(\ell_m)}{2^t} \geq \frac{d(p+q_e)}{2^k}$, as long as the total number of primitive and encryption queries is bounded as $p + q_e \leq \frac{\ell_m+1}{d} \cdot 2^{153}$, even if $q_v = 1$ (when, e.g., applying the bound to a rekeying connection of a reliable-transport Internet security protocol like TLS [Res18] which terminates upon the first verification error).

Improving the nonce-randomizer bound. When moving from the basic multi-user security result for restricted, d-repeating adversaries to general adversaries, the analysis of the nonce-randomizer transform XN by [HTT18] introduces an additive loss term of $2^{-\mu/2}$ for nonce length μ. Through our improved balls-into-bins lemma, we instead obtain a parameterized term $2^{-\delta\mu}$, for which $\delta = 1/2$ as in [HTT18] is just one instantiation. Indeed, choosing $\delta = 2$ allows us to ensure that this term is not dominant in the bound of Theorem 7.2 for nonce-randomized ChaCha20-Poly1305 for adversaries making up to even around $q_e \approx 2^{\mu} = 2^{96}$ encryption queries.[7]

Notably, our improved result for the XN transform also readily improves the multi-user security for nonce-randomized GCM [HTT18, Theorem 4.3], allowing improvements to IETF/IRTF draft AEAD limits [GTW21, Section 6.1]: while the advantage bound in [HTT18] cannot become smaller than $2^{-\mu/2} = 2^{-48}$ (for the GCM nonce length of $\mu = 96$), our result entirely lifts this restriction, similar to the ChaCha20-Poly1305 case.

Improving over the standard hybrid loss in real-world settings. The only prior multi-user security bound for ChaCha20-Poly1305 is outlined in [LP17] and is based on a standard hybrid security loss in the number of user u over the single-user bound [Pro14]. This hybrid bound was reflected in early versions of the IETF/IRTF draft on AEAD limits [GTW21] as $\frac{u \cdot q_{v/u} \cdot \epsilon(\ell_m)}{2^t}$, where $q_{v/u}$ is the maximum failed verification attempts *per user*. In comparison, our bound is more fine-grained, bounding the *total* number q_v of verification attempts across *all* users, where always $q_v \leq u \cdot q_{v/u}$; this gap can become relatively large when no tight upper bounds for attempts per user can be derived. Our multi-user bound is now reflected in [GTW21, Section 6.2.1] and confirms the approach taken in DTLS 1.3 [RTM21, Section 4.5.3] and QUIC [TT21, Section 6.6] to derive integrity limits from summing the number of forgery attempts across multiple keys in a connection to counter the security degradation of repeated forgeries over unreliable transports [FGJ20].

7.5 Increasing the Hash Size

As discussed in Section 7.4, the dominant term in the multi-user security bounds for ChaCha20-Poly1305 in essentially all practical scenarios is $\frac{q_v \cdot \epsilon(\ell_m)}{2^t}$, making—relatively speaking—the tag length the scheme's weakest point. The natural question then is whether we can improve this term and obtain a stronger bound by increasing the tag size of ChaCha20-Poly1305. An obvious solution would be to use a wider almost Δ-universal hash function. This appears as an even more appealing solution when considering

[7] Observe that Theorem 7.2 for $\delta = 2$ allows the adversary to make up to $q_e \leq \mu \cdot 2^{\mu-1}$ encryption queries while ensuring, via Theorem 7.1, that $d \leq \lceil \mu \cdot (\delta + 1) \rceil - 1 = 287 \leq 2^{128} = 2^t$ as required for Theorem 6.1.

that, in ChaCha20-Poly1305, twice as much key material is computed than is used (namely, only half of the first CC_block call output), leaving the other half available as possible extra key material without needing any extra computation. To illustrate one potential approach, we give a construction doubling the tag size and then discuss how this affects the security of ChaCha20-Poly1305.

To double the tag size, we propose as an easy solution the following almost Δ-universal hash function that reuses the almost Δ-universal hash function H component of the Poly1305_Mac one-time MAC of ChaCha20-Poly1305. The main idea of our construction $\overline{H}^c$ in Definition 7.1 below is to concatenate two instantiations of the hash function H using two independent hash keys. This new hash function $\overline{H}^c$, when augmented with a $2t$-bit blinding value, creates a one-time MAC that we call cPoly1305 in reference to it arising from concatenation. This one-time MAC cPoly1305 takes a $2t$-bit hash key and a $2t$-bit blinding value. Note again that due to the unused key material, we can obtain the needed key and blinding value for this new almost Δ-universal hash function in the ChaCha20-Poly1305 construction without additional computation and with little modification to the original scheme.

Definition 7.1 (The hash function $\overline{H}^c$ in cPoly1305**).** *Let r, γ be two t-bit strings, H be the ϵ-almost Δ-universal hash function of* Poly1305_Mac *and (AD, C) be a pair of byte strings. Define the hash function used in* cPoly1305 *as*

$$\overline{H}^c_{r\|\gamma}(AD, C) = H_r(AD, C)\|H_\gamma(AD, C).$$

We now give a bound on the almost Δ-universal property of $\overline{H}^c$.

Theorem 7.4 ($\overline{H}^c$ is AΔU). *Let $\epsilon(\ell) = 2^{25} \cdot (\ell + 1)$ and s be a $2t$-bit string, then for any two distinct byte-string pairs (AD, C) and (AD', C') it holds that:*

$$\Pr_{r\|\gamma \leftarrow\$\{0,1\}^{2t}}\left[\overline{H}^c_{r\|\gamma}(AD, C) = \overline{H}^c_{r\|\gamma}(AD', C') \overset{(t)}{+} s\right] \leq \left(\frac{\epsilon(\max(|AD|_t + |C|_t, |AD'|_t + |C'|_t))}{2^t}\right)^2.$$

Proof. Let H be the ϵ-almost Δ-universal hash function of Poly1305_Mac. Let (AD, C) and (AD', C') be distinct pairs of byte strings. If $\overline{H}^c_{r\|\gamma}(AD, C) = \overline{H}^c_{r\|\gamma}(AD', C') \overset{(t)}{+} s$ then

$$H_r(AD, C) = H_r(AD', C') \overset{(t)}{+} s[1{:}t] \qquad \text{and} \qquad H_\gamma(AD, C) = H_\gamma(AD', C') \overset{(t)}{+} s[t + 1{:}2t].$$

Thus if we sample two keys r, γ independently and uniformly at random, then $\Pr_{r\|\gamma \leftarrow\$\{0,1\}^{2t}}\left[\overline{H}^c_{r\|\gamma}(AD, C) = \overline{H}^c_{r\|\gamma}(AD', C') \overset{(t)}{+} s\right]$ is equal to

$$\Pr_{r \leftarrow\$\{0,1\}^t}\left[H_r(AD, C) = H_r(AD', C') \overset{(t)}{+} s[1{:}t]\right] \cdot \Pr_{\gamma \leftarrow\$\{0,1\}^t}\left[H_\gamma(AD, C) = H_\gamma(AD', C') \overset{(t)}{+} s[t + 1{:}2t]\right].$$

The final bound is obtained by applying Theorem 3.2 to each factor of this product. □

We now discuss how using this almost Δ-universal function $\overline{H}^c$ in the ChaCha20-Poly1305 construction impacts the scheme's security (the argument can be generalized to any universal function doubling the tag size). Applying Theorem 6.1 (with a slight modification), we obtain the following upper bound on the multi-user security of ChaCha20-cPoly1305, the AEAD scheme obtained using $\overline{H}^c$ in place of H in ChaCha20-Poly1305:

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-cPoly1305}[\pi]}(\mathcal{A}) \leq \frac{q_v((\epsilon(\ell_m))^2 + 3)}{2^{2t}} + \frac{d(p + q_e)}{2^k} + \frac{2p \cdot (n - k)}{2^k} + \frac{2q_v \cdot (n - k + 8t)}{2^k} + \frac{(\sigma_e + q_e)^2}{2^{n+1}} + \frac{1}{2^{4t-2}} + \frac{1}{2^{n-k-2}}.$$

The first observation on this changed bound is that by doubling the tag size, we obtain a more uniform bound, with denominators in each term being at least $2^{2t} = 2^k = 2^{256}$, increasing the security by $t - \log_2(\epsilon(\ell_m))$ bits. The second and perhaps more interesting observation is that the most significant term in the bound would likely become $\frac{d \cdot p}{2^k}$, making offline computation the most probable attack against the scheme. This term corresponds to a key recovery attack (see Proposition 7.2) and is most likely inherent to any nonce based scheme against d-repeating adversaries (see [BT16] for the equivalent attack in GCM and [Bih02] for block ciphers). Thus choosing a tag size equal to the key length as we have done is probably the best tradeoff in terms of selecting the smallest tag size with the best security.

Our construction ChaCha20-cPoly1305 for doubling the tag size and increasing security allows parallelization and reuse of current implementations of Poly1305. However, if we use instead a dedicated construction with an almost Δ-universal hash function similar in structure to the one in Poly1305 but with a bigger prime number, we could improve even more the security bound over what we obtain for ChaCha20-cPoly1305. For a given maximum message length ℓ_m, the corresponding term in our bound is quadratic in ℓ_m, and in such a construction would be linear in ℓ_m. A good candidate for the bigger prime number would be the pseudo-Mersenne prime $p = 2^{255} - 19$. We leave the development of such an alternative construction to future work.

8 Conclusions

We have given a detailed security analysis of ChaCha20-Poly1305, an increasingly important AEAD scheme. Our analysis is in the multi-user setting and assumes the permutation underlying the scheme is ideal. This enables us to capture offline computation in our model and make a detailed comparison with the corresponding analysis of GCM by Hoang, Tessaro, and Thiruvengadam [HTT18]. Amongst other things, our analysis surfaces that the security limits for ChaCha20-Poly1305 are dominated by the limits of its MAC component. This is in contrast to GCM, where the limiting factor is the AES block size. We have proposed a lightweight way to strengthen ChaCha20-Poly1305 by doubling its MAC tag length. In future work, we plan to investigate alternative MAC constructions and their performance/security characteristics. We will also bring our work to the attention of the TLS and QUIC working groups of the IETF and collaborate with them to establish safe data limits for ChaCha20-Poly1305 in the context of the TLS, DTLS and QUIC protocols.

Acknowledgments

Jean Paul Degabriele and Jérôme Govinden were supported by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. Felix Günther has been supported in part by Research Fellowship grant GU 1859/1-1 of the German Research Foundation (DFG). Kenneth G. Paterson's research is supported in part by a gift from VMware.

References

ABBT15. Mohamed Ahmed Abdelraheem, Peter Beelen, Andrey Bogdanov, and Elmar Tischhauser. Twisted polynomials and forgery attacks on GCM. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 762–786. Springer, Heidelberg, April 2015. 16

AM18. Divesh Aggarwal and Priyanka Mukhopadhyay. Improved Algorithms for the Shortest Vector Problem and the Closest Vector Problem in the Infinity Norm. In Wen-Lian Hsu, Der-Tsai Lee, and Chung-Shou Liao, editors, *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:13, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 42

Ber05a. Daniel J Bernstein. The poly1305-aes message-authentication code. In *International Workshop on Fast Software Encryption*, pages 32–49. Springer, 2005. 1, 7, 8

Ber05b. Daniel J Bernstein. Salsa20 specification. *eSTREAM Project algorithm description, http://www.ecrypt.eu.org/stream/salsa20pf.html*, 2005. 7

Ber08. Daniel J Bernstein. Chacha, a variant of salsa20. In *Workshop Record of SASC*, volume 8, pages 3–5, 2008. 1, 7

BGM04. Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, 2004. `https://eprint.iacr.org/2004/309`. 27

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, Heidelberg, April / May 2018. 10, 17, 30, 33

Bih02. Eli Biham. How to decrypt or even substitute des-encrypted messages in 228 steps. *Information Processing Letters*, 84(3):117–124, 2002. 2, 22

BMS06. Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs with multiple data. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pages 110–127, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 2

BR96. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996. 19

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Vaudenay [Vau06], pages 409–426. 4, 25, 26

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, Heidelberg, August 2016. 1, 2, 5, 18, 22

CS14. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, Heidelberg, May 2014. 6

FGJ20. Marc Fischlin, Felix Günther, and Christian Janson. Robust channels: Handling unreliable networks in the record layers of QUIC and DTLS 1.3. Cryptology ePrint Archive, Report 2020/718, 2020. `https://eprint.iacr.org/2020/718`. 3, 21

GG21. Shoni Gilboa and Shay Gueron. The advantage of truncated permutations. *Discrete Applied Mathematics*, 294:214–223, 2021. 46

GTW21. Felix Günther, Martin Thomson, and Christopher A. Wood. Usage Limits on AEAD Algorithms – draft-irtf-cfrg-aead-limits-03. `https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-03`, July 2021. 21

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. 1, 2, 3, 11, 18, 19, 20, 21, 23

IOM12. Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 31–49. Springer, Heidelberg, August 2012. 1

Kry21. KryptosLogic. Faster poly1305 key multicollisions. Kryptos Logic Blog. `https://www.kryptoslogic.com/blog/2021/01/faster-poly1305-key-multicollisions`, Jan 2021. 16

Lan13. A Langley. Chacha20 and poly1305 based cipher suites for tls, draft-agl-tls-chacha20poly1305-00. IETF Internet Draft. `https://tools.ietf.org/html/draft-agl-tls-chacha20poly1305-00`, 2013. 1

LMP17. Atul Luykx, Bart Mennink, and Kenneth G. Paterson. Analyzing multi-key security degradation. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 575–605. Springer, Heidelberg, December 2017. 1, 2

LP17. Atul Luykx and Kenneth G Paterson. Limits on authenticated encryption use in tls. *Personal webpage: http://www. isg. rhul. ac. uk/˜ kp/TLS-AEbounds. pdf*, 2017. 2, 21

MV04. David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, December 2004. 1

Nir15. Yoav Nir. ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec. RFC 7634, August 2015. 20

NL18. Y. Nir and A. Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 8439 (Informational), June 2018. 6

NOMI15. Yuichi Niwa, Keisuke Ohashi, Kazuhiko Minematsu, and Tetsu Iwata. GCM security bounds reconsidered. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 385–407. Springer, Heidelberg, March 2015. 1

Pat09. Jacques Patarin. The "coefficients H" technique (invited talk). In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, Heidelberg, August 2009. 6

PC15. Gordon Procter and Carlos Cid. On weak keys and forgery attacks against polynomial-based MAC schemes. *Journal of Cryptology*, 28(4):769–795, October 2015. 16

Pro14. Gordon Procter. A security analysis of the composition of ChaCha20 and Poly1305. Cryptology ePrint Archive, Report 2014/613, 2014. `https://eprint.iacr.org/2014/613`. 1, 2, 9, 21, 25

RD08. Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. 20

Res18. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Proposed Standard), August 2018. 1, 2, 3, 18, 21

RS98. Martin Raab and Angelika Steger. "balls into bins" — a simple and tight analysis. In Michael Luby, José D. P. Rolim, and Maria Serna, editors, *Randomization and Approximation Techniques in Computer Science*, pages 159–170, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. 10

RS06. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Vaudenay [Vau06], pages 373–390. 4

RTM21. Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3 – draft-ietf-tls-dtls13-43. `https://tools.ietf.org/html/draft-ietf-tls-dtls13-43`, April 2021. 2, 3, 18, 21

TT21. M. Thomson (Ed.) and S. Turner (Ed.). Using TLS to Secure QUIC. RFC 9001 (Proposed Standard), May 2021. 2, 3, 18, 21

Vau06. Serge Vaudenay, editor. *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer, Heidelberg, May / June 2006. 24

VM05. John Viega and David McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106, June 2005. 20

A Proof of The Single-User Security of ChaCha20-Poly1305

We first comment on an error present in the previous single-user security proof of ChaCha20-Poly1305 by Procter [Pro14]. Then we give a new single-user proof retaining the same security bound as the one originally claimed.

A.1 A Note on Procter's Prior Security Proof

A proof for the single-user setting was previously proposed by Procter [Pro14]. However, we noticed an error in this proof that we briefly describe here.

