A Robust Variant of ChaCha20-Poly1305

Tim Beyne, Yu Long Chen, and Michiel Verbauwhede

COSIC, KU Leuven, Belgium
name.lastname@esat.kuleuven.be

Abstract. The ChaCha20-Poly1305 AEAD scheme is widely used as an alternative for AES-GCM on platforms without AES hardware instructions. Although recent analysis by Degabriele et al. shows that ChaCha20-Poly1305 provides adequate security in the conventional multi-user model, the construction is totally broken when a single nonce is repeated – a real-world scenario that can occur due to faulty implementations or the desire to use random nonces.

We present a new nonce-misuse resistant and key-committing authenticated encryption scheme, called ChaCha20-Poly1305-PSIV, that is based on carefully combining the ChaCha20-Poly1305 building blocks into the NSIV paradigm proposed by Peyrin and Seurin (CRYPTO 2016) without performance loss. We analyze the security of the underlying mode PSIV in the multi-user faulty-nonce model assuming that the underlying permutation is ideal, and prove its key-committing security in the cmt-1 model. Rust and C implementations are provided, and benchmarks confirm that performance is comparable to the ChaCha20-Poly1305 implementation in libsodium.

In terms of security and efficiency (without hardware support), our proposal compares favorably to AES-GCM-SIV. Since we reuse the ChaCha20-Poly1305 building blocks, we expect ChaCha20-Poly1305-PSIV to benefit from existing analysis and to be easy to deploy in practice.

Keywords: Authenticated encryption · Nonce-misuse resistance · Key commitment · ChaCha20-Poly1305

1 Introduction

Many applications rely on symmetric-key cryptography to provide a combination of confidentiality and integrity. This has led to the formalization of a single security notion that combines both of these properties: *authenticated encryption* [3, 30].

Due to its excellent performance, AES-GCM [25] is one of the most widely used authenticated encryption modes. For example, on platforms with hardware support for AES, it is often the default choice for TLS. However, native AES instructions are not always available. This led Langley to propose ChaCha20-Poly1305 [23, 32] as an alternative to AES-GCM with often better software performance on platforms without AES instructions. It combines the Poly1305 message authentication code [4] and the ChaCha20 encryption scheme [5], both of

which are due to Bernstein. Its competitive performance derives from extensive use of instruction-level parallelism on architectures with SIMD instructions such as AVX. As of TLS 1.3, implementations are strongly recommended to support ChaCha20-Poly1305 [29, §9.1].

Both AES-GCM and ChaCha20-Poly1305 are *nonce-based authenticated encryption schemes*, meaning that the encryption and decryption operations take an additional input (the nonce) that should be unique in normal operation. Even a single nonce repetition leads to universal forgery attacks and, if one of the two messages encrypted with the same nonce is known, plaintext recovery. However, ensuring the uniqueness of nonces implies that the implementation must be stateful. This is not always feasible, for example when several devices share the same key but are unable to synchronize reliably. Faulty implementations can also lead to nonce repetitions. For example, in an Internet-wide scan, Böck et al. [6] found several HTTPS servers that always use the same nonce and many others that use random 64-bit nonces. Due to the birthday problem, the latter approach severely limits the number of messages that can be safely encrypted.

To make authenticated encryption more robust to nonce repetitions, Rogaway and Shrimpton [31] introduced the notion of nonce-misuse security. Several constructions achieving this security goal were subsequently proposed. The CAESAR portfolio for "defense in depth" includes two modes with some form of nonce-misuse resistance: Deoxys-II and COLM. They are both based on variants of the AES, making them suitable alternatives for AES-GCM. During but independently of the CAESAR competition, Gueron and Lindell proposed the nonce-misuse resistant mode AES-GCM-SIV. A revised version of their original proposal has been standardized for use in IETF protocols [17].

In addition to nonce reuse, there is another way in which incorrect usage of authenticated encryption has led to vulnerabilities. Most authenticated encryption schemes are not key-committing: a given ciphertext can decrypt to a valid plaintext for two different keys, and finding such a key pair is often computationally feasible. However, users sometimes implicitly rely on key-commitment. For example, at Crypto 2017, Grubbs et al. studied the importance of key-commitment for the message franking system used by Facebook Messenger for abuse reporting. At Crypto 2018 Dodis et al. [13] demonstrated a vulnerability in the same system, based on the lack of key-commitment in AES-GCM. In addition, Len, Grubbs and Ristenpart [24] have introduced 'partitioning oracle attacks' on AES-GCM and ChaCha20-Poly1305. Further attacks against the key-commitment security of AES-GCM, AES-GCM-SIV and ChaCha20-Poly1305 were presented by Albertini et al. [1].

1.1 Related work

Most nonce-misuse resistant authenticated encryption schemes, including Deoxys-II and AES-GCM-SIV, are based on the synthetic initialization vector (SIV) approach of Rogaway and Shrimpton [31]. This approach involves two passes over the message: a first pass to produce the tag, and a second pass to encrypt the message. The tag is used as the initialization vector of the encryption scheme in

the second pass. Two-pass schemes typically result in lower encryption performance. For example, AES-GCM-SIV encryption requires four additional AES calls per message compared to AES-GCM. However, even for long messages, encryption could be significantly slower in practice because AES-GCM implementations can interleave the evaluation of the universal hash function and AES to exploit instruction-level parallelism. COLM does not follow the SIV approach and needs only a single pass, but it aims for the weaker notion of online nonce-misuse resistance [15, 19].

The generic security of AES-GCM and AES-GCM-SIV has been extensively analyzed. The AES-GCM-SIV proposal [18] included a security analysis in the standard model (reduction to the prp-security of AES), but several flaws were pointed out by Iwata and Seurin [21]. As a result, limits on the number of messages that can be safely encrypted turned out to be quite stringent. Bose et al. [7] provide a refined multi-user security analysis.

Until recently, the security analysis of ChaCha20-Poly1305 had received relatively little attention. A single-user standard model proof was claimed by Procter [28] but it was shown to be flawed [11, 20]. Degabriele et al. also provide the first detailed multi-user security analysis in the ideal permutation model. Since ChaCha20-Poly1305 is based on a public permutation in feed forward, the ideal permutation model is more appropriate than the standard model. If q_v verification attempts can be made and the combination of the associated data and the message consists of ℓ blocks, then the advantage upper bound is dominated by $\ell q_v / 2^{103}$. Here q_v denotes the number of verification attempts.

As AES-GCM and AES-GCM-SIV are not key-committing, Bellare and Hoang [2] have proposed secure variants of these schemes at Eurocrypt 2022. Although their modifications are sufficient to achieve key-committing security up to the birthday bound in the tag length, the performance impact for short messages is not completely clear as no implementation is available.

1.2 Contributions

We propose the PSIV construction, a nonce-misuse resistant and key-committing authenticated encryption mode based on Poly1305 and the ChaCha20 permutation, with the same interface and performance as ChaCha20-Poly1305. The construction is based on the NSIV approach [27], a variant of the SIV approach where the encryption scheme is based on the tag *and* the input nonce. However, for ChaCha20-Poly1305-PSIV, we are able to avoid any additional calls to the ChaCha20 permutation by exploiting its large state size. In fact, when part of the associated data is constant, performance is better than for ChaCha20-Poly1305.

To support our security claims, we provide a complete multi-user security analysis in the ideal permutation model that takes into account the number of repeated nonces per user. To do this, we rely on the "faulty nonce model" formalized by Dutta et al. [14]. In the faulty nonce model, one labels a query as "faulty" if it is performed for a repeated nonce. This contrasts with the older multicollision model, which bounds the number of times each nonce can be used. We believe that the faulty nonce model is more realistic for queries made by the

same user. For example, when a counter is implemented in an aperiodic manner (such as timely nonces [6] used in TLS 1.2), a simple reset does not lead to a large number of faulty nonces. An interesting aspect of our proof is that the nonce-misuse analysis combines the multicollision (for queries across different users) and faulty-nonce (for queries made by the same user) approaches. To the best of our knowledge, this is the first time both models have been combined to obtain a bound in the faulty-nonce model.

In the nonce-respecting setting, the security of PSIV is dominated by the same term ($q_v\ell/2^{103}$) as ChaCha20-Poly1305. The analysis shows that our construction provides significantly more security against nonce-misuse than AES-GCM-SIV, leading to less stringent limits on the number of messages that can be safely encrypted. In addition, we show that it is safe to use the construction with random nonces (both 96-bit and 64-bit).

ChaCha20-Poly1305-PSIV is shown to be key-committing up to the birthday bound in the tag length, which is best-possible. We use the security definition that was introduced by Bellare and Hoang [2]. Unlike ChaCha20-Poly1305, we are able to achieve key-commitment at no additional cost because our construction protects the output of Poly1305 using the ChaCha20 permutation with a feed-forward. Another advantage of our construction is that it is straightforward to increase the tag length.

In general, ChaCha20-Poly1305-PSIV offers better security with no performance loss. Since existing implementations of ChaCha20-Poly1305 can easily be adapted to implement the PSIV construction, we argue that ChaCha20-Poly1305 should be widely replaced by ChaCha20-Poly1305-PSIV. Moreover, there are applications where replacing ChaCha20-Poly1305 by ChaCha20-Poly1305-PSIV is necessary to avoid potential security risks such as nonce reuse when synchronization is not possible or when key-commitment is required. In particular, ChaCha20-Poly1305-PSIV currently appears to be the only construction that reuses the components of a widely used construction to achieve key-committing security, whereas other existing constructions (including ChaCha20-Poly1305) were shown to be insecure with potential real world attacks on some applications [1, 16, 24]. Table 1 compares ChaCha20-Poly1305-PSIV with widely used authenticated encryption modes for general applications.

Table 1: Comparison of ChaCha20-Poly1305-PSIV with widely used authenticated encryption modes for general applications. Only the leading term of the security bound is given, with ℓ the message length, q the total number of encryption and decryption queries, and q_v the number of verification/decryption queries.

	security ($q_v = 1$)	primitive calls	nonce misuse	key commitment	note
AES-GCM	$\ell/2^{63}$	$\ell + 1$	✗	✗	[25]
AES-GCM-SIV	$q\ell^2/2^{129}$	$\ell + 1 + 4^{\dagger}$	✓	✗	[18]
ChaCha20-Poly1305	$\ell/2^{103}$	$\ell + 1$	✗	✗	[23, 32]
ChaCha20-Poly1305-PSIV	$\ell/2^{103}$	$\ell + 1$	✓	✓	§3

† Four additional calls for each fresh nonce.

A reference implementation in Rust is provided as supplementary material. To support our performance claims, we also provide an optimized C implementation that is compatible with libsodium. Benchmarks indicate that performance is comparable to the ChaCha20-Poly1305 implementation of libsodium.

Overview Section 2 introduces the security notions that are used in this work and reviews the main technical results that our security analysis relies on (the H-coefficient technique and a balls-into-bins lemma). We introduce our construction PSIV in Section 3 and explain all important design choices. Its multi-user security is discussed in Section 4 and compared to that of ChaCha20-Poly1305 in the nonce-respecting setting. Furthermore, nonce-misuse security is examined in detail and the impact of using random nonces is analyzed. The security bounds that Section 4 relies on are proven in Section 5. Key-committing security is proven in Section 6. Finally, Section 7 discusses some aspects of the implementation and contains benchmarking results.

2 Preliminaries

The set of bit strings of length $n \geq 1$ is denoted by $\{0,1\}^n$. Furthermore, $\{0,1\}^*$ denotes the set of all arbitrary-length bit strings. Let X be a bit string. The length of X is denoted by $|X|$, and the size of X in n-bit blocks by $|X|_n$. The empty string is assigned a size of one so that $|X|_n = \max(1, \lfloor |X| / n \rfloor))$. The concatenation of X and a second bitstring Y is another bitstring $X \| Y$. For $X, Y \in \{0,1\}^n$ with $|X| = |Y|$, the bitstring $X \oplus Y \in \{0,1\}^n$ is their bitwise exclusive-or. We define a truncation function $\mathsf{trunc}_n(X)$ that returns the first n bits of X, and $\mathsf{trunc}_n^*(X)$ that returns the last n bits of X. Finally, if $|X|$ and $|Y|$ are multiples of 32, then $X + Y$ and $X - Y$ denote the strings that result from individually adding and subtracting the 32-bit subwords of X and Y modulo 2^{32}, where each subword is interpreted as an unsigned integer.

For a finite set $\mathcal{S}$, we denote by $S \leftarrow\$ \mathcal{S}$ that S is a random variable uniformly distributed on $\mathcal{S}$. In particular, let $\mathrm{Perm}(n)$ be the set of all permutations on

$\{0,1\}^n$. All constructions in this paper are based on a single public permutation. For the security proof, this permutation is assumed to be uniform random: $\pi \leftarrow\$ \operatorname{Perm}(n)$.

Adversaries are algorithms with access to oracles. The result of running algorithm $\mathcal{A}$ with oracle $\mathcal{O}$ is denoted by $\mathcal{A}^{\mathcal{O}}$. A distinguisher is an adversary $\mathcal{D}$ that attempts to distinguish between oracles $\mathcal{O}$ and $\mathcal{P}$ by returning zero or one. The advantage of a distinguisher $\mathcal{D}$ is

$$\Delta_{\mathcal{D}}\big(\mathcal{O}\,;\mathcal{P}\big) = \big|\Pr[\mathcal{D}^{\mathcal{O}} = 1] - \Pr[\mathcal{D}^{\mathcal{P}} = 1]\big|\,.$$

2.1 Nonce-Based MAC Algorithms

A nonce-based message authentication code (MAC) F is a family of functions $F_K\colon \{0,1\}^b \times \{0,1\}^* \to \{0,1\}^t$, one for every key $K \in \{0,1\}^k$. It maps a b-bit nonce and a message to a tag.

The verification oracle $\mathrm{Ver}_K\colon \{0,1\}^b \times \{0,1\}^* \times \{0,1\}^t \to \{0,1\}$ takes as input a nonce, message, and tag and outputs 1 if the tag is correct and 0 otherwise. The rejection oracle, which returns zero for every nonce, message and tag, is denoted by Rej. In the following, it is assumed that both F_K and Ver_K are based on the same primitive π.

We use the pseudorandom mac security notion introduced by Jean et al. [22]. The multi-user pseudorandom message authentication code (mu-mac) security of F is measured by considering an adversary with access to multiple verification oracles with random keys and the primitive oracle. The goal of the adversary is to forge a tag for a new message. This security notion can be quantified in terms of the advantage of a distinguisher $\mathcal{D}$. Let $K_1, \ldots, K_u$ be $u \geq 1$ uniform random keys and let $\$_1, \ldots, \$_u$ be oracles that return uniform random outputs of length t. The mu-mac advantage of $\mathcal{D}$ is equal to

$$\mathbf{Adv}^{\text{mu-mac}}_F(\mathcal{D}) = \Delta_{\mathcal{D}}\Big((F_{K_j}, \mathrm{Ver}_{K_j})_{j=1}^u, \pi^{\pm}\,; (\$_j, \mathsf{Rej})_{j=1}^u, \pi^{\pm}\Big)\,.$$

The notation $\pi^{\pm}$ indicates that $\mathcal{D}$ has bi-directional access to π. For $u = 1$, this definition reduces to the single-user security of F and the corresponding advantage is denoted by $\mathbf{Adv}^{\text{mac}}_F(\mathcal{D})$.

The distinguisher $\mathcal{D}$ is non-trivial if, for every j, it never makes a query to its jth verification oracle with the result of a previous query to its jth MAC oracle.

A MAC query to the jth user is a faulty query if $\mathcal{D}$ has already queried F_{K_j} with the same nonce and a different message. Suppose that $\mathcal{D}$ makes at most μ faulty MAC queries. The distinguisher $\mathcal{D}$ is called nonce-respecting if $\mu = 0$, and nonce-misusing if $\mu \geq 1$. We stress that $\mathcal{D}$ may always repeat nonces in its verification queries.

2.2 Nonce-and-IV-Based Encryption Schemes

A nonce-and-IV-based encryption mode $\Pi = (\mathsf{E}, \mathsf{D})$ consists of a family of encryption and decryption functions E_K and D_K respectively, one for every key

$K \in \{0,1\}^k$. The encryption function E_K takes a b-bit nonce N, a t-bit uniform random initialization vector T, and an arbitrary-length message M, and outputs a ciphertext $C = \mathsf{E}_K(N,T,M) \in \{0,1\}^{|M|}$. The decryption function D_K takes a b-bit nonce N, a t-bit IV T, an arbitrary-length ciphertext C, and outputs a message $M = \mathsf{D}_K(N,T,C) \in \{0,1\}^{|C|}$. In the following, it is assumed that Π is based on the primitive π.

Let $\mathsf{E}^{\$}_K$ denote the probabilistic algorithm which takes as input $(K,N,M) \in \{0,1\}^k \times \{0,1\}^b \times \{0,1\}^*$, internally generates a uniformly random $T \leftarrow\!\!\$\ \{0,1\}^t$, computes $C = \mathsf{E}_K(N,T,M)$, and outputs $(T,C) \in \{0,1\}^t \times \{0,1\}^*$.

We rely on the nivE security notion introduced by Peyrin and Seurin [27]. The multi-user nonce-and-IV-based encryption security of Π can be measured in terms of the advantage of a distinguisher $\mathcal{D}$. Let $K_1, \ldots, K_u$ be $u \geq 1$ uniform random keys and $\$_1, \ldots, \$_u$ oracles that return a uniform random output of length $|M|$ given an input nonce N and an arbitrary-length message M. The mu-nivE advantage of $\mathcal{D}$ is equal to

$$\mathbf{Adv}^{\text{mu-nivE}}_{\Pi}(\mathcal{D}) = \Delta_{\mathcal{D}}\Big((\mathsf{E}^{\$}_{K_j})^u_{j=1}, \pi^{\pm}; (\$_j)^u_{j=1}, \pi^{\pm}\Big)\,.$$

Note that nivE security only considers forward construction queries (the encryption oracles). The case $u = 1$ corresponds to single-user security and the corresponding advantage is denoted by $\mathbf{Adv}^{\text{nivE}}_{\Pi}(\mathcal{D})$.

