Security Proofs for Oribatida

Arghya Bhattacharjee[1], Eik List[2],
Cuauhtemoc Mancillas López[3] and Mridul Nandi[1]

[1]Indian Statistical Institute Kolkata, India
bhattacharjeearghya29(at)gmail.com, mridul.nandi(at)gmail.com
[2]Bauhaus-Universität Weimar, Germany
<firstname>.<lastname>(at)uni-weimar.de
[3]Computer Science Department, CINVESTAV-IPN, Mexico
cuauhtemoc.mancillas83(at)gmail.com

Abstract. Permutation-based modes have been established for lightweight authenticated encryption (AE). While encryption can be performed in an on-line manner, authenticated decryption assumes that the resulting plaintext is buffered and never released if the corresponding tag is incorrect. Since lightweight devices may lack resources, additional robustness guarantees, such as integrity under release of unverified plaintexts (Int-RUP), are desirable. The Int-RUP security of previous permutation-based AE schemes is limited by $O(q_p q_d/2^c)$, where q_d and q_p are the limits for decryption and primitive queries, respectively, which motivates novel approaches.
Oribatida is a permutation-based duplex-like AE scheme that derives s-bit masks from previous permutation outputs to mask ciphertext blocks. As a result, Oribatida can provide an AE security of $O(r\sigma^2/2^{c+s})$, which allows smaller permutations and can therefore save area for the same level of security. In the Int-RUP setting, it provides a security level dominated by $O(\sigma_d^2/2^c)$, which eliminates the dependency on primitive queries. We show that our Int-RUP bound for Oribatida is tight and show general attacks on previous constructions.

Keywords: Authenticated encryption · permutation · provable security

1 Introduction

Permutation-based Modes, such as the Sponge [9] or duplex [8] modes, transform an internal n-bit state iteratively with a public permutation. In both modes, an n-bit permutation absorbs the data in r-bit chunks, where $r < n$ is called the rate. The hidden inner state of $c = n - r$ bits is called the capacity; where r and c represent the trade-off between performance and security. Keyed Sponge Modes [9] can be categorized into inner-keyed, outer-keyed, and full-keyed variants, cf. [21]. Recently, Dobraunig and Mennink added the suffix-keyed sponge [17].

Numerous Works have studied the security of the sponge and duplex in the past. The resources of a distinguisher $\mathbf{A}$ are usually quantified by q_e encryption queries, q_v verification queries, q_p construction queries, and σ blocks over all construction queries. Bertoni et al. [7] showed that the sponge is indifferentiable from a random oracle [20] for up to $O(2^{c/2})$ calls to the permutation. [10] improved the bounds for the unkeyed sponge to $O(\frac{q_p\sigma}{2^c} + \frac{q_c}{2^k})$ if $\sigma \ll 2^{c/2}$.
Permutation-based modes for AE started with the Duplex construction and SpongeWrap [8], as well as with MonkeyDuplex [11]. For SpongeWrap, Bertoni et al. [8] had shown a

privacy bound of $O(\frac{q}{2^k}+\frac{\sigma^2}{2^c})$ and an authenticity bound of $O(\frac{q}{2^k}+\frac{\sigma^2}{2^c}+\frac{q}{2^\tau})$. Jovanovic et al. [19] considered general integrity bounds, showing asymptotic security of $O(\max(\frac{\sigma^2}{2^n}, \frac{\sigma}{2^c}, \frac{q}{2^k}))$. Though, their result limited the number of decryption queries to $\sigma \ll 2^{c/2}$. For more decryption queries, their bound reduced to $O(\max(\frac{\sigma^2}{2^c}, \frac{q}{2^k}))$. Many further works studied the security of permutation-based schemes over time, e.g., [4, 15, 18, 22, 23]. Mennink [21] summarized many previous works by showing that keyed sponges achieve PRF security of about $O(\frac{q_c^2+q_cq_p}{2^c}) + \mathbf{Adv}^{\mathrm{KP}}_{\Pi}$, where the latter term is the key-prediction security, that is, e.g., in $O(\frac{q_p}{2^k})$ for full-keyed sponges if $k < n$. Improvements to those general bounds appear unlikely, which has motivated the search for novel construction approaches. For instance, the duplex-based scheme Beetle [12] added a transform to the output so that the input block that is added to the rate part and the block added to the ciphertext output block differed; Beetle offered a bound of $O(\frac{rq_p+r\sigma}{2^c} + \frac{q_v+q_p}{2^r} + \frac{\sigma^2+q_p^2}{2^n})$.

While Sponges and Duplex-based AE Schemes encrypt in an on-line fashion, authenticated decryption usually demands the entire plaintext to be buffered until the tag has been verified successfully. Though, such a behavior can exceed the available memory and induce unacceptable latency. Andreeva et al. [3] introduced security notions when unverified plaintext (RUP) material is released. While it is hard to impossible for on-line schemes to protect the privacy when unverified plaintexts are released, they can ensure integrity (Int-RUP), which is a valuable level of robustness.
While Beetle improved the security for nonce-based AE, there exists an attack with advantage $O(\frac{q_pq_v}{2^c})$ in the Int-RUP setting, as we will describe. The question arises whether higher Int-RUP security is achievable. This is motivated by practical relevance: in resource-constrained environments, buffering long decryption results may be infeasible and Int-RUP security advantageous. The ongoing NIST lightweight competition [24] requests 112-bit integrity security for AE schemes and support of at least $2^{50}-1$ encrypted bytes of data. So, the known bounds seem to imply permutations over ≥ 224 bits plus a plausible rate. Higher Int-RUP security could therefore lead to smaller permutations.

Contributions. This work contains two main contributions. First, it answers the research question above in the affirmative by showing that Oribatida achieves a security bound of $O(\frac{q_p}{2^k} + \frac{q_d}{2^\tau} + \frac{q_p\sigma}{2^{c+s}})$. From the NIST requirements, $c+s \geq 192$ should hold. Moreover, we prove that the Int-RUP advantage is upper bounded by $O(q_d^2/2^c)$, which depends only on the number of on-line queries, which contrasts the bound $O(q_dq_p/2^c)$ that holds for the generic duplex and previous constructions. While the difference may appear small, eliminating the dependency of off-line (i.e., primitive) queries by the adversary is a valuable gain. Second, we show attacks on previous permutation-based AE schemes, as well as a matching attack that shows that the Int-RUP bound of Oribatida is tight and also applies to other masked duplex-based designs.

Outline. After a brief recall of the necessary preliminaries, Section 3 briefly recaps Oribatida. We analyze the security on Oribatida for the standard nonce-based AE setting in Section 4 and in the Int-RUP setting in Section 5. Section 6 describes an Int-RUP attack on unmasked schemes for the Duplex, and an Int-RUP attack on Oribatida that matches our bound. The appendix provides various further Int-RUP attacks.

2 Preliminaries

General Notations. Uppercase letters (e.g., X, Y) denote functions and variables, lowercase letters (e.g., x, y) indices and lengths, and calligraphic uppercase letters (e.g., $\mathcal{X}, \mathcal{Y}$) sets and spaces. $\mathbb{F}_2$ denotes the field of characteristic 2 and $\mathbb{F}_2^n$ vectors over $\mathbb{F}_2$. $|X|$

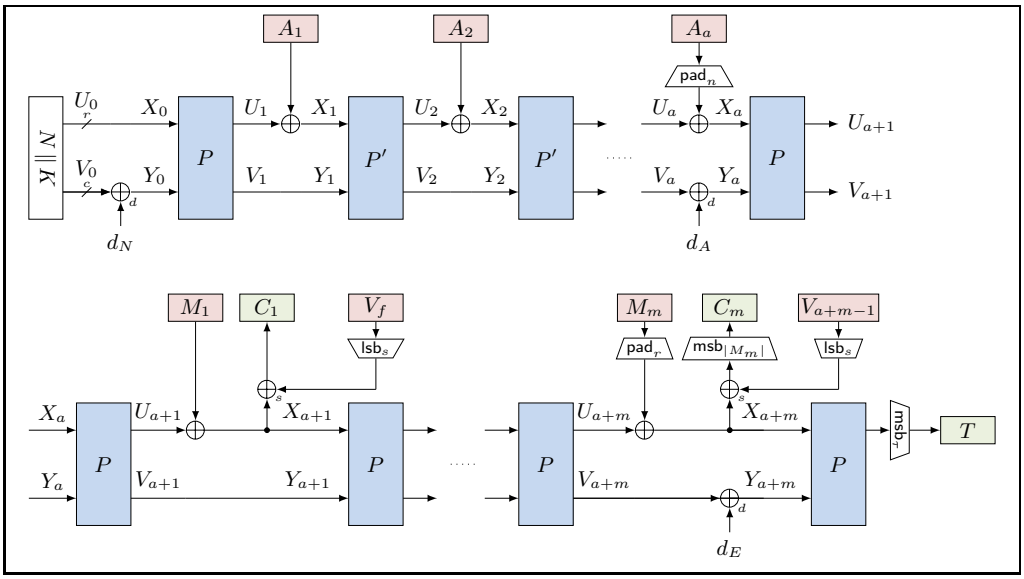

Figure 1: Schematic illustration of the authentication of an a-block associated data A and the encryption of an m-block plaintext M with Oribatida, for $m > 1$. P and P' are permutations, K the secret key, N the nonce, C the resulting ciphertext, and T the resulting authentication tag.