The error resides in the transition between Game 1 and 2 in [Pro14]. The assumption that, in Game 1, inputs to the random function URF never repeat is false: an encryption query with a nonce N followed by a decryption using the same nonce N are permitted queries, which result in a repeated call to $URF_k(0\|N)$. Therefore Game 1 and Game 2 are not identical.

Fixing this seemingly small error led us to a significantly more complex proof than Procter's analysis would indicate is necessary. In order to reduce the ciphertext integrity security to the almost universal hash property of the function H_r used in Poly1305, one needs to end up in a game where the hash keys r are sampled independently from the previous queries, during decryption. To accomplish this, Procter made the false assumption that nonces never repeat across encryption and decryption queries, and hence he sampled r independently across encryption and decryption queries. We address this in our proof by first reducing ciphertext forgery in a game G_3 with multiple decryption queries to ciphertext forgery in a game $G_3^{(1)}$ with only one decryption query. This allows us to then transition to a game G'' in which, during the only decryption query, either the tag is sampled independently at random or the hash key r is sampled independently at random. In the latter case, we are then able to correctly use the almost universal hash property of the function H_r.

A.2 Proof of Theorem 4.1 (Single-User Security Proof)

We use the code-based game-playing technique of [BR06] and prove the Theorem 4.1 using a sequence of games, starting from a game that corresponds to the real world in the security definition and ending up with a game corresponding to the ideal world. Consider the ChaCha20-Poly1305 nonce-based AEAD scheme described in Section 3 and let $\mathcal{A}$ be a valid nonce-respecting adversary making at most q_v decryption queries. The associated data, messages, and ciphertexts queried by $\mathcal{A}$ are restricted to be byte strings, as required for the use of the almost Δ-universal property of H.

G_1: Game G_1 is the real Game $G^{\text{Real-AE}}_{\text{ChaCha20-Poly1305}}$ (Fig. 2) instantiated with ChaCha20-Poly1305 i.e., the game where an adversary has oracle access to the encryption and decryption algorithm of the ChaCha20-Poly1305 AEAD Scheme.

G_2: Game G_2 is the same as G_1 with the only difference being that the ChaCha20 block function CC_block is replaced by a truly random function TRF. We also introduce a flag forge in the decryption procedure, which is set to true if a valid forgery is submitted to the decryption oracle. This flag is used in the next game G_3 to bound the probability of a valid forgery happening.
For any adversary $\mathcal{A}$ distinguishing between G_1 and G_2, we can construct another adversary $\mathcal{A}_{\text{prf}}$ which breaks the PRF security of the ChaCha20 block function. The adversary $\mathcal{A}_{\text{prf}}$ has oracle access to either the ChaCha20 block function or a truly random function. It runs $\mathcal{A}$ and answers to $\mathcal{A}$ encryption and decryption queries by simulating the encryption and decryption algorithm of ChaCha20-Poly1305 using its own PRF oracle instead of the Chacha20 block function. If $\mathcal{A}$ stops and returns a value, $\mathcal{A}_{\text{prf}}$ stops and returns the output of $\mathcal{A}$. When $\mathcal{A}_{\text{prf}}$ is playing the real PRF game $G^{\text{Real-PRF}}_{\text{CC\_block}}$, it has oracle access to the ChaCha20 block function and provides to $\mathcal{A}$ an exact simulation of the game G_1. Thus $\Pr\left[\mathcal{A}_{\text{prf}}^{G^{\text{Real-PRF}}_{\text{CC\_block}}} \Rightarrow 1\right] = \Pr\left[\mathcal{A}^{G_1} \Rightarrow 1\right]$. When $\mathcal{A}_{\text{prf}}$ is playing the ideal

Game G_1

```
procedure INITIALIZE
$K \leftarrow\$ \{0,1\}^k$

procedure ENC($N, AD, M$)
$M_1 \| \cdots \| M_\ell \leftarrow M$
for $j = 1$ to $\ell - 1$ do
    $C_j \leftarrow M_j \oplus \text{CC\_block}(K, N, j)$
$C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}(\text{CC\_block}(K, N, \ell), |M_\ell|)$
$C \leftarrow C_1 \| \cdots \| C_\ell$
$r \| s \leftarrow \mathsf{trunc}(\text{CC\_block}(K, N, 0), 2t)$
$T \leftarrow H_r(AD, C) \overset{(t)}{+} s$
return $C \| T$

procedure DEC($N, AD, C \| T$)
$r \| s \leftarrow \mathsf{trunc}(\text{CC\_block}(K, N, 0), 2t)$
$T' \leftarrow H_r(AD, C) \overset{(t)}{+} s$
if $T \neq T'$ then return $\perp$
$C_1 \| \cdots \| C_\ell \leftarrow C$
for $j = 1$ to $\ell - 1$ do
    $M_j \leftarrow C_j \oplus \text{CC\_block}(K, N, j)$
$M_\ell \leftarrow C_\ell \oplus \mathsf{trunc}(\text{CC\_block}(K, N, \ell), |C_\ell|)$
return $M_1 \| \cdots \| M_\ell$
```

Game G_2

```
procedure ENC($N, AD, M$)
$M_1 \| \cdots \| M_\ell \leftarrow M$
for $j = 1$ to $\ell - 1$ do
    $C_j \leftarrow M_j \oplus \text{TRF}(N, j)$
$C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}(\text{TRF}(N, \ell), |M_\ell|)$
$C \leftarrow C_1 \| \cdots \| C_\ell$
$r \| s \leftarrow \mathsf{trunc}(\text{TRF}(N, 0), 2t)$
$T \leftarrow H_r(AD, C) \overset{(t)}{+} s$
return $C \| T$

procedure DEC($N, AD, C \| T$)
$r \| s \leftarrow \mathsf{trunc}(\text{TRF}(N, 0), 2t)$
$T' \leftarrow H_r(AD, C) \overset{(t)}{+} s$
if $T \neq T'$ then return $\perp$
$\mathsf{forge} \leftarrow$ true
$C_1 \| \cdots \| C_\ell \leftarrow C$
for $j = 1$ to $\ell - 1$ do
    $M_j \leftarrow C_j \oplus \text{TRF}(N, j)$
$M_\ell \leftarrow C_\ell \oplus \mathsf{trunc}(\text{TRF}(N, \ell), |C_\ell|)$
return $M_1 \| \cdots \| M_\ell$
```

Fig. 10. Games G_1 and G_2 used for proving the single-user security of ChaCha20-Poly1305. Highlighting indicates changes from the respective prior game.

PRF game $G^{\text{Ideal-PRF}}_{\text{CC\_block}}$, it has oracle access to a truly random function and provides to $\mathcal{A}$ an exact simulation of the game G_2. Thus $\Pr\left[\mathcal{A}_{\text{prf}}^{G^{\text{Ideal-PRF}}_{\text{CC\_block}}} \Rightarrow 1\right] = \Pr\left[\mathcal{A}^{G_2} \Rightarrow 1\right]$. Therefore:

$$\Pr\left[\mathcal{A}^{G_1} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G_2} \Rightarrow 1\right] = \Pr\left[\mathcal{A}_{\text{prf}}^{G^{\text{Real-PRF}}_{\text{CC\_block}}} \Rightarrow 1\right] - \Pr\left[\mathcal{A}_{\text{prf}}^{G^{\text{Ideal-PRF}}_{\text{CC\_block}}} \Rightarrow 1\right] = \mathsf{Adv}^{\text{PRF}}_{\text{CC\_block}}(\mathcal{A}_{\text{prf}}). \quad (7)$$

The adversary $\mathcal{A}_{\text{prf}}$ makes at most $\sigma_e + q_e + \sigma_v + q_v$ queries.

G_3: Game G_3 has the same encryption oracle as Game G_2, but its decryption oracle always returns $\perp$. We can see that Game G_3 is identical to Game G_2 unless the flag forge is set to true (identical until forge). The differences between these games happen only after, and if, the flag forge is set to true. Using the fundamental lemma of game playing [BR06]:

$$\Pr\left[\mathcal{A}^{G_2} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{G_3} \Rightarrow 1\right] \leq \Pr\left[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\right]. \quad (8)$$

The flag forge is set to true in G_3, when the adversary $\mathcal{A}$ provides a valid forgery to the decryption oracle of G_3. We are going to bound the probability of this event, by transforming the game into another one where only one decryption query is allowed, and by bounding the probability of forgery in this new game.

$G_3^{(1)}$: Game $G_3^{(1)}$ is identical to Game G_3, except that the adversary $\mathcal{A}$ is allowed to make only one query to the decryption oracle (but still multiple queries to the encryption oracle). For the subsequent analysis of $G_3^{(1)}$, observe that anything happening after the decryption query in $G_3^{(1)}$ does not influence the probability of forge being set, as forge can only be set during this unique decryption query. Therefore, as we only want to bound the probability of forge being set in $G_3^{(1)}$, without loss of generality, we are going to consider for the encryption oracle of $G_3^{(1)}$, only the encryption queries made before the decryption query.

```
Game $G_3$
procedure $\mathrm{Enc}(N, AD, M)$
1 : $M_1 \| \cdots \| M_\ell \leftarrow M$
2 : for $j = 1$ to $\ell - 1$ do
3 :     $C_j \leftarrow M_j \oplus \mathrm{TRF}(N, j)$
4 : $C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}(\mathrm{TRF}(N, \ell), |M_\ell|)$
5 : $C \leftarrow C_1 \| \cdots \| C_\ell$
6 : $r \| s \leftarrow \mathsf{trunc}(\mathrm{TRF}(N, 0), 2t)$
7 : $T \leftarrow H_r(AD, C) \overset{(t)}{+} s$
8 : return $C \| T$

procedure $\mathrm{Dec}(N, AD, C \| T)$
1 : $r \| s \leftarrow \mathsf{trunc}(\mathrm{TRF}(N, 0), 2t)$
2 : $T' \leftarrow H_r(AD, C) \overset{(t)}{+} s$
3 : if $T \neq T'$ then return $\perp$
4 : $\mathsf{forge} \leftarrow$ true
5 : return $\perp$
```

```
Game $G'$
procedure $\mathrm{Enc}(N, AD, M)$
1 : $M_1 \| \cdots \| M_\ell \leftarrow M$
2 : for $j = 1$ to $\ell - 1$ do
3 :     $C_j \leftarrow M_j \oplus \mathrm{TRF}(N, j)$
4 : $C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}(\mathrm{TRF}(N, \ell), |M_\ell|)$
5 : $C \leftarrow C_1 \| \cdots \| C_\ell$
6 : $r \leftarrow\$ \{0,1\}^t, s \leftarrow\$ \{0,1\}^t$
7 : $\mathsf{VS}[N] \leftarrow (r, s)$
8 : $T \leftarrow H_r(AD, C) \overset{(t)}{+} s$
9 : return $C \| T$

procedure $\mathrm{Dec}(N, AD, C \| T)$
1 : $(r, s) \leftarrow \mathsf{VS}[N]$
2 : if $(r, s) = (\perp, \perp)$ then
3 :     $r \leftarrow\$ \{0,1\}^t, s \leftarrow\$ \{0,1\}^t$
4 : $T' \leftarrow H_r(AD, C) \overset{(t)}{+} s$
5 : if $T \neq T'$ then return $\perp$
6 : $\mathsf{forge} \leftarrow true$
7 : return $\perp$
```

```
Game $G''$
procedure $\mathrm{Enc}(N, AD, M)$
1 : $M_1 \| \cdots \| M_\ell \leftarrow M$
2 : for $j = 1$ to $\ell - 1$ do
3 :     $C_j \leftarrow M_j \oplus \mathrm{TRF}(N, j)$
4 : $C_\ell \leftarrow M_\ell \oplus \mathsf{trunc}(\mathrm{TRF}(N, \ell), |M_\ell|)$
5 : $C \leftarrow C_1 \| \cdots \| C_\ell$
6 : $T \leftarrow\$ \{0,1\}^t$
7 : $\mathsf{VS}[N] \leftarrow (T, AD, C)$
8 : return $C \| T$

procedure $\mathrm{Dec}(N, AD, C \| T)$
1 : $(T^*, AD^*, C^*) \leftarrow \mathsf{VS}[N]$
2 : if $T^* = \perp$ then
3 :     $T' \leftarrow\$ \{0,1\}^t$
4 : else
5 :     $r \leftarrow\$ \{0,1\}^t, s \leftarrow T^* \overset{(t)}{-} H_r(AD^*, C^*)$
6 :     $T' \leftarrow H_r(AD, C) \overset{(t)}{+} s$
7 : if $T \neq T'$ then return $\perp$
8 : $\mathsf{forge} \leftarrow true$
9 : return $\perp$
```

Fig. 11. Games G_3, G', and G'' used for proving the single-user security of ChaCha20-Poly1305. Highlighting indicates changes from the respective prior game.

Let $I_G^{\mathcal{A}}$ denote the random variable indicating at which decryption query the adversary $\mathcal{A}$ forges for the first time in some game G, where $I_G^{\mathcal{A}} = 0$, if $\mathcal{A}$ does not forge. Let $\mathcal{A}$ be an adversary playing game G_3 and making q_v decryption queries. Then $I_{G_3}^{\mathcal{A}}$ denote the number of decryption queries made by $\mathcal{A}$ before a decryption query (the $I_{G_3}^{\mathcal{A}}$-th) set forge to true for the first time. Thus $\Pr\big[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\big] = \sum_{i=1}^{q_v} \Pr\big[I_{G_3}^{\mathcal{A}} = i\big]$. Similarly to the reduction used in [BGM04, Thm B.2], we construct from $\mathcal{A}$ an adversary $\mathcal{B}$ playing $G_3^{(1)}$ and making only one decryption query. Then using a reduction argument, we prove that $\Pr\big[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\big] \leq q_v \cdot \Pr\Big[\mathcal{B}^{G_3^{(1)}} \text{ sets } \mathsf{forge}\Big]$. The adversary $\mathcal{B}$ is specified in figure 12.

Let S be the simulated game for $\mathcal{A}$ in the reduction. The adversary $\mathcal{B}$ provides to $\mathcal{A}$ an exact simulation of game G_3 until the $I_{\mathrm{S}}^{\mathcal{A}}$-th decryption query, thus $\Pr\big[I_{\mathrm{S}}^{\mathcal{A}} = i \,\big|\, \mathsf{guess} = i\big] = \Pr\big[I_{G_3}^{\mathcal{A}} = i\big]$. Furthermore, if $\mathcal{A}^{\mathrm{S}}$ sets forge (i.e., $I_{\mathrm{S}}^{\mathcal{A}} \geq 1$) and guess is equal to $I_{\mathrm{S}}^{\mathcal{A}}$, then $\mathcal{B}$ sets forge. Therefore

$$\Pr\Big[\mathcal{B}^{G_3^{(1)}} \text{ sets } \mathsf{forge}\Big] \geq \Pr\big[I_{\mathrm{S}}^{\mathcal{A}} = \mathsf{guess}\big]. \tag{9}$$

```
Adversary $\mathcal{B}^{G_3^{(1)}}$
$\eta_{\mathsf{dquery}} \leftarrow 0, \mathsf{guess} \leftarrow\!\!\$ \{1,\dots,q_v\}$
Run $\mathcal{A}$ and reply to its oracle queries as follows:
- If $\mathcal{A}$ makes a query to its encryption oracle, reply with the encryption oracle of $\mathcal{B}$.
- If $\mathcal{A}$ makes a query to its decryption oracle, do:
    $\eta_{\mathsf{dquery}} \leftarrow \eta_{\mathsf{dquery}} + 1$
    if $\eta_{\mathsf{dquery}} \neq \mathsf{guess}$ then reply $\perp$ to $\mathcal{A}$
    if $\eta_{\mathsf{dquery}} = \mathsf{guess}$ then reply with the decryption oracle of $\mathcal{B}$.
If $\mathcal{A}$ stops and returns a value, $\mathcal{B}$ stops and returns the output of $\mathcal{A}$.
```

Fig. 12. Description of the adversary $\mathcal{B}$ used in the reduction from G_3 to $G_3^{(1)}$.