2.3 Nonce-Based Authenticated Encryption

An authenticated encryption scheme $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ consists of a pair of encryption and decryption function families. In this work we focus on nonce-based authenticated encryption schemes. For every key $K \in \{0,1\}^k$, the encryption function Enc_K takes as input a nonce $N \in \{0,1\}^b$, associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$, and outputs a ciphertext $C \in \{0,1\}^{|M|}$ and tag $T \in \{0,1\}^t$. That is, $\mathsf{Enc}_K(N,A,M) = (C,T)$. The corresponding decryption function Dec_K takes as input a nonce $N \in \{0,1\}^b$, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^t$, and outputs a message $M = \mathsf{Dec}(N,A,C,T) \in \{0,1\}^{|C|}$ if the tag T is correct and $\perp$ otherwise. We require that

$$\mathsf{Dec}_K(N,A,\mathsf{Enc}_K(N,A,M)) = M\,.$$

In the following, it is assumed that $\mathcal{E}$ is based on the primitive π.

The multi-user authenticated encryption security of $\mathcal{E}$ is quantified in terms of the advantage of a distinguisher $\mathcal{D}$. Let $K_1, \ldots, K_u$ be $u \geq 1$ uniform random keys and define oracles $\$_1, \ldots, \$_u$ that output a uniform random bitstring of length $|M|+t$ given a nonce N, arbitrary-length associated data, and a message M. Denote the rejection oracle that returns $\perp$ for every decryption query by Rej. The mu-ae advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\text{mu-ae}}_{\mathcal{E}}(\mathcal{D}) = \Delta_{\mathcal{D}}\Big((\mathsf{Enc}_{K_j}, \mathsf{Dec}_{K_j})^u_{j=1}, \pi^{\pm}; (\$_j, \mathsf{Rej})^u_{j=1}, \pi^{\pm}\Big)\,.$$

For $u = 1$, the above corresponds to the single-user security of $\mathcal{E}$. The corresponding advantage is denoted by $\mathbf{Adv}^{\mathrm{ae}}_{\mathcal{E}}(\mathcal{D})$.

The distinguisher $\mathcal{D}$ is allowed to make at most μ faulty encryption queries to the same user, and in total at most d encryption queries with the same nonce to all users. We consider non-trivial distinguishers: repeated encryption queries are disallowed, and forwarding an output of an encryption query to the corresponding decryption oracle is also disallowed.

2.4 Universal Hash Functions

For a positive integer n, a universal hash function H is a family of functions $H_L\colon \{0,1\}^* \to \{0,1\}^n$, one for every key $L \in \mathcal{K}$. A universal hash function H is ϵ_{reg}-regular if for all $M \in \{0,1\}^*$ and all $C \in \{0,1\}^n$,

$$\Pr\left[L \leftarrow\$\, \mathcal{K}\colon H_L(M) = C\right] \leq \epsilon_{\mathrm{reg}}\,.$$

A universal hash function H is ϵ_{Δ}-almost Δ universal if for all distinct $M, M' \in \{0,1\}^*$ and all $C \in \{0,1\}^n$,

$$\Pr\left[L \leftarrow\$\, \mathcal{K}\colon H_L(M) - H_L(M') = C\right] \leq \epsilon_{\Delta}\,.$$

In the bounds below, these parameters depend on the input length; when all relevant inputs have at most ℓ blocks, we write the corresponding quantity as $\epsilon(\ell)$.

2.5 Patarin's H-Coefficient Technique

We use the H-coefficient technique of Patarin [26], and our description of it follows Chen and Steinberger [8].

Consider a deterministic distinguisher $\mathcal{D}$ that is given access to either a real world oracle $\mathcal{O}$ or an ideal world oracle $\mathcal{P}$. The query-response tuples learned by $\mathcal{D}$ during its interaction with the oracle $\mathcal{O}$ or $\mathcal{P}$ can be summarized in a transcript τ. Let $X_{\mathcal{O}}$ (respectively $X_{\mathcal{P}}$) be a random variable equal to the transcript produced by the interaction between $\mathcal{D}$ and $\mathcal{O}$ (respectively $\mathcal{P}$). A particular transcript τ is called attainable if $\Pr[X_{\mathcal{P}} = \tau] > 0$ and the set of all attainable transcripts is denoted by $\mathcal{T}$.

Lemma 1 (H-coefficient technique). *Let $\mathcal{D}$ be a deterministic distinguisher. Define a partition $\mathcal{T} = \mathcal{T}_{\mathrm{good}} \cup \mathcal{T}_{\mathrm{bad}}$, where $\mathcal{T}_{\mathrm{good}}$ is the subset of attainable transcripts $\mathcal{T}$ which contains all the "good" transcripts and $\mathcal{T}_{\mathrm{bad}}$ is the subset with all the "bad" transcripts. If there exists an $\epsilon \geq 0$ such that for all attainable $\tau \in \mathcal{T}_{\mathrm{good}}$,*

$$\frac{\Pr[X_{\mathcal{O}} = \tau]}{\Pr[X_{\mathcal{P}} = \tau]} \geq 1 - \epsilon\,,$$

then $\Delta_{\mathcal{D}}\big(\mathcal{O}\,;\mathcal{P}\big) \leq \epsilon + \Pr[X_{\mathcal{P}} \in \mathcal{T}_{\mathrm{bad}}]$.

2.6 A Balls-Into-Bins Lemma

To bound the bad transcript probabilities in our security analysis, the following lemma of Degabriele et al. [11] will be used.

Lemma 2 (Balls-into-bins [11], Lemma C.1). *Consider an experiment where Q balls are thrown into a set of bins at random, where each throw may depend on the outcome of the prior ones. Let $1/D \leq 1$ be an upper bound on the probability that, when conditioned on prior throws, a ball lands into any bin. If $Q \leq D \log_2(D)/3$, then the probability that the heaviest bin contains $2\log_2(D)/\max(1, \log_2(D/Q))$ or more balls is at most $1/D$.*

3 Construction

Our generic construction combines a universal hash function and a public permutation using the synthetic initialization vector (SIV) approach of Rogaway and Shrimpton [31] to build a nonce-misuse resistant authenticated encryption scheme. The specific instantiation based on the Poly1305 [1] hash function and the ChaCha20 permutation is called ChaCha20-Poly1305-PSIV. Similar to ChaCha20-Poly1305, encryption is performed using a variant of counter mode. However, the MAC function of our construction is considerably different.

The interface of our proposal is identical to that of ChaCha20-Poly1305 with 96-bit nonces. Specifically, the key length is 256 bits and the algorithm produces 128-bit tags. A high-level overview of the construction is given in Section 3.1. Section 3.2 discusses the rationale behind the design and its main novelties. Further details about the construction are given in Section 3.3.

3.1 Overview

A high-level overview of our construction PSIV is shown in Figure 1 and Algorithm 1-4. It is based on a universal hash function H and two functions φ_m and φ_e, both of which are based on a public permutation π. The hash function key L is static and generated from the master key K using a third function: $L = \varphi_k(K)$. The detailed construction of the functions φ_k, φ_m and φ_e is given in Section 3.3.

The PSIV AEAD scheme internally uses two subprocedures, the MAC function F and the IV-based encryption scheme Π. In the MAC algorithm F, the function $\varphi_m : \{0,1\}^k \times \{0,1\}^b \times \{0,1\}^t \to \{0,1\}^t$ maps the key K, a nonce N and the hash output H to a tag $T = \varphi_m(K, N, H)$ [2]. The tag is used as the initialization vector of the keystream generator of the encryption mode Π. The keystream

[1] The Poly1305 algorithm designed by Bernstein [4], takes a key consisting of two 128-bit strings (r, s); the tag corresponding to a message M is computed as $H_r(M)(t) + s$. This construction was also adopted in the original ChaCha20-Poly1305 AEAD scheme [23, 32].

[2] The MAC algorithm can be seen as a permutation-based variant of the NaK construction proposed in [10]. One of the main differences is that we benefit from the large permutation size and the nonce is not xored to the universal hash output.

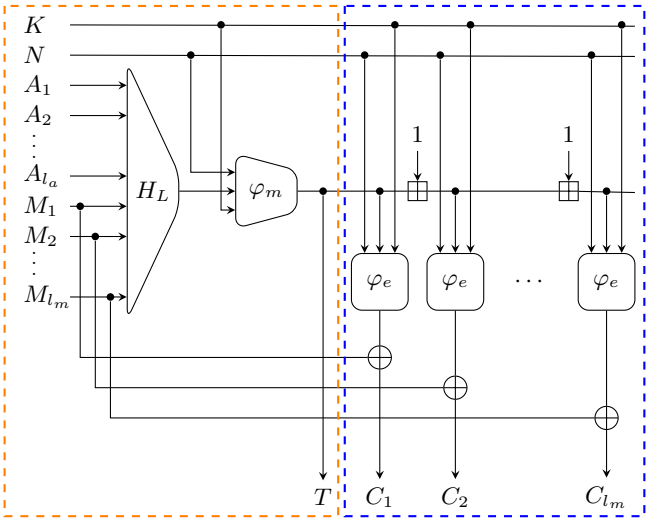

Fig. 1: The PSIV authenticated encryption scheme.

is generated by using the tag as a counter, incrementing it by one for every block. More specifically, the function $\varphi_e : \{0,1\}^k \times \{0,1\}^b \times \{0,1\}^t \to \{0,1\}^n$ takes the key K, a nonce N and a counter $T + i - 1$, and produces the ith keystream block $\varphi_e(K, N, T + i - 1)$. The ciphertext blocks are obtained as $C_i = M_i \oplus \varphi_e(K, N, T + i - 1)$, with M_i the ith message block. Note that for our IV-based encryption mode Π, the decryption function is the same as the encryption function: $\mathsf{D}_K = \mathsf{E}_K$.

Figure 1 does not include the padding scheme for the associated data and message. It is the same as in ChaCha20-Poly1305; additional details are given in Section 3.3.

3.2 Rationale

Reusing a nonce in ChaCha20-Poly1305 immediately leads to loss of authenticity and confidentiality. Confidentiality is lost because the same nonce always produces the same keystream. This issue can be avoided by using SIV, since the keystream then depends on the tag and hence indirectly on the associated data and message. However, this means that – unlike in ChaCha20-Poly1305 – the tag must be computed from the message rather than from the ciphertext. As discussed below, our construction in fact achieves this without a loss in performance.

It is not possible to directly use a Poly1305 hash of the associated data and message as the tag, because this does not resolve the loss of authenticity when nonces are repeated. Instead, a nonce-misuse resistant MAC is necessary. However, at the same time, additional calls to ChaCha20 should be avoided. We achieve this by protecting the output of Poly1305 with a call to the function

Algorithm 1 MAC function F_K with key $K \in \{0,1\}^k$.

Input: $N \in \{0,1\}^b$, $A \in \{0,1\}^*$, $M \in \{0,1\}^*$
Output: $T \in \{0,1\}^t$
1: $L \leftarrow \varphi_k(K)$ ▷ Can be precomputed.
2: $H \leftarrow H_L(A \| \mathsf{pad}(A) \| M \| \mathsf{pad}(M) \| \mathsf{len}(A) \| \mathsf{len}(M))$
3: $T \leftarrow \varphi_m(K, N, H)$
4: **return** T

Algorithm 2 Encryption algorithm E_K with key $K \in \{0,1\}^k$.

Input: $T \in \{0,1\}^t$, $N \in \{0,1\}^b$, $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^{|M|}$
1: $M_1 \| \ldots \| M_{\ell_m} \leftarrow M$
2: **for** $i = 1, \ldots, \ell_m - 1$ **do**
3: $\quad C_i \leftarrow M_i \oplus \varphi_e(K, N, T + i - 1)$
4: **end for**
5: $C_{\ell_m} \leftarrow M_{\ell_m} \oplus \mathsf{trunc}_{|M_{\ell_m}|}(\varphi_e(K, N, T + \ell_m - 1))$
6: **return** $C_1 \| \ldots \| C_{\ell_m}$

Algorithm 3 AEAD encryption Enc_K with key $K \in \{0,1\}^k$.

Input: $N \in \{0,1\}^b$, $A \in \{0,1\}^*$, $M \in \{0,1\}^*$
Output: $C \in \{0,1\}^{|M|}, T \in \{0,1\}^t$
1: $T \leftarrow F_K(N, A, M)$
2: $C \leftarrow \mathsf{E}_K(N, T, M)$
3: **return** (C, T)

Algorithm 4 AEAD decryption Dec_K with key $K \in \{0,1\}^k$.

Input: $N \in \{0,1\}^b$, $A \in \{0,1\}^*$, $C \in \{0,1\}^*$, $T \in \{0,1\}^t$
Output: $M \in \{0,1\}^{|C|}$ or $\perp$
1: $M \leftarrow \mathsf{E}_K(N, T, C)$
2: $T' \leftarrow F_K(N, A, M)$
3: **if** $T \neq T'$ **then**
4: $\quad$ **return** $\perp$
5: **else**
6: $\quad$ **return** M
7: **end if**

Fig. 2: Description of the PSIV mode in pseudocode.

φ_m. This approach makes it possible to reuse the universal hash function key L instead of generating it from the nonce, thereby saving one call to ChaCha20. As a result, nonce-misuse resistance is achieved using the same number of ChaCha20 calls as in ChaCha20-Poly1305. Using a fixed hash function key L has the additional benefit that the contribution of constant associated data or message blocks can be precomputed.

It is worth noting that the modified order of operations (compared to ChaCha20-Poly1305) does not have a negative impact on performance. This is unlike for AES-GCM, where applying the SIV approach as in AES-GCM-SIV decreases encryption speed for long messages by up to 50% on some platforms. This is because, unlike ChaCha20-Poly1305 implementations, AES-GCM implementations interleave the evaluation of the universal hash function and AES to exploit instruction level parallelism.

3.3 Additional Details

The functions φ_k, φ_m and φ_e are constructed from the public permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ as follows (we assume $n \geq k + t + b$):

$$\begin{aligned}
\varphi_k(K) &= \mathsf{trunc}_k(\pi(f_k(K)\|00\cdots0) + (f_k(K)\|00\cdots0))\,, \\
\varphi_m(K,N,H) &= \mathsf{trunc}_t(\pi(f_m(K)\|N\|H) + (f_m(K)\|N\|H))\,, \\
\varphi_e(K,N,T+i-1) &= \pi(f_e(K)\|N\|T+i-1) + (f_e(K)\|N\|T+i-1)\,.
\end{aligned}$$

For the concrete construction ChaCha20-Poly1305-PSIV, we have $n = 512$, $k = 256$, $b = 96$ and $t = 128$. Figure 3 illustrates the structure of the input to ChaCha20. The injective functions f_k, f_m and $f_e : \{0,1\}^k \to \{0,1\}^{n-t-b}$ expand the master key by inserting constant bytes at specific positions. For generic permutations, the position of the constants does not matter. However, for ChaCha20, we choose the positions to ensure that all input states are in different orbits of any column rotation. This is necessary because ChaCha20 satisfies $\mathsf{ChaCha20}(\rho(x)) = \rho(\mathsf{ChaCha20}(x))$ for any rotation ρ of the columns. In this way, our construction specifically accounts for the symmetry properties of ChaCha20.

Remark 1 (ChaCha20 symmetry). The informal claim above should be understood in the following sense. Let G be the group generated by the column rotations of the ChaCha20 state. If the public permutation is sampled uniformly from the set of permutations π satisfying $\pi(\rho(x)) = \rho(\pi(x))$ for all ρ in G, then the proof still applies because the constants in f_k, f_m and f_e ensure that all permutation inputs lie in distinct G-orbits. Hence, the column-rotation symmetry does not introduce additional input collisions beyond the bad events already bounded in the ideal-permutation proof.

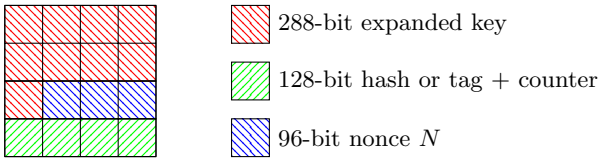

Fig. 3: Modified input structure of ChaCha20.

Specifically, we choose f_k, f_m and f_e as follows for ChaCha20-Poly1305-PSIV:

$$\begin{aligned} f_k(K) &= K_1\|K_2\|K_3\|\mathsf{03}\|K_5\|K_6\|K_7\|\mathsf{0c}\|K_9\|K_{10}\|K_{11}\|\mathsf{30}\|K_4\|K_8\|K_{12} \\ &\quad \|\mathsf{c0}\|K_{13}\|K_{14}\|\cdots\|K_{32}\,, \\ f_m(K) &= K_1\|K_2\|K_3\|\mathsf{05}\|K_5\|K_6\|K_7\|\mathsf{0a}\|K_9\|K_{10}\|K_{11}\|\mathsf{50}\|K_4\|K_8\|K_{12} \\ &\quad \|\mathsf{a0}\|K_{13}\|K_{14}\|\cdots\|K_{32}\,, \\ f_e(K) &= K_1\|K_2\|K_3\|\mathsf{06}\|K_5\|K_6\|K_7\|\mathsf{09}\|K_9\|K_{10}\|K_{11}\|\mathsf{60}\|K_4\|K_8\|K_{12} \\ &\quad \|\mathsf{90}\|K_{13}\|K_{14}\|\cdots\|K_{32}\,, \end{aligned}$$

with $K_1, \ldots, K_{32}$ the 32 bytes of the master key K.

Compared to ChaCha20-Poly1305, the adversary-controlled words are mostly in the third rather than the fourth row of the input state – but due to the symmetry of the permutation this does not have a significant impact on existing cryptanalysis. Note that in most cases, only the nonce can be freely chosen by the adversary. The adversary does not know and has little control over the output of Poly1305. The tag is known, but not chosen by the adversary.

Finally, the input to the Poly1305 hash function is padded as in ChaCha20-Poly1305: the associated data A and message M are padded with zeros up to a multiple of the block length, and their lengths are appended. Specifically, the input to Poly1305 is equal to

$$A\|\mathsf{pad}(A)\|M\|\mathsf{pad}(M)\|\mathsf{len}(A)\|\mathsf{len}(M)\,,$$

with $\mathsf{pad}(X)$ a minimum-length string of zeros such that $|X\|\mathsf{pad}(X)|$ is a multiple of 128. Here, $\mathsf{len}(X)$ is the unsigned 64-bit integer representation of the byte length of X.