denotes the number of bits of X. Given $X \in \mathbb{F}_2^n$, $X[i]$ is the i-th (least significant) bit of X, and the bit order is $X = (X[n-1] \| \ldots \| X[1] \| X[0])$; ε means the empty string. We denote by $X[x..y]$ the range of $X[x], \ldots, X[y]$ for non-zero integers x and y. Given binary strings X and Y, we denote their concatenation by $X \| Y$ and their bitwise XOR by $X \oplus Y$ when $|X| = |Y|$. For positive integers x and y and bit strings of different lengths $X \in \mathbb{F}_2^x$ and $Y \in \mathbb{F}_2^y$ with $x \geq y$, we define $X \oplus_y Y =^{\text{def}} X \oplus (0^{x-y} \| Y)$.
We write $X \twoheadleftarrow \mathcal{X}$ for X being chosen uniformly at random and independent from other variables from a set $\mathcal{X}$. We consider $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ to be the set of all mappings $F : \mathcal{X} \to \mathcal{Y}$, and $\mathsf{Perm}(\mathcal{X})$ to be the set of all permutations over $\mathcal{X}$. We denote the invalid symbol by $\perp$. Moreover, we denote by $(n)_k =^{\text{def}} \prod_{i=0}^{k-1}(n-i)$ the falling factorial.
For $X \in \mathbb{F}_2^*$, $(X_1, X_2, \ldots, X_x) \xleftarrow{n} X$ denotes the splitting of X into n-bit strings $X_1, \ldots, X_{x-1}$, and $|X_x| \leq n$, in form of $X_1 \| \ldots \| X_x = X$. Moreover, for $Y \in \mathbb{F}_x$, we write $(X_1, X_2, \ldots, X_m) \xleftarrow{x_1, x_2, \ldots, x_m} Y$ to denote the splitting of Y into $X_1 = Y[x-1..x-x_1]$, $X_2 = Y[x-x_1-1..x-x_1-x_2]$, $\ldots$, $X_m = Y[x_m-1..0]$, where $x = x_1 + x_2 + \ldots + x_m$.
For a set $\mathcal{X}$ and non-negative integer x, $\mathcal{X}^{\leq x}$ denotes the union set $\cup_{i=0}^{x} \mathcal{X}^i$. Given a non-negative integer $x < 2^n$, $\langle x \rangle_n$ denotes its conversion into an n-bit binary string with the most significant bit left, e.g., $\langle 135 \rangle_8 = (10000111)$. We omit n if clear from the context.

Definitions of Nonce-based Authenticated Encryption. For the sake of space limitations, the definitions and notions for nonce-based authenticated encryption (nAE), integrity under release of unverified plaintexts (Int-RUP), and the H-coefficient technique that will be employed in our security analysis are in Appendix A. Our analyses consider information-theoretic distinguishers $\mathbf{A}$ and idealized primitives, where $\mathbf{A}$'s resources are bounded only by the maximal numbers of queries and blocks they can ask to their available oracles. One can derive the computation-theoretic counterparts easily by adding a parameter of the distinguishers' maximal computational resources.

3 Brief Specification of Oribatida

Let $P, P' \in \mathsf{Perm}(\mathbb{F}_2^n)$ be permutations over $\mathbb{F}_2^n$. In Oribatida, P' will only be used to process non-final associated-data blocks since no output will be released for them; P will be used at all other locations where a call to a primitive is needed. Thus, good differential bounds for P' are sufficient in practice, whereas P should be indistinguishable from a random permutation.

General Definitions. Let n denote the state size, k the key size, r the rate, c the capacity, s the mask size, ν the nonce size, d a domain size, and τ a tag size in bits, all of which are non-negative integers. We define:

- The key space $\mathcal{K} = \mathbb{F}_2^k$, with $k \leq n$.
- We denote by r the rate and by c the capacity of the Oribatida mode, where $r+c = n$ bits.
- The nonce space $\mathcal{N} = \mathbb{F}_2^\nu$, with $\nu \leq r$. Oribatida requires $\nu + k = n$.
- A finite set of domains $\mathbb{F}_2^d$ for $d = 4$ bits.
- The associated-data space $\mathcal{A} = \mathbb{F}_2^{\leq a_{\max}}$.
- Message and ciphertext spaces $\mathcal{M} = \mathcal{C} = \mathbb{F}_2^{\leq m_{\max}}$.
- Moreover, we define the space of authentication tags $\mathcal{T} = \mathbb{F}_2^\tau$ with $\tau \leq r$.

We define $s \leq c$ for the mask size in bits. We denote the state after the i-th call to the permutations by $S_i = (U_i \,\|\, V_i)$, and the state after XORing the subsequent associated-data block A_i or message block M_{i-a} to it by $(X_i \,\|\, Y_i)$, where a denotes the number of associated-data blocks after padding. We say that A is integral if its length is a multiple of r bits, and say that it is partial otherwise. Similarly, we say that M (or C) is integral if its length is a multiple of r bits, and call it partial otherwise.

The Core Idea. Oribatida is a variant of the monkey-wrap design [11], as used before, e.g., in Ascon [16] or NORX [5]. Oribatida extends previous designs by a ciphertext-block masking that increases the resilience against release of unverified plaintext material. Unlike the usual sponge, an s-bit part of the capacity is used to mask the subsequent ciphertext block. The definition is given in Algorithm 1. In the following, explanations and details are presented.

Domain Separation. For the purpose of domain separation, Oribatida defines a set of domain constants d_N, d_A and d_E. Note that $d = 4$ bits suffice in practice. The domains are XORed with the least significant byte of the state at three stages. Domains are encoded as bit strings, e.g., $\langle 12 \rangle_d = (1100)_2$. The value depends on the presence of A and M and whether their final blocks are absent, partial, or integral. This ensures that there exist no trivial collisions of inputs to P among blocks of A and M.
The constants are determined by the four control bits (t_3, t_2, t_1, t_0) that reflect inputs in the hardware API, similar to, e.g., [12]. The rationale behind them is the following:

- **EOI:** t_3 is the **end-of-input** control bit. This bit is set to 1 iff the current data block is the final block of the input. For all other cases, t_3 is set to 0.
- **EOT:** t_2 is the **end-of-type** control bit. This bit is set to 1 iff the current data block is the final block of the same type, i.e., it is the last block of the message/associated data. Note that, if the associated data is empty, the nonce is treated as the final block of the associated data. So, t_2 is set to 1. For all other cases, t_2 is set to 0.

Algorithm 1 Specification of Oribatida.

```
101: function E_K^{N,A}(M)
102:   ℓ_A ← |A|
103:   ℓ_E ← |M|
104:   d_N ← GetDomainForN(ℓ_A, ℓ_E)
105:   d_A ← GetDomainForA(ℓ_A, ℓ_E)
106:   d_E ← GetDomainForE(ℓ_E)
107:   A ← pad_r(A)
108:   M ← pad_r(M)
109:   (S_1, V_f) ← Init(K, N, d_N, ℓ_A)
110:   S_{a+1} ← ProcessAD(S_1, A, d_A)
111:   (C, T) ← Encrypt(S_{a+1}, M, V_f, d_E, ℓ_E)
112:   return (C, T)

121: function GetDomainForN(ℓ_A, ℓ_E)
122:   if ℓ_A = 0 ∧ ℓ_E = 0 then return ⟨9⟩_n
123:   return ⟨5⟩_n

131: function GetDomainForA(ℓ_A, ℓ_E)
132:   if ℓ_A = 0 then return ⟨4⟩_n
133:   if ℓ_E > 0 ∧ ℓ_A mod r = 0 then return ⟨4⟩_n
134:   if ℓ_E > 0 ∧ ℓ_A mod r ≠ 0 then return ⟨6⟩_n
135:   if ℓ_E = 0 ∧ ℓ_A mod r = 0 then return ⟨12⟩_n
136:   if ℓ_E = 0 ∧ ℓ_A mod r ≠ 0 then return ⟨14⟩_n

141: function GetDomainForE(ℓ_E)
142:   if ℓ_E = 0 then return ⟨0⟩_n
143:   if ℓ_E mod r = 0 then return ⟨13⟩_n
144:   if ℓ_E mod r ≠ 0 then return ⟨15⟩_n

151: function pad_x(X)
152:   if |X| mod x = 0 then return X
153:   return X ‖ 1 ‖ 0^{x−(|X| mod x)−1}

161: function Init(K, N, d_N, ℓ_A)
162:   V_0 ← lsb_s(N ‖ K)
163:   S_1 ← P((N ‖ K) ⊕_d d_N)
164:   V_1 ← lsb_s(S_1)
165:   if ℓ_A = 0 then return (S_1, V_0)
166:   if ℓ_A ≠ 0 then return (S_1, V_1)

171: function ProcessAD(S_1, A, d_A)
172:   (A_1, ··· , A_a) ←r A
173:   for i = 1..a − 1 do
174:     S_{i+1} ← P'(S_i ⊕ (A_i ‖ 0^c))
175:   S_{a+1} ← P(S_a ⊕ (A_a ‖ 0^c) ⊕_d d_A)
176:   return S_{a+1}

181: function lsb_x(X)
182:   if |X| ≤ x then return X
183:   return X[(|X| − x − 1)..0]

191: function msb_x(X)
192:   if |X| ≤ x then return X
193:   return X[(|X| − 1)..(|X| − x)]

201: function D_K^{N,A}(C, T)
202:   ℓ_A ← |A|
203:   ℓ_E ← |C|
204:   d_N ← GetDomainForN(ℓ_A, ℓ_E)
205:   d_A ← GetDomainForA(ℓ_A, ℓ_E)
206:   d_E ← GetDomainForE(ℓ_E)
207:   A ← pad_r(A)
208:   C ← pad_r(C)
209:   (S_1, V_f) ← Init(K, N, d_N, ℓ_A)
210:   S_{a+1} ← ProcessAD(S_1, A, d_A)
211:   (M, T') ← Decrypt(S_{a+1}, C, V_f, d_E, ℓ_E)
212:   if T = T' then return M
213:   else return ⊥

221: function Encrypt(S_{a+1}, M, V_f, d_E, ℓ_E)
222:   x ← ℓ_E mod r
223:   (M_1, ··· , M_m) ←r M
224:   V ← V_f
225:   for i = 1..m do
226:     (U_{a+i}, V_{a+i}) ←r,c S_{a+i}
227:     X_{a+i} ← M_i ⊕ U_{a+i}
228:     C_i ← X_{a+i} ⊕_s lsb_s(V)
229:     Y_{a+i} ← V_{a+i}
230:     if i = m then
231:       Y_{a+i} ← Y_{a+i} ⊕_d d_E
232:       C_m ← msb_x(C_m)
233:     V ← V_{a+i}
234:     S_{a+i+1} ← P(X_{a+i} ‖ Y_{a+i})
235:   C ← (C_1 ‖ C_2 ‖ ··· ‖ C_m)
236:   T ← msb_τ(S_{a+m+1})
237:   return (C, T)

241: function Decrypt(S_{a+1}, C, V_f, d_E, ℓ_E)
242:   x ← ℓ_E mod r
243:   if ℓ_E = 0 then
244:     T' ← msb_τ(S_{a+1})
245:     return (ε, T')
246:   (C_1, ··· , C_m) ←r C
247:   V ← V_f
248:   for i = 1..m do
249:     (U_{a+i}, V_{a+i}) ←r,c S_{a+i}
250:     X_{a+i} ← C_i ⊕_s lsb_s(V)
251:     Y_{a+i} ← V_{a+i}
252:     M_i ← U_{a+i} ⊕ X_{a+i}
253:     if i = m then
254:       Y_{a+i} ← Y_{a+i} ⊕_d d_E
255:       M_m ← msb_x(M_m)
256:     V ← V_{a+i}
257:     S_{a+i+1} ← P(X_{a+i} ‖ Y_{a+i})
258:   M ← (M_1 ‖ M_2 ‖ ··· ‖ M_m)
259:   T' ← msb_τ(S_{a+m+1})
260:   return (M, T')
```