Moreover :

$$\Pr\left[I_{\mathrm{S}}^{\mathcal{A}} = \mathsf{guess}\right] = \sum_{i=1}^{q_v} \Pr\left[I_{\mathrm{S}}^{\mathcal{A}} = i \wedge \mathsf{guess} = i\right] = \sum_{i=1}^{q_v} \Pr\left[I_{\mathrm{S}}^{\mathcal{A}} = i \;\middle|\; \mathsf{guess} = i\right] \cdot \Pr[\mathsf{guess} = i]$$
$$= \sum_{i=1}^{q_v} \Pr\left[I_{G_3}^{\mathcal{A}} = i\right] \cdot \frac{1}{q_v} = \Pr\left[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\right] \cdot \frac{1}{q_v}. \tag{10}$$

By combining (9) and (10), we get

$$\Pr\left[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\right] \leq q_v \cdot \Pr\left[\mathcal{B}^{G_3^{(1)}} \text{ sets } \mathsf{forge}\right]. \tag{11}$$

G': Game G' is identical to game $G_3^{(1)}$, with only one decryption query allowed, but with the difference that in Game G', we now lazily sample the output of the random function TRF on inputs $(N, 0)$ (for any N), in both the encryption and decryption oracle. For this, we keep track of the values already sampled in a vector VS indexed by N, where we assume that all entries in VS are initalized to $(\perp, \perp)$ at the beginning of the game. As in Game $G_3^{(1)}$, we are only interested in bounding the probability of forge being set in G' (which can only be done during the unique decryption query), and hence only need to consider those encryption queries made before the decryption query. (Indeed, the lazy sampling might become inconsistent after the decryption query, which will not burden us.) For each of these encryption queries, a new nonce N is used and we sample lazily a new pair (r, s) of one time keys. For the decryption query, (r, s) are sampled only if they were not already sampled during a previous encryption query. The lazy sampling does not change the behavior of the game up to the decryption query, therefore the probability of setting forge is identical in Games $G_3^{(1)}$ and G':

$$\Pr\left[\mathcal{B}^{G_3^{(1)}} \text{ sets } \mathsf{forge}\right] = \Pr\left[\mathcal{B}^{G'} \text{ sets } \mathsf{forge}\right]. \tag{12}$$

G'': Game G'' is similar to game G', except for two changes that leave the overall behavior of the game before the decryption query unchanged. (Again, we are only concerned with bounding the probability of forge being set in the single decryption query.) In Game G'', instead of sampling r and s, and computing T (or T') as $H_r(AD, C) \overset{(t)}{+} s$, we instead sample r and T (or T') and compute s as $T \overset{(t)}{-} H_r(AD, C)$ (or $T' \overset{(t)}{-} H_r(AD, C)$). Since r and s are never reused within encryption queries (due to the changes from G'), we can further defer the sampling of r and computation of s to when they would be reused in the decryption query. To do so, we have VS store (T, AD, C), which, after sampling r for decryption, allows us to compute s as $T \overset{(t)}{-} H_r(AD, C)$. In case the values (T^*, AD^*, C^*), corresponding to the nonce N queried to decryption, are not stored in VS, we can directly sample T' at random, which is equivalent to sampling r and s at random as in G'.

The difference between Games G' and G'' (swapping s with T and deferring the computation of r and s) being only conceptual and indistinguishable to the adversary, we have:

$$\Pr\left[\mathcal{B}^{G'} \text{ sets } \mathsf{forge}\right] = \Pr\left[\mathcal{B}^{G''} \text{ sets } \mathsf{forge}\right]. \tag{13}$$

During the decryption query, either it is the first use of the nonce N, and T' is sampled uniformly from $\{0,1\}^t$, or it is the second use of the nonce N (the first one was an encryption query), and r is sampled uniformly from $\{0,1\}^t$. In the latter case, T' is computed as $H_r(AD, C) \overset{(t)}{+} T^* \overset{(t)}{-} H_r(AD^*, C^*)$ for $(AD, C) \neq (AD^*, C^*)$. Therefore:

$$\begin{aligned}
\Pr\Big[\mathcal{B}^{G''} \text{ sets } \mathsf{forge}\Big] &= \Pr[T = T' \text{ during the only decryption query of } G''] \\
&\leq \max\left(\frac{1}{2^t}, \Pr_{r \leftarrow\$ \{0,1\}^t}\Big[T = H_r(AD, C) \overset{(t)}{+} T^* \overset{(t)}{-} H_r(AD^*, C^*)\Big]\right) \\
&\leq \frac{\epsilon(\ell_m)}{2^t}. \quad \text{(Thm. 3.2)} \qquad (14)
\end{aligned}$$

Combining Equations (11), (12), (13) and (14), we can conclude our sequence of games bounding the difference between G_2 and G_3 as

$$\Pr\big[\mathcal{A}^{G_2} \Rightarrow 1\big] - \Pr\big[\mathcal{A}^{G_3} \Rightarrow 1\big] \leq \Pr\big[\mathcal{A}^{G_3} \text{ sets } \mathsf{forge}\big] \leq q_v \cdot \frac{\epsilon(\ell_m)}{2^t}. \qquad (15)$$

Game G_4

```
procedure Enc(N, AD, M)
$M_1 \| \cdots \| M_\ell \leftarrow M$
for $j = 1$ to $\ell - 1$ do
    $Y \leftarrow\$ \{0,1\}^n$
    $C_j \leftarrow M_j \oplus Y$
$Y \leftarrow\$ \{0,1\}^{|M_\ell|}$
$C_\ell \leftarrow M_\ell \oplus Y$
$C \leftarrow C_1 \| \cdots \| C_\ell$
$r \leftarrow\$ \{0,1\}^t, s \leftarrow\$ \{0,1\}^t$
$T \leftarrow H_r(AD, C) \overset{(t)}{+} s$
return $C \| T$

procedure Dec(N, AD, C||T)
return $\perp$
```

Game G_5

```
procedure Enc(N, AD, M)
$M_1 \| \cdots \| M_\ell \leftarrow M$
for $j = 1$ to $\ell - 1$ do
    $C_j \leftarrow\$ \{0,1\}^n$
$C_\ell \leftarrow\$ \{0,1\}^{|M_\ell|}$
$C \leftarrow C_1 \| \cdots \| C_\ell$
$T \leftarrow\$ \{0,1\}^t$
return $C \| T$

procedure Dec(N, AD, C||T)
return $\perp$
```

Fig. 13. Games G_4 and G_5 used for proving the single-user security of ChaCha20-Poly1305. Highlighting indicates changes from the respective prior game.

G_4: Game G_4 is identical to Game G_3, except that we sample lazily the random function TRF. To obtain Game G_4 from Game G_3, observe that the decryption oracle of Game G_3 does not have any side effects and always returns $\perp$, so in addition to the lazy sampling, we have the decryption oracle directly return $\perp$. As a nonce can only be used once in an encryption query, the values of the random function are only used once and therefore do not need to be stored. These modifications being indistinguishable for any adversary, we get:

$$\Pr\big[\mathcal{A}^{G_3} \Rightarrow 1\big] = \Pr\big[\mathcal{A}^{G_4} \Rightarrow 1\big]. \qquad (16)$$

G_5: Finally, Game G_5 is the ideal AE game $G^{\text{Ideal-AE}}_{\text{ChaCha20-Poly1305}}$ (Fig. 2), i.e., its encryption procedure returns a random bit-string of length equal to the length of the message plus t (the bit-length of the tag), and its decryption procedure always returns $\bot$. In this game, we make two changes that leave the overall behavior of the game unchanged. First, we swap the dependent and independent variables C_j and Y: for all j, instead of sampling $Y \leftarrow\!\!\$ \{0,1\}^n$ and defining $C_j \leftarrow M_j \oplus Y$, we sample $C_j \leftarrow\!\!\$ \{0,1\}^n$ and define $Y \leftarrow M_j \oplus C_j$. As Y is not being used in G_5 anymore, we remove it. The distribution of the ciphertexts returned by the encryption oracle is unchanged from the adversary's point of view. Secondly, instead of sampling s and computing T as $H_r(AD, C) \overset{(t)}{+} s$, we sample T and compute s as $T \overset{(t)}{-} H_r(AD, C)$. These processes are equivalent and leave the distribution of s and T unchanged. With those changes, r and s are not used anymore, thus again we remove them. The differences between Game G_4 and Game G_5 (swapping Y with C_j and s with T) being only conceptual and indistinguishable to the adversary, we obtain:

$$\Pr\big[\mathcal{A}^{G_4} \Rightarrow 1\big] = \Pr\big[\mathcal{A}^{G_5} \Rightarrow 1\big]. \tag{17}$$

The combination of (7), (8), (15), (16) and (17) yields the bound of the theorem. □

B Proof of Balls-Into-Bins Theorem

We prove the balls-into-bins theorem in two stages via a preliminary lemma. The improvement over [BHT18] is a direct consequence of this lemma. Lemma B.1 is additionally used to prove Theorem 7.1, resulting in an improved bound for nonce randomization.

B.1 Preliminary Lemma

The proof of Theorem 5.1 and Theorem 7.1 is based on the following lemma.

Lemma B.1. *Let $m, Q \in \mathbb{N}^*$, $D \in (0,1]$ and $\widetilde{m}, \lambda \in \mathbb{R}_{>0}$, with $\lambda > 1$. Let* e *denote Euler's number. For any of the following cases:*

1. $m = \left\lceil \frac{\log_\lambda(D^{-1})+\widetilde{m}}{\log_\lambda((QD)^{-1})} \right\rceil$ *and* $Q \le \frac{1}{D}$,
2. $m = \lceil \log_\lambda(D^{-1}) + \widetilde{m} \rceil$ *and* $Q \le \frac{\log_\lambda(D^{-1})+\widetilde{m}}{D\lambda \mathrm{e}}$,
3. $m = \lceil QD\lambda\mathrm{e} \rceil$ *and* $Q \ge \frac{\log_\lambda(D^{-1})+\widetilde{m}}{D\lambda \mathrm{e}}$,
4. $m = \left\lceil \frac{\max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1})+\widetilde{m})}{\max(1, \log_\lambda((QD)^{-1}))} \right\rceil$,

it holds that $\binom{Q}{m} \cdot D^{m-1} \le \lambda^{-\widetilde{m}}$.

Note that Cases 2 and 3 can be combined into $m = \lceil \max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1}) + \widetilde{m}) \rceil$, and if $Q \le \frac{\log_\lambda(D^{-1})+\widetilde{m}}{D\lambda\mathrm{e}}$, Case 4 gives us $m = \left\lceil \frac{\log_\lambda(D^{-1})+\widetilde{m}}{\max(1,\log_\lambda((QD)^{-1}))} \right\rceil \le \lceil \log_\lambda(D^{-1}) + \widetilde{m} \rceil$.

Proof. We consider the different cases in turn.

Case 1: $m = \left\lceil \frac{\log_\lambda(D^{-1})+\widetilde{m}}{\log_\lambda((QD)^{-1})} \right\rceil$ and $Q \le \frac{1}{D}$.
For this case,

$$\binom{Q}{m} \cdot D^{m-1} \le Q^m \cdot D^{m-1} = D^{-1} \cdot (QD)^m = D^{-1} \cdot \lambda^{-m \cdot \log_\lambda((QD)^{-1})}.$$

As $\lambda > 1$, $QD \le 1$ and $m \ge \frac{\log_\lambda(D^{-1})+\widetilde{m}}{\log_\lambda((QD)^{-1})}$, then

$$\lambda^{-m \cdot log_\lambda((QD)^{-1})} \le \lambda^{-\log_\lambda(D^{-1})-\widetilde{m}} = D \cdot \lambda^{-\widetilde{m}}.$$

Hence $\displaystyle\binom{Q}{m} \cdot D^{m-1} \le \lambda^{-\widetilde{m}}$.

Cases 2 and 3: $m = \lceil \max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1}) + \widetilde{m}) \rceil$.
For these cases, as $\binom{Q}{m} < \left(\frac{Q\cdot \mathrm{e}}{m}\right)^m$, we have

$$\binom{Q}{m} \cdot D^{m-1} < \left(\frac{Q\cdot \mathrm{e}}{m}\right)^m \cdot D^{m-1} = D^{-1} \cdot \left(\frac{QD\cdot \mathrm{e}}{m}\right)^m .$$

Moreover, as $m \geq QD\lambda\mathrm{e}$,

$$D^{-1} \cdot \left(\frac{QD\cdot \mathrm{e}}{m}\right)^m \leq D^{-1} \cdot \lambda^{-m}.$$

Finally, as $m \geq \log_\lambda(D^{-1}) + \widetilde{m}$, we obtain

$$D^{-1} \cdot \lambda^{-m} \leq D^{-1} \cdot D \cdot \lambda^{-\widetilde{m}} = \lambda^{-\widetilde{m}}.$$

Hence $\binom{Q}{m} \cdot D^{m-1} \leq \lambda^{-\widetilde{m}}$.

Case 4: $m = \left\lceil \frac{\max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1})+\widetilde{m})}{\max(1, \log_\lambda((QD)^{-1}))} \right\rceil$.
When $(QD)^{-1} < \lambda$, then $\max(1, \log_\lambda((QD)^{-1})) = 1$. Thus $m = \lceil \max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1}) + \widetilde{m}) \rceil$ and we can reuse results from Cases 2 and 3.
When $(QD)^{-1} \geq \lambda$, then $\max(1, \log_\lambda((QD)^{-1})) = \log_\lambda((QD)^{-1})$ and $m = \left\lceil \frac{\max(QD\lambda\mathrm{e}, \log_\lambda(D^{-1})+\widetilde{m})}{\log_\lambda((QD)^{-1})} \right\rceil$.
Thus $m \geq \frac{\log_\lambda(D^{-1})+\widetilde{m}}{\log_\lambda((QD)^{-1})}$ and similarly to Case 1,

$$\binom{Q}{m} \cdot D^{m-1} \leq D^{-1} \cdot \lambda^{-m\cdot \log_\lambda((QD)^{-1})} \leq D^{-1} \cdot \lambda^{-\log_\lambda(D^{-1}) - \widetilde{m}} = \lambda^{-\widetilde{m}}.$$

□

B.2 Proof of Theorem 5.1 (Biased Balls-Into-Bins Theorem)

We are now ready to show how Theorem 5.1 is derived from Lemma B.1

Since $m \geq 1$, when $Q = 0$ the probability that the heaviest bin contains m or more balls is clearly 0. Thus, we can focus on the case where at most $Q > 0$ balls are thrown into bins and where m satisfies one of the theorem conditions. If $m > Q$, the probability that the heaviest bin contains m or more balls is 0. If $m \leq Q$, there are at most $\binom{Q}{m}$ subsets of m balls among the set of all balls thrown. If the heaviest bin contains m or more balls, then at least one of these subsets of m balls is in the same bin. Hence, to bound the probability that the heaviest bin contains m or more balls in this case, we only need to bound the probability that at least one of these subsets is in the same bin. For each of these subsets, the probability that all m balls in that subset are thrown into the same bin is at most D^{m-1}. Thus the probability that at least one of these subsets of m balls is in the same bin is at most $\binom{Q}{m} \cdot D^{m-1}$. Using Lemma B.1, $\binom{Q}{m} \cdot D^{m-1} \leq \lambda^{-\widetilde{m}}$ for the four different cases. Hence the probability that the heaviest bin contains m or more balls is also at most $\lambda^{-\widetilde{m}}$. □

C Missing Components in the Proof of Theorem 6.1

In this section, we prove the remaining pieces that are necessary to complete the proof of the multi-user security of ChaCha20-Poly1305 outlined in Section 6. In Appendix C.1 we prove Proposition 6.1, which lower bounds the ratio of good transcripts, and in Appendix C.2 we prove Lemmas C.2–C.7, which upper bound the probabilities of bad transcripts in the ideal world.

C.1 Proof of Proposition 6.1 (Good Transcript Ratio)

Let $\tau \in \mathbb{T}_{\mathrm{good}}$ be a good transcript and let τ_{key}, τ_{prim}, τ_{enc}, and τ_{vf} be the sets of revealed-key, ideal-permutation, encryption, and verification entries in τ, respectively. Let $u = |\tau_{\mathsf{key}}|$ be the number of key entries. Let $\mathsf{P}_{\mathrm{ideal}}(\tau)$, resp. $\mathsf{P}_{\mathrm{real}}(\tau)$, be the probability that if we make the queries described in τ (in the same order), we receive in the ideal game, resp. the real game, the corresponding answers recorded in τ.