3.4 Generic Construction

The construction described above can be generalized to arbitrary parameter sizes. In particular, PSIV is the authenticated encryption mode with k-bit keys, t-bit tags, b-bit nonces and n-bit blocks (with $n \geq k+t+b$) corresponding to the pseudocode in Figure 2. More specifically, PSIV instantiates the construction in Figures 1 and 2 with functions φ_k, φ_m and φ_e based on an n-bit permutation π in feed-forward (with proper domain separation as in Section 3.3), and an ϵ-regular and ϵ-almost Δ universal hash function H. If π is instantiated with ChaCha20 and H with Poly1305, then the resulting mode is called ChaCha20-Poly1305-PSIV. The security analysis in Section 5 considers the generic mode PSIV, with $\pi \leftarrow\$ \mathrm{Perm}(n)$.

4 Multi-user Security

In Section 5, we prove the following result about the multi-user security of PSIV in the nonce-misuse setting.

Theorem 1. *Let $\mathcal{D}$ be a multi-user authenticated encryption adversary for $u \leq 2^{(n-k)/2-1}$ users against* PSIV *with $\pi \leftarrow\$ \mathrm{Perm}(n)$ and H an ϵ-regular and ϵ-almost Δ universal hash function. Assume that $\mathcal{D}$ makes $q_e \leq (n-k-t)/3 \cdot 2^{n-k-t}$ encryption queries (in total over all users) totaling $\sigma \leq 2^{(n-k)/2-1}$ blocks, $q_v \leq 2^t$ decryption queries (in total over all users), and p primitive queries. Denote the maximum number (across all users) of faulty nonces within the same user [3] by $\mu < 2^{t/2-1}$, and let $d \leq t/3 \cdot 2^t$ be the total number of queries per nonce. If ℓ is the maximum joint number of blocks of the associated data and message and ℓ_m the maximum number of blocks of the message, then*

$$\begin{aligned}\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \epsilon(\ell) \left(\frac{2n\,(p+q_v)}{2^k} + \frac{pq_v}{2^k} + 4q_v + \mu^2 + \frac{d(q_e+p)}{2^k} \right) \\ + \frac{(\sigma+q_e)^2 + u(\sigma+2q_e+u-1)}{2^{n+1}} + \frac{2p(8+d)+u(2p+u-1)}{2^{k+1}} \\ + \frac{1}{2^t}\left(3q_v + \mu^2\ell_m + \frac{q_e(d-1)\ell_m}{2^k}\right) + \frac{3}{2^{n-k-t}} + \frac{3}{2^t}\,,\end{aligned}$$

assuming that $d\ell_m \leq t/3 \cdot 2^t$, $q_e + u \leq 2^n$ and $\sigma + p \leq 2^n$.

For ChaCha20-Poly1305-PSIV, Theorem 1 can be simplified as follows. Corollary 1 is based on the fact that Poly1305 is an ϵ-regular and ϵ-almost Δ universal hash function with $\epsilon(\ell) = (\ell+1)/2^{103}$ [4, Theorem 3.3]. The additional 16-byte block is due to the padding scheme described in Section 3.3.

Corollary 1. *Let $\mathcal{D}$ be a multi-user authenticated encryption adversary for $u \leq 2^{64}$ users against ChaCha20-Poly1305-PSIV, with ChaCha20 modeled as $\pi \leftarrow\$ \mathrm{Perm}(512)$. Assume that $\mathcal{D}$ makes at most 2^{127} encryption queries (in total over all users) totaling no more than 2^{127} blocks, $q_v \leq 2^{128}$ decryption queries (in total over all users), and $p \leq 2^{128}$ primitive queries. Denote the maximum number (across all users) of faulty nonces within the same user by μ, and suppose the total number of queries per nonce is at most 2^{63}. If $\ell \leq 2^{33}$ is the maximum joint number of blocks of the associated data and message, then*

$$\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \frac{\ell + 1.001}{2^{103}}\left(4q_v + \mu^2\right) + \frac{up}{2^{255}} + \frac{1}{2^{100}}\,.$$

The following three sections discuss the practical implications of Corollary 1. The nonce-respecting case ($\mu = 0$) is compared to ChaCha20-Poly1305 in Section 4.1. Section 4.2 examines the effect of reusing nonces, and compares ChaCha20-Poly1305-PSIV to AES-GCM-SIV. Finally, Section 4.3 shows that it is safe to use random nonces – even when nonces are restricted to 64 bits.

[3] For example, assume there are two users using respectively μ_1 and μ_2 faulty nonces, then μ is a bound on $\mu_1 + \mu_2$.

4.1 Nonce-Respecting Setting

In the nonce-respecting setting, the bound in Corollary 1 is dominated by the term $q_v\,\ell/2^{101}$. Up to a factor of four, this term is exactly the same as for ChaCha20-Poly1305 [11, Theorem 6.1]. In fact, the factor four loss is only due to the simplifications we made to obtain a more compact bound. The more detailed bounds in Section 5 and Lemma 3 in particular show that the dominant term is equal to $t/(t-1)q_v\,\ell/2^{103} \approx q_v\,\ell/2^{103}$ when $\mu = 0$.

The term $up/2^{255}$ corresponds to an offline attack on the hash function key L. It can be avoided by generating L uniformly at random, instead of using $L = \varphi_k(K)$. In any case, it is usually negligible compared to $q_v\,\ell/2^{101}$.

4.2 Nonce-Misuse Setting

If the number of queries per nonce for the same user is large, then the term $\mu^2\,\ell/2^{103}$ may dominate the advantage bound in Corollary 1. With messages of length $\ell = 2^{33}$ or 512 GiB, the bound becomes $\mu^2/2^{70}$. That is, security is lost only when a user makes about 2^{35} queries with the same nonce.

Due to our use of the faulty nonce model, as opposed to the older multicollision model that was used for the security analysis of AES-GCM-SIV, a direct comparison with the latter mode is not straightforward. To make a reasonable comparison, we consider a range of possible values for $\lambda = \mu/q_e$. The parameter λ can be interpreted as the rate at which faulty nonces occur in the implementation.

As illustrated in Figure 4 for $\lambda \in \{1, 2^{-8}, 2^{-20}, 2^{-32}\}$, our bound leads to less stringent data limits than the AES-GCM-SIV bound. A specific downside of AES-GCM-SIV, which was first pointed out by Iwata and Seurin [21], is that the dominant term in its insecurity bound is proportional to the square of the message length – as opposed to the message length itself as in our Corollary 1. The extreme case $\lambda = 1$ corresponds to a completely static nonce. In this case $\mu = q_e$, and the nonce-misuse term in Corollary 1 becomes $q_e^2\ell/2^{103}$. Hence, the data limit obtained from this term is $q_e \le 2^{51.5}/\ell$, where ℓ is the message length measured in ChaCha20 blocks. For example, even for $\ell = 2^{33}$ blocks, $q_e = 2^{32}$ static-nonce encryption queries contribute only 2^{-6} to this term, whereas ChaCha20-Poly1305 loses confidentiality after a single nonce reuse.

4.3 Random Nonces

In implementations that are unable to keep state, it is often convenient to use random nonces. In this case, our security bound in Theorem 1 remains the same except for the terms that involve μ or d. The effect of random nonces on these terms is analyzed in Appendix D. The analysis of the terms involving d introduces a new bad event to ensure that $d < 2^{t/2-1}$ and upper bounds its probability. For the terms that depend on μ, the corresponding bad events are rebounded. Under the same assumptions but for random b-bit nonces, the bound

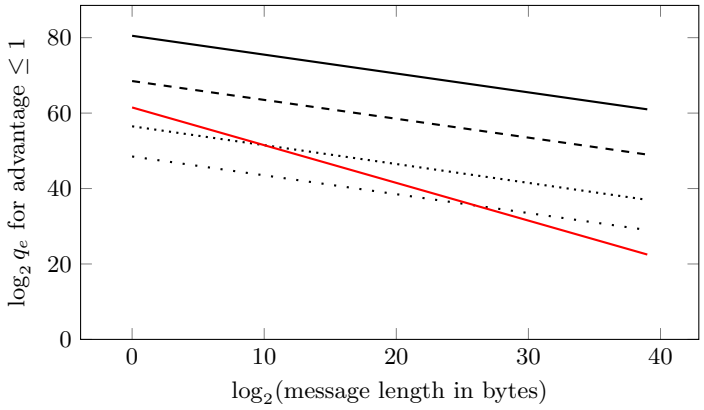

Fig. 4: Maximum amount of data as a function of message length for faulty-nonce rate $\lambda = \mu/q_e$, for ChaCha20-Poly1305-PSIV ($\cdots$ with $\lambda = 1$, $\cdots\cdots$ with $\lambda = 2^{-8}$, - - - with $\lambda = 2^{-20}$ and —— with $\lambda = 2^{-32}$) and AES-GCM-SIV (——).

in Theorem 1 becomes

$$\begin{aligned}\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \epsilon(\ell) \left(\frac{2n\,(p+q_v)}{2^k} + \frac{pq_v}{2^k} + 4q_v + \frac{q_e^2}{2^{b+1}} + \frac{q_e+p}{2^{k-t/2+1}} \right) \\ + \frac{(\sigma+q_e)^2 + u(\sigma+2q_e+u-1)}{2^{n+1}} + \frac{2p(8+d)+u(2p+u-1)}{2^{k+1}} \\ + \frac{1}{2^t} \left(3q_v + \frac{q_e^2\ell_m}{2^{b+1}} + \frac{2^{t/2-1}q_e\ell_m}{2^k} \right) + \frac{4q_e(q_e-1)}{2^{b+t/2}} + \frac{3}{2^{n-k-t}} + \frac{3}{2^t}\,.\end{aligned}$$

Using the same assumptions as in Corollary 1, this simplifies to

$$\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \frac{\ell+1.001}{2^{101}}\,q_v + \frac{q_e^2}{2^{b+61}} + \frac{up}{2^{255}} + \frac{1}{2^{100}}\,.$$

This bound shows that for 96-bit nonces, up to 2^{78} encryptions can be tolerated for any message length. That is because the above bound is dominated by the term $l \cdot q_v/2^{101}$. In practice the number of allowed verification queries is often low, so that the real limit here would be the length of the counter field. For 64-bit nonces, this reduces to 2^{62} encryptions. The latter is still comparable to the maximum acceptable number of *nonce-respecting* queries for many block cipher-based modes including AES-GCM.

The random-nonce insecurity bounds for a given message length $|M|$ (in bits) and a total number of encryptions q_e for AES-GCM-SIV and ChaCha20-Poly1305-PSIV are compared in Table 2. We assume that $q_v = 1$ like in many protocols, including TLS. With 96-bit nonces, ChaCha20-Poly1305-PSIV provides better security than AES-GCM-SIV (which also uses 96-bit nonces) – especially for long messages.

Table 2: Insecurity bounds for AES-GCM-SIV and ChaCha20-Poly1305-PSIV with random 96-bit nonces.

Scheme	$\log_2 q_e$	$\log_2 \lvert M\rvert$	$\log_2$(Bound)
AES-GCM-SIV ($b = 96$)	32	14	-61
		28	-52
		42	-24
	64	14	-29
		28	-19
		42	≥ 0
ChaCha20-Poly1305-PSIV ($b = 96$)	32	14	-92
		28	-82
		42	-67
	64	14	-29
		28	-29
		42	-29

5 Security Analysis

In this section, we prove the multi-user security of the PSIV construction in the ideal permutation model, i.e., when the ChaCha20 permutation is replaced by a uniform random permutation. This allows us to capture the local computation of the adversary in terms of the number of offline queries that it makes to the permutation.

Our proof is modular: we first prove the multi-user pseudorandom mac security of F described in Algorithm 1 and the nivE security of $\Pi = (\mathsf{E}, \mathsf{D})$ described in Algorithm 2. The following lemma upper bounds the adversarial distinguishing advantage against F.

Lemma 3. *Let $\mathcal{D}$ be a multi-user MAC adversary for $u \leq 2^{(n-k)/2-1}$ users against the MAC described in Algorithm 1 with $\pi \leftarrow\$ \operatorname{Perm}(n)$, and H an ϵ-regular and ϵ-almost Δ universal hash function. Assume that $\mathcal{D}$ makes $q_e \leq (n-k-t)/3 \cdot 2^{n-k-t}$ MAC queries (in total over all users), $q_v \leq 2^t$ verification queries (in total over all users), and p primitive queries. Denote the maximum number (across all users) of faulty nonces within the same user by $\mu < 2^{t/2-1}$, and let $d \leq t/3 \cdot 2^t$ be the total number of queries per nonce. If ℓ is the maximum joint number of blocks of the associated data and message, then*

$$\mathbf{Adv}_F^{\text{mu-mac}}(\mathcal{D}) \leq \epsilon(\ell) \left(\frac{2n\,(p+q_v)}{2^k} + \frac{pq_v}{2^k} + 4q_v + \mu^2 + \frac{d(q_e+p)}{2^k} \right) + \frac{3q_v}{2^t} + \frac{q_e^2 + u(2q_e+u-1)}{2^{n+1}} + \frac{8p + u(2p+u-1)}{2^{k+1}} + \frac{24}{2^{n-k-t}} + \frac{2}{2^t}\,,$$

where we require that $q_e + u \leq 2^n$.

The proof of Lemma 3 is given in Section 5.1, with some details moved to Supplementary Material A for brevity. The following lemma upper bounds the distinguishing advantage against the IV-based encryption scheme Π.

Lemma 4. *Let $\mathcal{D}$ be a multi-user nonce-and-IV encryption adversary for $u \leq 2^{(n-k)/2-1}$ users against Π described in Algorithm 2 with $\pi \leftarrow\$ \operatorname{Perm}(n)$. Assume that $\mathcal{D}$ makes q_e encryption queries (in total over all users) totaling $\sigma \leq 2^{(n-k)/2-1}$ blocks, and p primitive queries. Denote the maximum number (across all users) of faulty nonces within the same user by μ, and let d be the total number of queries per nonce. If ℓ_m is the maximum number of message blocks, then*

$$\mathbf{Adv}^{\text{mu-nivE}}_{\Pi}(\mathcal{D}) \leq \frac{(d+4)p}{2^k} + \frac{\ell_m}{2^t}\left(\mu^2 + \frac{q_e(d-1)}{2^k}\right) + \frac{\sigma^2}{2^{n+1}} + \frac{1}{2^{n-k}} + \frac{1}{2^t}\,.$$

where we require that $d\ell_m \leq t/3 \cdot 2^t$ and $\sigma + p \leq 2^n$.

The proof of Lemma 4 is given in Section 5.2, with some details worked out in Supplementary Material B. Finally, Theorem 1 is obtained by combining Lemma 3 and Lemma 4 using the following lemma. The proof of Lemma 5 is given in Appendix C.

Lemma 5. *Let $\mathcal{D}$ be a multi-user authenticated encryption adversary for $u \leq 2^{(n-k)/2-1}$ users against PSIV with $\pi \leftarrow\$ \operatorname{Perm}(n)$ and H an ϵ-regular and ϵ-almost Δ universal hash function. Assume that $\mathcal{D}$ makes $q_e \leq (n-k-t)/3{\cdot}2^{n-k-t}$ encryption queries (in total over all users) totaling $\sigma \leq 2^{(n-k)/2-1}$ blocks, $q_v \leq 2^t$ decryption queries (in total over all users), and p primitive queries. Denote the maximum number of all faulty nonces for the same user by $\mu < 2^{t/2-1}$, and let $d \leq t/3 \cdot 2^t$ be the total number of queries per nonce. If ℓ is the maximum joint number of blocks of the associated data and message and ℓ_m the maximum number of blocks of message, then there exist distinguishers $\mathcal{D}'$ with the same $(u, q_e, q_v, p, \mu, d, \ell)$ and $\mathcal{D}''$ with the same $(u, q_e, \sigma, p, \mu, d, \ell_m)$ such that*

$$\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \mathbf{Adv}^{\text{mu-mac}}_{F}(\mathcal{D}') + \mathbf{Adv}^{\text{mu-nivE}}_{\Pi}(\mathcal{D}'') + \frac{\sigma(q_e+u)}{2^n}\,.$$

5.1 Security Proof of Lemma 3

In this section we prove Lemma 3. Let $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$ and $\pi \leftarrow\$ \operatorname{Perm}(n)$. As explained in Section 2.1, we consider a computationally unbounded and deterministic distinguisher $\mathcal{D}$ that has access to either the real world oracles $((F_{K_j}, \operatorname{Ver}_{K_j})_{j=1}^u, \pi^{\pm})$, or the ideal world oracles $((\$_j, \mathsf{Rej})_{j=1}^u, \pi^{\pm})$. Recall that when using the H-coefficient technique, we only need to consider attainable transcripts, i.e., those that occur in the ideal world with nonzero probability.