- **Partial:** t_1 is the **partial-control** bit. It is set to 1 if the current data block is partial, i.e. if its size is less than the required block size. For all other data blocks, t_1 is 0.

- **Type:** t_0 is called the **type-control** bit. It identifies the type of the current data block. For the nonce and the processing of the final message block, t_0 is set to 1. For all other cases, t_0 is set to 0.

While processing a data block, the domain values are set as the integer representation of $t_3 \,\|\, t_2 \,\|\, t_1 \,\|\, t_0$. For example, if we are processing the nonce (which is always a complete r-bit block), where the associated data is empty, and the message is not empty, it holds that $d_N = (t_3 t_2 t_1 t_0) = (0101)_2 = 5$.

4 nAE Security Analysis

This section analyzes the nAE security of Oribatida. In the following, let $K \twoheadleftarrow \mathcal{K}$ and $\pi \twoheadleftarrow \mathsf{Perm}(\mathbb{F}_2^n)$. We use $\Pi[\pi, \pi]_K = \Pi[\pi]_K$ as short form of Oribatida, instantiated with π for P *and* for P', and keyed by K.
Let $\mathbf{A}$ be a nonce-respecting nAE adversary w.r.t. $\Pi[\pi]_K$. We denote by q_p, q_f, q_b, q_e, q_d, σ_e, σ_d the number of primitive queries, forward primitive queries, backward primitive queries, encryption queries, decryption queries, blocks summed over all encryption queries, and blocks summed over all decryption queries, respectively. Clearly, it holds that $q_p = q_f + q_b$. For simplicity, we define a function ρ as

$$\rho(i,j) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } j = 1 \wedge a^i = 0 \\ 1 & \text{if } j = 1 \wedge a^i \neq 0 \\ a^i + j - 1 & \text{otherwise} \end{cases}$$

So, $V^i_{\rho(i,j)}$ denotes the used block for masking the ciphertext block C^i_j.

Theorem 1 (nAE Security of Oribatida)**.** Let $\mathbf{A}$ be a nonce-respecting adversary w.r.t. $\Pi[\pi]_K$. Then, $\mathbf{Adv}^{\text{nAE}}_{\Pi[\pi]_K}(\mathbf{A})$ is upper bounded by

$$\frac{\binom{\sigma}{r} + 2\binom{q_p}{r}}{2^{r(r-1)}} + \frac{\sigma^2}{2^n} + \frac{3q_p}{2^k} + \frac{r(q_d + \sigma_d) + 2\sigma_e q_p + q_p q_c + q_d(\sigma_e + q_p)}{2^{c+s}} + \frac{2rq_p}{2^{n-\tau}} + \frac{q_d}{2^\tau}\,.$$

Proof. We follow the strategy of the nAE proof of Beetle [12]. The queries by $\mathbf{A}$ and their corresponding answers are collected in a transcript $\tau = (\tau_e, \tau_d, \tau_p)$. Encryption construction queries are stored as tuples $\tau_e = \{(N^i, A^i, M^i, C^i, T^i)\}$, for $1 \leq i \leq q_e$, decryption construction queries as tuples $\tau_d = \{(N^i, A^i, M^i, C^i, T^i)\}$, for $1 \leq i \leq q_d$, and primitive queries are stored as tuples $\tau_p = \{(Q^i, R^i)\}$, where $\pi(Q^i) = R^i$, for $1 \leq i \leq q_p$.

Sampling. We define the ideal oracle to consist of an on-line and an off-line phase. In the on-line phase, the ideal oracle samples the responses (C^i, T^i) uniformly at random from the bit strings of expected lengths for encryption queries. For decryption queries, it always outputs $\perp$. For forward primitive queries Q^i, it forwards the result of $\pi(Q^i)$ to $\mathbf{A}$; for backward primitive queries R^i, it forwards the result of $\pi^{-1}(R^i)$.
In the off-line phase, the ideal oracle samples the internal chaining values $V^i_j \twoheadleftarrow \{0,1\}^c$ and $U^i_j \twoheadleftarrow \{0,1\}^r$ uniformly at random for all construction queries in encryption direction. It derives the analogous internal chaining values V^i_j, for $1 \leq j \leq k$ for all construction queries in decryption direction (N^i, A^i, C^i, T^i) that share $N^i = N^{i'}$ and $A^i = A^{i'}$, and for which $C^i_1 = C^{i'}_1, \ldots, C^i_k = C^{i'}_k$ holds for some i'-th construction query, where $i \neq i'$. Moreover, for construction queries whose plaintext or ciphertext length is not a multiple of r bits, the oracle samples exactly the missing bits C^i_m uniformly independently at random that are not fixed from previous queries, at most at most $r - |C^i_{m^i}|$ bits at a time. The so-sampled values for the final blocks $C^i_{m^i}$ are stored also in the transcript. Moreover, the random key K is revealed to the adversary at the end of the off-line phase.

Bad Events. We define the following bad events. If any of them occurs, the adversary aborts, and we define that it wins in this case.

- bad_1: Multi-collision on the rate part X in encryption construction queries. For some $w \geq r$, there exist indices (i_1, j_1), (i_2, j_2), …, (i_w, j_w) with i_1, i_2, …, $i_w \in [1..q_e]$, and $j_1 \in [1..a^{i_1}]$, $j_2 \in [1..a^{i_2}]$, etc., s. t. $X^{i_1}_{j_1} = X^{i_2}_{j_2} = \ldots = X^{i_w}_{j_w}$.
- bad_2: Collision of two permutation inputs in encryption construction queries. There exist indices $(i,j) \neq (i',j')$ with $i, i' \in [1..q_e]$, $j \in [1..m^i]$, and $j' \in [1..m^{i'}]$ s. t. $\left(X^i_j \,\|\, Y^i_j\right) = \left(X^{i'}_{j'} \,\|\, Y^{i'}_{j'}\right)$.

- bad_3: Collision of two permutation outputs in encryption construction queries. There exist indices $(i,j) \neq (i',j')$ with $i,i' \in [1..q_e]$, $j \in [1..m^i]$, and $j' \in [1..m^{i'}]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = \left(U_{j'}^{i'} \,\|\, V_{j'}^{i'}\right)$.
- bad_4: Collision of permutation inputs between a construction and a primitive query. There exist indices (i,j,i') with $i \in [1..q_e]$, $j \in [1..m^i]$, and $i' \in [1..q_p]$ s. t. $\left(X_j^i \,\|\, Y_j^i\right) = Q^{i'}$.
- bad_5: Collision of permutation outputs between a construction and a primitive query. There exist indices (i,j,i') with $i \in [1..q_e]$, $j \in [1..m^i]$, and $i' \in [1..q_p]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = R^{i'}$.
- bad_6: Initial-state collision with a primitive query. There exist indices (i,i') with $i \in [1..q_e]$ and $i' \in [1..q_p]$ s. t. $\left(X_0^i \,\|\, Y_0^i\right) = Q^{i'}$.
- bad_7: Multi-collision in the rate part of w outputs of forward primitive queries. So, for some $w \geq r$, there exist i_1, i_2, …, $i_w \in [1..q_p]$ s. t. $\mathsf{msb}_r(R^{i_1}) = \mathsf{msb}_r(R^{i_2}) = \cdots = \mathsf{msb}_r(R^{i_w})$.
- bad_8: Multi-collision in the rate part of w outputs of backward primitive queries. So, for some $w \geq r$, there exist i_1, i_2, …, $i_w \in [1..q_p]$ s. t. $\mathsf{msb}_r(Q^{i_1}) = \mathsf{msb}_r(Q^{i_2}) = \cdots = \mathsf{msb}_r(Q^{i_w})$.