In the ideal world, the revealed keys and the answers to the ideal permutation, encryption, and verification queries are generated independently. Thus,

$$\mathsf{P}_{\text{ideal}}(\tau) = \mathsf{P}_{\text{ideal}}(\tau_{\mathsf{key}}) \cdot \mathsf{P}_{\text{ideal}}(\tau_{\mathsf{prim}}) \cdot \mathsf{P}_{\text{ideal}}(\tau_{\mathsf{enc}}) \cdot \mathsf{P}_{\text{ideal}}(\tau_{\mathsf{vf}}).$$

Recall that for good transcripts, the calls to the ideal permutation π and random blocks induced by ideal permutation and encryption entries are distinct. Therefore, there are exactly $|S_1(\tau)|$ distinct calls to the ideal permutation in τ_{prim} and $|S_2(\tau)|$ independently sampled random blocks generated in τ_{enc}. Hence $\mathsf{P}_{\text{ideal}}(\tau_{\mathsf{prim}}) = \prod_{i=0}^{|S_1(\tau)|-1} \frac{1}{2^n - i}$ and $\mathsf{P}_{\text{ideal}}(\tau_{\mathsf{enc}}) = \prod_{i=0}^{|S_2(\tau)|-1} \frac{1}{2^n}$. Moreover, as the u-many k-bit user keys are sampled at random and verification queries always return false, we obtain $\mathsf{P}_{\text{ideal}}(\tau_{\mathsf{key}}) = 2^{-ku}$ and $\mathsf{P}_{\text{ideal}}(\tau_{\mathsf{vf}}) = 1$. Consequently,

$$\mathsf{P}_{\text{ideal}}(\tau) = 2^{-ku} \cdot \prod_{i=0}^{|S_1(\tau)|-1} \frac{1}{2^n - i} \cdot \prod_{i=0}^{|S_2(\tau)|-1} \frac{1}{2^n}. \tag{18}$$

In the real world, the user keys are also sampled at random and $\mathsf{P}_{\text{real}}(\tau_{\mathsf{key}}) = 2^{-ku}$. Once the user keys have been sampled, the probability of query outputs depends on the number of distinct ideal permutation calls made. For good transcripts in the real world, the calls to the ideal permutation π in ideal permutation and encryption entries are distinct. Therefore, there are exactly $|S_1(\tau)| + |S_2(\tau)|$ distinct calls to the ideal permutation in $\tau_{\mathsf{prim}} \cup \tau_{\mathsf{enc}}$, and the input-output of these calls are also entirely determined in these sets. Moreover, there are at most $|S_3(\tau)|$ more calls to the ideal permutation done during verification queries and distinct from the ones done during ideal permutation and encryption queries. However, the input-output of these calls is not entirely determined in τ_{vf}, only the inputs are. For the outputs, as τ is a good transcript, they are required to be distinct from the ones in $\tau_{\mathsf{prim}} \cup \tau_{\mathsf{enc}}$ and not to result in a forgery for queries in τ_{vf}. If q is the exact number of distinct ideal permutation calls in τ, the probability to get these calls is $\prod_{i\in I} \frac{1}{2^n - i} \cdot \prod_{j \in J} \left(1 - \frac{f_j}{2^n - j}\right)$, where I, J is a partition of $\{0, \ldots, q-1\}$ fixed by the order of queries in τ and f_j is the number of values that would result in a forgery while sampling the associated ideal permutation call during verification queries. I contains the index/order in which the associated ideal permutation calls from $\tau_{\mathsf{prim}} \cup \tau_{\mathsf{enc}}$ are sampled and J contains the remaining ones from τ_{vf}. Note that $|I| = |S_1(\tau)| + |S_2(\tau)|$ and $|J| \leq |S_3(\tau)|$. Thus,

$$\mathsf{P}_{\text{real}}(\tau) = 2^{-ku} \cdot \prod_{i \in I} \frac{1}{2^n - i} \cdot \prod_{j \in J} \left(1 - \frac{f_j}{2^n - j}\right).$$

Furthermore, we can reorder and minimize the products in the following way:

$$\prod_{i \in I} \frac{1}{2^n - i} \cdot \prod_{j \in J} \left(1 - \frac{f_j}{2^n - j}\right) \geq \prod_{i=0}^{|I|-1} \frac{1}{2^n - i} \cdot \prod_{j=|I|}^{|I|+|S_3(\tau)|-1} \left(1 - \frac{F}{2^n - j}\right),$$

where F is a bound on the number of values that would result in a forgery while sampling the associated ideal permutation call of one verification query, i.e. for a verification query $\text{V}{\scriptstyle\text{F}}(i, N, AD, C\|T)$, the number of $(r\|s\|W) \in \{0,1\}^n$ such that $H_r(AD, C) \overset{(t)}{+} s = T$. There are 2^{n-2t} possible values for W, and each of the 2^t possible values for r yield exactly one value s such that $H_r(AD, C) \overset{(t)}{+} s = T$. Therefore, we can consider $F = 2^{n-2t} \cdot 2^t = 2^{n-t}$ in the previous inequality. Consequently,

$$\mathsf{P}_{\text{real}}(\tau) \geq 2^{-ku} \cdot \prod_{i=0}^{|S_1(\tau)|+|S_2(\tau)|-1} \frac{1}{2^n - i} \cdot \prod_{j=0}^{|S_3(\tau)|-1} \left(1 - \frac{2^{n-t}}{2^n - |S_1(\tau)| - |S_2(\tau)| - j}\right). \tag{19}$$

Combining (18) and (19), we can now calculate the probability ratio of a good transcript:

$$\begin{aligned} \frac{\mathsf{P}_{\text{real}}(\tau)}{\mathsf{P}_{\text{ideal}}(\tau)} &\geq \prod_{j=0}^{|S_3(\tau)|-1} \left(1 - \frac{2^{n-t}}{2^n - |S_1(\tau)| - |S_2(\tau)| - j}\right) \\ &\geq \prod_{j=0}^{|S_3(\tau)|-1} \left(1 - \frac{2^{n-t}}{2^n - |S_1(\tau)| - |S_2(\tau)| - |S_3(\tau)|}\right). \end{aligned}$$

As $|S_1(\tau)| \leq p \leq \frac{n-k-1}{6} \cdot 2^{n-k}$, $|S_2(\tau)| \leq \sigma_e + q_e \leq \frac{n-k}{3} \cdot 2^{n-k}$ and $|S_3(\tau)| \leq q_v \leq 2^{n-2}$, then

$$|S_1(\tau)| + |S_2(\tau)| + |S_3(\tau)| \leq p + \sigma_e + q_e + q_v \leq (n-k) \cdot 2^{n-k} + 2^{n-2} \leq 2^{n-1}.$$

Hence,

$$\frac{\mathsf{P}_{\text{real}}(\tau)}{\mathsf{P}_{\text{ideal}}(\tau)} \geq \prod_{j=0}^{|S_3(\tau)|-1} \left(1 - \frac{2^{n-t}}{2^{n-1}}\right) = \left(1 - \frac{1}{2^{t-1}}\right)^{|S_3(\tau)|} \geq 1 - \frac{|S_3(\tau)|}{2^{t-1}} \geq 1 - \frac{q_v}{2^{t-1}} = 1 - \frac{2q_v}{2^t}. \qquad \square$$

C.2 Proofs of Bad Transcript Probabilities

In this subsection, we bound the probabilities of the six sets of Bad transcripts in the ideal world through Lemma C.2–C.7. However, we first give a corollary of our balls-into-bins theorem to simplify its application to our lemmas.

Recall that all the transcripts are generated by a valid nonce-respecting adversary $\mathcal{A}$ that is d-repeating. We also recall here that in the ideal world, the keys K_i are uniformly sampled at the end of the execution, during the last oracle query to Reveal, and are therefore independent of any other previous queries. Moreover, in the ideal world, all the V_j values are independent and uniformly distributed. As for $V_0 = (r \| (T \overset{(t)}{-} H_r(AD, C)) \| W)$, the values r, T and W are uniformly distributed, for $V_j = (M_j \oplus C_j)$, the value C_j is uniformly distributed and for $V_\ell = ((M_\ell \oplus C_\ell) \| W')$, the values C_ℓ and W' are uniformly distributed.

C.2.1 Balls-into-bins corollary.

C.2.1 Balls-into-bins corollary. The following lemma is a direct corollary of our balls-into-bins theorem. It will be used below to bound bad transcript probabilities, specifically in Lemmas C.3, C.6, and C.7. To simplify the computed bounds, we apply our generalized balls-into-bins theorem in the proof of the lemma only for a bounded number of balls. It should be noted that we could lift some of the restrictions on the number of queries in Theorem 6.1 and 7.2 by considering an unrestricted number of balls, however at the expense of a more complicated bound.

Lemma C.1. *Consider an experiment where at most Q balls are thrown into a set of bins, where each throw may depend on the outcome of the prior ones. Let $D \in (0, 1]$ be an upper bound on the probability that, when conditioned on prior throws, a ball lands into any bin. If $Q \leq D^{-1} \cdot \frac{\log_2(D^{-1})}{3}$, the probability that the heaviest bin contains $m = \left\lceil \frac{2\log_2(D^{-1})}{\max(1, \log_2((QD)^{-1}))} \right\rceil$ or more balls is at most D.*

Proof. We simply apply the Case 4 of Theorem 5.1, with $\lambda = 2$, $\widetilde{m} = \log_\lambda(D^{-1})$ and $Q \leq D^{-1} \cdot \frac{\log_2(D^{-1})}{3}$. $\square$

Note that compared to [BHT18, Lemma 11], our maximum load m can be smaller than $\log_2(D^{-1})$. Also, compared to [BHT18, Lemma 10], our maximal number of queries can be bigger than D^{-1} (when $\frac{\log_2(D^{-1})}{3} > 1$) and our maximum load m is always smaller than $\lceil 2\log_2(D^{-1}) \rceil$.

C.2.2 Bounding $\mathbf{Bad_1}$ transcripts probability.

Lemma C.2 (Probability of Bad_1 transcripts).
Let Bad_1 be the set of all attainable transcripts that contain two entries $(\mathsf{prim}, x, y, \cdot)$ and $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\| \cdots \|V_\ell)$ such that $x \in \{Z\|K\|0\|N, \ldots, Z\|K\|\ell\|N\}$ and $K_i = K$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_1] \leq \frac{pd}{2^k}.$$

Proof. The main idea we use to bound $\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_1]$ is to count the number of entry pairs of the form $((\mathsf{prim}, Z\|K\| \cdot \|N, \cdot, \cdot), (\mathsf{enc}, i, N, \cdot, \cdot, \cdot, \cdot))$ and use a union bound over the events that for such a pair, the independently sampled key K_i is equal to K.

If a transcript generated by the adversary $\mathcal{A}$ in the ideal augmented game is in Bad_1, then $\mathcal{A}$ has made an encryption query Enc(i, N, AD, M) and either a query Prim$(Z\|K\| \cdot \|N)$ or a query Prim$^{-1}(y)$ with answer $Z\|K\| \cdot \|N$, and finally a query to Reveal that returned the key K_i to be equal to K. Hence the probability that a transcript generated by $\mathcal{A}$ in the ideal augmented game is in Bad_1 is bounded by the probability that $\mathcal{A}$ makes the previously described queries.

We are going to consider the case where $\mathcal{A}$ is just about to query REVEAL, but has already made all its other oracle queries. For each of the at most p ideal permutation queries $\textsc{Prim}(Z\|K\|\cdot\|N)$ or $\textsc{Prim}^{-1}(y)$ with answer $Z\|K\|\cdot\|N$ done by $\mathcal{A}$, there are at most d encryption queries $\textsc{Enc}(i, N', AD, M)$ with $N' = N$ done by $\mathcal{A}$. Thus there are at most pd possible pairs of such queries done by $\mathcal{A}$. When querying REVEAL, for each of these pairs, the probability that $K_i = K$ is $\frac{1}{2^k}$, as the keys are uniformly sampled, independently from any previous queries. Hence, using a union bound, the probability that for at least one of these pairs $K_i = K$, is at most $\frac{pd}{2^k}$. Thus,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_1] \leq \frac{pd}{2^k}. \qquad \square$$

C.2.3 Bounding $\mathbf{Bad_2}$ transcripts probability.

Lemma C.3 (Probability of Bad_2 transcripts).
Let Bad_2 be the set of all attainable transcripts that contain two entries $(\mathsf{prim}, x, y, \cdot)$ and $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\|\cdots\|V_\ell)$ such that $y \in \{V_0 \overset{(32)}{-} (Z\|K_i\|0\|N), \cdots, V_\ell \overset{(32)}{-} (Z\|K_i\|\ell\|N)\}$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2] \leq \frac{p \cdot \overline{2(n-k)}^{\sigma_e}}{2^k} + \frac{1}{2^{n-k}}.$$

Proof. The main idea we use to bound $\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2]$ is to count the number of entry pairs of the form $((\mathsf{prim}, \cdot, V_j \overset{(32)}{-} (Z\|K\|j\|N), \cdot), (\mathsf{enc}, i, N, \cdot, \cdot, \cdot, V_0\|\cdots\|V_\ell))$ and use a union bound over the events that for such a pair, the independently sampled key K_i is equal to K. However, compared to Bad_1, counting the pairs is more complex. It is done by looking at the primitive entries as bins and all the encryption entries $(\mathsf{enc}, i, N, \cdot, \cdot, \cdot, V_0\|\cdots\|V_\ell)$ as throwing balls of the form $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$ into bins. Using our balls-into-bins corollary, we then obtain for each primitive entry a bound on the maximum number m_1 of associated encryption entries. This yields an upper bound of $p \cdot m_1$ on the number of considered pairs.