The distinguisher makes q_e MAC queries to $(F_{K_j})_{j=1}^u$ or $(\$_j)_{j=1}^u$, with at most μ faulty queries per user and in total at most d queries per nonce. For the ith such query, there is a corresponding user index $j_i \in \{1, \ldots, u\}$, nonce input N_i, associated data input A_i, message input M_i, and output tag T_i. For convenience, we also let

$$H_{j_i} = H_{L_{j_i}}(A_i \| \mathsf{pad}(A_i) \| M_i \| \mathsf{pad}(M_i) \| \mathsf{len}(A_i) \| \mathsf{len}(M_i))\,.$$

In addition to $T_1, \ldots, T_{q_e}$ with $T_i = \varphi_m(K_{j_i} \| H_{j_i} \| N_i)$, we give the distinguisher access to the internal values $V_1, \ldots, V_{q_e}$ of φ_m (values output by the uniform random permutation π before truncation) in the real world, where the first t bits of these values (equal to $T_1, \ldots, T_{q_e}$) are given as the output of every query, and the remaining $n - t$ bits are revealed only at the end of the interaction but before $\mathcal{D}$ makes its final decision. Including $V_1, \ldots, V_{q_e}$ in the transcript makes it easier to define the tuple of bad transcripts in our proof, while only making the distinguisher more powerful. In the ideal world, $V_1, \ldots, V_{q_e}$ are dummy values, with their first t bits returned by oracles $\$_1, \ldots, \$_u$ and the remaining $n-t$ bits generated uniformly at random at the end of the interaction. To summarize, the q_e MAC queries are contained in a transcript

$$\tau_e = \{(\mathsf{con}, j_1, N_1, A_1, M_1, V_1), \ldots, (\mathsf{con}, j_{q_e}, N_{q_e}, A_{q_e}, M_{q_e}, V_{q_e})\} .$$

The distinguisher $\mathcal{D}$ also makes q_v verification queries to $(\mathrm{Ver}_{K_j})_{j=1}^{u}$ or $(\mathsf{Rej})_{j=1}^{u}$. As before, the ith such query corresponds to a user index j'_i, nonce input N'_i, associated data input A'_i, message input M'_i, tag T'_i, and an output decision bit b'_i. We also let

$$H'_{j'_i} = H_{L_{j'_i}}(A'_i \| \mathsf{pad}(A'_i) \| M'_i \| \mathsf{pad}(M'_i) \| \mathsf{len}(A'_i) \| \mathsf{len}(M'_i)) .$$

The q_v verification queries are summarized in a transcript

$$\tau_v = \{(\mathsf{ver}, j'_1, N'_1, A'_1, M'_1, T'_1, b'_1), \ldots, (\mathsf{ver}, j'_{q_v}, N'_{q_v}, A'_{q_v}, M'_{q_v}, T'_{q_v}, b'_{q_v})\} .$$

In addition to construction queries, $\mathcal{D}$ also makes p primitive queries to $\pi^{\pm}$, and these are summarized in the transcript

$$\tau_p = \{(\mathsf{prim}, x_1, y_1), \ldots, (\mathsf{prim}, x_p, y_p)\} .$$

After $\mathcal{D}$'s interaction with the oracles, but before it outputs its decision, we disclose the key material used in the interaction. This can only increase the advantage of the distinguisher. The first values that are revealed are values $K_1, \ldots, K_u$. In the real world, these are the keys $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$ that are actually used by the construction oracles (MAC and verification); in the ideal world, these are dummy keys $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$. The second values that are revealed are $U_1, \ldots, U_u \in \{0,1\}^n$. The first k bits of these values are the universal hash function keys $L_1, \ldots, L_u$. In the real world, these are the internal values $U_1, \ldots, U_u \in \{0,1\}^n$ of φ_k (values output by the uniform random permutation π before truncation) with $L_j = \varphi_k(K_j)$ for $j = 1, \ldots, u$; in the ideal world, these are dummy values $U_1, \ldots, U_u \leftarrow\$ \{0,1\}^n$. The revealed keys are summarized in a transcript

$$\tau_k = \{(K_1, U_1), \ldots, (K_u, U_u)\} .$$

Note that in the single-user setting, where $u = 1$, τ_k is a singleton.

The complete transcript is denoted by τ. We assume that the distinguisher never makes any duplicate query, hence τ_e, τ_v and τ_p contain no duplicate elements.

Bad Events In the real world, all tuples in τ_p define exactly one input-output pair for π. Likewise, every $(K, U) \in \tau_k$ defines one input-output pair for π, namely $(K\|00\cdots0, U - (K\|00\cdots0))$. Finally, every tuple $(\mathsf{con}, j_i, N_i, A_i, M_i, V_i) \in \tau_e$ also defines one input-output pair for π, namely

$$((f_m(K_{j_i})\|N_i\|H_{j_i}), (V_i - (f_m(K_{j_i})\|N_i\|H_{j_i}))) \ .$$

On the other hand, every tuple $(\mathsf{ver}, j'_i, N'_i, A'_i, M'_i, T'_i, b'_i) \in \tau_v$ with $b'_i = 0$ implies that the following *cannot* be an input-output pair for π:

$$\Big((f_m(K_{j'_i})\|N'_i\|H'_{j'_i}), (T'_i\|W'_i - (f_m(K_{j'_i})\|N'_i\|H'_{j'_i}))\Big) \ ,$$

for all $W'_i \in \{0,1\}^{n-t}$. If among all these $q_e + p + u$ input-output pairs defined by τ_e, τ_p, and τ_k there are two that have colliding input or output values, or if one of the $q_e + p + u$ input values and the first t bits of its corresponding output value collide with the q_v input-output pairs defined by τ_v, then we say τ is a bad transcript. Hence, $\tau \in \mathcal{T}_{\text{bad}}$ if and only if one of the following cases happens:

1. There exist two distinct MAC queries $(\mathsf{con}, j, N, A, M, V)$ and $(\mathsf{con}, j^*, N^*, A^*, M^*, V^*)$ in τ_e and a primitive query $(\mathsf{prim}, x, y) \in \tau_p$ such that one of the following conditions holds:

$$\begin{aligned}
&\text{bad}_1 \colon x = f_m(K_j)\|N\|H_j \ , && (1)\\
&\text{bad}_2 \colon y = V - (f_m(K_j)\|N\|H_j) \ , && (2)\\
&\text{bad}_3 \colon K_j = K_{j^*} \text{ and } N = N^* \text{ and } H_j = H_{j^*} \ , && (3)\\
&\text{bad}_4 \colon (K_j\|N\|H_j) \neq (K_{j^*}\|N^*\|H_{j^*}) \text{ and} \\
&\quad V - (f_m(K_j)\|N\|H_j) = V^* - (f_m(K_{j^*})\|N^*\|H_{j^*}) \ . && (4)
\end{aligned}$$

2. There exist a verification query $(\mathsf{ver}, j', N', A', M', T', 0) \in \tau_v$, a MAC query $(\mathsf{con}, j, N, A, M, V) \in \tau_e$, and a primitive query $(\mathsf{prim}, x, y) \in \tau_p$ such that one of the following conditions holds:

$$\begin{aligned}
&\text{bad}_5 \colon x = f_m(K_{j'})\|N'\|H'_{j'} \text{ and } \mathsf{trunc}_t(x + y) = T' \ , && (5)\\
&\text{bad}_6 \colon N = N' \text{ and } H_j = H'_{j'} \text{ and } K_j = K_{j'} \text{ and } \mathsf{trunc}_t(V) = T' \ . && (6)
\end{aligned}$$

3. There exist a construction MAC query $(\mathsf{con}, j, N, A, M, V) \in \tau_e$, a primitive query $(\mathsf{prim}, x, y) \in \tau_p$, and two distinct key pairs $(K, U), (K^*, U^*) \in \tau_k$ such that one of the following conditions holds:

$$\begin{aligned}
&\text{bad}_7 \colon x = f_k(K)\|00\cdots0 \ , && (7)\\
&\text{bad}_8 \colon y = U - (f_k(K)\|00\cdots0) \ , && (8)\\
&\text{bad}_9 \colon V - (f_m(K_j)\|N\|H_j) = U - (f_k(K)\|00\cdots0) \ . && (9)\\
&\text{bad}_{10} \colon K = K^* \ . && (10)\\
&\text{bad}_{11} \colon U = U^* \ . && (11)
\end{aligned}$$

Any attainable transcript τ for which $\tau \notin \mathcal{T}_{\text{bad}}$ will be called a good transcript. In Supplementary Material A, we prove the following lemma.

Lemma 6. *If* $q_e \le (n-k-t)/3 \cdot 2^{n-k-t}$, $\mu + 1 \le 2^{t/2-1}$, $nt \le t/3 \cdot 2^t$, *and* $u \le 2^{(n-k)/2-1}$, *then the ideal world probability of the bad transcripts satisfies*

$$\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\mathrm{bad}}] \le \epsilon(\ell) \left(\frac{2n\,(p+q_v)}{2^k} + \frac{pq_v}{2^k} + 4q_v + \mu^2 + \frac{d(q_e+p)}{2^k} \right) + \frac{q_v}{2^t} + \frac{q_e^2 + u(2q_e+u-1)}{2^{n+1}} + \frac{8p + u(2p+u-1)}{2^{k+1}} + \frac{2}{2^{n-k-t}} + \frac{2}{2^t}\,.$$

Ratio for Good Transcripts Consider an attainable transcript $\tau \in \mathcal{T}_{\mathrm{good}}$. We now lower bound $\Pr[X_{\mathcal{O}} = \tau]$ and upper bound $\Pr[X_{\mathcal{P}} = \tau]$ in order to obtain a lower bound for the ratio of these probabilities.

Ideal World. For the ideal world oracle $\mathcal{P} = ((\$_j, \mathsf{Rej})_{j=1}^u, \pi^{\pm})$, the transcript $\tau = (\tau_e, \tau_v, \tau_p, \tau_k)$ consists of four lists of independent tuples: τ_e consists of q_e random n-bit strings $V_1, \ldots, V_{q_e} \leftarrow\$\ \{0,1\}^n$ (where the first t bits are the output of $\$_1, \ldots, \$_u$), τ_p defines exactly p input-output pairs for π, τ_k consists of u random tuples $(K_1, U_1), \ldots, (K_u, U_u) \leftarrow\$\ \{0,1\}^k \times \{0,1\}^n$, and τ_v only contains rejected verification queries. The probability of any good transcript τ is equal to

$$\begin{aligned}
\Pr[X_{\mathcal{P}} = \tau] &= \Pr\left(K'_1, \ldots, K'_u \leftarrow\$\ \{0,1\}^k : K'_1 = K_1 \wedge \cdots \wedge K'_u = K_u\right) \\
&\quad \times \Pr\left(U'_1, \ldots, U'_u \leftarrow\$\ \{0,1\}^n : U'_1 = U_1 \wedge \cdots \wedge U'_u = U_u\right) \\
&\quad \times \Pr\left(\$_1, \ldots, \$_u : (\$_j)_{j=1}^u \vdash \tau_e\right) \\
&\quad \times \Pr\Big(\mathsf{trunc}^*_{n-t}(V'_1), \ldots, \mathsf{trunc}^*_{n-t}(V'_{q_e}) \leftarrow\$\ \{0,1\}^{n-t} : \\
&\qquad\quad \mathsf{trunc}^*_{n-t}(V_i \oplus V'_1) = 0 \wedge \cdots \wedge \mathsf{trunc}^*_{n-t}(V_{q_e} \oplus V'_{q_e}) = 0\Big) \\
&\quad \times \Pr\left(\pi \leftarrow\$\ \mathrm{Perm}(n) : \pi \vdash \tau_p\right) \\
&= \frac{1}{2^{(k+n)u}} \cdot \frac{1}{2^{nq_e}} \cdot \frac{(2^n-p)!}{2^n!} \\
&= \frac{1}{2^{ku}} \cdot \left(\frac{(2^n-1)!}{2^n!}\right)^u \cdot \left(\frac{(2^n-1)!}{2^n!}\right)^{q_e} \cdot \frac{(2^n-p)!}{2^n!} \\
&\le \frac{1}{2^{ku}} \cdot \frac{(2^n-q_e-p-u)!}{2^n!}\,,
\end{aligned} \tag{12}$$

where we use the fact that $(2^n-q_e)! \cdot (2^n-u)!/2^n! \le (2^n-q_e-u)!$ if $q_e + u \le 2^n$. The first factor in the second equality is due to the number of possible user key pairs $(K_1, U_1), \ldots, (K_u, U_u) \leftarrow\$\ \{0,1\}^k \times \{0,1\}^n$. The second factor corresponds to the probability that $\$_1, \ldots, \$_u$ are consistent with the transcript τ_e and the number of possible $\mathsf{trunc}_{n-t}(V_1), \ldots, \mathsf{trunc}_{n-t}(V_{q_e}) \leftarrow\$\ \{0,1\}^{n-t}$. Finally, the third factor corresponds to the probability that the uniform random permutations π is consistent with the transcript τ_p. Note that since the verification oracle in the ideal world always returns zero, it does not affect the probability of good transcripts in the ideal world.

Real World. Similarly, a good transcript $\tau = (\tau_e, \tau_p, \tau_k)$ compatible with the real world oracle $\mathcal{O} = ((F_{K_j}, \mathrm{Ver}_{K_j})_{j=1}^u, \pi)$ defines exactly $q_e + p + u$ distinct input-output pairs for π. Furthermore, τ_k consists of random values $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$. Moreover, there are at most q_v more calls to π done during verification queries, and these are distinct from the ones in τ_e, τ_p, and τ_k. However, the input-output pairs for π corresponding to these calls are not completely specified in τ_v; only the inputs are. Since τ is a good transcript, the first t bits of the outputs are required to be distinct from those defined by τ_e, τ_p, and τ_k. Therefore, we obtain:

$$\begin{aligned}
&\Pr[X_{\mathcal{O}} = \tau] = \Pr\left(K'_1, \ldots, K'_u \leftarrow\$ \{0,1\}^k : K'_1 = K_1 \wedge \cdots \wedge K'_u = K_u\right) \\
&\times \Pr\left(\pi \leftarrow\$ \mathrm{Perm}(n) : (F_{K_j})_{j=1}^u \vdash \tau_e \wedge \pi \vdash (\tau_p, \tau_k)\right) \\
&\times \Pr\left(\pi \leftarrow\$ \mathrm{Perm}(n) : (\mathrm{Ver}_{K_j})_{j=1}^u \vdash \tau_v \,\middle|\, (F_{K_j})_{j=1}^u \vdash \tau_e \wedge \pi \vdash (\tau_p, \tau_k)\right) \\
&\geq \frac{1}{2^{uk}} \cdot \frac{(2^n - q_e - p - u)!}{2^n!} \cdot \prod_{i=0}^{q_v - 1} \left(1 - \frac{B}{2^n - q_e - p - u - i}\right), \qquad (13)
\end{aligned}$$

where the probability is taken with respect to $\pi \leftarrow\$ \mathrm{Perm}(n)$ and conditional on the keys. As before, the first factor corresponds to the number of possible user keys $K_1, \ldots, K_u$. The second factor is the probability that $(F_{K_j})_{j=1}^u$ is consistent with τ_e and that π is consistent with τ_p and τ_k. The third factor is the probability that $(\mathrm{Ver}_{K_j})_{j=1}^u$ is consistent with τ_v under the condition that $(F_{K_j})_{j=1}^u$ is consistent with τ_e and that π is consistent with τ_p and τ_k. Here, the inequality (13) is bounded using $q_v < 2^t$.

The value B in (13) is an upper bound on the number of $(T \| W) \in \{0,1\}^n$ such that $T = T'$ for a verification query $(\mathsf{ver}, j', N', A', M', T', 0)$. Since there are 2^{n-t} possible values for W, we can take $B = 2^{n-t}$. This is only an upper bound: the exact number can be smaller because outputs of π already fixed by the transcript and matching T' should be subtracted. Combining (12) and (13) with $B = 2^{n-t}$, we find that every good transcript $\tau \in \mathcal{T}_{\mathrm{good}}$ satisfies

$$\begin{aligned}
\frac{\Pr[X_{\mathcal{O}} = \tau]}{\Pr[X_{\mathcal{P}} = \tau]} &\geq \prod_{i=0}^{q_v - 1} \left(1 - \frac{2^{n-t}}{2^n - q - p - u - i}\right) \\
&\geq \prod_{i=0}^{q_v - 1} \left(1 - \frac{2^{n-t}}{2^{n-1}}\right) \\
&\geq \left(1 - \frac{1}{2^{t-1}}\right)^{q_v} \\
&\geq 1 - \frac{2q_v}{2^t}. \qquad (14)
\end{aligned}$$

Conclusion Using Patarin's H-Coefficient technique (Lemma 1), we obtain the desired result:

$$\mathbf{Adv}_F^{\text{mu-mac}}(\mathcal{D}) \le \epsilon(\ell)\left(\frac{2n\,(p+q_v)}{2^k} + \frac{pq_v}{2^k} + 4q_v + \mu^2 + \frac{d(q_e+p)}{2^k}\right) + \frac{3q_v}{2^t} + \frac{q_e^2 + u(2q_e+u-1)}{2^{n+1}} + \frac{8p + u(2p+u-1)}{2^{k+1}} + \frac{2}{2^{n-k-t}} + \frac{2}{2^t}\,,$$

for $q_e \le (n-k-t)/3 \cdot 2^{n-k-t}$, $\mu + 1 \le 2^{t/2-1}$, $d \le t/3 \cdot 2^t$, $u \le 2^{(n-k)/2-1}$, $q_e + u \le 2^n$, and $q_v < 2^{2^t}$.

5.2 Security Proof of Lemma 4

In this section we prove Lemma 4. Let $K_1, \ldots, K_u \xleftarrow{\$} \{0,1\}^k$ and $\pi \xleftarrow{\$} \mathrm{Perm}(n)$. Note that mu-nivE security requires outputs indistinguishable from random. If n keystream bits are pseudorandom, this remains true for shorter streams. Without loss of generality, we can assume that $\mathcal{D}$ makes queries of length a multiple of n bits.

Let $\mathsf{Rand}_1, \ldots, \mathsf{Rand}_u$ be oracles that for every input (N, T, M) return a random bitstring of length $|M|$. As discussed in Section 2.2, we consider a computationally unbounded deterministic distinguisher $\mathcal{D}$ with access to either the real world oracles $((\mathsf{E}_{K_j})_{j=1}^u, \pi^{\pm})$ or the ideal world oracles $((\mathsf{Rand}_j)_{j=1}^u, \pi^{\pm})$.