We define that the adversary is provided with all internal chaining values V_j^i and U_j^i after its interaction, but before it outputs its decision bit. Clearly, this fact only strengthens the adversary.
We define the set of bad transcripts BadT, to contain exactly those attainable transcripts τ for which at least one of the bad events occurred. All other attainable transcripts are in GoodT. It holds that $\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \sum_{i=1}^{8} \Pr[\mathsf{bad}_i]$. The probability of bad transcripts in the ideal world is then treated in the proof of Lemma 1. The ratio of obtaining a good transcript is bounded in Lemma 2. Our bound in Theorem 1 follows then from them and the application of Lemma 5. We apply $w = r$ in the bounds of Lemma 1 and 2 to obtain our bound from Theorem 1.

Lemma 1. Let $w \geq r$ be a positive integer. It holds that

$$\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \frac{\binom{\sigma}{w} + 2\binom{q_p}{w}}{2^{r(w-1)}} + \frac{\sigma^2}{2^n} + \frac{3q_p}{2^k} + \frac{2\sigma_e q_p + q_p q_c}{2^{c+s}} + \frac{2w \cdot q_p}{2^{n-\tau}}.$$

Lemma 2. Let $\tau \in \text{GoodT}$. Then

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \leq 1 - \left(\frac{q_d}{2^\tau} + \frac{q_d(q_p + \sigma_e)}{2^{c+s}} + \frac{(\sigma_d + q_d) \cdot w}{2^{c+s}}\right).$$

The proofs of both Lemma 1 and 2 are deferred to Appendix D.

5 Int-RUP Analysis

This section provides our Int-RUP result for Oribatida. We employ the same notations as in the nAE analysis in Section 4, but add several notations. We denote by q_d and σ_d the number of decryption queries and blocks over all decryption queries, respectively, and by q_v and σ_v the analogues for verification queries. Again, we replace $\pi \twoheadleftarrow \mathsf{Perm}(\mathbb{F}_2^n)$, assume $K \twoheadleftarrow \mathcal{K}$, and denote $\Pi[\pi]_K$ for Oribatida, instantiated with π and K.
We recall the notion of a longest common prefix from Bellare et al. [6]. Let $Q = (N, A, M, C, T)$ denote a query of $\mathbf{A}$, including the oracle response. Let $\mathcal{Q}$ denote a set of

such queries without Q, i.e., $Q \notin \mathcal{Q}$. We define the length of the longest common prefix of M and another message M' as $\mathsf{LCP}(M, M') =^{\text{def}} \max_i\{1 \leq j \leq i : M_j = M'_j\}$. Given $\mathcal{Q}$ and M, we overload the notation by considering the longest common prefix of M with the queries in $Q' = (N', A', M', C', T') \in \mathcal{Q}$: $\mathsf{LCP}(M, \mathcal{Q}) =^{\text{def}} \max_{M' \in \mathcal{Q}} \{\mathsf{LCP}(M, M')\}$. Moreover, we define the notion

$$\mathsf{LCP}^{N,A}(M, \mathcal{Q}) \stackrel{\text{def}}{=} \max_{\substack{(N', A', M', C', T') \in \mathcal{Q} \\ N'=N \wedge A'=A}} \{\mathsf{LCP}(M, M')\} \ .$$

Theorem 2 (INT-RUP Security of Oribatida)**.** Let $\mathbf{A}$ be a nonce-respecting adversary w.r.t. $\Pi[\pi]_K$. Then

$$\mathbf{Adv}^{\text{INT-RUP}}_{\Pi[\pi]_K}(\mathbf{A}) \leq \frac{\sigma_e^2}{2^n} + \frac{4\sigma_e\sigma_d + 4\sigma q_p + q_c q_p + q_p + r(\sigma_d + q_d)}{2^{c+s}} + \frac{q_d^2 + \binom{q_d+q_v}{2}}{2^c} + \frac{\binom{q_e}{r}}{2^{\tau(r-1)}} + \frac{3rq_p}{2^{n-\tau}} + \frac{3q_p}{2^k} + \frac{2\binom{q_p}{r}}{2^{r(r-1)}} + \frac{2q_v}{2^\tau} \ .$$

Proof. The INT-RUP analysis of Oribatida follows a similar proof strategy as our nAE analysis. However, this time, the adversary has access to three oracles for encryption, decryption and verification. Moreover, the encryption and decryption oracles are the same in both the real *and* the ideal world. Both worlds differ only in the verification oracle. To alleviate the task, we replace the oracles for encryption and decryption $\widetilde{\mathcal{E}}[\pi]_K$, $\widetilde{\mathcal{D}}[\pi]_K$ with a pair of **consistent** pseudo-random oracles $\$_{\widetilde{\mathcal{E}}}[\pi]$ and $\$_{\widetilde{\mathcal{D}}}[\pi]$ (we define our intent of consistency for encryption and decryption in a moment). The advantage between both settings can be bounded by

$$\underset{\mathbf{A}_1}{\Delta}\left(\widetilde{\mathcal{E}}[\pi]_K, \widetilde{\mathcal{D}}[\pi]_K, \widetilde{\mathcal{V}}[\pi]_K, \pi^{\pm}; \$_{\widetilde{\mathcal{E}}}, \$_{\widetilde{\mathcal{D}}}, \perp, \pi^{\pm}\right).$$

Note that the oracles $\$_{\widetilde{\mathcal{E}}}$ and $\$_{\widetilde{\mathcal{D}}}$ differ from the *independent* random oracles in the stronger RUPAE notion. In the RUPAE notion, they sample **independently** without considering common prefixes between queries, which would be impossible to achieve for on-line AE. Again, we employ the H-coefficient technique. We define several bad events and bad as well good transcripts. If any of the bad events occurs, the adversary aborts, and we say, it wins in this case. Next, we consider the probability of forgeries under those idealized oracles. So, we can exclude the previous bad events and study the probability of forgeries. Finally, we study the ratio of interpolation probabilities for good transcripts as usual.

Sampling Consistently in the On-line Phase. The on-line phase contains much from the off-line phase from the nAE analysis. We define the ideal encryption oracle as in the nAE proof: it samples the responses (C^i, T^i) uniformly at random from the bit strings of expected lengths for encryption queries. The ideal decryption oracle, however, must sample plaintext outputs consistently. For this purpose, the ideal encryption oracle has to sample also the internal chaining values $V^i_j \twoheadleftarrow \{0,1\}^c$ and $U^i_j \twoheadleftarrow \{0,1\}^r$ uniformly at random for all construction queries already in the on-line phase. It stores the values of C^i_j, V^i_j, and U^i_j also internally, but does not release U^i_j and V^i_j in this phase.
On each input (N^i, A^i, C^i, T^i), the ideal decryption oracle looks up the length of the longest common prefix of the query $p \leftarrow \mathsf{LCP}^{N^i,A^i}(C^i, \mathcal{Q})$ with all previous queries $\mathcal{Q}$. For all blocks in the common prefix $1 \leq j \leq p$, it uses the same outputs M^i_j that have been fixed from previous queries. Since the oracle has sampled V^i_{p+1} for the $(p+1)$-th block, it can deduce all bits that are not fixed from previous query outputs. Assume, $i \neq i'$, $(N^i, A^i) = (N^{i'}, A^{i'})$, and $p = \mathsf{LCP}^{N^i,A^i}(C^i, C^{i'})$ where $p < m^i, m^{i'}$. Then, $C^i_{p+1} = C^i_{p+1} \oplus \Delta$ is the block directly after the common prefix. Sampling V^i_j and deriving

U_j^i ensures consistent sampling. This means that $M_{p+1}^i = M_{p+1}^{i'} \oplus \Delta$ holds, for all such queries that share a common prefix.
From the $(p+2)$-th block, the ideal decryption oracle samples the responses $M_j^i \twoheadleftarrow \{0,1\}^r$, $V_j^i \twoheadleftarrow \{0,1\}^c$, and $U_j^i \twoheadleftarrow \{0,1\}^r$ uniformly and independently at random from the bit strings of expected lengths, for $p+2 \leq j \leq a^i + m^i$. Queries whose ciphertext lengths are not multiples of r bits are answered consistently since the oracle samples V_j^i, and all bits that are fixed from previous queries are used. For verification queries, the ideal verification oracle always outputs $\perp$. For forward primitive queries Q^i, the ideal oracle forwards the result of $\pi(Q^i)$; for backward primitive queries R^i, it returns $\pi^{-1}(R^i)$.