Let E_1 be the event that there exist $w \in \{0,1\}^{n-k}$ such that among all encryption queries, there are $m_1 = \left\lceil \frac{2(n-k)}{\max(1, n-k-\log_2(\sigma_e+q_e))} \right\rceil$ or more values of $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$ that are equal to w, where N is the nonce associated to the encryption query of V_j. Then,

$$\begin{aligned}\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2] &= \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2 \wedge \overline{E_1}\right] + \Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2 \wedge E_1] \\ &\leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2 \;\middle|\; \overline{E_1}\right] + \Pr[E_1].\end{aligned}$$

We will now bound the probability of E_1, so that we only have to consider the event $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2$ conditioned by $\overline{E_1}$ afterward. Recall that in the ideal world, the V_j values are independent and uniformly distributed. We can view each encryption query $\textsc{Enc}(i, N, AD, M_1\|\cdots\|M_\ell)$ with answer $C_1\|\cdots\|C_\ell\|T$ together with the sampling of its associated r, W, W' parameters, as throwing $\ell+1$ balls $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$, for $0 \leq j \leq \ell$, uniformly at random into 2^{n-k} bins. Thus if we consider all encryption queries, we throw at most $\sigma_e + q_e$ balls uniformly at random into 2^{n-k} bins. Using Lemma C.1, with $Q = \sigma_e + q_e$ and $D = 2^{-(n-k)}$, the probability that the heaviest bin contains m_1 or more balls is at most $2^{-(n-k)}$. Therefore, the probability of E_1 is bounded by $2^{-(n-k)}$. Hence,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2] \leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2 \;\middle|\; \overline{E_1}\right] + 2^{-(n-k)}. \tag{20}$$

Note that $\overline{E_1}$ is the event that for all $w \in \{0,1\}^{n-k}$, among all encryption queries, there are strictly less than m_1 values of $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$ that are equal to w. To bound the remaining term, we now consider the case where $\mathcal{A}$ is querying the REVEAL oracle and all r, W, W' parameters have been sampled, but no user keys K_i have been sampled yet. In the following, we mean by y the value associated to a primitive query and by j the block index associated to an encryption query. The union $\bigcup_{y,j}$ and sum $\sum_y \sum_j$ are over all primitive and encrypted blocks already queried. A transcript is in Bad_2, if there exist a y from a primitive query and a $V_j \overset{(32)}{-} (Z\|K_i\|j\|N)$ from an encryption query that are equal. To bound the probability of this event, we split it into two, one for the event that the key part of $V_j \overset{(32)}{-} (Z\|K_i\|j\|N)$ and y are equal and a second one for the event that the remaining parts are equal. Let $E_2(j, y)$ be the event that $[V_j]^{K+} \overset{(32)}{-} K_i = [y]^{K+}$ and $E_3(j,y)$ be the event that $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N) = [y]^{K\text{-}}$. Then, using a union bound,

$$\begin{aligned}\Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2 \;\middle|\; \overline{E_1}\right] &\leq \Pr\left[\bigcup_{y,j} E_2(j,y) \wedge E_3(j,y) \;\middle|\; \overline{E_1}\right] \\ &\leq \sum_y \sum_j \Pr\left[E_2(j,y) \;\middle|\; \overline{E_1} \wedge E_3(j,y)\right] \cdot \Pr\left[E_3(j,y) \;\middle|\; \overline{E_1}\right]. \qquad (21)\end{aligned}$$

When querying Reveal, for any pair (j,y), the probability that $[V_j]^{K+} \overset{(32)}{-} K_i = [y]^{K+}$ is the probability that when the key is sampled $K_i = [V_j]^{K+} \overset{(32)}{-} [y]^{K+}$. As the keys are uniformly sampled in Reveal, and independently from any previous queries and parameters, this probability is $\frac{1}{2^k}$. This event is also independent from $\overline{E_1}$ and $E_3(j,y)$. Thus

$$\Pr\left[E_2(j,y) \;\middle|\; \overline{E_1} \wedge E_3(j,y)\right] = \Pr[E_2(j,y)] = \frac{1}{2^k}. \qquad (22)$$

Note that conditioned on event $\overline{E_1}$ there are strictly less than m_1 values $[V_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$ that are equal to one $[y]^{K\text{-}}$, thus for a fix y, there are strictly less than m_1 block indexes j such that $\Pr\left[E_3(j,y) \;\middle|\; \overline{E_1}\right]$ is not zero. Thus

$$\sum_y \sum_j \Pr\left[E_3(j,y) \;\middle|\; \overline{E_1}\right] < \sum_y m_1 \leq p \cdot m_1. \qquad (23)$$

Moreover, $m_1 = \left\lceil \frac{2(n-k)}{\max(1, n-k-\log_2(\sigma_e+q_e))} \right\rceil \leq 2\left\lceil \frac{(n-k)}{\max(1,n-k-\log_2(2\sigma_e))} \right\rceil = 2\overline{(n-k)}^{\sigma_e}$ and in combination with (20), (21), (22), (23), we obtain

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_2] \leq \frac{p \cdot 2\overline{(n-k)}^{\sigma_e}}{2^k} + \frac{1}{2^{n-k}}. \qquad \square$$

C.2.4 Bounding $\mathbf{Bad}_3$ transcripts probability.

Lemma C.4 (Probability of Bad_3 transcripts).
Let Bad_3 be the set of all attainable transcripts that contain two entries $(\mathsf{enc}, i, N, AD, M, C\|T, V)$ and $(\mathsf{enc}, i', N', AD', M', C'\|T', V')$ with $N = N'$, $i \neq i'$ and $K_i = K_{i'}$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_3] \leq \frac{q_e(d-1)}{2^k}.$$

Proof. The main idea we use to bound $\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_3]$ is to count the number of entry pairs of the form $((\mathsf{enc}, i, N, \cdot,\cdot,\cdot,\cdot), (\mathsf{enc}, i', N, \cdot,\cdot,\cdot,\cdot))$ with $i \neq i'$ and use a union bound over the events that for such a pair, the independently sampled key K_i is equal to $K_{i'}$.

If a transcript generated by the adversary $\mathcal{A}$ in the ideal augmented game is in Bad_3, then this adversary $\mathcal{A}$ has made a pair of encryption queries $\textsc{Enc}(i, N, AD, M)$ and $\textsc{Enc}(i', N', AD', M')$ with $N = N'$ and $i \neq i'$, and a query to Reveal that returned the two corresponding K_i and $K_{i'}$ being equal. Hence the probability that a transcript generated by $\mathcal{A}$ in the ideal augmented game is in Bad_3 is bounded by the probability that $\mathcal{A}$ makes the previously described queries.

We are again going to consider the case where $\mathcal{A}$ is just about to query Reveal, but has already made all its other oracle queries. For each of the at most q_e encryption queries $\textsc{Enc}(i, N, AD, M)$ done by $\mathcal{A}$, there are at most $d-1$ other encryption queries $\textsc{Enc}(i', N', AD', M')$ with $N = N'$ and $i \neq i'$ done by $\mathcal{A}$. Thus there are at most $q_e(d-1)$, possible pairs of such encryption queries done by $\mathcal{A}$. When querying Reveal, for each of these pairs, the probability that $K_i = K_{i'}$ is $\frac{1}{2^k}$, as the keys are uniformly sampled, independently from any previous queries. Hence, using a union bound, the probability that for at least one of these pairs $K_i = K_{i'}$, is at most $\frac{q_e(d-1)}{2^k}$. Thus,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_3] \leq \frac{q_e(d-1)}{2^k}. \qquad \square$$

C.2.5 Bounding $\mathbf{Bad_4}$ transcripts probability.

Lemma C.5 (Probability of Bad_4 transcripts).
Let Bad_4 be the set of all attainable transcripts that contain two entries $(\mathsf{enc}, i, N, AD, M, C\|T, V_0\| \cdots \|V_\ell)$ and $(\mathsf{enc}, i', N', AD', M', C'\|T', V'_0\| \cdots \|V'_{\ell'})$ such that $(K_i, j, N) \neq (K_{i'}, j', N')$ and $V_j \overset{(32)}{-} (Z\|K_i\|j\|N) = V'_{j'} \overset{(32)}{-} (Z\|K_{i'}\|j'\|N')$ for $0 \leq j \leq \ell$ and $0 \leq j' \leq \ell'$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_4] \leq \frac{(\sigma_e + q_e)^2}{2^{n+1}}.$$

Proof. The main idea we use to bound $\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_4]$ is to look at the probability of a collision through the randomly generated V blocks (minus the feed forward) in the encryption entries.

If a transcript generated by the adversary $\mathcal{A}$ in the ideal augmented game is in Bad_4, then $\mathcal{A}$ has made a Reveal query that returned values verifying $V_j \overset{(32)}{-} (Z\|K_i\|j\|N) = V'_{j'} \overset{(32)}{-} (Z\|K_{i'}\|j'\|N')$. As established above, the V_j values are all independent and uniformly distributed. Moreover, the keys K_i are also independent and uniformly distributed. Hence, for each encryption query $\text{Enc}(i, N, AD, M_1\| \cdots \|M_\ell)$, the ℓ values $V_0 \overset{(32)}{-} (Z\|K_i\|0\|N), \cdots, V_\ell \overset{(32)}{-} (Z\|K_i\|\ell\|N)$ are also independent and uniformly distributed. If we consider all encryption queries, we obtain at most $\sigma_e + q_e$ independent and uniformly distributed values, and the probability that at least two of them are equal is at most the birthday bound, i.e., $\frac{(\sigma_e+q_e)^2}{2^{n+1}}$. Thus,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_4] \leq \frac{(\sigma_e + q_e)^2}{2^{n+1}}. \qquad \square$$

C.2.6 Bounding $\mathbf{Bad_5}$ transcripts probability.

Lemma C.6 (Probability of Bad_5 transcripts).
Let Bad_5 be the set of all attainable transcripts that contain two entries $(\mathsf{vf}, i, N, AD, C\|T, \mathsf{false})$ and $(\mathsf{prim}, x, y, \cdot)$ such that $x = (Z\|K_i\|0\|N)$ and $\exists r \in \{0,1\}^t, W \in \{0,1\}^{n-2t}$ such that $y \overset{(32)}{+} x = (r\|(T \overset{(t)}{-} H_r(AD, C))\|W)$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5] \leq \frac{q_v \cdot 2\left(\overline{(n-k)}^p + \overline{2t}^p\right)}{2^k} + \frac{1}{2^{n-k-1}} + \frac{1}{2^{2t-1}}.$$

Proof. The probability calculation for this case will follow the same outline as for Bad_2. The main idea we use to bound $\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5]$ is to count the number of entry pairs of the form $((\mathsf{vf}, i, N, AD, C\|T, \mathsf{false}),$ $(\mathsf{prim}, (Z\|K\|0\|N), (r\|(T \overset{(t)}{-} H_r(AD, C))\|W) \overset{(32)}{-} (Z\|K\|0\|N), \cdot))$ and use a union bound over the events that for such a pair, the independently sampled key K_i is equal to K. Again, we look at the verification entries as bins and all the primitive entries $(\mathsf{prim}, x, y, \cdot)$ as throwing balls of the form $[x]^{K\text{-}}$ for inverse primitive entries and of the form $[y \overset{(32)}{+} x]^r\|([y \overset{(32)}{+} x]^s \overset{(t)}{+} H_r(AD, C))$ for forward primitive entries, into bins. Using our balls-into-bins corollary, we then obtain for each verification entry a bound on the maximum number $m_4 + m_5$ of associated primitive entries. This yields an upper bound of $q_v \cdot (m_4 + m_5)$ on the number of considered pairs.

Let E_4 be the event that there exist $w \in \{0,1\}^{n-k}$, such that the number of inverse ideal permutation queries verifying $[x]^{K\text{-}} = w$, is greater or equal to $m_4 = \left\lceil \frac{2(n-k-1)}{\max(1, n-k-1-\log_2(p))} \right\rceil$ and E_5 be the event that there exist $AD^*, C^* \in \{0,1\}^*$ and $T^*, r^* \in \{0,1\}^t$ such that, the number of forward ideal permutation queries verifying $[y \overset{(32)}{+} x]^s \overset{(t)}{+} H_{r^*}(AD^*, C^*) = T^*$ and $r^* = [y \overset{(32)}{+} x]^r$, is greater or equal to $m_5 = \left\lceil \frac{2\cdot(2t-1)}{\max(1, 2t-1-\log_2(p))} \right\rceil$. Then, in a similar way as for Bad_2,

$$\begin{aligned}
\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5] &= \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \wedge \overline{E_4} \wedge \overline{E_5}\right] + \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \wedge \overline{\overline{E_4} \wedge \overline{E_5}}\right] \\
&\leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right] + \Pr[E_4 \vee E_5] \\
&\leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right] + \Pr[E_4] + \Pr[E_5]. \qquad (24)
\end{aligned}$$

We will again bound the probability of E_4 and E_5, leaving us afterward with event $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5$ conditioned by $\overline{E_4} \wedge \overline{E_5}$.

We can view each inverse ideal permutation query $\textsc{Prim}^{-1}(y)$ with answer x as throwing a ball $[x]^{K\text{-}}$ into one of 2^{n-k} possible bins, with probability at most $\frac{2^k}{2^n-p}$. As $p \leq 2^{n-1}$, then $\frac{2^k}{2^n-p} \leq \frac{2^k}{2^{n-1}} = \frac{1}{2^{n-k-1}}$. Thus if we consider all inverse ideal permutation queries, we throw at most p balls with conditional probability at most $2^{-(n-k-1)}$ into 2^{n-k} bins. Using Lemma C.1, with $Q = p$ and $D = 2^{-(n-k-1)}$, the probability that the heaviest bin contains m_4 or more balls is at most $2^{-(n-k-1)}$. Therefore, the probability of E_4 is bounded by $2^{-(n-k-1)}$:

$$\Pr[E_4] \leq 2^{-(n-k-1)}. \tag{25}$$

For any $AD, C \in \{0,1\}^*$, let $\mathsf{Throw}(AD, C)$ be the throwing experiment, where we view each forward ideal permutation $\textsc{Prim}(x)$ with answer y, as throwing a ball $[y \overset{(32)}{+} x]^r \| ([y \overset{(32)}{+} x]^s \overset{(t)}{+} H_r(AD, C))$, where $r = [y \overset{(32)}{+} x]^r$ into one of 2^{2t} possible bins. For any bin $r \| T \in \{0,1\}^{2t}$, each throw has a conditional probability of at most

$$\begin{aligned}
&\Pr_{y \leftarrow \$ \textsc{Prim}(x)}\left[\bigcup_{W \in \{0,1\}^{n-2t}} (y \overset{(32)}{+} x) = r \| (T \overset{(t)}{-} H_r(AD, C)) \| W\right] \\
&\leq \sum_{W \in \{0,1\}^{n-2t}} \Pr_{y \leftarrow \$ \textsc{Prim}(x)}\left[(y \overset{(32)}{+} x) = r \| (T \overset{(t)}{-} H_r(AD, C)) \| W\right] \\
&\leq \frac{2^{n-2t}}{2^n - p}.
\end{aligned}$$

Therefore, each throw has a conditional probability of at most $\frac{2^{n-2t}}{2^n-p}$. As $p \leq 2^{n-1}$, then $\frac{2^{n-2t}}{2^n-p} \leq \frac{2^{n-2t}}{2^{n-1}} = \frac{1}{2^{2t-1}}$. If we consider all forward ideal permutation queries, we throw at most p balls with conditional probability at most $2^{-(2t-1)}$ into 2^{2t} bins. Using Lemma C.1, with $Q = p$ and $D = 2^{-(2t-1)}$, the probability that the heaviest bin of $\mathsf{Throw}(AD, C)$ contains m_5 or more balls is at most $2^{-(2t-1)}$. If E_5 happens, then the bin $r^* \| T^*$ in experiment $\mathsf{Throw}(AD^*, C^*)$ contains m_5 or more balls, therefore, the number of balls in the heaviest bin of $\mathsf{Throw}(AD^*, C^*)$ contains m_5 or more balls. Thus, the probability of E_5 is also bounded by $2^{-(2t-1)}$:

$$\Pr\left[E_5\right] \leq 2^{-(2t-1)}. \tag{26}$$

Combining (24), (25) and (26), we can bound the probability of a transcript being in Bad_5:

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5] \leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right] + \frac{1}{2^{n-k-1}} + \frac{1}{2^{2t-1}}. \tag{27}$$

Note that $\overline{E_4}$ is the event that for all $w \in \{0,1\}^{n-k}$, there are strictly less than m_4 inverse ideal permutation queries such that $[x]^{K\text{-}} = w$ and $\overline{E_5}$ is the event that for any $AD, C \in \{0,1\}^*$ and $T \in \{0,1\}^t$, the number of forward ideal permutation queries verifying $[y \overset{(32)}{+} x]^s \overset{(t)}{+} H_r(AD, C) = T$, where $r = [y \overset{(32)}{+} x]^r$, is strictly less than m_5. To bound the remaining term, we now consider the case where $\mathcal{A}$ is just about to query $\textsc{Reveal}$, but has already made all its other oracle queries. In the following, we mean by $\textsc{Prim}$ and $\textsc{Vf}$, a primitive and verification query already done by $\mathcal{A}$. The union $\bigcup_{\textsc{Vf},\textsc{Prim}}$ and sum $\sum_{\textsc{Vf}} \sum_{\textsc{Prim}}$ are over all primitive and verification queries already done by $\mathcal{A}$.