The distinguisher makes q_e construction queries to $(\mathsf{E}_{K_j})_{j=1}^u$ or $(\mathsf{Rand}_j)_{j=1}^u$, with each query of length at most ℓ_m blocks and totaling at most σ encrypted blocks. It can make at most μ faulty queries per user, and in total at most d queries per nonce. The ith such query corresponds to a user index j_i, nonce input N_i, tag input T_i, message input M_i, and ciphertext C_i. We also include the values $W_1, \ldots, W_{q_e}$ with $W_i = M_i \oplus C_i$ for $i = 1, \ldots, q_e$ in the transcript. Due to the assumption that $|M|$ is a multiple of n, it holds that $|W_i| = |M| = \ell_m$. Below, the n-bit blocks in W_i will be denoted by $W_i[1], \ldots, W_i[\ell_m]$. Note that in the real world, $W_i[l] = \varphi_e(K_{j_i} \| N_i \| T + l - 1)$ for $l = 1, \ldots, \ell_m$. Including $W_1, \ldots, W_{q_e}$ in the transcript makes it easier to define the tuple of bad transcripts for our proof. The q_e queries are summarized in a transcript

$$\tau_e = \{(\mathsf{con}, j_1, N_1, T_1, M_1, C_1, W_1), \ldots, (\mathsf{con}, j_{q_e}, N_{q_e}, T_{q_e}, M_{q_e}, C_{q_e}, W_{q_e})\}\,.$$

The distinguisher $\mathcal{D}$ also makes p primitive queries to $\pi^{\pm}$, and these are summarized in a transcript

$$\tau_p = \{(\mathsf{prim}, x_1, y_1), \ldots, (\mathsf{prim}, x_p, y_p)\}\,.$$

After $\mathcal{D}$'s interaction with the oracles, but before it outputs its decision, we disclose the keys $K_1, \ldots, K_u$ to the distinguisher. This can only increase its advantage. In the real world, these are the keys used in the construction. In the ideal world, $K_1, \ldots, K_u$ are uniform random dummy keys. The complete transcript is denoted by τ.

Bad Events In the real world, all tuples in τ_p define exactly one input-output pair for π. Furthermore, any $(\mathsf{con}, j_i, N_i, T_i, M_i, C_i, W_i) \in \tau_e$ also defines at most ℓ_m input-output pairs for π, namely

$$(f_e(K_{j_i})\|N_i\|T_i + l - 1, W_i[l] - (f_e(K_{j_i})\|N_i\|T_i + l - 1)) \ ,$$

for $l = 1, \ldots, \ell_m$. If two of the $\sigma + p$ input-output pairs defined by τ have colliding input or output values, we say that τ is a bad transcript. Hence, $\tau \in \mathcal{T}_{\text{bad}}$ if and only if there exist construction queries $(\mathsf{con}, j, N, T, M, C, W) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ in τ_e and a primitive query $(\mathsf{prim}, x, y) \in \tau_p$ such that one of the following conditions holds:

$$\text{bad}_1 \colon x \in \left\{f_e(K_j)\|N\|T, \ldots, f_e(K_j)\|N\|T + \ell_m - 1\right\}, \tag{15}$$

$$\text{bad}_2 \colon y \in \left\{W[1] - (f_e(K_j)\|N\|T), \ldots, W[\ell_m] - (f_e(K_j)\|N\|T + \ell_m - 1)\right\}, \tag{16}$$

$$\text{bad}_3 \colon N = N^* \text{ and } T = T^* + l^* - 1 \text{ and } K_j = K_{j^*} \text{for } l^* = 1, \ldots, \ell_m \ , \tag{17}$$

$$\begin{aligned}\text{bad}_4 \colon &(K_j\|N\|T + l - 1) \neq (K_{j^*}\|N^*\|T^* + l^* - 1) \text{ and} \\ &W[l] - (f_e(K_j)\|N\|T + l - 1) = W^*[l^*] - (f_e(K_{j^*})\|N^*\|T^* + l^* - 1) \\ &\qquad \text{for } l^* = 1, \ldots, \ell_m \, .\end{aligned} \tag{18}$$

Any attainable transcript τ for which $\tau \notin \mathcal{T}_{\text{bad}}$ will be called a good transcript. In Supplementary Material B, we prove the following lemma.

Lemma 7. *If $d\ell_m \leq t/3 \cdot 2^t$ and $\sigma \leq 2^{(n-k)/2-1}$, then the ideal world probability of the bad transcripts satisfies*

$$\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}] \leq \frac{(d+4)p}{2^k} + \frac{\ell_m}{2^t}\left(\mu^2 + \frac{q_e(d-1)}{2^k}\right) + \frac{\sigma^2}{2^{n+1}} + \frac{1}{2^{n-k}} + \frac{1}{2^t} \, .$$

Ratio for Good Transcripts Consider an attainable transcript $\tau \in \mathcal{T}_{\text{good}}$. We lower bound $\Pr[X_{\mathcal{O}} = \tau]$ and upper bound $\Pr[X_{\mathcal{P}} = \tau]$ to obtain a lower bound for the ratio of these probabilities.

Ideal World. For the ideal world oracle $\mathcal{P} = ((\mathsf{Rand}_j)_{j=1}^u, \pi^{\pm})$, the transcript $\tau = (\tau_e, \tau_v, K_1, \ldots, K_u)$ consists of the tuple τ_e containing σ random n-bit blocks returned by $\mathsf{Rand}_1, \ldots, \mathsf{Rand}_u$, the tuple τ_p that defines exactly p input-output pairs for π, and u random bitstrings $K_1, \ldots, K_u \leftarrow\$\ \{0,1\}^k$. The probability of a good transcript τ is

$$\begin{aligned}\Pr[X_{\mathcal{P}} = \tau] &= \Pr\left(K_1', \ldots, K_u' \leftarrow\$\ \{0,1\}^k : K_1' = K_1 \wedge \cdots \wedge K_u' = K_u\right) \\ &\quad \times \Pr\left(\mathsf{Rand}_1, \ldots, \mathsf{Rand}_u : (\mathsf{Rand}_j)_{j=1}^u \vdash \tau_e\right) \\ &\quad \times \Pr\left(\pi \leftarrow\$\ \text{Perm}(n) : \pi \vdash \tau_p\right) \\ &= \frac{1}{2^{uk}} \cdot \frac{1}{2^{n\sigma}} \cdot \frac{(2^n - p)!}{2^n!} \\ &\leq \frac{1}{2^{uk}} \cdot \frac{(2^n - \sigma - p)!}{2^n!} \, ,\end{aligned} \tag{19}$$

using the fact that $1/2^{n\sigma} \leq (2^n - \sigma)!/2^n!$ and $(2^n - \sigma)! \cdot (2^n - p)!/2^n! \leq (2^n - \sigma - p)!$ if $\sigma + p \leq 2^n$. The first factor is due to the number of possible user keys $K_1, \ldots, K_u$. The second factor corresponds to the probability that the $\mathsf{Rand}_1, \ldots, \mathsf{Rand}_u$ are consistent with the transcript τ_e. Finally, the third factor corresponds to the probability that the uniform random permutation π is consistent with the transcript τ_p.

Real World. Similarly, a good transcript $\tau = (\tau_e, \tau_p, K_1, \ldots, K_u)$ compatible with the real world oracle $\mathcal{O} = ((\mathsf{E}_{K_j})_{j=1}^u, \pi^{\pm})$ defines exactly $\sigma + p$ input-output pairs for π, and no two of them collide on the input or output. Therefore, we obtain

$$\begin{aligned}\Pr[X_{\mathcal{O}} = \tau] &= \Pr\left(K'_1, \ldots, K'_u \leftarrow\$ \{0,1\}^k : K'_1 = K_1 \wedge \cdots \wedge K'_u = K_u\right) \\ &\quad \times \Pr\left(\pi \leftarrow\$ \mathrm{Perm}(n) : (\mathsf{E}_{K_j})_{j=1}^u \vdash \tau_e \wedge \pi \vdash \tau_p\right) \\ &= \frac{1}{2^{uk}} \cdot \frac{(2^n - \sigma - p)!}{2^n!}\,, \end{aligned} \tag{20}$$

where the probability is taken with respect to $\pi \leftarrow\$ \mathrm{Perm}(n)$ and conditional on the keys. As before, the first factor corresponds to the number of possible user keys $K_1, \ldots, K_u$. The second factor is the probability that $(\mathsf{E}_{K_j})_{j=1}^u$ is consistent with τ_e and that π is consistent with τ_p.

Combining (19) and (20), we obtain that every good transcript $\tau \in \mathcal{T}_{\mathrm{good}}$ satisfies

$$\frac{\Pr[X_{\mathcal{O}} = \tau]}{\Pr[X_{\mathcal{P}} = \tau]} \geq 1\,. \tag{21}$$

Conclusion Using Patarin's H-Coefficient technique (Lemma 1), we obtain the desired result:

$$\mathbf{Adv}^{\text{mu-nivE}}_{\Pi}(\mathcal{D}) \leq \frac{(d+4)p}{2^k} + \frac{\ell_m}{2^t}\left(\mu^2 + \frac{q_e(d-1)}{2^k}\right) + \frac{\sigma^2}{2^{n+1}} + \frac{1}{2^{n-k}} + \frac{1}{2^t}\,,$$

for $d\ell_m \leq t/3 \cdot 2^t$, $\sigma \leq 2^{(n-k)/2-1}$, and $\sigma + p \leq 2^n$.

6 Key-Committing Security

We present provable security result for the key-committing security of the proposed PSIV construction, assuming its underlying permutation is ideal. We prove a birthday-type security bound in the tag length. Note that AES-GCM, AES-GCM-SIV and ChaCha20-Poly1305 do not provide key-committing security [1].

6.1 Security Notion

We use the following "committing security" notion by Bellare and Hoang [2].

Definition 1. *Let $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ be a nonce-based AE scheme with key space $\{0,1\}^k$, nonce space $\{0,1\}^b$, message space $\{0,1\}^*$, and AD space $\{0,1\}^*$. The CMT-1 insecurity is the maximum advantage over all adversaries whose goal is to find two input tuples $(K, N, A, M), (K', N', A', M') \in \{0,1\}^k \times \{0,1\}^b \times \{0,1\}^* \times \{0,1\}^*$ with $K \neq K'$ and $\mathsf{Enc}(K, N, A, M) = \mathsf{Enc}(K', N', A', M')$. The CMT-1 advantage of an adversary $\mathcal{D}$ is defined as*

$$\mathbf{Adv}^{\text{cmt-1}}_{\mathcal{E}}(\mathcal{D}) := \Pr[((K, N, A, M), (K', N', A', M')) \leftarrow \mathcal{D} \\ \textit{s.t. } K \neq K', \mathsf{Enc}(K, N, A, M) = \mathsf{Enc}(K', N', A', M')].$$

There is an even stronger security notion called cmt-4, which is identical to cmt-1 except that only $(K, N, A, M) \neq (K', N', A', M')$ is required instead of $K \neq K'$. As shown in [2], a cmt-1 scheme can be turned into a cmt-4 scheme using their proposed HtE transform that requires a suitable hash function. Whether or not ChaCha20-Poly1305-PSIV can achieve cmt-4 security using only additional calls to the ChaCha20 permutation is left as future work.

6.2 Proof of cmt-1 Security for ChaCha20-Poly1305-PSIV

Our proof is based on the notion of collision resistance. Let $H\colon \mathcal{X} \to \mathcal{Y}$ be a function. An collision for H is a pair (X_1, X_2) of distinct points in $\mathcal{X}$ such that $H(X_1) = H(X_2)$. For an adversary $\mathcal{D}$, define its advantage in breaking the collision resistance of H built on a primitive π as

$$\mathbf{Adv}^{\text{col}}_{H}(\mathcal{D}) = \Pr[(X_1, X_2) \leftarrow \mathcal{D}^{\pi^{\pm}} : X_1 \neq X_2,\ H(X_1) = H(X_2)]\,.$$

Here $\pm$ means that the adversary is getting both forward and inverse access to this primitive.

We rely on the following result on the collision resistance of the permutation-based variant of the truncated Davies-Meyer (pDM). The result follows from the multicollision resistance bound by Chen et al. [9]. In particular, let $m \leq n$ be an integer and define $\mathsf{pDM}_m(X) := (\pi(X) + X)[1..m]$ based on a permutation $\pi\colon \{0,1\}^n \to \{0,1\}^n$. For the special case $m = n$ (no truncation), we write pDM instead of pDM_m.

Lemma 8 (Collision resistance of pDM_m [9]). *Let $\pi\colon \{0,1\}^n \to \{0,1\}^n$ be an ideal permutation and $m \leq n$ a positive integer. For an adversary $\mathcal{D}$ that makes at most p ideal permutation queries, we have*

$$\mathbf{Adv}^{\text{col}}_{\mathsf{pDM}_m}(\mathcal{D}) \leq \frac{p(p-1)}{2^m}\,.$$

We now prove the following result about the cmt-1 security of PSIV.

Theorem 2. *Let $\mathcal{D}$ be a cmt-1 adversary against PSIV with $\pi \leftarrow\$ \operatorname{Perm}(n)$. If $\mathcal{D}$ makes at most $p \leq 2^{n-1} - 2$ primitive queries, then*

$$\mathbf{Adv}^{\text{cmt-1}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \frac{p(p-1)}{2^t}\,.$$

Proof. We prove cmt-1 security by showing that the collision probability of the *tag* is sufficiently small. Indeed, the tag is almost the same as the output of the pDM_t construction. For distinct input tuples of PSIV, (K, N, A, M) and (K', N', A', M') such that $K \neq K'$, the tag is given by $T = \varphi_m(K\|H\|N)$. Hence, we obtain

$$\mathbf{Adv}^{\text{cmt-1}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \Pr[((K, N, A, M), (K', N', A', M')) \leftarrow \mathcal{D} \text{ s.t. } K \neq K', T = T'] .$$

Here, $\mathcal{D}$ has both forward and backward (inverse) access to the ideal permutation π. Next, we evaluate $\mathbf{Adv}^{\text{cmt-1}}_{\mathsf{PSIV}}(\mathcal{D})$ for any adversary $\mathcal{D}$. Without loss of generality, we can assume that $\mathcal{D}$ makes no repeated queries. Hence, there exists an adversary $\mathcal{D}'$ such that

$$\mathbf{Adv}^{\text{cmt-1}}_{\mathsf{PSIV}}(\mathcal{D}) \leq \mathbf{Adv}^{\text{col}}_{\mathsf{pDM}_t}(\mathcal{D}') .$$

The adversary $\mathcal{D}'$ runs $\mathcal{D}$ and makes a total of p queries to the primitive. By Lemma 8 with $m = t$, it follows that

$$\mathbf{Adv}^{\text{col}}_{\mathsf{pDM}_t}(\mathcal{D}') \leq \frac{p(p-1)}{2^t} ,$$

which concludes the proof. □

For ChaCha20-Poly1305-PSIV, Theorem 2 implies generic security up to approximately 2^{64} queries to ChaCha20. This is the best possible cmt-1 security that can be achieved with a 128-bit tag. Note that it is straightforward to increase the tag length of PSIV.

7 Implementation

In this section, the performance of ChaCha20-Poly1305-PSIV is discussed. A reference implementation in Rust and an optimized implementation in C for libsodium 1.0.18 [12] are provided as supplementary material, which are available online. The implementation for libsodium was straightforward as all the SIMD optimizations present in the library could be reused for ChaCha20-Poly1305-PSIV.

The encryption (top half) and decryption (bottom half) speed of ChaCha20-Poly1305-PSIV and several AEAD modes implemented in libsodium 1.0.18 are given in Table 3, and additional benchmarking results are discussed in Tables 4 and 5 in Appendix E. The measurements in Table 3 were made on an Intel Xeon Gold 6320 processor at 2.10 GHz with Intel Turbo Boost Technology disabled. Libsodium was configured with the default options for our processor, which enable the AES, PCLMULQDQ, SSSE3 and AVX2 instructions set extensions during compilation. Each message was encrypted with 13 bytes of additional data[4]. Additional measurements on different platforms (AMD Ryzen 7, Intel i7) are given in Appendix E.

[4] This corresponds to the length of the additional data used in the AEAD benchmarks of the `speed` utility of OpenSSL, results for other lengths are in Appendix E.

Table 3: Encryption (top) and decryption (bottom) speed of AES256-GCM, ChaCha20-Poly1305 and ChaCha20-Poly1305-PSIV (Ours) in MiB/s. The measurements were performed on an Intel Xeon Gold 6320 processor at 2.10 Ghz with Intel Turbo Boost Technology disabled. The 95% confidence interval is given.

$\|M\|_n$	AES-256-GCM	ChaCha20-Poly1305	Ours
2	$7.56 - 7.67$	$3.66 - 3.67$	$3.28 - 3.29$
32	$112 - 117$	$52.3 - 52.4$	$50.1 - 50.2$
512	$822 - 837$	$432 - 435$	$432 - 435$
8192	$1073 - 1081$	$737 - 743$	$736 - 742$
16384	$1083 - 1091$	$755 - 761$	$755 - 761$
2	$8.10 - 8.13$	$3.59 - 3.59$	$3.28 - 3.28$
32	$118 - 118$	$49.0 - 52.3$	$49.8 - 50.4$
512	$820 - 848$	$429 - 431$	$425 - 427$
8192	$1042 - 1051$	$737 - 743$	$735 - 741$
16384	$1039 - 1053$	$756 - 761$	$756 - 761$

Table 3 (and Tables 4 and 5 in Appendix E) shows that their is no measurable difference in speed between ChaCha20-Poly1305 and ChaCha20-Poly1305-PSIV, except for small messages. This small difference can be attributed to small differences in the implementations, such as in the initialization of the input to ChaCha20, the generation of the tag or to the extra memory overhead in using multiple keys.

8 Conclusions

We presented PSIV, a new nonce-misuse resistant authenticated encryption mode based on a public permutation and a universal hash function. When PSIV is instantiated with the ChaCha20 permutation and the Poly1305 hash function, it is as efficient as ChaCha20-Poly1305 while being robust against nonce repetitions. We provided a full security analysis of PSIV in the multi-user faulty nonce setting. Using the ideal permutation model allowed us to capture offline computation of adversaries. Our analysis showed that ChaCha20-Poly1305-PSIV compares favorably to AES-GCM-SIV in the nonce-respecting, nonce-misuse, and random nonce settings – especially for long messages. We expect that ChaCha20-Poly1305-PSIV can serve as a good alternative to AES-GCM-SIV in applications where AES is not available, or where higher security is required.

Acknowledgments. This work was supported in part by the Research Council KU Leuven: GOA TENSE (C16/15/058). Tim Beyne and Yu Long Chen are recipients of postdoctoral Fellowships 1274724N and 1264825N, respectively, funded by the Research Foundation - Flanders (FWO).