Off-line phase. Here, the ideal oracle releases the internal chaining values (U_j^i, V_j^i), after the considered adversary made all queries, but before outputting the decision bit. The ideal oracle also reveals a random key $K \twoheadleftarrow \mathcal{K}$ then.

Bad Events. We define trivial collisions of chaining values to be collisions in the longest common prefix. All other collisions are non-trivial collisions. Whenever we consider a non-trivial collision between blocks or chaining values at block indices j, j' of two messages, we assume that at least one of them exceeds the longest common prefix.

- bad_1: Non-trivial collision of two permutation inputs in construction queries. There exist indices $(i,j) \neq (i',j')$ with $i, i' \in [1..q_c]$, $j \in [1..m^i]$, and $j' \in [1..m^{i'}]$ s. t. $\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)$.
- bad_2: Non-trivial collision of two permutation outputs in construction queries. There exist indices $(i,j) \neq (i',j')$ with $i, i' \in [1..q_c]$, $j \in [1..m^i]$, and $j' \in [1..m^{i'}]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = \left(U_{j'}^{i'} \,\|\, V_{j'}^{i'}\right)$.
- bad_3: Multi-collision between w tags. For some $w \geq r$, there exist indices $i_1, i_2, \ldots, i_w$ with $i_1, i_2, \ldots, i_w \in [1..q_e]$, s. t. $T^{i_1} = T^{i_2} = \ldots = T^{i_w}$.
- bad_4: Non-trivial collision of permutation inputs between a construction and a primitive query. There exist indices (i,j,i') with $i \in [1..q_c]$, $j \in [1..m^i]$, and $i' \in [1..q_p]$ s. t. $\left(X_j^i \,\|\, Y_j^i\right) = Q^{i'}$.
- bad_5: Non-trivial collision of permutation outputs between a construction and a primitive query. There exist indices $i \in [1..q_c]$, $j \in [1..a^i + m^i]$, and $i' \in [1..q_p]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = R^{i'}$.
- bad_6: Initial-state collision with a primitive query. There exist indices $i \in [1..q_c]$ and $i' \in [1..q_p]$ s. t. $\left(X_0^i \,\|\, Y_0^i\right) = Q^{i'}$.
- bad_7: Multi-collision in the rate part of w outputs of forward primitive queries. So, for some $w \geq r$, there exist $i_1, i_2, \ldots, i_w \in [1..q_p]$ s. t. $\mathsf{msb}_r\left(R^{i_1}\right) = \mathsf{msb}_r\left(R^{i_2}\right) = \cdots = \mathsf{msb}_r\left(R^{i_w}\right)$.
- bad_8: Multi-collision in the rate part of w outputs of backward primitive queries. So, for some $w \geq r$, there exist $i_1, i_2, \ldots, i_w \in [1..q_p]$ s. t. $\mathsf{msb}_r\left(Q^{i_1}\right) = \mathsf{msb}_r\left(Q^{i_2}\right) = \cdots = \mathsf{msb}_r\left(Q^{i_w}\right)$.
- bad_9: Forgery in decryption queries if all blocks are old: There exists an index $i \in [1..q_d]$ s. t. for all blocks $0 \leq j \leq a^i + m^i$, there exist indices i', j' with $i' \in [1..q_c]$, $j' \in [1..m^{i'}]$ or $i' \in [1..q_p]$ s. t. $(X_j^i \,\|\, Y_j^i) = (X_{j'}^{i'} \,\|\, Y_{j'}^{i'})$ or $(X_j^i \,\|\, Y_j^i) = Q^{i'}$ and the provided tag is valid: $\mathsf{msb}_\tau\left(\pi\left(X_{a^i+m^i}^i \,\|\, Y_{a^i+m^i}^i\right)\right) = T^i$

We define the set of bad transcripts BadT, to contain exactly those attainable transcripts τ for which at least one of the bad events occurred. All other attainable transcripts are in GoodT. It holds that $\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \sum_{i=1}^{8} \Pr[\mathsf{bad}_i]$.
The probability of bad transcripts in the ideal world is then treated in the proof of Lemma 3. The ratio of obtaining a good transcript is bounded in Lemma 4. Our bound in Theorem 1 follows then from them and the application of Lemma 5. We apply $w = r$ in the bound of Lemma 3.

Lemma 3. Let $w \geq r$ be a positive integer. It holds that

$$\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \frac{\sigma_e^2}{2^n} + \frac{3\sigma_e\sigma_d + 3\sigma q_p + q_c q_p + q_p + w(\sigma_d + q_d)}{2^{c+s}} + \frac{q_d^2 + \binom{q_d+q_v}{2}}{2^c} + \frac{\binom{q_e}{w}}{2^{\tau(w-1)}} + \frac{3wq_p}{2^{n-\tau}} + \frac{3q_p}{2^k} + \frac{2\binom{q_p}{w}}{2^r(w-1)} + \frac{q_v}{2^\tau}.$$

Lemma 4. Let $\tau \in \text{GoodT}$. Then

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \leq 1 - \left(\frac{q_d}{2^\tau} + \frac{\sigma_d(\sigma_e + q_p)}{2^{c+s}}\right).$$

The proofs of Lemma 3 and 4 are given in Appendix E.

6 Int-RUP Attacks on Existing AE Schemes

This section outlines an Int-RUP attack on the (unmasked) duplex mode. We provide further attacks on Beetle, SPoC, and a hybrid of both Beetle and SPoC (that have genuine properties that are general enough to use them as specific examples) in Appendix 6. For each construction, we briefly recall the necessary parts of their definition. As summarized in Table 1, the proposed attacks possess an advantage of $O(\frac{q_p q_d}{2^c})$ on the previous constructions. Thus, the improved bounds of Beetle or SPoC do not carry over to the Int-RUP setting. For all attacks, we consider $\pi \twoheadleftarrow \mathsf{Perm}(\mathcal{B})$ and $K \twoheadleftarrow \mathcal{K}$ and a nonce-respecting $\mathbf{A}$. The core idea of all attacks is as follows: $\mathbf{A}$ asks q_d decryption queries s. t. some predetermined r bits (e.g., the first r bits) of the input to one of the permutations of the construction are fixed and known (say X). The remaining $n - r = c$ bits may vary. Next, $\mathbf{A}$ asks q_p primitive queries $Q^1, \ldots, Q^{q_p}$ with the first r bits fixed to X, but with pairwise distinct c bits, and receives $R^1, \ldots, R^{q_p}$. When $q_d \cdot q_p \approx O(2^c)$, $\mathbf{A}$ can expect a state collision between an on-line input to the permutation and an (off-line) permutation query. This collision can be detected from the first r bits of the outputs of the corresponding construction queries, which will match the colliding inputs. Once $\mathbf{A}$ knows the full state at the input to the permutation of the construction, it can revert the permutation calls in the construction and finally recover the key.

The Int-RUP Attack on the Duplex Mode. In the following, we briefly adapt this strategy for the duplex mode. Consider the common duplex mode [8]. $\mathbf{A}$ asks q_d decryption queries (N, A^1, C), (N, A^2, C), $\ldots$, (N, A^{q_d}, C), and receives M^1, M^2, $\ldots$, M^{q_d}. The associated data A^i consist of a single block, the ciphertexts $C = (C_1, C_2)$ are fixed to the same two blocks for each query. Now, $\mathbf{A}$ can follow the generic idea to complete the attack. The attack complexity is $q_d q_p \approx O(2^c)$.
In general, unmasked sponge-based AE schemes allow Int-RUP attacks whose advantage can depend linearly on the number of off-line primitive queries. Appendix C provides further attacks on strengthened versions of the schemes to show that they can be similarly strengthened by a ciphertext masking.

Table 1: Comparison of the security bounds for INT-RUP attacks.

	Bound	
Scheme	Unmasked	Masked
Generic Duplex [8]	$O(q_d q_p/2^c)$	$O(q_d^2/2^c)$
Beetle [12]	$O(q_d q_p/2^c)$	$O(q_d^2/2^c)$
SPoC [1]	$O(q_d q_p/2^c)$	$O(q_d^2/2^r)$
Oribatida [This work]	–	$O(q_d^2/2^c)$

The Int-RUP Attack on Oribatida. For Oribatida as well as for masked variants of Beetle or SPoC, the attack above does not apply directly. Though, there still exists an attack on each of the three modes with an advantage of $O(q_d^2/2^c)$. Here, we consider the application to Oribatida that shows that our INT-RUP bound of $O(q_d^2/2^c)$ is tight:

1. **A** asks q_d decryption queries (N^i, A^i, C^i, T^i), where N^i and C^i is static for all queries. We assume that the associated data A^i are pairwise distinct and consist of a single block for all queries. **A** obtains M^i from the encryption oracle, for $1 \leq i \leq q_d$. Note that the rate part $X_2^i \leftarrow C_1^i \oplus_s \mathsf{lsb}_s(V_1^i)$ is constant for all queries.
2. For $q_d \approx O(2^{c/2})$, **A** can expect a collision in the capacity part of the input: $V_2^i \leftarrow V_2^j$ for some distinct $i \neq j$, $i, j \in [1..q_d]$. Then, this collision leads to a full-state collision that can be detected when $M_1^i = M_1^j$.
3. **A** asks the encryption oracle with (N, A^i, M^i) for (C^i, T) for some tag T.
4. (N, A^j, C^i, T) is a valid forgery and yields M^j.

The attack is successful w.h.p. for $q_d^2 \approx O(2^c)$.