A transcript is in Bad_5, if there exists a verification and primitive query such that $x = (Z \| K_i \| 0 \| N)$ and $[y \overset{(32)}{+} x]^s = T \overset{(t)}{-} H_r(AD, C)$, where $r = [y \overset{(32)}{+} x]^r$. To bound the probability of this event, we split it into two, one for the event that the key part of x is equal to the key of the verification query and a second one for the rest of the event that doesn't depend on the key. Let $E_6(\textsc{Vf}, \textsc{Prim})$ be the event that when querying the $\textsc{Reveal}$ oracle, the key K_i of the verification query $\textsc{Vf}$ is equal to the value $[x]^{K+}$ associated to the primitive query $\textsc{Prim}$, and $E_7(\textsc{Vf}, \textsc{Prim})$ be the event that $[x]^{K\text{-}} = (Z \| 0 \| N)$ and $[y \overset{(32)}{+} x]^s = T \overset{(t)}{-} H_r(AD, C)$, where $r = [y \overset{(32)}{+} x]^r$, (N, AD, C, T) are the values associated to the verification query $\textsc{Vf}$, and (x, y) are the values associated to the primitive query $\textsc{Prim}$. Then, using a union bound,

$$\begin{aligned}
\Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5 \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right] &\leq \Pr\left[\bigcup_{\textsc{Vf},\textsc{Prim}} E_6(\textsc{Vf}, \textsc{Prim}) \wedge E_7(\textsc{Vf}, \textsc{Prim}) \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right] \\
&\leq \sum_{\textsc{Vf}} \sum_{\textsc{Prim}} \Pr\left[E_6(\textsc{Vf}, \textsc{Prim}) \;\middle|\; \overline{E_4} \wedge \overline{E_5} \wedge E_7(\textsc{Vf}, \textsc{Prim})\right] \cdot \Pr\left[E_7(\textsc{Vf}, \textsc{Prim}) \;\middle|\; \overline{E_4} \wedge \overline{E_5}\right].
\end{aligned} \tag{28}$$

When querying Reveal, the users' keys are uniformly sampled, independently from any previous queries and parameters. Thus, for any pair Vf, Prim, the probability that $K_i = [x]^{K+}$ is $\frac{1}{2^k}$. This event is also independent from $\overline{E_4}$, $\overline{E_5}$ and $E_7(\text{Vf}, \text{Prim})$, which both depend only on parameters already fixed before querying Reveal. Thus

$$\Pr\left[E_6(\text{Vf}, \text{Prim}) \mid \overline{E_4} \wedge \overline{E_5} \wedge E_7(\text{Vf}, \text{Prim})\right] = \Pr[E_6(\text{Vf}, \text{Prim})] = \frac{1}{2^k}. \tag{29}$$

Note that conditioned on event $\overline{E_4} \wedge \overline{E_5}$, for any verification query Vf there are strictly less than $m_4 + m_5$ primitive queries Prim such that $[x]^{K\text{-}} = (Z \| 0 \| N)$ and $[y \overset{(32)}{+} x]^s = T \overset{(t)}{-} H_r(AD, C)$, where $r = [y \overset{(32)}{+} x]^r$, (N, AD, C, T) are the values associated to Vf, and (x, y) are the values associated to Prim. Hence, for a fix verification query Vf, there are strictly less than $m_4 + m_5$ primitive queries Prim such that $\Pr\left[E_7(\text{Vf}, \text{Prim}) \mid \overline{E_4} \wedge \overline{E_5}\right]$ is not zero. Thus

$$\sum_{\text{Vf}} \sum_{\text{Prim}} \Pr\left[E_7(\text{Vf}, \text{Prim}) \mid \overline{E_4} \wedge \overline{E_5}\right] < \sum_{\text{Vf}} (m_4 + m_5) \leq q_v \cdot (m_4 + m_5)\,. \tag{30}$$

Moreover,

$$\begin{aligned} m_4 &= \left\lceil \frac{2(n-k-1)}{\max(1, n-k-1-\log_2(p))} \right\rceil \leq 2 \cdot \left\lceil \frac{(n-k)}{\max(1, n-k-\log_2(2p))} \right\rceil = 2 \cdot \overline{(n-k)}^p \\ m_5 &= \left\lceil \frac{2(2t-1)}{\max(1, 2t-1-\log_2(p))} \right\rceil \leq 2 \cdot \left\lceil \frac{2t}{\max(1, 2t-\log_2(2p))} \right\rceil = 2 \cdot \overline{2t}^p \end{aligned}$$

and in combination with (27), (28), (29) and (30), we obtain

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_5] \leq \frac{q_v \cdot 2\left(\overline{(n-k)}^p + \overline{2t}^p\right)}{2^k} + \frac{1}{2^{n-k-1}} + \frac{1}{2^{2t-1}}. \qquad \square$$

C.2.7 Bounding $\mathbf{Bad_6}$ transcripts probability.

Lemma C.7 (Probability of Bad_6 transcripts).
Let Bad_6 be the set of all transcripts that contain two entries $(\mathsf{vf}, i, N, AD, C \| T, \mathsf{false})$ and $(\mathsf{enc}, i', N, AD', M', C' \| T', V'_0 \| \cdots \| V'_\ell)$ such that $K_{i'} = K_i$ and $\exists r \in \{0,1\}^t, W \in \{0,1\}^{n-2t}$ such that $V'_0 = (r \| (T \overset{(t)}{-} H_r(AD, C)) \| W)$. Then,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_6] \leq \frac{q_v}{2^t} + \frac{q_v \cdot \epsilon(\ell_m)}{2^t} + \frac{q_v \cdot 2 \cdot \overline{2t}^d}{2^k} + \frac{1}{2^{2t}}.$$

Proof. Recall that for an entry $(\mathsf{enc}, i', N, AD', M', C' \| T', V'_0 \| \cdots \| V'_\ell)$, in the ideal world, $V'_0 = (r \| (T' \overset{(t)}{-} H_r(AD', C')) \| W)$, with $r \leftarrow\!\!\$\; \{0,1\}^t$ and $W \leftarrow\!\!\$\; \{0,1\}^{n-2t}$. Two cases are possible. The first one is when the two queries are made to the same user, i.e., $i = i'$, and the second one is when the two queries are made to different users, i.e., $i \neq i'$. The main idea we use to bound the first case is to use the ϵ-almost Δ-universal property of the function H and the fact that encryption returns random authentication tags in the ideal world. For the second case, we count the number of entry pairs of the form $((\mathsf{vf}, i, N, AD, C \| T, \mathsf{false}), (\mathsf{enc}, i', N, \cdot, \cdot, \cdot, V'_0 \| \cdots \| V'_\ell))$ with $V'_0 = (r \| (T \overset{(t)}{-} H_r(AD, C)) \| W)$ and use a union bound over the events that for such a pair, the independently sampled key K_i is equal to $K_{i'}$. To count such pairs, we look at the verification entries as bins and all the encryption entries $(\mathsf{enc}, i', N, \cdot, \cdot, \cdot, V'_0 \| \cdots \| V'_\ell)$ as throwing balls of the form $[V'_0]^r \| ([V'_0]^s \overset{(t)}{+} H_r(AD, C))$ into bins $r \| T$. Using our balls-into-bins corollary, we then obtain for each verification entry a bound on the maximum number m_8 of associated encryption entries. This yields an upper bound of $q_v \cdot m_8$ on the number of considered pairs.

- $\mathsf{Bad}_{6\text{-}1}$**: case** $i = i'$**.**

In the ideal world, Bad_6 for this case could be redefined as the set of all transcripts τ that contain two entries $(\mathsf{vf}, i, N, AD, C \| T, \mathsf{false})$ and $(\mathsf{enc}, i', N, AD', M', C' \| T', V'_0 \| \cdots \| V'_\ell)$ such that $T \overset{(t)}{-} H_r(AD, C) = T' \overset{(t)}{-} H_r(AD', C')$, where $r = [V'_0]^r$. To bound the probability of this case, we are going to use the ϵ-almost Δ-universal property of the function H. However, this property require for (AD, C) and (AD', C') to be distinct. We are therefore first going to bound the probability of transcripts verifying this case when $(AD, C) = (AD', C')$.

When $(AD, C) = (AD', C')$, if a transcript is in this case then $T \overset{(t)}{-} H_r(AD, C) = T' \overset{(t)}{-} H_r(AD', C')$, i.e., $T = T'$. Depending on the order of the queries, we consider two subcases for when $(AD, C) = (AD', C')$. The first one is for when the verification query is done after the encryption query, and the second one is for the inverse order. For the first subcase, when the verification query is done after the encryption query and $(AD, C) = (AD', C')$, we cannot have that $T = T'$, as it would result in a non valid query. Thus the probability of this subcase is zero. For the second subcase, when the encryption query is done after the verification query and $(AD, C) = (AD', C')$, the probability that $T' = T$ is $\frac{1}{2^t}$, as encryption queries return uniform random strings. For each of the at most q_v verification queries $\textsc{Vf}(i, N, AD, C\|T)$, there can be at most one following encryption query $\textsc{Enc}(i, N, AD', M')$ with answer $C'\|T'$ and the same (i, N), and the probability that $T' = T$ is $\frac{1}{2^t}$. Thus the probability of this subcase is bounded by $\frac{q_v}{2^t}$.

When $(AD, C) \neq (AD', C')$, if a transcript generated by the adversary $\mathcal{A}$ in the ideal augmented game is in this case, then $\mathcal{A}$ has made a verification query $\textsc{Vf}(i, N, AD, C\|T)$, an encryption query $\textsc{Enc}(i, N, AD', M')$ with answer $C'\|T'$, and finally a query to $\textsc{Reveal}$ that returned a V_0' associated to the previous encryption query such that $r = [V_0']^r$ is a uniform random string. Hence the probability that a transcript generated by $\mathcal{A}$ in the ideal augmented game is in this case is bounded by the probability that $\mathcal{A}$ makes the previously described queries. We are going to consider the case where $\mathcal{A}$ is just about to query $\textsc{Reveal}$, but has already made all its other oracle queries.

For each of the at most q_v verification queries $\textsc{Vf}(i, N, AD, C\|T)$ done by $\mathcal{A}$, there are at most one other encryption query $\textsc{Enc}(i, N, AD', M')$ with answer $C'\|T'$ and the same (i, N) done by him. Thus there are at most q_v possible pairs of such queries done by $\mathcal{A}$. When querying $\textsc{Reveal}$, for each of these pairs, the value V_0' associated to the encryption query is computed as $(r\|(T' \overset{(t)}{-} H_r(AD', C'))\|W)$ with r being sampled uniformly at random and independently from any previous queries. Hence, as $(AD, C) \neq (AD', C')$ and H is a ϵ-almost Δ-universal hash function, the probability that $T \overset{(t)}{-} H_r(AD, C) = T' \overset{(t)}{-} H_r(AD', C')$ is bounded by $\frac{\epsilon(\max(|AD|_t + |C|_t, |AD'|_t + |C'|_t))}{2^t} \leq \frac{\epsilon(\ell_m)}{2^t}$, for $r = [V_0']^r$ and V_0' being the associated value to the encryption query. Therefore, using a union bound, the probability that for at least one of these pairs $T \overset{(t)}{-} H_r(AD, C) = T' \overset{(t)}{-} H_r(AD', C')$ for $r = [V_0']^r$, is at most $\frac{q_v \cdot \epsilon(\ell_m)}{2^t}$. Note that to use the ϵ-almost Δ-universal property of the function H, the associated data AD, messages M, and ciphertexts C considered, are restricted to byte strings.

Combining the two sub-cases using a union bound, we obtain:

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}1}] \leq \frac{q_v}{2^t} + \frac{q_v \cdot \epsilon(\ell_m)}{2^t}. \tag{31}$$

- $\mathsf{Bad}_{6\text{-}2}$**: case** $i \neq i'$**.**

The probability calculation for this case will follow the same outline as for Bad_2.
Let E_8 be the event that there exist $AD^*, C^* \in \{0,1\}^*$ and $(N^*, T^*, r^*) \in \{0,1\}^\mu \times \{0,1\}^t \times \{0,1\}^t$ such that, the number of encryption queries $\textsc{Enc}(\cdot, N^*, \cdot, \cdot)$ with nonce N^* and associated V_0' verifying $[V_0']^s \overset{(t)}{+} H_{r^*}(AD^*, C^*) = T^*$ and $r^* = [V_0']^r$, is greater or equal to $m_8 = \left\lceil \frac{4t}{\max(1, 2t - \log_2(d))} \right\rceil$. Then,

$$\begin{aligned}\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2}] &= \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2} \wedge \overline{E_8}\right] + \Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2} \wedge E_8] \\ &\leq \Pr\left[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2} \;\middle|\; \overline{E_8}\right] + \Pr[E_8].\end{aligned}$$

We will now bound the probability of E_8, so that we only have to consider the event $\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2}$ conditioned by $\overline{E_8}$ afterward. For any $AD, C \in \{0,1\}^*$ and nonce N, let $\mathsf{Throw}(AD, C, N)$ be the throwing experiment, where we view each encryption query $\textsc{Enc}(\cdot, N, \cdot, \cdot)$ with nonce N and associated V_0', as throwing a ball $[V_0']^r\|([V_0']^s \overset{(t)}{+} H_r(AD, C))$ where $r = [V_0']^r$ into 2^{2t} possible bins. For any bin $r\|T \in \{0,1\}^{2t}$, each throw has a conditional probability of at most

$$\begin{aligned}&\Pr_{V_0' \leftarrow \$\textsc{Enc}(\cdot, N, \cdot, \cdot)}\left[[V_0']^r\|[V_0']^s = r\|T \overset{(t)}{-} H_r(AD, C)\right] \\ &= \Pr\left[[V_0']^s = T \overset{(t)}{-} H_r(AD, C) \;\middle|\; [V_0']^r = r\right] \cdot \Pr[[V_0']^r = r] \\ &= \frac{1}{2^t} \cdot \frac{1}{2^t} = \frac{1}{2^{2t}}.\end{aligned}$$

If we consider all encryption queries with the nonce N, we throw at most d balls uniformly at random into 2^{2t} bins. Using Lemma C.1, with $Q = d$ and $D = 2^{-2t}$, the probability that the heaviest bin of $\mathsf{Throw}(AD, C, N)$ contains m_8 or more balls is at most 2^{-2t}. If E_8 happens, then the bin $r^* \| T^*$ in experiment $\mathsf{Throw}(AD^*, C^*, N^*)$ contains m_8 or more balls, therefore, the number of balls in the heaviest bin of $\mathsf{Throw}(AD^*, C^*, N^*)$ contains m_8 or more balls. Thus, the probability of E_8 is also bounded by 2^{-2t}. Hence,

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2}] \leq \Pr\big[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2} \bigm| \overline{E_8}\big] + 2^{-2t}. \tag{32}$$

Note that $\overline{E_8}$ is the event that for any $AD, C \in \{0,1\}^*$ and $(N, T) \in \{0,1\}^\mu \times \{0,1\}^t$, the number of encryption queries $\textsc{Enc}(\cdot, N, \cdot, \cdot)$ with nonce N and associated V_0' verifying $[V_0']^s \overset{(t)}{+} H_r(AD, C) = T$ where $r = [V_0']^r$, is strictly less than m_8. To bound the remaining term, we now consider the case where $\mathcal{A}$ is querying the $\textsc{Reveal}$ oracle and all parameters r have already been sampled, but no user keys have been sampled yet. In the following, we mean by $\textsc{Enc}$ and $\textsc{Vf}$, an encryption and verification query already done by $\mathcal{A}$. The union $\bigcup_{\textsc{Vf},\textsc{Enc}}$ and sum $\sum_{\textsc{Vf}} \sum_{\textsc{Enc}}$ are over all verification queries already done by $\mathcal{A}$ and all encryption queries with the same nonce as the verification query.