References

1. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to abuse and fix authenticated encryption without key commitment. pp. 3291–3308
2. Bellare, M., Hoang, V.T.: Efficient schemes for committing authenticated encryption. pp. 845–875. LNCS
3. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545
4. Bernstein, D.J.: The poly1305-AES message-authentication code. In: FSE 2005. LNCS, vol. 3557, pp. 32–49
5. Bernstein, D.J., et al.: Chacha, a variant of salsa20. In: Workshop record of SASC. vol. 8, pp. 3–5
6. Böck, H., Zauner, A., Devlin, S., Somorovsky, J., Jovanovic, P.: Nonce-disrespecting adversaries: Practical forgery attacks on GCM in TLS. In: 10th USENIX Workshop on Offensive Technologies, WOOT 16, Austin, TX, USA, August 8-9, 2016. `https://www.usenix.org/conference/woot16/workshop-program/presentation/bock`
7. Bose, P., Hoang, V.T., Tessaro, S.: Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In: EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 468–499
8. Chen, S., Steinberger, J.P.: Tight security bounds for key-alternating ciphers. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 327–350
9. Chen, Y.L., Flórez-Gutiérrez, A., Inoue, A., Ito, R., Iwata, T., Minematsu, K., Mouha, N., Naito, Y., Sibleyras, F., Todo, Y.: Key committing security of AEZ and more. ToSC **2023**(4), 452–488
10. Cogliati, B., Lee, J., Seurin, Y.: New constructions of macs from (tweakable) block ciphers. IACR Trans. Symm. Cryptol. **2017**(2), 27–58
11. Degabriele, J.P., Govinden, J., Günther, F., Paterson, K.G.: The security of ChaCha20-Poly1305 in the multi-user setting. pp. 1981–2003
12. Denis, F.: libsodium. `https://github.com/jedisct1/libsodium` (2013)
13. Dodis, Y., Grubbs, P., Ristenpart, T., Woodage, J.: Fast message franking: From invisible salamanders to encryptment. In: CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 155–186
14. Dutta, A., Nandi, M., Talnikar, S.: Beyond birthday bound secure MAC in faulty nonce model. In: EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 437–466
15. Fleischmann, E., Forler, C., Lucks, S.: McOE: A family of almost foolproof on-line authenticated encryption schemes. In: FSE 2012. LNCS, vol. 7549, pp. 196–215
16. Grubbs, P., Lu, J., Ristenpart, T.: Message franking via committing authenticated encryption. In: CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 66–97
17. Gueron, S., Langley, A., Lindell, Y.: AES-GCM-SIV: Nonce misuse-resistant authenticated encryption, `https://datatracker.ietf.org/doc/html/rfc8452`
18. Gueron, S., Lindell, Y.: Better bounds for block cipher modes of operation via nonce-based key derivation. In: ACM CCS 2017. pp. 1019–1036
19. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 493–517
20. Imamura, K., Minematsu, K., Iwata, T.: Integrity analysis of authenticated encryption based on stream ciphers. Int. J. Inf. Sec. **17**(5), 493–511
21. Iwata, T., Seurin, Y.: Reconsidering the security bound of AES-GCM-SIV. IACR Trans. Symm. Cryptol. **2017**(4), 240–267

22. Jean, J., Nikolic, I., Peyrin, T., Seurin, Y.: The deoxys AEAD family. Journal of Cryptology **34**(3), 31
23. Langley, A.: Chacha20 and poly1305 based cipher suites for tls, https://datatracker.ietf.org/doc/html/draft-agl-tls-chacha20poly1305-00
24. Len, J., Grubbs, P., Ristenpart, T.: Partitioning oracle attacks. pp. 195–212
25. McGrew, D.A., Viega, J.: The security and performance of the Galois/counter mode (GCM) of operation. In: INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355
26. Patarin, J.: The "coefficients H" technique (invited talk). In: SAC 2008. LNCS, vol. 5381, pp. 328–345
27. Peyrin, T., Seurin, Y.: Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In: CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 33–63
28. Procter, G.: A security analysis of the composition of ChaCha20 and Poly1305. Cryptology ePrint Archive, Report 2014/613 (2014), https://eprint.iacr.org/2014/613
29. Rescorla, E.: The transport layer security (TLS) protocol version 1.3, https://www.rfc-editor.org/rfc/rfc8446
30. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM CCS 2002. pp. 98–107
31. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390
32. Yoav Nir, A.L.: Chacha20 and poly1305 for ietf protocols, https://datatracker.ietf.org/doc/html/rfc8439

Supplementary Material

For any n-bit string w, we use $[w]^{K+}$, $[w]^{H+}$, and $[w]^{K,H+}$ to denote the substring of w corresponding to the key portion (the 256-bit key value without the constant), the hash portion, and the substring that contains the key and the hash portion, when w is interpreted as an input to π. Similarly, we use $[w]^{K-}$, $[w]^{H-}$, and $[w]^{K,H-}$ to denote the complementary portion of the string. Thus, for example, $[x]^{H+}$ denotes the hash-coordinate substring of the input x.

A Proof of Lemma 6

We want to bound the probability $\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}]$ that an ideal world transcript τ satisfies one of (1)-(11). Therefore, by the union bound, the probability that $X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}$ can be bounded as

$$\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}] \leq \sum_{i=1}^{11} \Pr[\text{bad}_i]\,.$$

A.1 1st Bad Event

We first consider the bad event bad_1. For any construction MAC query $(\mathsf{con}, j, N, A, M, V) \in \tau_e$ and any primitive queries (prim, x, y), we rely on the randomness of K_j and the ϵ-regularity property of the function H based on the hash key L_j, the probability that $[x]^{H+} = H_j$ is bounded by $\epsilon(\ell)$. Since K_j and L_j are dummy keys generated independently of τ_e and τ_p, the probability that (1) holds for fixed construction MAC query $(\mathsf{con}, j, N, A, M, V)$ and primitive query (prim, x, y) is $\epsilon(\ell)/2^k$. Summing over the p possible primitive queries and d possible construction MAC queries with repeated nonces, we get

$$\Pr[\text{bad}_1] \leq \frac{dp\epsilon(\ell)}{2^k}\,.$$

A.2 2nd Bad Event

We now consider the event bad_2. Let E_2 be the event that there exists a bitstring $w \in \{0,1\}^{n-k-t}$ such that among all construction MAC queries, there are $m_2 = \lceil 2(n-k-t)/\max(1, n-k-t-\log_2 q_e)\rceil$ or more values of $[V]^{K,H-} - (Z\|N)$ that are equal to w, where N is the nonce associated to the MAC query of V and Z is the constant part of $f_m(K_j)$ that is independent of K_j. Then,

$$\begin{aligned}\Pr[\text{bad}_2] &= \Pr[\text{bad}_2 \wedge \overline{E_2}] + \Pr[\text{bad}_2 \wedge E_2] \\ &\leq \Pr[\text{bad}_2 | \overline{E_2}] + \Pr[E_2]\,.\end{aligned}$$

We now bound the probability of E_2, so that we only have to consider the event bad_2 conditioned on $\overline{E_2}$ afterwards. Recall that in the ideal world, V values

are dummy values generated uniformly at random and independently from each other. Every construction MAC query $(\mathsf{con}, j, N, A, M, V)$ can be considered as throwing one ball, $[V]^{K,H-} - (Z\|N)$, uniformly at random into one of 2^{n-k-t} bins. Thus if we consider all construction MAC queries, we throw at most q_e balls uniformly at random into 2^{n-k-t} bins. Using Lemma 2, with $Q = q_e$ and $D = 2^{-(n-k-t)}$, the probability that the heaviest bin contains m_2 or more balls is at most $2^{-(n-k-t)}$. Therefore, the probability of E_2 is bounded by $2^{-(n-k-t)}$. Hence,

$$\Pr[\mathrm{bad}_2] \leq \Pr[\mathrm{bad}_2|\overline{E_2}] + 2^{-(n-k-t)}\,.$$

Note that $\overline{E_2}$ is the event that for all $w \in \{0,1\}^{n-k-t}$, among all MAC queries, there are strictly less than m_2 values of $[V]^{K,H-} - (Z\|N)$ that are equal to w. For any of these m_2 construction MAC query $(\mathsf{con}, j, N, A, M, V) \in \tau_e$ and any primitive queries (prim, x, y), we rely on the randomness of K_j and the ϵ-regularity property of the function H with hash key L_j. In particular, the probability that $H_j = [V]^{H+} - [y]^{H+}$ is bounded by $\epsilon(\ell)$. Since K_j and L_j are dummy keys generated independently of τ_v and τ_p, the probability that (2) holds for a fixed construction MAC query $(\mathsf{con}, j, N, A, M, V)$ and primitive query (prim, x, y) is $\epsilon(\ell)/2^k$. Moreover,

$$m_2 = \left\lceil \frac{2(n-k-t)}{\max(1, n-k-t-\log_2 q_e)} \right\rceil \leq 2 \left\lceil \frac{(n-k-t)}{\max(1, n-k-t-\log_2 2q_e)} \right\rceil \leq 2n\,.$$

Summing over the p possible primitive queries and m_2 possible construction MAC queries, we get

$$\Pr[\mathrm{bad}_2] \leq \frac{2n\,p\epsilon(\ell)}{2^k} + \frac{1}{2^{n-k-t}}\,.$$

A.3 3rd Bad Event

Next, we consider the bad event bad_3. For this event, two cases are possible. The first one is when the two queries are made to the same user, i.e., $j = j^*$, and the second one is when the two queries are made to different users, i.e., $j \neq j^*$.

- bad_{3_1}: case $j = j^*$. In the ideal world, bad_3 for this case could be redefined by the tuple of all transcripts τ that contain two MAC queries $(\mathsf{con}, j, N, M, A, V) \neq (\mathsf{con}, j^*, N^*, M^*, A^*, V^*)$ such that $H_{L_j}(A, M) = H_{L_j}(A^*, M^*)$ and $N = N^*$. To bound the probability of this case, we can use the ϵ-almost Δ-universal property of the function H. However, this property requires that (A, M) and (A^*, M^*) are distinct. However, if $(A, M) = (A^*, M^*)$ then we cannot have that $N = N^*$, as it would result in a repeated query. Thus, the probability of this subcase is zero. If $(A, M) \neq (A^*, M^*)$, then the probability that $H_{L_j}(A, M) = H_{L_j}(A^*, M^*)$ is bounded by $\epsilon(\ell)$ because H is the output of an ϵ-almost Δ-universal hash function with key L_j. For every construction MAC query using a faulty nonce, there are at

most μ other MAC queries using the same nonce. So the number of pairs $(\mathsf{con}, j, N, A, M, V) \neq (\mathsf{con}, j, N^*, A^*, M^*, V^*)$ such that $N = N^*$ is at most μ^2. Hence,

$$\Pr[\mathrm{bad}_{3_1}] \leq \mu^2 \epsilon(\ell)\,. \tag{22}$$

– bad_{3_2}: case $j \neq j^*$. For the first condition, we rely on the ϵ-regularity property of the function H based on the hash key L_j: the probability that $H_j = H_{j^*}$ is bounded by $\epsilon(\ell)$. For the second condition, we rely on the randomness of the key K_j. Since K_j and L_j are dummy keys generated independently of τ_e, the probability that the event holds for fixed $(\mathsf{con}, j, N, A, M, V) \neq (\mathsf{con}, j^*, N^*, A^*, M^*, V^*)$ is $\epsilon(\ell)/2^k$. Summing over q_e possible construction queries $(\mathsf{con}, j, N, A, M, V)$ and $d-1$ possible construction queries $(\mathsf{con}, j^*, N^*, A^*, M^*, V^*)$ such that $N = N^*$, we get

$$\Pr[\mathrm{bad}_{3_2}] \leq \frac{(d-1)q_e\epsilon(\ell)}{2^k}\,. \tag{23}$$

Combining the two cases (22) and (23) using the union bound, we obtain the following result:

$$\Pr[\mathrm{bad}_3] \leq \epsilon(\ell)\left(\mu^2 + \frac{q_e(d-1)}{2^k}\right).$$

A.4 4th Bad Event

Next, we consider the bad event bad_4. Since the values V and V^* are generated uniformly and independent in the ideal world, the probability that the event holds for fixed $(\mathsf{con}, j, N, A, M, V) \neq (\mathsf{con}, j^*, N^*, A^*, M^*, V^*)$ is $1/2^n$. Summing over $\binom{q_e}{2}$ possible choices of construction MAC query pairs, we have

$$\Pr[\mathrm{bad}_4] \leq \frac{q_e^2}{2^{n+1}}\,.$$

A.5 5th Bad Event

This event is very similar to the bad event bad_1. For any construction verification query $(\mathsf{ver}, j', N', A', M', T', 0) \in \tau_v$ and any primitive query (prim, x, y), we rely on the randomness of $K_{j'}$ and the ϵ-regularity property of H with key $L_{j'}$. Specifically, the probability that $[x]^{H+} = H_{L_{j'}}(A', M')$ is bounded by $\epsilon(\ell)$. Since $K_{j'}$ and $L_{j'}$ are dummy keys generated independently of τ_v and τ_p, the probability that (5) holds for a fixed construction verification query $(\mathsf{ver}, j', N', A', M', T', 0)$ and primitive query (prim, x, y) is $\epsilon(\ell)/2^k$. Summing over the p possible primitive queries and q_v possible construction verification queries, we get

$$\Pr[\mathrm{bad}_5] \leq \frac{q_v p \epsilon(\ell)}{2^k}\,.$$

A.6 6th Bad Event

Consider the bad event bad_6. For this event, two cases are possible. The first case is when the MAC and verification queries are made to the same user, i.e., $j = j'$, and the second one is when the two queries are made to different users, i.e., $j \neq j'$.

- bad_{6_1}: case $j = j'$. Note that for a verification query, we can always choose $N = N'$ for any N used in a MAC query. In the ideal world, bad_6 for this case could be redefined by the tuple of all transcripts τ that contain a verification query $(\mathsf{ver}, j, N, A', M', T', 0)$ and a MAC query $(\mathsf{con}, j, N, A, M, V)$ such that $H_j = H'_{j'}$ and $T' = \mathsf{trunc}_t(V)$. To bound the probability of this case, we use the ϵ-almost Δ-universal property of H with hash key L_j. However, this property requires that (A, M) and (A', M') are distinct. Hence, we first consider the case with $(A, M) = (A', M')$.
  Depending on the order of the queries, we consider two subcases when $(A, M) = (A', M')$. The first subcase is when the verification query is made after the encryption query, and the second subcase is for the reverse order. For the first subcase, we cannot have that $\mathsf{trunc}_t(V) = T'$, as it would result in an invalid query. Hence, the probability of this subcase is zero. For the second subcase, when the encryption query is made after the verification query, the probability that $\mathsf{trunc}_t(V) = T'$ is $1/2^t$, as encryption queries return uniform random strings. For each of the q_v verification queries $(\mathsf{ver}, j, N, A', M', T', 0)$, there can be at most one MAC query with the same (j, N) and answer $T = \mathsf{trunc}_t(V)$ when $(A, M) = (A', M')$, and the probability that $T = T'$ is $1/2^t$. Hence, the probability of this subcase is bounded by $q_v/2^t$.
  Let $\text{bad}^*_{6_1}$ be the subcase when $(A, M) \neq (A', M')$. Let E_6 be the event that there exist a $w \in \{0,1\}^t$ such that among all construction MAC queries *with the same nonce input*, there are $m_6 = \lceil 2t/\max(1, t - \log_2(\mu + 1)) \rceil$ or more values of $\mathsf{trunc}_t(V)$ that are equal to w. It is important to consider queries that use the same nonce in order to maximize the number of queries where $\mathsf{trunc}_t(V)$ takes the same value, thereby increasing the probability of the desired event. Since the distinguisher can choose its queries arbitrarily, we assume the optimal case in which it selects as many identical nonces as possible to maximize its success probability. Then,

$$\begin{aligned}\Pr[\text{bad}^*_{6_1}] &= \Pr[\text{bad}^*_{6_1} \wedge \overline{E_6}] + \Pr[\text{bad}^*_{6_1} \wedge E_6] \\ &\leq \Pr[\text{bad}^*_{6_1} | \overline{E_6}] + \Pr[E_6]\,.\end{aligned}$$

  We first bound the probability of E_6, so that we only have to consider the event bad_6 conditioned on $\overline{E_6}$ afterwards. Recall that in the ideal world, the values $\mathsf{trunc}_t(V)$ are actually the tag values output by the oracles $\$, \ldots, \$_u$. We can consider every construction MAC query $(\mathsf{con}, j, N, A, M, V)$ with the same (j, N), as throwing one ball $\mathsf{trunc}_t(V)$, uniformly at random into one of 2^t bins. Hence, if we consider all faulty construction MAC queries for one user, we throw at most $\mu + 1$ balls (the first query with nonce N

is not considered as faulty, and there can be at most μ additional repeated nonces.) uniformly at random into 2^t bins. Using Lemma 2, with $Q = \mu + 1$ and $D = 2^t$, the probability that the heaviest bin contains m_6 or more balls is at most 2^{-t}. Therefore, the probability of E_6 is bounded by 2^{-t}. Hence,

$$\Pr[\mathrm{bad}_{6_1}^*] \leq \Pr[\mathrm{bad}_{6_1}^* | \overline{E_6}] + 2^{-t}\,.$$

Note that $\overline{E_6}$ is the event that for all $w \in \{0,1\}^t$, among all MAC queries, there are strictly less than m_6 values of $\mathsf{trunc}_t(V)$ that are equal to w. For any of these m_6 construction MAC query $(\mathsf{con}, j, N, A, M, V) \in \tau_e$ and q_v verification queries $(\mathsf{ver}, j, N, A', M', T', 0)$, the probability that $H_{L_j}(A, M) = H_{L_j}(A', M')$ for a fixed MAC query $(\mathsf{con}, j, N, A, M, V)$ and verification query $(\mathsf{ver}, j, N, A', M', T', 0)$ is bounded by $\epsilon(\ell)$. Moreover,

$$m_6 = \left\lceil \frac{2t}{\max(1, t - \log_2(\mu+1))} \right\rceil \leq 2 \left\lceil \frac{t}{\max(1, t - \log_2(2\mu+2))} \right\rceil \leq 4\,,$$

if $\mu + 1 \leq 2^{t/2-1}$. Summing over the q_v possible verification queries and m_6 possible MAC queries, we get

$$\Pr[\mathrm{bad}_{6_1}^*] \leq 4q_v\epsilon(\ell) + \frac{1}{2^t}\,.$$

Combining the two subcases using a union bound, we obtain

$$\Pr[\mathrm{bad}_{6_1}] \leq \frac{q_v}{2^t} + 4q_v\epsilon(\ell) + \frac{1}{2^t}\,. \tag{24}$$