7 Version Notes and Effective Future Tweaks

Version Notes. Initially, this submission targeted a slightly modified variant of Oribatida, where we adapted the domains for the sake of consistency, and decided to always use at least one block of associated data. After this work was under submission, the NIST announced the criteria for second-round candidates of the lightweight-competition standardization process. Since design tweaks were prohibited for the second round, we revised our analysis to our second-round candidate Oribatida v1.2 that has no tweaks compared to our first-round submission, as requested by the NIST.
While this work was under submission, Rohit and Sarkar pointed out an attack against Oribatida-192 with complexity $O(2^{96})$ on the NIST lightweight mailing list. According to their observation, an adversary could guess the 96-bit hidden part (the capacity) that is not returned with the tag, and compute backwards until the initial state to find the correct secret key. We acknowledge their observation; we note that the security of Oribatida-256 was unaffected by the observation since the capacity matches the key length. We found that we already addressed this event in bad_4 and bad_5 in our analysis, and also bounded its probability correctly in our INT-RUP analysis. Though, we revised the denominator in the probabilities of bad_4 and bad_5 of our NAE analysis. We addressed their observation in this revised submission by updating the bound for those bad events in the NAE analysis and updated the security guarantees.

Possible Future Tweaks. We propose two simple future tweaks that can easily increase the key-recovery security of Oribatida-192 back to $128 - \log_2(r) \approx 121$ bits. The first

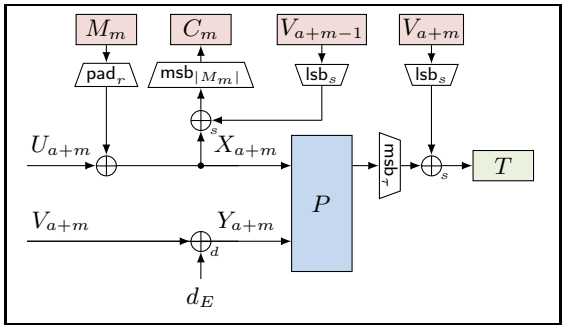

Figure 2: A possible tweak to increase the security of our secondary variant Oribatida-192.

approach would be to restrict the tag length of Oribatida-192 to 64 bits, which would increase the capacity at the end to 128 bits. A more generic approach would be to mask the final authentication tag. This could be realized by using the secret key as mask, as has been employed already, e.g., for PRIMATEs [2]. Though, it is preferable to use the previous capacity V_{a+m} as mask, as is done also for the ciphertext blocks: $T \leftarrow \mathsf{msb}_\tau(P(X_{a+m} \,\|\, Y_{a+m})) \oplus_s \mathsf{msb}_s(V_{a+m})$, assuming $s \leq \tau$ and that Y_{a+m} is the capacity after the domain has been XORed to it. This method is illustrated in Figure 2. It not only spares the need of buffering the input key until the end, but also unifies the masking for ciphertext and tag, which is helpful for hardware implementations. Both approaches would thwart any attack effectively in the sense that the current term of $rq_p/2^{n-\tau}$ would become $rq_p/2^{n-\tau+s}$ in the analysis of bad_4 and bad_5, assuming $s \leq \tau$.
In this document, as well as for the second round of the NIST lightweight competition, we followed the NIST submission guidelines and avoided tweaks. Thus, we suggest masking the tag as simple but effective tweak for a potential third-round submission.

References

[1] Riham AlTawy, Guang Gong, Morgan He, Ashwin Jha, Kalikinkar Mandal, Mridul Nandi, and Raghvendra Rohit. SpoC: An Authenticated Cipher, Feb 24 2019. First-round submission to the NIST Lightweight Cryptography Competition.

[2] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1.0 Submission to the CAESAR Competition. September 15 2016.

[3] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT I*, volume 8873 of *LNCS*, pages 105–125. Springer, 2014.

[4] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In Gregor Leander, editor, *FSE*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[5] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX: Parallel and Scalable AEAD. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS II*, volume 8713 of *LNCS*, pages 19–36. Springer, 2014. https://doi.org/10.1007/978-3-319-11212-1_2.

[6] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 292–309. Springer, 2001.

[7] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

[8] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In Ali Miri and Serge Vaudenay, editors, *SAC*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[9] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop*, volume 2007. Citeseer, 2007.

[10] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the security of the keyed sponge construction. In *SKEW*, 2011.

[11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. *Directions in Authenticated Ciphers*, 2012.

[12] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):218–241, 2018. Updated version at https://eprint.iacr.org/2018/805.

[13] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. Cryptology ePrint Archive, Report 2018/805, 2018.

[14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014.

[15] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT II*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2 Submission to the CAESAR Competition. September 15 2016. Submission to the CAESAR competition. http://competitions.cr.yp.to/caesar-submissions.html.

[17] Christoph Dobraunig and Bart Mennink. Security of the Suffix Keyed Sponge. Cryptology ePrint Archive, Report 2019/573, 2019.

[18] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO I*, volume 9215 of *LNCS*, pages 368–387. Springer, 2015.

[19] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT I*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[20] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *TCC*, volume 2951 of *LNCS*, pages 21–39. Springer, 2004.

[21] Bart Mennink. Key Prediction Security of Keyed Sponges. *IACR Trans. Symmetric Cryptol.*, 2018(4):128–149, 2018.

[22] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed sponge and duplex: Applications to authenticated encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT II*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015.

[23] Yusuke Naito and Kan Yasuda. New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 3–22. Springer, 2016.

[24] NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process, August 27 2018.

[25] Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *LNCS*, pages 328–345. Springer, 2008.

A Notions and Definitions for AE and Int-RUP

Definitions of Nonce-based Authenticated Encryption. Let $\mathcal{K}$ be a set of keys, $\mathcal{N}$ be a set of nonces, $\mathcal{A}$ a set of associated data, $\mathcal{M}$ a set of messages, $\mathcal{C}$ a set of ciphertexts, and $\mathcal{T}$ a set of authentication tags. A nonce $N \in \mathcal{N}$ is an input that must be unique for each authenticated encryption query. A nonce-based authenticated encryption scheme (with associated data) $\Pi = (\mathcal{E}, \mathcal{D})$ is a tuple of deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$ and deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \times \{\bot\}$ with associated key space $\mathcal{K}$. The encryption algorithm $\mathcal{E}$ takes a tuple (K, N, A, M) and outputs (C, T), where C is a ciphertext and T an authentication tag. We assume that $|C| = |M|$ holds for all inputs (K, N, A, M) and their corresponding ciphertexts. The associated data is authenticated, but not encrypted. The decryption function $\mathcal{D}$ takes a tuple (K, N, A, C, T) and outputs either the unique plaintext M for which $\mathcal{E}_K(N, A, M) = (C, T)$ holds, or outputs $\bot$ if the input is invalid. We introduce $\mathcal{E}_K^{N,A}(M)$ as short form of $\mathcal{E}_K(N, A, M)$ and $\mathcal{D}_K^{N,A}(C, T)$ for $\mathcal{D}_K(N, A, C, T)$, respectively.

Standard Notions. The ideal AE scheme provides two oracles $\$: \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$ and $\bot : \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \times \{\bot\}$ for encryption and verification. We overload the $\bot$ notation to mean the oracle and the symbol of invalid decryption. Given a tuple (N, A, M), the ideal encryption oracle outputs ciphertext-tag tuples (C, T) that are random bits of the expected length, i.e., computes $(C', T') = \mathcal{E}_K^{N,A}(M)$ and samples $C \twoheadleftarrow \{0,1\}^{|C'|}$ and $T \twoheadleftarrow \{0,1\}^{\tau}$. The ideal decryption oracle ensures that, given an input (N, A, C, T) where (C, T) had been the output to a previous encryption query (N, A, M), the decryption oracle outputs the corresponding message M. Otherwise, the decryption always returns the invalid symbol $\bot$ for every new decryption query that had not been the answer of an earlier encryption query.

The Ideal-permutation Model is useful to study schemes based on public permutations. Therein, the adversary has an additional oracle $\pi^{\pm}$ that provides access to the public permutation $\pi \in \mathsf{Perm}(\mathcal{B})$ for $\mathcal{B} = \mathbb{F}_2^n$ in for- and backward direction. We write $\Pi[\pi]$, $\mathcal{E}[\pi]$, $\mathcal{D}[\pi]$, etc. to denote that an AE scheme Π and its algorithms are based on π.

Definition 1 (nAE Security)**.** Let $K \twoheadleftarrow \mathcal{K}$, $\pi \twoheadleftarrow \mathsf{Perm}(\mathcal{B})$, and let $\Pi[\pi] = (\mathcal{E}[\pi]_K, \mathcal{D}[\pi]_K)$ be a nonce-based AE scheme. Let $\mathbf{A}$ be a nonce-respecting adversary. Then, $\mathbf{Adv}^{\mathrm{nAE}}_{\Pi[\pi]}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \Delta_{\mathbf{A}}\left(\mathcal{E}[\pi]_K, \mathcal{D}[\pi]_K, \pi^{\pm}; \$, \bot, \pi^{\pm}\right)$.