A transcript is in $\mathsf{Bad}_{6\text{-}2}$, if there exists a verification query $\textsc{Vf}(i, N, AD, C\|T)$ and an encryption query $\textsc{Enc}(i', N, AD', M')$ with the same nonce N and an associated V_0' such that $[V_0']^s \overset{(t)}{+} H_r(AD, C) = T$ where $r = [V_0']^r$, and $K_{i'} = K_i$. To bound the probability of this event, we split it into two, one for the event that the keys are equals and a second one for the rest of the event that doesn't depend on the key. Let $E_9(\textsc{Vf}, \textsc{Enc})$ be the event that the key K_i of the verification query $\textsc{Vf}$ is equal to the key $K_{i'}$ of the encryption query $\textsc{Enc}$, and $E_{10}(\textsc{Vf}, \textsc{Enc})$ be the event that $[V_0']^s \overset{(t)}{+} H_r(AD, C) = T$, where $r = [V_0']^r$, (AD, C, T) are the values associated to the verification query $\textsc{Vf}$, and V_0' is the value associated to the encryption query $\textsc{Enc}$. Then, using a union bound,

$$\begin{aligned}\Pr\big[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2} \bigm| \overline{E_8}\big] &\leq \Pr\left[\bigcup_{\textsc{Vf},\textsc{Enc}} E_9(\textsc{Vf}, \textsc{Enc}) \wedge E_{10}(\textsc{Vf}, \textsc{Enc}) \,\middle|\, \overline{E_8}\right] \\ &\leq \sum_{\textsc{Vf}} \sum_{\textsc{Enc}} \Pr\big[E_9(\textsc{Vf}, \textsc{Enc}) \bigm| \overline{E_8} \wedge E_{10}(\textsc{Vf}, \textsc{Enc})\big] \cdot \Pr\big[E_{10}(\textsc{Vf}, \textsc{Enc}) \bigm| \overline{E_8}\big]. \end{aligned} \tag{33}$$

When querying $\textsc{Reveal}$, the users' keys are uniformly sampled, independently from any previous queries and parameters. Thus, for any pair $\textsc{Vf}, \textsc{Enc}$, the probability that $K_i = K_{i'}$ is $\frac{1}{2^k}$. This event is also independent from $\overline{E_8}$ and $E_{10}(\textsc{Vf}, \textsc{Enc})$, which both depend only on parameters already fixed before sampling the users' keys. Thus

$$\Pr\big[E_9(\textsc{Vf}, \textsc{Enc}) \bigm| \overline{E_8} \wedge E_{10}(\textsc{Vf}, \textsc{Enc})\big] = \Pr[E_9(\textsc{Vf}, \textsc{Enc})] = \frac{1}{2^k}. \tag{34}$$

Note that conditioned on event $\overline{E_8}$, for any verification query $\textsc{Vf}$ there are strictly less than m_8 encryption queries $\textsc{Enc}$ with the same nonce N and an associated V_0' such that $[V_0']^s \overset{(t)}{+} H_r(AD, C) = T$, where $r = [V_0']^r$, (AD, C, T) are the values associated to $\textsc{Vf}$, and V_0' is the value associated to $\textsc{Enc}$. Hence, for a fix verification query $\textsc{Vf}$, there are strictly less than m_8 encryption queries $\textsc{Enc}$ such that $\Pr\big[E_{10}(\textsc{Vf}, \textsc{Enc}) \bigm| \overline{E_8}\big]$ is not zero. Thus

$$\sum_{\textsc{Vf}} \sum_{\textsc{Enc}} \Pr\big[E_{10}(\textsc{Vf}, \textsc{Enc}) \bigm| \overline{E_8}\big] < \sum_{\textsc{Vf}} m_8 \leq q_v \cdot m_8. \tag{35}$$

Moreover,

$$m_8 = \left\lceil \frac{4t}{\max(1, 2t - \log_2(d))} \right\rceil \leq 2 \cdot \left\lceil \frac{2t}{\max(1, 2t - \log_2(2d))} \right\rceil = 2 \cdot \overline{2t}^{\,d}$$

and in combination with (32), (33), (34) and (35), we obtain

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_{6\text{-}2}] \leq \frac{q_v \cdot 2 \cdot \overline{2t}^{\,d}}{2^k} + \frac{1}{2^{2t}}. \tag{36}$$

Combining the two cases (31) and (36) using a union bound, we obtain the following result:

$$\Pr[\mathcal{T}_{\text{ideal}} \in \mathsf{Bad}_6] \leq \frac{q_v}{2^t} + \frac{q_v \cdot \epsilon(\ell_m)}{2^t} + \frac{q_v \cdot 2 \cdot \overline{2t}^{\,d}}{2^k} + \frac{1}{2^{2t}}. \qquad \square$$

D Attacks and Proofs of Lower Bounds

In this section, we give the details of the attacks against ChaCha20-Poly1305 briefly described in Section 7.1. We also lower bound their advantages as stated through Propositions 7.1–7.6.

D.1 Proof of Proposition 7.1 (Forgery Attack)

We define an adversary $\mathcal{A}$ that makes one encryption query and q_v verification queries, attempting each time a forgery. If a verification query returns true, then it outputs 1. As a forgery is impossible in the ideal world, then

$$\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] = 0.$$

Thus

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) = \Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right].$$

We now describe in detail this adversary $\mathcal{A}$ against ChaCha20-Poly1305 and how it constructs its queries. Let $\mathsf{p} = 2^{130} - 5$ and note that $\mathsf{p}^{\frac{3}{4}} < 2^t < \mathsf{p}$. In the following, we will use the integer representation for t-bit strings. The adversary $\mathcal{A}$ makes as its first query an encryption query $\textsc{Enc}(i, N, AD, M)$ with nonce N and inputs of exactly ℓ_m blocks, where $M = \varepsilon$ is an empty string and $AD = AD_1\| \cdots \|AD_{\ell_m-1}$, where $|AD_j| = t$ and $AD_j = 2^{t-1}$. It receives a tag T as answer. Then, the adversary $\mathcal{A}$ constructs q_v forgery polynomials (described below). From each polynomial, it crafts a string AD' and an associated verification query $\textsc{Vf}(i, N, AD', T)$ that returns true with probability at least $\frac{3(\ell_m-5)}{2^{t-20}}$.

We denote by r_N and s_N the hash key (after clamping) and the blinding value, associated to the nonce N queried during encryption, i.e., $r_N\|s_N = \text{CC\_block}(K_i, N, 0)[1{:}2t]$. Let $\mathcal{R}$ denote the set of all possible hash keys for Poly1305_Mac (after clamping). Let $\mathcal{R}_1, \ldots, \mathcal{R}_{q_v} \subset \mathcal{R}$ be q_v disjoint sets, each of $3\lfloor\frac{\ell_m-1}{4}\rfloor$ hash keys. For each $\mathcal{R}_q$, we will construct a forgery polynomial and the associated pair AD', T, that will be a valid forgery with nonce N, if r_N is in $\mathcal{R}_q$. Let

$$R(x, \mathcal{R}_q) = x^2 \cdot g(x) \cdot \prod_{r\in\mathcal{R}_q} (x - r) \mod \mathsf{p}$$

be a polynomial with the set $\mathcal{R}_q$ as roots and where $g(x)$ is a non zero polynomial of degree at most $\lfloor\frac{\ell_m-1}{4}\rfloor - 1$ defined such that when we rewrite $R(x, \mathcal{R}_q)$ as $\sum_{j=2}^{\ell_m} a_{\ell_m-j+1} \cdot x^j \mod \mathsf{p}$ (or equivalently as $\sum_{j=2}^{\ell_m} a'_{j-2} \cdot x^j \mod \mathsf{p}$ later in the proof), the coefficients a_j satisfies $(2^{t-1} + a_j \mod \mathsf{p}) < 2^t$ for all $j < \ell_m$. Note that we consider the coefficients a_j (and a'_j) as integers that can take negative values. We will show shortly how to construct such a polynomial $g(x)$ and will assume for now the existence of a $g(x)$ ensuring that $(2^{t-1} + a_j \mod \mathsf{p}) < 2^t$ for all $j < \ell_m$. Recall that

$$H_x(AD, \varepsilon) = (c_1x^{\ell_m} + \cdots + c_{\ell_m-1}x^2 + c_{\ell_m}x^1 \mod \mathsf{p}) \mod 2^t$$

where $c_j = 2^t + AD_j = 2^t + 2^{t-1}$ (i.e., the integer representation of $AD_j\|1$) for $j < \ell_m$, and c_{ℓ_m} is the integer representation of $\text{len}(AD)\|\text{len}(\varepsilon)\|1$. Note that $H_{r_N}(AD, \varepsilon) \overset{(t)}{+} s_N = T$ and if r_N is in $\mathcal{R}_q$, then $R(r_N, \mathcal{R}_q) = 0$. We can construct the string AD' and its associated polynomial as $AD' = AD'_1\| \cdots \|AD'_{\ell_m-1}$, where $|AD'_j| = t$, $AD'_j = (2^{t-1} + a_j \mod \mathsf{p})$, and

$$H_x(AD', \varepsilon) = (c'_1x^{\ell_m} + \cdots + c'_{\ell_m-1}x^2 + c'_{\ell_m}x^1 \mod \mathsf{p}) \mod 2^t,$$

where $c'_j = 2^t + AD'_j = 2^t + (2^{t-1} + a_j \mod \mathsf{p})$ (i.e., the integer representation of $AD'_j\|1$) for $j < \ell_m$, and c'_{ℓ_m} is the integer representation of $\text{len}(AD')\|\text{len}(\varepsilon)\|1$. Thanks to our assumption, $(2^{t-1}+a_j \mod \mathsf{p}) < 2^t$ and therefore, the string AD' and its associated polynomial is well defined. Moreover, as we constructed our polynomial so that $c'_j = (c_j + a_j \mod \mathsf{p})$ for $j < \ell_m$ and $c'_{\ell_m} = c_{\ell_m}$, then

$$H_x(AD', \varepsilon) = (c_1x^{\ell_m} + \cdots + c_{\ell_m}x^1 + R(x, \mathcal{R}_q) \mod \mathsf{p}) \mod 2^t.$$

Furthermore, since when r_N is in $\mathcal{R}_q$, we obtain $R(r_N, \mathcal{R}_q) = 0$, it follows that, if r_N is in $\mathcal{R}_q$, then:

$$H_{r_N}(AD', \varepsilon) \overset{(t)}{+} s_N = H_{r_N}(AD, \varepsilon) \overset{(t)}{+} s_N = T.$$

Hence, $\textsc{Vf}(i, N, AD', T)$ is a valid forgery query if r_N is in $\mathcal{R}_q$. As r_N is sampled uniformly and independently from the choices of $\mathcal{R}_q$, the probability that $r_N \in \mathcal{R}_q$ is $3\lfloor\frac{\ell_m-1}{4}\rfloor \cdot \frac{1}{2^{t-22}} \geq \frac{3(\ell_m-5)}{2^{t-20}}$. Thus, the probability that $\mathrm{VF}(i, N, AD', T)$ is a valid forgery query is at least $\frac{3(\ell_m-5)}{2^{t-20}}$. The adversary $\mathcal{A}$ makes q_v such verification queries, one for each $\mathcal{R}_q$. As the $\mathcal{R}_q$ are disjoint, the probability that at least one of these q_v verification queries is a valid forgery is at least $\frac{3q_v(\ell_m-5)}{2^{t-20}}$. Hence the bound in the proposition follows.

We now show how to construct a polynomial $g(x)$ so that the coefficients a'_j of the polynomial

$$R(x, \mathcal{R}_q) = x^2 \cdot g(x) \cdot \prod_{r\in\mathcal{R}_q} (x-r) \mod \mathsf{p} = \sum_{j=2}^{\ell_m} a'_{j-2} \cdot x^j \mod \mathsf{p}$$

satisfies $(2^{t-1} + a'_j \mod \mathsf{p}) < 2^t$ for all $j < \ell_m - 1$. Let $\psi = \left\lfloor \frac{\ell_m-1}{4} \right\rfloor$, we can rewrite $\prod_{r\in\mathcal{R}_q}(x-r)$ as $x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j$ and $g(x)$ as $\sum_{\mathsf{i}=0}^{\psi-1} g_{\mathsf{i}} \cdot x^{\mathsf{i}}$. Then

$$\begin{aligned} g(x) \cdot \prod_{r\in\mathcal{R}_q} (x-r) \mod \mathsf{p} = g_0 \cdot \left(x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j \right) + g_1 x^1 \cdot \left(x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j \right) + \cdots \\ + g_{\psi-1} x^{\psi-1} \cdot \left(x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j \right) \mod \mathsf{p}. \end{aligned}$$

We define how to construct the coefficients g_{i} (and therefore $g(x)$), by using a lattice generated by the ψ polynomials

$$x^{\mathsf{i}} \cdot \left(x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j \right)$$

where $\mathsf{i} < \psi$. Let $\mathcal{L}$ be the lattice generated by the rows of the following matrix

$$\begin{bmatrix} 1 & b_{3\psi-1} & b_{3\psi-2} & \cdots & b_0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & b_{3\psi-1} & b_{3\psi-2} & \cdots & b_0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & b_{3\psi-1} & b_{3\psi-2} & \cdots & b_0 \\ & & & & \mathsf{p} \cdot I_{4\psi-1} & & & & \end{bmatrix}$$

where, for $\mathsf{i} \le \psi$, the i-th row represents the polynomial $x^{\psi-\mathsf{i}} \cdot \left(x^{3\psi} + \sum_{j=0}^{3\psi-1} b_j \cdot x^j \right)$ (and the columns represents the coefficients of this polynomial) and where the last $4\psi - 1$ rows represents the reductions modulo p of the coefficients, i.e., $\mathsf{p} \cdot I_{4\psi-1}$ where $I_{4\psi-1}$ is the identity matrix of size $4\psi - 1$. We denote by $(a'_{4\psi-2}, \cdots, a'_0)$ the shortest vector in this lattice. It arises as a linear combination of the rows of the previous matrix and, as such, defines the coefficients g_{i} (that we don't need to know). More importantly, $R(x, \mathcal{R}_q) = \sum_{j=2}^{\ell_m} a'_{j-2} \cdot x^j \mod \mathsf{p}$, where we set $a'_j = 0$ for $j > 4\psi - 2$. We are left to show that $(2^{t-1} + a'_j \mod \mathsf{p}) < 2^t$ for all $j \le 4\psi - 2$. We can compute a basis (and determinant) of the lattice $\mathcal{L}$, by looking at the row echelon form of the previous matrix:

$$\begin{bmatrix} I_\psi & A \\ 0 & \mathsf{p} \cdot I_{3\psi} \end{bmatrix}$$

where A is a $\psi \times 3\psi$ matrix. From this matrix, we can observe that the rank of $\mathcal{L}$ is 4ψ, and its determinant is $\mathsf{p}^{3\psi}$. Minkowski's theorem with the infinity norm yields that, for the shortest vector in $\mathcal{L}$, $|a'_j| \le \mathsf{p}^{\frac{3\psi}{4\psi}}$ for all $j \le 4\psi - 2$. Thus $|a'_j| \le \mathsf{p}^{\frac{3}{4}} < 2^{t-1}$ for all $j \le 4\psi - 2$. Hence $-2^{t-1} < a'_j < 2^{t-1}$ and $(2^{t-1} + a'_j \mod \mathsf{p}) < 2^t$ for all $j \le 4\psi - 2$.

Therefore, as the adversary $\mathcal{A}$ is unbounded, it can compute the smallest vector of $\mathcal{L}$ and construct $R(x, \mathcal{R}_q) = \sum_{j=2}^{\ell_m} a'_{j-2} \cdot x^j \mod \mathsf{p}$ such that $(2^{t-1} + a'_j \mod \mathsf{p}) < 2^t$ for all $j < \ell_m - 1$. It then proceeds to the above-described attack.

Note that [AM18] provides an algorithm for solving SVP with the infinity norm used in the proof. Also note that the complexity of this attack may be exponential when using deterministic SVP algorithms (or slightly better when using SVP approximation algorithms), making the attack impractical, especially for large ℓ_m, but still valid in the model we use here, where the adversary is computationally unbounded. For simplicity, we queried only one user for a specific encryption query in our attack. The approach can be extended to an attack querying multiple users, which then requires at least one encryption query per user. Moreover, the encryption query we use, in fact, only authenticates the associated data $AD = 0^{t\cdot(\ell_m-1)}$; the message M queried is empty. The attack should be extendable to arbitrary values AD and M of maximum length ℓ_m by considering a closest vector (CVP) algorithm rather than a shortest vector (SVP) one. □

D.2 Key-recovery Attacks

D.2.1 Proof of Proposition 7.2. To prove the claim we describe the following d-repeating adversary $\mathcal{A}$. It makes d encryption queries $\textsc{Enc}(i, N, AD, M)$ with the same inputs across d different users (indexed by i) such that $n \geq |M| \geq k+2$ and receives in return d ciphertexts $C_1\|T_1, \ldots, C_d\|T_d$. The adversary then makes p permutation queries $\textsc{Prim}(Z\|K'_{i'}\|1\|N)$ using distinct and independently-chosen keys $K'_{i'}$, which it can then use to construct p distinct ciphertext guesses of the form

$$C'_{i'} = M \oplus (\textsc{Prim}(Z\|K'_{i'}\|1\|N) \overset{(32)}{+} (Z\|K'_{i'}\|1\|N)).$$

It then checks whether there exist i, i' such that $C_i = C'_{i'}$, and if so returns 1, otherwise it returns 0.