– bad_{6_2}: case $j \neq j'$. Using the same approach as for bad_{6_1}, but with $Q = d$ and $D = 2^t$, we have

$$m_6 = \left\lceil \frac{2t}{\max(1, t - \log_2 d)} \right\rceil \leq 2 \left\lceil \frac{t}{\max(1, t - \log_2 2d)} \right\rceil \leq 2n\,.$$

For the first condition, we rely on the ϵ-regularity property of the function H with hash key L_j, i.e. the probability that $H_j = H'_{j'}$ is bounded by $\epsilon(\ell)$. For the third condition, we rely on the randomness of the key K_j. Since K_j and L_j are dummy keys generated independently of τ_e and τ_v, the probability that the event holds for a fixed construction MAC query $(\mathsf{con}, j, N, A, M, V)$ and construction verification query $(\mathsf{ver}, j', N, A', M', T', 0)$ is $\epsilon(\ell)/2^k$. Summing over q_v possible construction verification queries and m_6 possible construction MAC queries with repeated (N, T) pairs, we get

$$\Pr[\mathrm{bad}_{6_2}] \leq \frac{2n\, q_v\epsilon(\ell)}{2^k} + \frac{1}{2^t}\,. \tag{25}$$

Combining the two cases (24) and (25) using the union bound, we obtain the following result:

$$\Pr[\mathrm{bad}_6] \leq \frac{q_v}{2^t} + 2q_v\epsilon(\ell)\left(2 + \frac{n}{2^k}\right) + \frac{2}{2^t}\,.$$

A.7 7th Bad Event

This event is very similar to bad events bad_1 and bad_5. For all key pairs $(K, U) \in \tau_k$ and all primitive queries (prim, x, y), we rely on the randomness of K_j. Since K_j is a dummy key generated independently of τ_k and τ_p, the probability that (7) holds for fixed key pair (K, U) and primitive query (prim, x, y) is $1/2^k$. Summing over p possible primitive queries and u possible key pairs, we get

$$\Pr[\text{bad}_7] \leq \frac{up}{2^k} .$$

A.8 8th Bad Event

The analysis of this event is similar to that of bad event bad_2. Let E_8 be the event that there exists a $w \in \{0,1\}^{n-k}$ such that among all users, there are $m_8 = \lceil 2(n-k)/\max(1, n-k-\log_2 u)\rceil$ or more values of $[U]^{K-}$ that are equal to w. Then,

$$\begin{aligned}\Pr[\text{bad}_8] &= \Pr[\text{bad}_8 \wedge \overline{E_8}] + \Pr[\text{bad}_8 \wedge E_8] \\ &\leq \Pr[\text{bad}_8 | \overline{E_8}] + \Pr[E_8] .\end{aligned}$$

We first bound the probability of E_8, so that we only have to consider the event bad_8 conditioned on $\overline{E_8}$ afterwards. Recall that in the ideal world, the values U are dummies generated uniformly at random and independent from each other. We can consider each user key pair (K, U) as throwing a ball $[U]^{K-}$ uniformly at random into one of 2^{n-k} bins. Hence, if we consider all key pairs in τ_k, we throw at most u balls uniformly at random into 2^{n-k} bins. Using Lemma 2, with $Q = u$ and $D = 2^{n-k}$, the probability that the heaviest bin contains m_8 or more balls is at most $2^{-(n-k)}$. Therefore, the probability of E_8 is bounded by $2^{-(n-k)}$. Hence,

$$\Pr[\text{bad}_8] \leq \Pr[\text{bad}_8 | \overline{E_8}] + 2^{-(n-k)} .$$

Note that $\overline{E_8}$ is the event that for all $w \in \{0,1\}^{n-k}$, among all key pairs in τ_k, there are strictly less than m_8 values of $[U]^{K-}$ that are equal to w. For any of these m_8 key pairs $(K, U) \in \tau_k$ and any primitive queries (prim, x, y), we rely on the randomness of K. Since K is a dummy key generated independently of τ_k and τ_p, the probability that (8) holds for key pair (K, U) and primitive query (prim, x, y) is $1/2^k$. Moreover,

$$m_8 = \left\lceil \frac{2(n-k)}{\max(1, n-k-\log_2 u)} \right\rceil \leq 2 \left\lceil \frac{n-k}{\max(1, n-k-\log_2(2u))} \right\rceil \leq 4 ,$$

for $u \leq 2^{(n-k)/2-1}$. Summing over the p possible primitive queries and m_8 possible key pairs in τ_k, we get

$$\Pr[\text{bad}_8] \leq \frac{4p}{2^k} + \frac{1}{2^{n-k}} .$$

A.9 9th Bad Event

Since the values V and U are generated uniformly at random and independent in the ideal world, the probability that the event holds for fixed construction MAC query $(\mathsf{con}, j, N, A, M, V)$ and key pair (K, U) is $1/2^n$. Summing over q_e possible choices of construction MAC query and u possible choices of key pairs, we have

$$\Pr[\text{bad}_9] \leq \frac{uq_e}{2^n}\,.$$

A.10 10th Bad Event

Since the keys K and K^* are dummy keys generated uniformly and independent in the ideal world, the probability that the event holds for fixed $(K, U) \neq (K^*, U^*)$ is $1/2^k$. Summing over $\binom{u}{2}$ possible choices of user pairs, we have

$$\Pr[\text{bad}_{10}] \leq \frac{u(u-1)}{2^{k+1}}\,.$$

A.11 11th Bad Event

Since the values U and U^* are generated uniformly and independent in the ideal world, the probability that the event holds for fixed $(K, U) \neq (K^*, U^*)$ is $1/2^n$. Summing over $\binom{u}{2}$ possible choices of key pairs in τ_k, we have

$$\Pr[\text{bad}_{11}] \leq \frac{u(u-1)}{2^{n+1}}\,.$$

B Proof of Lemma 7

We want to bound the probability $\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}]$ that an ideal world transcript τ satisfies either of (15)-(18). Therefore, by the union bound, the probability that $X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}$ can be bounded as

$$\Pr[X_{\mathcal{P}} \in \mathcal{T}_{\text{bad}}] \leq \sum_{i=1}^{4} \Pr[\text{bad}_i]\,.$$

B.1 1st Bad Event

We first consider the bad event bad_1. For any $w \in \{0,1\}^t$, we determine the maximum number of occurrences where $T + l - 1 = w$ across all construction queries *sharing the same nonce input*, for $l = 1, \ldots, \ell_m$. Recall that the T values are independent and uniformly distributed. Thus, for every construction query $(\mathsf{con}, j, N, T, M, C, W)$ with the same nonce, the remaining $\ell_m - 1$ blocks $T+l-1$ of this query (for $l = 1, \ldots, \ell_m - 1$) will always be *distinct*. For all $w \in \{0,1\}^t$, among all construction queries with the same nonce input, there are strictly no

more than d values of $T+l-1$ that are equal to w. For any of these d construction query $(\mathsf{con}, j, N, T, M, C, V) \in \tau_e$ and any primitive queries (prim, x, y), the event bad_1 happens if the key is sampled as $K_j = [x]^{K+}$. Here, we rely on the randomness of K_j. Since K_j is a dummy key generated independently of τ_e and τ_p, the probability that (16) holds for fixed construction query $(\mathsf{con}, j, N, T, M, C, V)$ and primitive query (prim, x, y) is $1/2^k$. Summing over p primitive queries and d possible construction queries, we get

$$\Pr[\mathrm{bad}_1] \leq \frac{dp}{2^k}\,.$$

B.2 2nd Bad Event

Let E_2 be the event that there exists a $w \in \{0,1\}^{n-k}$ such that among all construction queries, there are $m_2 = \lceil 2(n-k)/\max(1, n-k-\log_2(\sigma))\rceil$ or more values of $[W[l]]^{K-} - (Z\|N\|T+l-1)$ that are equal to w, for $l = 1, \ldots, \ell_m$, where N and T are respectively the nonce and the tag associated to the encryption query of V, and Z is the constant part of $f_e(K_j)$ which is independent of K_j. Then,

$$\begin{aligned}\Pr[\mathrm{bad}_2] &= \Pr[\mathrm{bad}_2 \wedge \overline{E_2}] + \Pr[\mathrm{bad}_2 \wedge E_2] \\ &\leq \Pr[\mathrm{bad}_2 | \overline{E_2}] + \Pr[E_2]\,.\end{aligned}$$

We first bound the probability of E_2, so that we only have to consider the event bad_2 conditioned on $\overline{E_2}$ afterwards. Recall that in the ideal world, we have $W = M \oplus C$, hence W values are independent and uniformly distributed. We can view each construction query $(\mathsf{con}, j, N, T, M, C, W[1]\| \ldots \|W[\ell_m])$, as throwing ℓ_m balls $([W[1]]^{K-} - (Z\|N\|T)), \ldots, ([W[\ell_m]]^{K-} - (Z\|N\|T+\ell_m-1))$, uniformly at random into 2^{n-k} bins. Hence, if we consider all construction queries, we throw at most σ balls uniformly at random into 2^{n-k} bins. Using Lemma 2, with $Q = \sigma$ and $D = 2^{n-k}$, the probability that the heaviest bin contains m_2 or more balls is at most $2^{-(n-k)}$. Therefore, the probability of E_2 is bounded by $2^{-(n-k)}$. Hence,

$$\Pr[\mathrm{bad}_2] \leq \Pr[\mathrm{bad}_2 | \overline{E_2}] + 2^{-(n-k)}\,.$$

Note that $\overline{E_2}$ is the event that for all $w \in \{0,1\}^{n-k}$, among all construction queries, there are strictly less than m_2 values of $[W[l]]^{K-} - (Z\|N\|T+l-1)$ that are equal to w, for $l = 1, \ldots, \ell_m$. For any of these m_2 construction query $(\mathsf{con}, j, N, T, M, C, W[1]\| \ldots \|W[\ell_m]) \in \tau_e$ and any primitive queries (prim, x, y), the event bad_2 happens if the key is sampled as $K_j = [W[l]]^{K+} - [y]^{K+}$. Since K_j is a dummy key generated independently of τ_e and τ_p, the probability that (16) holds for fixed construction query $(\mathsf{con}, j, N, T, M, C, W[1]\| \ldots \|W[\ell_m])$ and primitive query (prim, x, y) is $1/2^k$. Moreover,

$$m_2 = \left\lceil \frac{2(n-k)}{\max(1, n-k-\log_2(\sigma))} \right\rceil \leq 2 \left\lceil \frac{n-k}{\max(1, n-k-\log_2(2\sigma))} \right\rceil \leq 4\,,$$

using $\sigma \leq 2^{(n-k)/2-1}$. Summing over p possible primitive queries and m_2 possible construction queries, we get

$$\Pr[\mathrm{bad}_2] \leq \frac{8p}{2^k} + \frac{1}{2^{n-k}}\,.$$

B.3 3rd Bad Event

Next, we consider the bad event bad_3. For this event, two cases are possible. The first one is when the two queries are made to the same user, i.e., $j = j^*$, and the second one is when the two queries are made to different users, i.e., $j \neq j^*$.

- bad_{3_1}: case $j = j^*$. In the ideal world, bad_3 for this case could be redefined by the tuple of all transcripts τ that contain two construction queries $(\mathsf{con}, j, N, T, M, C, W) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ such that $T = T^* + l^* - 1$ and $N = N^*$ for $l^* = 1, \ldots, \ell_m$. Since the input tags are generated uniformly at random, the probability that $T = T^* + l^* - 1$ for fixed $(\mathsf{con}, j, N, T, M, C, W) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ is bounded by $1/2^t$. For every construction query using a faulty nonce, there are at most μ other construction queries using the same nonce. Hence, the number of construction query pairs such that $N = N^*$ is at most μ^2 and since there are ℓ_m possible l^*, we get
$$\Pr[\mathrm{bad}_{3_1}] \leq \frac{\mu^2 \ell_m}{2^t}\,. \tag{26}$$
- bad_{3_2}: case $j \neq j^*$. For the first condition $T = T^* + l^* - 1$, since the input tags are generated uniformly at random, the probability that $T = T^* + l^* - 1$ for fixed $(\mathsf{con}, j, N, T, M, C, W) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ and $l^* = 1, \ldots, \ell_m$ is bounded by $1/2^t$. For the second condition, we rely on the randomness of the uniform key K. The probability that the event holds for fixed $(\mathsf{con}, j, N, T, M, C, W) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ is $1/2^{k+t}$. Summed over q_e possible construction queries $(\mathsf{con}, j, N, T, M, C, W)$, $d-1$ possible construction queries $(\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*)$ such that $N = N^*$, and ℓ_m possible l^*, we get
$$\Pr[\mathrm{bad}_{3_2}] \leq \frac{q_e(d-1)\ell_m}{2^{k+t}}\,. \tag{27}$$

Combining the two cases (22) and (27) using the union bound, we obtain the following result:

$$\Pr[\mathrm{bad}_3] \leq \frac{\ell_m}{2^t}\left(\mu^2 + \frac{q_e(d-1)}{2^k}\right).$$

B.4 4th Bad Event

Finally, we consider the bad event bad_4. Since the values $W[l]$ and $W^*[l^*]$ are generated uniformly and independent in the ideal world, the probability that the event holds for fixed $(\mathsf{con}, j, N, T, M, C, W[1] \| \ldots \| W[\ell_m]) \neq (\mathsf{con}, j^*, N^*, T^*, M^*,$

$C^*, W^*[1] \| \ldots \| W^*[\ell_m])$ is $1/2^n$. Summing over $\binom{\sigma}{2}$ possible choices of construction query pairs, we have

$$\Pr[\text{bad}_4] \leq \frac{\sigma^2}{2^{n+1}} .$$

C Security Proof of Lemma 5

In this section we prove Lemma 5. We reuse the notations introduced in Sections 5.1 and 5.2. Let $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$ and $\pi \leftarrow\$ \text{Perm}(n)$. Let $\mathsf{Rand}_1, \ldots, \mathsf{Rand}_u$ be functions that for each input (N, A, M) return a random string of size $|M| + t$ bits. As explained in Section 2.3, we consider a computationally unbounded and deterministic distinguisher $\mathcal{D}$ that has access to either the real world oracles $\mathcal{O} = ((\mathsf{Enc}_{K_j}, \mathsf{Dec}_{K_j})_{j=1}^u, \pi^{\pm})$, or the ideal world oracles $\mathcal{P} = ((\mathsf{Rand}_j, \mathsf{Rej}^*)_{j=1}^u, \pi^{\pm})$. The distinguisher makes p primitive queries to $\pi^{\pm}$.

Apart from the primitive queries, $\mathcal{D}$ also makes q_e construction encryption queries to $(\mathsf{Enc}_{K_j})_{j=1}^u$ or $(\mathsf{Rand}_j)_{j=1}^u$, each query with a message length of less than ℓ_m blocks, totaling at most σ encrypted blocks. It makes at most μ faulty queries for the same user and in total at most d queries per nonce. Enc uses two subprocedures: the MAC function F and the encryption function E of $\mathit{\Pi}$, therefore we will also denote the encryption oracle of j-th user as $\mathsf{Enc}_{K_j}[F, \mathsf{E}]$, for each user $j \in \{1, \ldots, u\}$. Then, $\mathsf{Enc}_{K_j}[F, \mathsf{E}]$ works as follows for $N_i \in \{0,1\}^b$, $A_i \in \{0,1\}^*$ and $M_i \in \{0,1\}^*$:

$$\mathsf{Enc}_{K_{j_i}}[F, \mathsf{E}](N_i, A_i, M_i) = (C_i, T_i)$$
$$\text{where } T_i = F_{K_{j_i}}(N_i, A_i, M_i) \text{ and } C_i = \mathsf{E}_{K_{j_i}}(N_i, T_i, M_i) ,$$

for $i = 1, \ldots, q_e$. The distinguisher also gets access to the values $V_1, \ldots, V_{q_e}$, where the first t bits of these values are given as the output tags of each query, and the remaining $n - t$ bits of these q_e values are revealed at the end of the interaction, but before $\mathcal{D}$ makes its final decision. As before, in the real world, $V_i = \varphi_m(K_{j_i} \| N_i \| H_{j_i})$. In the ideal world, $(V_1, \ldots, V_{q_e})$ are dummy values, where the first t bits are returned by oracles $\$_1, \ldots, \$_u$, and the remaining $n - t$ bits are generated uniformly at random at the end of the interaction.

The distinguisher $\mathcal{D}$ also makes q_v decryption queries to $(\mathsf{Dec}_{K_j})_{j=1}^u$ or $(\mathsf{Rej}^*)_{j=1}^u$. In the real world, Dec uses two subprocedures: the verification function Ver of the MAC F and the decryption function D of $\mathit{\Pi}$. Hence, we will denote the decryption oracle of jth user as $\mathsf{Dec}_{K_j}[\text{Ver}, \mathsf{D}]$. Then, $\mathsf{Dec}_{K_j}[F, \mathsf{E}]$ works as follows for $N'_i \in \{0,1\}^b$, $A'_i \in \{0,1\}^*$ and $C'_i \in \{0,1\}^*, T'_i \in \{0,1\}^t$:

$$\mathsf{Dec}_{K_{j'_i}}[\text{Ver}, \mathsf{D}](N'_i, T'_i, A'_i, C'_i) = \begin{cases} M'_i & \text{if } \text{Ver}_{K_{j'_i}}(N'_i, A'_i, M'_i, T'_i) = 1 \\ \perp & \text{if } \text{Ver}_{K_{j'_i}}(N'_i, A'_i, M'_i, T'_i) = 0 \end{cases}$$

where $M'_i = \mathsf{D}_{K_{j'_i}}(N'_i, T'_i, C'_i)$, for $i = 1, \ldots, q_v$. Note that we also grant the adversary the decrypted message M'_i in the transcript.