Notions under Release of Unverified Plaintext Material. In the RUP model, the adversary can always see the resulting plaintext from a decryption query. To formulate the forgery goal, the oracles are adapted. A verification oracle outputs 1 iff the input is valid, and 0 otherwise. A nonce-based RUP AE scheme $\widetilde{\Pi} = (\widetilde{\mathcal{E}}_K, \widetilde{\mathcal{D}}_K, \widetilde{\mathcal{V}}_K)$ is a three tuple of encryption algorithm $\widetilde{\mathcal{E}} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T}$, decryption algorithm $\widetilde{\mathcal{D}} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M}$, and verification algorithm $\widetilde{\mathcal{V}}_K : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \{0,1\}$. The signature of the encryption and decryption algorithms are unchanged.

Definition 2 (Int-RUP Security)**.** Let $K \twoheadleftarrow \mathcal{K}$, $\pi \twoheadleftarrow \mathsf{Perm}(\mathcal{B})$, and let $\widetilde{\Pi}[\pi] = (\widetilde{\mathcal{E}}[\pi]_K, \widetilde{\mathcal{D}}[\pi]_K, \widetilde{\mathcal{V}}[\pi]_K)$ be a nonce-based RUP AE scheme. Let $\mathbf{A}$ be nonce-respecting. Then $\mathbf{Adv}^{\text{Int-RUP}}_{\Pi[\pi]}(\mathbf{A}) =^{\mathrm{def}} \Delta_{\mathbf{A}}\left(\widetilde{\mathcal{E}}[\pi]_K, \widetilde{\mathcal{D}}[\pi]_K, \widetilde{\mathcal{V}}[\pi]_K, \pi^{\pm}; \widetilde{\mathcal{E}}[\pi]_K, \widetilde{\mathcal{D}}[\pi]_K, \bot, \pi^{\pm}\right)$.

H-coefficient Technique. We use Patarin's H-coefficient technique as proof method in the variant by Chen and Steinberger [14, 25]. The results of the interaction of an adversary $\mathbf{A}$ with its oracles are collected in a transcript τ. The oracles can sample randomness before the interaction, and are then deterministic throughout the experiment [14]. $\mathbf{A}$ shall distinguish the real world $\mathcal{O}_{\mathrm{real}}$ from the ideal world $\mathcal{O}_{\mathrm{ideal}}$. Θ_{real} and Θ_{ideal} denote random variables that represent the distribution of transcripts in the real and the ideal world, respectively. A transcript τ is called *attainable* if the probability to obtain τ in the ideal world – i.e. over Θ_{ideal} – is non-zero. The fundamental Lemma of the H-coefficients technique, whose proof is given in [14, 25], states that we can partition the set of all attainable transcripts into two disjoint sets GoodT and BadT.

Lemma 5 (Fundamental Lemma of the H-coefficient Technique [25])**.** Assume that there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in \text{GoodT}$, it holds that

$$\frac{\Pr\left[\Theta_{\mathrm{real}} = \tau\right]}{\Pr\left[\Theta_{\mathrm{ideal}} = \tau\right]} \geq 1 - \epsilon_1.$$

Moreover, assume that $\Pr\left[\Theta_{\mathrm{ideal}} \in \text{BadT}\right] \leq \epsilon_2$. Then, for all adversaries $\mathbf{A}$, it holds that $\Delta_{\mathbf{A}}\left(\mathcal{O}_{\mathrm{real}}; \mathcal{O}_{\mathrm{ideal}}\right) \leq \epsilon_1 + \epsilon_2$.

B Further Int-RUP Attacks on the Unmasked SPoC and Beetle

This section sketches attacks on Beetle, SPoC, and hybrids thereof.

Int-RUP Attack on Beetle. Beetle [12] is a recent permutation-based lightweight AE scheme; hereafter, we consider the updated variant from [13]. An overview of the encryption process is provided in Figure 3. The map $\rho : \{0,1\}^r \times \{0,1\}^r \to \{0,1\}^r \times \{0,1\}^r$ computes $\rho(I_1, I_2) \stackrel{\mathrm{def}}{=} (\mathsf{shuffle}(I_1) \oplus I_2, I_1 \oplus I_2)$ for all inputs $I_1, I_2 \in \{0,1\}^r$, where $\mathsf{shuffle}(x) =^{\mathrm{def}} (\mathsf{lsb}_{r/2}(x) \,\|\, \mathsf{lsb}_{r/2}(x) \oplus \mathsf{msb}_{r/2}(x))$. The results of ρ are ordered as $(X_{a+i}, C_i) \leftarrow \rho(Y_{a+i}, M_i)$ and $(Y_{a+i}, M_i) \leftarrow \rho^{-1}(X_{a+i}, C_i)$, respectively.

1. $\mathbf{A}$ asks q_d encryption queries (N^1, A^1, M), (N^2, A^2, M), $\ldots$, (N^{q_d}, A^{q_d}, M) and receives C^1, C^2, $\ldots$, C^{q_d}. M and A^i consist of one block for each i.

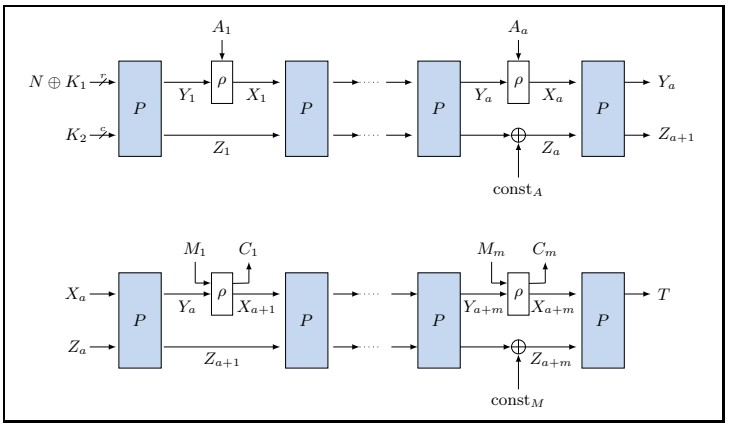

Figure 3: The Beetle authenticated encryption scheme.

2. Then, **A** asks q_d queries $(N^1, A^1, C^{1'})$, $(N^2, A^2, C^{2'})$, ... , $(N^{q_d}, A^{q_d}, C^{q'_d})$ to the decryption oracle where $C^{i'} \leftarrow Y_2^i \oplus \mathsf{shuffle}(Y_2^i)$. This ensures that the first r bits of the input to the third permutation is always equal to zero.

3. Now **A** can follow the generic idea to complete the attack.

The attack complexity is again $q_d q_p \approx O(2^c)$.

B.1 Int-RUP Attack on SPoC

SPoC is a recent permutation-based submission to the NIST Lightweight competition by AlTawy et al. [1]. In contrast to the common duplex mode, SPoC uses the capacity part of the state to derive ciphertext outputs from, while it still absorbs the message in the rate part. Figure 4 provides a schematic illustration. In SPoC, the adversary cannot fix any part of the state in contrast to the common duplex and Beetle. Though, there exists a similar attack as follows:

1. **A** asks q_d queries (N, A, C^1), (N, A, C^2), ... , (N, A, C^{q_d}) to the decryption oracle and receives $M^1, M^2, \ldots, M^{q_d}$. The associated data A and ciphertext C^i consist of a single block for every i. This ensures that the first r bits of the input to the third permutation is always equal to $M^1 \oplus C^1$.

2. Now **A** can follow the generic idea to complete the attack.

So, the attack needs $q_d q_p \in O(2^c)$ to work, as before.

B.2 Int-RUP Attack on A Hybrid of Beetle and SPoC

We can generalize our attacks to hybrid modes of Beetle and SPoC as well. Such a hybrid would use both modes Beetle and SPoC in parallel to process the queries. Each message block (say M) is parsed into two sections (say M^1 and M^2), where $|M^1| = r_1$ and $|M^2| = r_2$. M^1 is processed with Beetle to a ciphertext block C^1; M^2 is processed with SPoC to a ciphertext block C^2. The final ciphertext block generated is $C \leftarrow C^1 \,\|\, C^2$. The associated-data blocks and the ciphertext blocks for decryption are also processed in the same manner. Note that the hybrid mode is parameterized by r_1, r_2 and c with the condition $c \geq r_2$. The size of rate and capacity of the Beetle part are r_1 and $c - r_2$; the size of both rate and capacity of the SPoC part is r_2. As a result, the size of rate and capacity

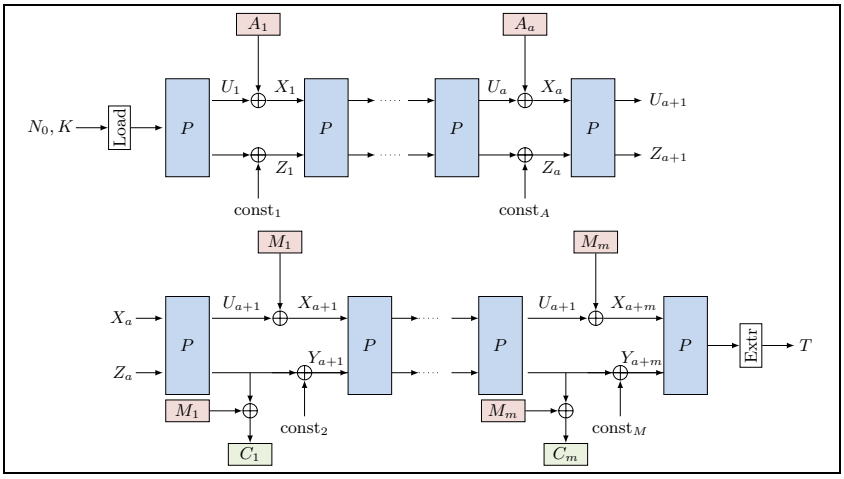

Figure 4: The SPoC mode of operation.

of the hybrid mode is $r = r_1 + r_2$ and c. When $r_2 = 0$, the hybrid mode translates to the Beetle mode. Similarly, when $r_1 = 0$, the hybrid mode is equivalent to the SPoC mode. An Int-RUP attack on such modes could be defined as follows:

1. **A** asks q_d decryption queries (N, A^1, C), (N, A^2, C), …, (N, A^{q_d}, C) to the decryption oracle and receives $M^1, M^2, \ldots, M^{q_d}$. The ciphertext C and associated data A^i consist of a single block for every i.