Now, in the real world, if there exist i, i' such that $K_i = K'_{i'}$ (i.e., the adversary correctly guessed a user key) it follows that $C_i = C'_{i'}$. Thus

$$\begin{aligned}\Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] &= \Pr\left[\bigcup_{i,i'}(C_i = C'_{i'})\right] \\ &\geq \Pr\left[\bigcup_{i,i'}(K_i = K'_{i'})\right] = 1 - \Pr\left[\bigcap_{i,i'}(K_i \neq K'_{i'})\right].\end{aligned}$$

Noting that the user keys K_i are sampled uniformly at random and using the fact that $1-x \leq \mathrm{e}^{-x} \leq 1-\frac{x}{2}$ for all $x \in [0,1]$, we obtain

$$\begin{aligned}&= 1 - \left(1 - \frac{p}{2^k}\right)^d \\ &\geq 1 - \mathrm{e}^{-\frac{pd}{2^k}} \\ &\geq \frac{pd}{2^{k+1}}.\end{aligned}$$

On the other hand, in the ideal world, the ciphertexts C_i are sampled uniformly at random. It then follows that

$$\begin{aligned}\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] &= \Pr\left[\bigcup_{i,i'}(C_i = C'_{i'})\right] \\ &\leq \sum_{i,i'} \Pr[(C_i = C'_{i'})] \\ &= \frac{pd}{2^{|M|}} \\ &\leq \frac{pd}{2^{k+2}}.\end{aligned}$$

Where the first inequality follows from the union bound and the second from the restriction on the message size. Combining the above it follows that

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) \geq \frac{pd}{2^{k+1}} - \frac{pd}{2^{k+2}} = \frac{pd}{2^{k+2}}.$$ □

D.2.2 Proof of Proposition 7.3. Consider the following adversary $\mathcal{A}$ making one permutation query $\textsc{Prim}(Z\|K\|0\|N)$, using an arbitrary nonce N and key K, to construct from it guess values for the hash key and the blinding value as follows

$$r\|s = (\textsc{Prim}(Z\|K\|0\|N) \overset{(32)}{+} (Z\|K\|0\|N))[1{:}2t].$$

It then constructs a guess tag $T = H_r(AD, \varepsilon) \overset{(t)}{+} s$ for arbitrary associated data AD and an empty ciphertext. Afterwards, it makes q_v verification queries $\textsc{Vf}(i, N, AD, T)$ with the same inputs across q_v different users (indexed by i) and returns 1 if any of these queries returns true.

In the ideal world, a verification query always returns false, and thus $\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] = 0$. In the real world, however, if the key K_i sampled by user i is equal to K, the verification query $\textsc{Vf}(i, N, AD, T)$ will be a valid forgery. Noting again that the user keys are sampled independently and uniformly at random and applying a similar analysis to that used in the previous attack, we obtain:

$$\begin{aligned}
\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) &= \Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] - 0 \\
&\geq \Pr\left[\bigcup_i (K_i = K)\right] \\
&= 1 - \left(1 - \frac{1}{2^k}\right)^{q_v} \\
&\geq \frac{q_v}{2^{k+1}}.
\end{aligned}$$

□

D.3 Proof of Proposition 7.4 (Key-collision Attacks)

We describe a d-repeating adversary $\mathcal{A}$ that makes q_e encryption queries across q_e different users, such that q_e is an integer multiple of d. It uses $\frac{q_e}{d}$ different nonces throughout its encryption queries and reuses each of these nonces precisely d times across different users. Thus each query can be uniquely identified by the user identity i and the set of queries can be partitioned according to the nonce value used in the query. In each encryption query $\textsc{Enc}(i, N, AD, M)$ it uses the same message M and associated data AD, where $|M| \geq k + 2$, and receives in turn a ciphertext C_i of length greater or equal to $|M|$. Then, if there exist two distinct queries i, i' using the same nonce such that $C_i = C_{i'}$, i.e., two colliding ciphertexts within the same partition, the adversary outputs 1 and outputs 0 otherwise.

Let N_i and K_i denote the nonce and key used in query i. In the real world, we expect the ciphertexts to collide whenever the keys collide if the same nonce and message are used. Thus, quantifying over all queries within the same partition, it follows that

$$\begin{aligned}
\Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] &= \Pr\left[\bigcup_{i,i' \,:\, N_i = N_{i'}} (C_i = C_{i'})\right] \\
&\geq \Pr\left[\bigcup_{i,i' \,:\, N_i = N_{i'}} (K_i = K_{i'})\right] \\
&= 1 - \Pr\left[\bigcap_{i,i' \,:\, N_i = N_{i'}} (K_i \neq K_{i'})\right].
\end{aligned}$$

To evaluate this, we can consider the probability of non-colliding keys for each partition separately and then multiply the probabilities together as they are independent of each other, thereby yielding

$$= 1 - \prod_N \prod_{i=1}^{d-1} \left(1 - \frac{i}{2^k}\right).$$

Then using the inequality $1 - x \le \mathrm{e}^{-x}$ for all x and simplifying, we obtain

$$\begin{aligned}
&\ge 1 - \prod_{N} \prod_{i=1}^{d-1} \mathrm{e}^{-\frac{i}{2^k}} \\
&= 1 - \prod_{N} \mathrm{e}^{-\sum_{i=1}^{d-1} \frac{i}{2^k}} \\
&= 1 - \prod_{N} \mathrm{e}^{-\frac{d(d-1)}{2^{k+1}}} \\
&= 1 - \left(\mathrm{e}^{-\frac{d(d-1)}{2^{k+1}}}\right)^{\frac{q_e}{d}} \\
&= 1 - \mathrm{e}^{-\frac{q_e(d-1)}{2^{k+1}}}. \qquad (37)
\end{aligned}$$

Finally, using that $\mathrm{e}^{-x} \le 1 - \frac{x}{2}$ for all $x \in [0, 1]$ we obtain

$$\ge \frac{q_e(d-1)}{2^{k+2}}.$$

Now, in the ideal world the ciphertexts are sampled at random. Thus for each nonce, the probability of having a collision in the ciphertexts is bounded above by the following birthday bound

$$\frac{d(d-1)}{2^{|C_i|+1}} \le \frac{d(d-1)}{2^{|M|+1}} \le \frac{d(d-1)}{2^{k+3}}.$$

Applying the union bound to sum up over all the queried nonces we obtain

$$\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] \le \frac{q_e}{d} \cdot \frac{d(d-1)}{2^{k+3}} = \frac{q_e(d-1)}{2^{k+3}}. \qquad (38)$$

Combining the above we get

$$\begin{aligned}
\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) &\ge \frac{q_e(d-1)}{2^{k+2}} - \frac{q_e(d-1)}{2^{k+3}} \\
&= \frac{q_e(d-1)}{2^{k+3}}.
\end{aligned}$$

□

D.4 Block-collision Attacks

D.4.1 Proof of Proposition 7.5. We describe a distinguishing adversary $\mathcal{A}$ against ChaCha20-Poly1305 that works by detecting collisions in the outputs of the ChaCha20 permutation. It will make encryption queries that result in distinct inputs to the ChaCha20 permutation and look for collisions across the outputs of the permutation corresponding to a specific user—an event that can only happen in the ideal world.

Specifically, the adversary encrypts B blocks of plaintext per user, totalling σ_e blocks across all users, such that σ_e is a multiple of B. It will query each of the $\frac{\sigma_e}{B}$ different users exactly once. In each encryption query $\text{Enc}(i, N_i, \varepsilon, M)$, it uses a fixed B-block message $M = M_1 \| M_2 \| \cdots \| M_B$ together with a distinct nonce N_i and obtains in return a ciphertext $C^i = C^i_1 \| C^i_2 \| \cdots \| C^i_B \| T^i$. For each ciphertext C^i it receives, it looks for two distinct block counters j, j' such that

$$C^i_j \oplus M_j \overset{(32)}{-} (Z \| 0^k \| j \| N_i) = C^i_{j'} \oplus M_{j'} \overset{(32)}{-} (Z \| 0^k \| j' \| N_i).$$

If such a collision is found for any i, it outputs 1; otherwise it outputs 0.

In the ideal world, the blocks C^i_j are sampled uniformly at random. Thus, the probability of a collision across any user is given by:

$$\begin{aligned}
\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] &= \Pr\left[\bigcup_{i} \bigcup_{j \ne j'} \left(C^i_j \oplus M_j \overset{(32)}{-} (Z \| 0^k \| j \| N_i) = C^i_{j'} \oplus M_{j'} \overset{(32)}{-} (Z \| 0^k \| j' \| N_i)\right)\right] \\
&= 1 - \Pr\left[\bigcap_{i} \bigcap_{j \ne j'} \left(C^i_j \oplus M_j \overset{(32)}{-} (Z \| 0^k \| j \| N_i) \ne C^i_{j'} \oplus M_{j'} \overset{(32)}{-} (Z \| 0^k \| j' \| N_i)\right)\right] \\
&= 1 - \prod_{i=1}^{\frac{\sigma_e}{B}} \prod_{j=1}^{B-1} \left(1 - \frac{j}{2^n}\right).
\end{aligned}$$

Then using the inequality $1 - x \leq \mathrm{e}^{-x}$ for all x and simplifying, we obtain

$$\begin{aligned}
&\geq 1 - \prod_{i=1}^{\frac{\sigma_e}{B}} \prod_{j=1}^{B-1} \mathrm{e}^{-\frac{j}{2^n}} \\
&= 1 - \prod_{i=1}^{\frac{\sigma_e}{B}} \mathrm{e}^{-\sum_{j=1}^{B-1} \frac{j}{2^n}} \\
&= 1 - \prod_{i=1}^{\frac{\sigma_e}{B}} \mathrm{e}^{-\frac{B(B-1)}{2^{n+1}}} \\
&= 1 - \left(\mathrm{e}^{-\frac{B(B-1)}{2^{n+1}}}\right)^{\frac{\sigma_e}{B}} \\
&= 1 - \mathrm{e}^{-\frac{\sigma_e(B-1)}{2^{n+1}}}.
\end{aligned}$$

Finally, using that $\mathrm{e}^{-x} \leq 1 - \frac{x}{2}$ for all $x \in [0,1]$ we obtain

$$\geq \frac{\sigma_e(B-1)}{2^{n+2}}.$$

On the other hand, in the real world, ciphertexts are generated by xoring the plaintexts with the ChaCha20 block function, and thus

$$C^i_j \oplus M_j = \text{CC\_block}(K_i, N_i, j) = \pi(Z\|K_i\|j\|N_i) \overset{(32)}{+} (Z\|K_i\|j\|N_i).$$

Then for any fixed i and any pair of distinct integers (j, j') the following condition:

$$C^i_j \oplus M_j \overset{(32)}{-} (Z\|0^k\|j\|N_i) = C^i_{j'} \oplus M_{j'} \overset{(32)}{-} (Z\|0^k\|j'\|N_i)$$

reduces to

$$\pi(Z\|K_i\|j\|N_i) = \pi(Z\|K_i\|j'\|N_i).$$

Since π is a permutation and $j \neq j'$ it follows that

$$\Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] = 0.$$

Combining the above, we obtain the following bound

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) \geq \frac{\sigma_e(B-1)}{2^{n+2}}. \qquad \square$$

D.4.2 Proof of Proposition 7.6. The attack from Proposition 7.5 only detects collisions within queries originating from the same user. In principle it is possible to detect collisions across queries from distinct users if the difference between the two user keys is known. However, in our security model this information is not available to the adversary and such an assumption cannot be justified. Nevertheless, if we ignore the part of the ChaCha20 block function's output where the key is xored, we essentially have access to a truncated output of the ChaCha20 permutation given by:

$$[\pi(Z\|K\|j\|N)]^{K\text{-}} = [C_j \oplus M_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N).$$

This way, distinguishing the real world from the ideal world can be reduced to distinguishing a truncated random permutation from a truncated random function with σ_e queries. This problem was studied in [GG21] where a lower bound for this distinguishing advantage was established. The intuition behind the attack is that a collision in the output of a truncated random permutation is less likely than a collision in the output of a random function. In contrast to [GG21], in our setting, the adversary does not have full control over the queried inputs—namely the user keys. Nevertheless, the attack and its analysis can be easily adapted to our setting, as shown below.

The adversary $\mathcal{A}$ queries the encryption oracle a total number of σ_e blocks across the different users, where a distinct nonce N is used for every encryption query (across all users). For each of the σ_e blocks queried, we denote by W_{J} the $(n-k)$-bit string $[C_j \oplus M_j]^{K\text{-}} \overset{(32)}{-} (Z\|j\|N)$, where C_j is the encryption of the j-th message block M_j with nonce N and J is an index uniquely identifying every distinct pair (N, j). Accordingly, because every encryption query employs a distinct nonce each value W_{J} is produced using a unique pair (N, j). The adversary outputs 1, if $W_{\mathsf{J}} \neq W_{\mathsf{J}'}$ for all $\mathsf{J} \neq \mathsf{J}'$ and it outputs 0 otherwise.

Now, in the ideal world, the strings W_{J} are independent and uniformly distributed, and thus the probability of the adversary outputs 1 is given by

$$\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] = \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 - \frac{\mathsf{J}}{2^{n-k}}\right).$$

By the Weierstrass product inequality we can bound this expression from below to obtain

$$\prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 - \frac{\mathsf{J}}{2^{n-k}}\right) \geq 1 - \sum_{\mathsf{J}=0}^{\sigma_e-1} \frac{\mathsf{J}}{2^{n-k}} = 1 - \frac{1}{2} \cdot \frac{\sigma_e(\sigma_e-1)}{2^{n-k}},$$

and then using that $\sigma_e \leq 2^{\frac{n-k}{2}}$ the above reduces to

$$\Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] \geq \frac{1}{2}. \tag{39}$$

On the other hand, in the real world the W_{J}'s are computed by evaluating a permutation over σ_e distinct inputs and then truncating k bits from each. Thus, the probability of all these values being distinct is given by

$$\begin{aligned}
\Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] &= \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 - \frac{\mathsf{J}(2^k-1)}{2^n - \mathsf{J}}\right) \\
&= \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(\frac{2^n - \mathsf{J}\cdot 2^k}{2^n - \mathsf{J}}\right) \\
&= \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 - \frac{\mathsf{J}}{2^{n-k}}\right) \cdot \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(\frac{2^n}{2^n - \mathsf{J}}\right) \\
&= \Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] \cdot \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(\frac{2^n}{2^n - \mathsf{J}}\right).
\end{aligned}$$

Now, the second term on the right-hand side can be bounded from below using the inequality $(2^n - \mathsf{J}) \cdot (2^n + \mathsf{J}) \leq 2^{2n}$, leading to the following inequality

$$\prod_{\mathsf{J}=0}^{\sigma_e-1} \left(\frac{2^n}{2^n - \mathsf{J}}\right) \geq \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(\frac{2^n + \mathsf{J}}{2^n}\right) = \prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 + \frac{\mathsf{J}}{2^n}\right).$$

Further applying the Weierstrass product inequality we obtain that

$$\prod_{\mathsf{J}=0}^{\sigma_e-1} \left(1 + \frac{\mathsf{J}}{2^n}\right) \geq 1 + \sum_{\mathsf{J}=0}^{\sigma_e-1} \frac{\mathsf{J}}{2^n} = 1 + \frac{\sigma_e(\sigma_e-1)}{2^{n+1}},$$

and thus,

$$\Pr\left[\mathcal{A}^{G^{\text{Real-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] \geq \Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right] \cdot \left(1 + \frac{\sigma_e(\sigma_e-1)}{2^{n+1}}\right).$$

It then follows that

$$\mathsf{Adv}^{\text{muAE}}_{\text{ChaCha20-Poly1305}[\pi]}(\mathcal{A}) \geq \frac{\sigma_e(\sigma_e-1)}{2^{n+1}} \cdot \Pr\left[\mathcal{A}^{G^{\text{Ideal-muAE}}_{\text{ChaCha20-Poly1305}[\pi]}} \Rightarrow 1\right],$$

where the final bound is obtained by combining (39) and this last inequality. □