The ideal world Rej^* creates a permutation $\tilde{\pi}_{j'_i}$ such that, $\tilde{\pi}_{j'_i}(\cdot)$ is sampled uniformly random from $\text{Perm}(n)$, subject to the constraint that we must have

$\tilde{\pi}_{j'_i}(x) = y$ for any primitive queries (prim, x, y), and that it must satisfy the input-output consistency of all $(\mathsf{Rand}_j)_{j=1}^u$ functions. For each decryption query to Rej^*, we run $M' = \mathsf{D}[\tilde{\pi}_{j'_i}](N'_i, T'_i, C'_i)$, and grant the adversary the decrypted message M' as above. Here $\mathsf{D}[\tilde{\pi}_{j'_i}]$ uses $\tilde{\pi}_{j'_i}$ as the underlying primitive. Without loss of generality, it can be assumed that the distinguisher does not make duplicate construction or primitive queries.

After $\mathcal{D}$'s interaction with the oracles, but before it outputs its decision, we disclose the key material used in the interaction. The first values that are revealed are values $K_1, \ldots, K_u$. In the real world, these are the keys $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$ that are actually used by the construction oracles (encryption and decryption); in the ideal world, these are dummy keys $K_1, \ldots, K_u \leftarrow\$ \{0,1\}^k$. The second values that are revealed are values $U_1, \ldots, U_u \in \{0,1\}^n$, where the first t bits of these values are the universal hash functions keys $L_1, \ldots, L_u$. In the real world, these are the values $U_j = \varphi_k(K_j)$ for $j = 1, \ldots, u$. In the ideal world, these are dummy values $U_1, \ldots, U_u \leftarrow\$ \{0,1\}^n$. All queries and revealed keys are summarized in a transcript τ.

Now, from such a transcript τ, we can extract a transcript $\mathcal{R}_m(\tau)$ for F (as defined in Section 5.1), and another transcript $\mathcal{R}_e(\tau)$ for Π (as defined in Section 5.2) as follows. The transcript $\mathcal{R}_m(\tau)$ consists of the revealed keys, all primitive queries in τ, and for every encryption (respectively decryption) query in τ, we store a query $(\mathsf{con}, j, N, A, M, V)$ (respectively $(\mathsf{ver}, j', N', A', M', T', b')$) in $\mathcal{R}_m(\tau)$. The transcript $\mathcal{R}_e(\tau)$ consists of the keys $K_1, \ldots, K_u$, all primitive queries in τ, and for every encryption query of τ a corresponding query $(\mathsf{con}, j, N, T, M, C, W)$ in $\mathcal{R}_e(\tau)$ with $W = M \oplus C$.

For a MAC query $(\mathsf{con}, j, N, A, M, V) \in \mathcal{R}_m(\tau)$, a key pair $(K, U) \in \mathcal{R}_m(\tau)$, and a construction encryption query $(\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*) \in \mathcal{R}_e(\tau)$, we define the following bad events:

$$\begin{aligned} \mathrm{bad}_1 \colon V - (f_m(K_j) \| N \| H_j) &= W^*[l^*] - (f_e(K_{j^*}) \| N^* \| T^* + l^* - 1) \\ &\text{for } l^* = 1, \ldots, \ell_m\,, \end{aligned} \tag{28}$$

$$\begin{aligned} \mathrm{bad}_2 \colon U - (f_k(K) \| 00 \cdots 0) &= W^*[l^*] - (f_e(K_{j^*}) \| N^* \| T^* + l^* - 1) \\ &\text{for } l^* = 1, \ldots, \ell_m\,. \end{aligned} \tag{29}$$

Note that because the same master key is used for both the encryption and MAC components of the mode, domain separation is required to ensure that the permutation inputs and outputs do not collide between a MAC query and an encryption query. However, this requirement does not apply in the ideal world. Hence, the two bad events mentioned above are essential to ensuring that the outputs do not collide in the ideal world. There is no separate bad event for a MAC-query or verification-query input colliding with an input from a key pair (K, U), because the outputs of f_k, f_m, and f_e are domain-separated.

Furthermore, we define $\mathsf{BAD} = \mathrm{bad}_1 \vee \mathrm{bad}_2$. Next, consider three worlds World_i for $i = 1, 2, 3$, defined as follows:

$$\begin{aligned}
\mathrm{World}_1 &= ((\mathsf{Enc}_{K_j}[F, \mathsf{E}], \mathsf{Dec}_{K_j}[\mathrm{Ver}, \mathsf{D}])_{j=1}^{u}, \pi^{\pm}) \,, \\
\mathrm{World}_2 &= ((\mathsf{Enc}_{K_j}[\$_j, \mathsf{E}], \mathsf{Dec}_{K_j}[\mathsf{Rej}, \mathsf{D}])_{j=1}^{u}, \pi^{\pm}) \,, \\
\mathrm{World}_3 &= ((\mathsf{Rand}_j, \mathsf{Rej})_{j=1}^{u}, \pi^{\pm}) \,,
\end{aligned}$$

where $\$_1, \ldots, \$_u$ are oracles that return uniform random outputs. Note that F and E are based on an ideal permutation π, hence domain separation is guaranteed by functions φ_k, φ_m and φ_e.

By the triangle inequality, the advantage of an adversary $\mathcal{D}$ against the mu-ae security of PSIV satisfies

$$\begin{aligned}
\mathbf{Adv}^{\text{mu-ae}}_{\mathsf{PSIV}}(\mathcal{D}) &= \Delta_{\mathcal{D}}\left(\mathrm{World}_1; \mathrm{World}_3\right) \\
&= \left|\Pr[\mathcal{D}^{\mathrm{World}_1} = 1] - \Pr[\mathcal{D}^{\mathrm{World}_3} = 1]\right| \\
&\le \left|\Pr[\mathcal{D}^{\mathrm{World}_1} = 1 \wedge \neg\mathsf{BAD}] - \Pr[\mathcal{D}^{\mathrm{World}_3} = 1 \wedge \neg\mathsf{BAD}]\right| \\
&\quad + \left|\Pr[\mathcal{D}^{\mathrm{World}_1} = 1 \wedge \mathsf{BAD}] - \Pr[\mathcal{D}^{\mathrm{World}_3} = 1 \wedge \mathsf{BAD}]\right| \\
&\le \left|\Pr[\mathcal{D}^{\mathrm{World}_1} = 1 \wedge \neg\mathsf{BAD}] - \Pr[\mathcal{D}^{\mathrm{World}_3} = 1 \wedge \neg\mathsf{BAD}]\right| \\
&\quad + \Pr[\mathrm{World}_3 \text{ sets } \mathsf{BAD}] \,. \qquad (30)
\end{aligned}$$

Note that for World_1, the inputs to the permutation calls made by F and those made by Π never collide due to the domain separation provided by f_k, f_m and f_e. Therefore, the event BAD never happens in World_1.

Denoting the first term in (30) by $\Delta^{\neg\mathsf{BAD}}_{\mathcal{D}}\left(\mathrm{World}_1; \mathrm{World}_3\right)$ for brevity, the triangle inequality shows that

$$\begin{aligned}
&\Delta^{\neg\mathsf{BAD}}_{\mathcal{D}}\left(\mathrm{World}_1; \mathrm{World}_3\right) \le \\
&\qquad \Delta^{\neg\mathsf{BAD}}_{\mathcal{D}}\left(\mathrm{World}_1; \mathrm{World}_2\right) + \Delta^{\neg\mathsf{BAD}}_{\mathcal{D}}\left(\mathrm{World}_2; \mathrm{World}_3\right) \,. \qquad (31)
\end{aligned}$$

Since World_1 and World_2 use (F, Ver) resp. $(\$_j, \mathsf{Rej})$ as subprocedures in their construction oracles, we have

$$\Delta^{\neg\mathsf{BAD}}_{\mathcal{D}}\left(\mathrm{World}_1; \mathrm{World}_2\right) \le \mathbf{Adv}^{\text{mu-mac}}_{F}(\mathcal{D}') \,, \qquad (32)$$

for an adversary $\mathcal{D}'$ against the multi-user mac security of F that makes q_e encryption queries with at most μ faulty nonces for the same user and in total at most d queries per nonce, q_v construction verification queries (in total to all u construction oracles), and p primitive queries in order to simulate $\mathcal{D}$'s oracles.

In both World_2 and World_3, the *decryption* construction oracles behave in the same way, always returning zero. Hence, without loss of generality, we can assume that $\mathcal{D}$ makes queries only to the encryption oracles and the primitive $\pi^{\pm}$. We can then construct an adversary $\mathcal{D}''$ against the multi-user ivE security of Π that makes q_e construction encryption queries (in total to all users), each query with a message length at most ℓ_m blocks, totaling at most σ encrypted

blocks, for at most μ faulty queries per user and in total at most d queries per nonce, and p primitive queries in order to simulate $\mathcal{D}$'s oracles. Then, we have

$$\Delta_{\mathcal{D}}^{\neg\mathsf{BAD}}\left(\mathrm{World}_2;\mathrm{World}_3\right) \leq \mathbf{Adv}_{\Pi}^{\text{mu-nivE}}(\mathcal{D}'')\,. \tag{33}$$

It follows from (31), (32) and (33) that

$$\Delta_{\mathcal{D}}^{\neg\mathsf{BAD}}\left(\mathrm{World}_1;\mathrm{World}_3\right) \leq \mathbf{Adv}_{F}^{\text{mu-mac}}(\mathcal{D}') + \mathbf{Adv}_{\Pi}^{\text{mu-nivE}}(\mathcal{D}'')\,.$$

Now consider the probability that World_3 sets BAD in (30). We prove that

$$\Pr\left[\mathrm{World}_3 \text{ sets } \mathsf{BAD}\right] \leq \frac{\sigma(q_e+u)}{2^n}\,. \tag{34}$$

Using the union bound, we get

$$\Pr\left[\mathrm{World}_3 \text{ sets } \mathsf{BAD}\right] \leq \Pr[\mathrm{bad}_1] + \Pr[\mathrm{bad}_2]\,.$$

1st Bad Event. Since the values V and $W^*[l^*]$ are generated uniformly and independent in the ideal world, the probability that the event holds for fixed MAC query $(\mathsf{con}, j, N, A, M, V) \in \mathcal{R}_m(\tau)$ and encryption query $(\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*[1]\|\cdots\|W^*[\ell_m]) \in \mathcal{R}_e(\tau)$ is $1/2^n$. Summing over q_e possible choices of MAC queries and σ possible choices of encryption queries, we have

$$\Pr[\mathrm{bad}_1] \leq \frac{q_e\sigma}{2^n}\,.$$

2nd Bad Event. Similarly, since the values U and $W^*[l^*]$ are generated uniformly and independent in the ideal world, the probability that the event holds for a fixed key pair $(K, U) \in \mathcal{R}_m(\tau)$ and encryption query $(\mathsf{con}, j^*, N^*, T^*, M^*, C^*, W^*[1]\|\ldots\|W^*[\ell_m]) \in \mathcal{R}_e(\tau)$ is $1/2^n$. Summing over u possible choices of user key pairs and σ possible choices of encryption queries, we have

$$\Pr[\mathrm{bad}_2] \leq \frac{u\sigma}{2^n}\,.$$

Summing over the probabilities of the bad events completes the proof of (34). Combining this result with (30) and (31) completes the proof.

D Analysis in the Random Nonce Case

When a random nonce is used, the value d in Theorem 1 is bounded by the maximum number of nonce collisions. The expected number of collisions is $q_e(q_e-1)/2^{b-1}$. By Markov's inequality, the probability that the number of collisions exceeds $2^{t/2-1}$ is at most $4q_e(q_e-1)/2^{b+t/2}$. Hence, the terms with d in Theorem 1 can be upper bounded by

$$\epsilon(\ell)\frac{d(q_e+p)}{2^k} + \frac{q_e(d-1)\ell_m}{2^{k+t}} \leq \epsilon(\ell)\frac{q_e+p}{2^{k-t/2+1}} + \frac{q_e\ell_m}{2^{k+t/2+1}}\,.$$

For terms with μ, we bound the related bad events again with random nonces. In particular, it follows from the analysis below that the terms related to μ can be replaced by

$$\frac{q_e^2\epsilon(\ell)}{2^{b+1}} + \frac{q_e^2\ell_m}{2^{t+b+1}}\,.$$

Only bad_{3_1} and bad_{6_1} in Supplementary Material A, and bad_{3_1} in Supplementary Material B are related to μ. The bound for bad_{6_1} in Supplementary Material A, is independent of μ apart from the condition $\mu + 1 \leq 2^{t/2-1}$. However, due to $\mu \leq d < 2^{t/2-1}$, this condition is always satisfied. Hence, we only have to consider the events bad_{3_1} from Supplementary Material A and B.

D.1 3rd Bad Event of Supplementary Material A

Only the case $j = j^*$ has to be reconsidered. In the ideal world, bad_3 for this case could be redefined by the tuple of all transcripts τ that contain two MAC queries $(\mathsf{con}, j, N, M, A, V) \neq (\mathsf{con}, j^*, N^*, M^*, A^*, V^*)$ such that $H_{L_j}(A, M) = H_{L_j}(A^*, M^*)$ and $N = N^*$. To bound the probability of this case, we rely on the ϵ-almost Δ-universal property of the function H, and the fact that the nonces are generated uniformly at random. Note that when the nonce collides with one of the previous nonces, we cannot have $(A, M) = (A^*, M^*)$, otherwise this will be a repeated query. When $(A, M) \neq (A^*, M^*)$ and H is an ϵ-almost Δ-universal hash function based on the hash key L_j, the probability that $H_{L_j}(A, M) = H_{L_j}(A^*, M^*)$ is bounded by $\epsilon(\ell)$. Summing over $\binom{q_e}{2}$ possible pairs of distinct MAC queries, we get

$$\Pr[\mathrm{bad}_{3_1}] \leq \frac{q_e^2\epsilon(\ell)}{2^{b+1}}\,.$$

D.2 3rd Bad Event of Supplementary Material B

Consider the case $j = j^*$. In the ideal world, bad_3 for this case could be redefined as the tuple of all transcripts τ that contain two construction queries $(\mathsf{con}, j, N, T, M, C, V) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, V^*)$ such that $T = T^* + l^*$ and $N = N^*$ for $l^* = 0, \ldots, \ell_m - 1$. Since the input tags and nonces are both generated uniformly at random, the probability that $T = T^* + l^*$ and $N = N^*$ for fixed $(\mathsf{con}, j, N, T, M, C, V) \neq (\mathsf{con}, j^*, N^*, T^*, M^*, C^*, V^*)$ is bounded by $1/2^{t+b}$. Summing over $\binom{q_e}{2}$ possible pair of distinct construction queries and at most ℓ_m possible l^*, we get

$$\Pr[\mathrm{bad}_{3_1}] \leq \frac{q_e^2\ell_m}{2^{t+b+1}}\,.$$

E Additional Benchmarks

Additional benchmarking results on an AMD Ryzen 7 5700U and on an Intel i7 10510U processor are given in Tables 4 and 5. In both cases, Turbo Boost Technology was turned off and Libsodium was compiled with the default options.

Table 4: Performance of AES256-GCM, ChaCha20-Poly1305 and ChaCha20-Poly1305-PSIV (ours) in MiB/s. The measurements were performed on an AMD Ryzen 7 5700U processor at 1.796 Ghz with Turbo Boost Technology disabled. The median value is given.

(a) Encryption speed in MiB/s.

A (bytes)	M (bytes)	AES-256-GCM	ChaCha20-Poly1305	Ours
2	2	6.561	3.142	2.736
	32	101.0	41.83	42.59
	512	700.8	362.1	354.9
	8192	880.0	591.4	590.4
	16384	886.0	605.5	604.7
32	2	6.735	2.966	2.919
	32	104.8	41.01	42.82
	512	708.4	360.5	366.8
	8192	880.8	590.5	593.5
	16384	886.0	605.6	604.9
512	2	4.109	1.936	1.871
	32	64.26	28.19	28.71
	512	562.0	287.0	291.9
	8192	861.5	577.0	575.9
	16384	877.0	597.1	597.6

(b) Decryption speed in MiB/s.

A (bytes)	M (bytes)	AES-256-GCM	ChaCha20-Poly1305	Ours
2	2	6.775	3.017	2.771
	32	103.2	42.14	41.92
	512	657.7	363.0	351.8
	8192	800.9	590.6	590.0
	16384	808.2	603.6	605.1
32	2	6.906	2.903	2.801
	32	104.6	41.84	42.30
	512	657.2	360.8	362.3
	8192	802.0	591.0	591.3
	16384	808.6	606.1	605.6
512	2	4.190	1.902	1.849
	32	64.71	28.38	28.31
	512	531.8	289.8	291.0
	8192	786.6	578.5	577.8
	16384	800.9	598.5	598.5

Table 5: Performance of AES256-GCM, ChaCha20-Poly1305 and ChaCha20-Poly1305-PSIV (ours) in MiB/s. The measurements were performed on an Intel i7 10510U processor at 1.80 Ghz with Turbo Boost Technology disabled. The median value is given.

(a) Encryption speed in MiB/s.

A (bytes)	M (bytes)	AES-256-GCM	ChaCha20-Poly1305	Ours
2	2	6.885	3.216	2.876
	32	102.6	45.02	44.28
	512	711.7	388.6	384.0
	8192	904.7	681.7	679.6
	16384	907.7	694.8	698.3
32	2	7.244	3.064	2.984
	32	107.6	43.30	44.63
	512	721.2	380.8	390.5
	8192	900.2	679.8	681.1
	16384	910.4	695.8	700.0
512	2	4.865	1.921	1.881
	32	74.00	28.21	28.71
	512	609.2	306.0	308.2
	8192	892.8	656.9	660.9
	16384	900.9	685.8	689.6

(b) Decryption speed in MiB/s.

A (bytes)	M (bytes)	AES-256-GCM	ChaCha20-Poly1305	Ours
2	2	7.152	3.017	2.771
	32	102.9	44.08	43.83
	512	724.5	363.0	351.8
	8192	899.5	680.5	678.6
	16384	894.9	603.6	605.1
32	2	7.549	3.131	2.842
	32	110.4	42.48	44.06
	512	614.0	377.0	386.6
	8192	901.2	674.5	679.8
	16384	903.3	695.8	699.7
512	2	4.978	1.881	1.865
	32	74.10	27.64	28.56
	512	614.0	300.4	307.9
	8192	887.6	656.2	662.3
	16384	894.0	685.8	685.3