2. There exists at least one value of the last r_2 bits of the input to the third permutation which remains same for at least $q = \frac{q_d}{2^{r_2}}$ queries. Suppose q such queries are $(N, A^{1'}, C)$, $(N, A^{2'}, C)$, …, $(N, A^{q'}, C)$.

3. **A** can detect the previous step as it knows the value of the last r_2 bits of the input to the third permutation because that will be equal to the last r_2 bits of $C \oplus M^i$.

4. **A** retains those q queries and discards the rest.

5. For each of the above queries, **A** updates the value of the first r_1 bits of the ciphertext to $Y_2 \oplus \mathsf{shuffle}(Y_2)$ and varies the remaining r_2 bits. This ensures that the first r_1 bits of the input to the third permutation is always equal to zero.

6. In the above mentioned way, **A** can ask q_d more decryption queries to the decryption oracle. This time, it is ensured that a total of r bits (first r_1 bits and last r_2 bits) of the input to the third permutation are fixed and known to **A**.

7. Then, **A** can follow the generic idea to complete the attack.

The attack needs again $q_d q_p \in O(2^c)$ as before.

C Int-RUP Attacks on Schemes with Masked Ciphertexts

This section outlines more attacks on masked variants of the permutation-based AE schemes Beetle and SPoC.

C.1 Int-RUP Attack on The Masked Beetle

1. The adversary **A** asks q_d encryption queries (N^1, A^1, M), (N^2, A^2, M), ... , (N^{q_d}, A^{q_d}, M) to the encryption oracle, and receives C^1, C^2, ..., C^{q_d}. The associated data A^i consist of a single block for each i; the message M contains $\lceil \frac{c}{r} \rceil$ blocks.

2. **A** asks $q_d - 1$ decryption queries, one corresponding to each encryption query except the first encryption query, to the decryption oracle. The decryption query corresponding to the i-th encryption query is $(N^i, A^i, C^{i'})$, where

$$C^{i'} = Y_2^1 \oplus Y_2^i \oplus \mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i)\,.$$

   A can calculate the right-hand side of the equation as

$$\mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i) = C^1 \oplus C^i\,,$$

   and $Y_2^1 \oplus Y_2^i$ can be computed directly from $\mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i)$ with the definition of the function $\mathsf{shuffle}$. This makes the first r bits of the input to the third permutation always equal to that of the first encryption query.

3. Afterwards, **A** repeats Step 2 to 6 from Section 6 to complete the attack.

The attack is successful for $q_d^2 \approx O(2^c)$. However, the attack strategy differs for SPoC.

C.2 Int-RUP Attack on The Masked SPoC

Here, **A** has to perform the attack in two stages.

1. First, **A** asks q_d decryption queries (N, A^1, C), (N, A^2, C), ... , (N, A^{q_d}, C) to the decryption oracle, and receives M^1, M^2, ..., M^{q_d}. The associated data A^i consists of a single block for each i; the ciphertext C consists of two blocks.

2. When $q_d \approx \mathcal{O}(2^r)$, **A** expects a collision in the first r bits of the input to the third permutation.

3. **A** can detect this collision by looking at the first message block because it will be equal only for the two colliding queries.

4. Suppose the associated data of the two colliding queries are A^i and A^j.

5. **A** makes q_1 decryption queries (N, A^i, C^1), (N, A^i, C^2), ... , (N, A^i, C^{q_1}), and q_2 decryption queries (N, A^j, C^1), (N, A^j, C^2), ... , (N, A^j, C^{q_2}) to the decryption oracle.

6. When $q_1 \cdot q_2 \approx \mathcal{O}(2^r)$, **A** expects a full state collision at the input to the third permutation, between one query with associated data A^i and another query with associated data A^j.

7. Suppose the two ciphertexts corresponding to the two colliding queries are C^p and C^q, and the corresponding messages are M^p and M^q.

8. **A** can detect this collision in the following way: **A** identifies those pairs of queries $(N, A^i, C^x), (N, A^j, C^y), 1 \le x \le q_1, 1 \le y \le q_2$, for which the sum of the second message blocks is equal to the sum of the first message blocks. For each such pair, **A** updates C^x and C^y by appending $\lceil \frac{c}{r} \rceil - 1$ ciphertext blocks to each of them such that, $C_2^x = C_2^y, \ldots, C^x_{\lceil \frac{c}{r} \rceil} = C^y_{\lceil \frac{c}{r} \rceil}$, and makes decryption queries using these updated ciphertext values. The ciphertexts C^p and C^q, for all $k > 2$, M_k^p, will be equal to M_k^q.

9. Next, $\mathbf{A}$ asks (N, A^i, M^p) to the encryption oracle; suppose, the tag is T.
10. Then, $\mathbf{A}$ submits the forgery query (N, A^j, C^q, T) to the verification oracle, which is a successful forgery.

Again, the probability for forgeries becomes non-negligible when $q_d^2 \approx O(2^c)$.

D Details of the nAE Analysis of Oribatida

D.1 Proof of Lemma 1

Proof. In the following, we upper bound the probabilities of the individual bad events.

bad_1: Multi-collision on X in encryption construction queries. In the ideal world, the ciphertext blocks are sampled independently uniformly at random from the strings of expected length. The internal values X_j^i can be computed by $\mathbf{A}$ once it is given the transcript including the internal chaining values V_j^i. It must hold that $X_j^i \leftarrow C_{j-a^i}^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j-a^i)}^i)$. The random sampling of C implies that the probability of the values X_j^i to take any specific r-bit value is $1/2^r$. Note that in the case of a padded ciphertext block, each padded bit of $C_{m^i}^i$ is also sampled once randomly and given in the transcript. Hence, the probability for $X_{a^i+m^i}^i$ to take any r-bit value is also 2^{-r} in the ideal world. For fixed indices (i_1, j_1), (i_2, j_2), $\ldots$, (i_w, j_w), it holds that

$$\Pr\left[X_{j_1}^{i_1} = X_{j_2}^{i_2} = \ldots = X_{j_w}^{i_w}\right] \leq 2^{-r(w-1)}.$$

Over all queries and blocks of τ_e, it follows that

$$\Pr\left[\mathsf{bad}_1\right] \leq \frac{\binom{\sigma}{w}}{2^{r(w-1)}}.$$

bad_2: Collision of two permutation inputs in encryption construction queries. Here, we consider

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right].$$

All ciphertext blocks and the internal chaining values $V_{\rho(i,j-a^i)}^i$, $j > a^i$ are sampled independently and uniformly at random. Moreover, padded bits of ciphertexts are sampled also independently and uniformly at random. Though, we have to consider two cases;

- $j = 0 \wedge j' = 0$: since X_0^i and $X_0^{i'}$ contain nonces, and since we assume $\mathbf{A}$ to be nonce-respecting, the probability for a collision is zero in this case.
- $j > 0$: In this case, $Y_j^i = V_j^i \oplus \text{const}$, where $\text{const} \in \{d_N, d_A, d_E\}$ is a public constant. Moreover, X_j^i is derived from C_j^i; so, both X_j^i and Y_j^i are chosen independently and uniformly at random, and the probability for a collision in this case is at most 2^{-n}.

Therefore, for fixed indices $(i, j) \neq (i', j')$, the probability is

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq 2^{-n}.$$

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_2\right] \leq \frac{\binom{\sigma}{2}}{2^n}.$$

bad_3: Collision of two permutation outputs in encryption construction queries. This case is analogous to bad_2. The permutation outputs V_j^i are sampled randomly. In all cases, it holds that

$$\Pr\left[\left(U_j^i \,\|\, V_j^i\right) = \left(U_{j'}^{i'} \,\|\, V_{j'}^{i'}\right)\right] \leq 2^{-n}.$$

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_3\right] \leq \frac{\binom{\sigma}{2}}{2^n}.$$

bad_4: Collision of permutation inputs between a construction and a primitive query. Again, we consider $X_j^i \leftarrow C_{j-a^i}^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j-a^i)}^i)$ and $Y_j^i \leftarrow V_j^i \oplus \mathrm{const}$, where const is a public constant. The values $C_{j-a^i}^i$ and $V_{\rho(i,j-a^i)}^i$ are sampled randomly, the values $(X_j^i \,\|\, Y_j^i)$ take any value with probability at most 2^{-n}.

a) Assume, the primitive query was asked before the construction query. If the construction query was in encryption direction, the collision probability for fixed queries is at most 2^{-n}, for $q_p \cdot q_c$ combinations.

b) If the primitive query was asked after an encryption query, then, the latter one produced a tag. If the primitive query starts at any other block, $\mathbf{A}$ can see $r-s$ bits. Hence, the probability is at most $2^{-(c+s)}$ for $q_p \cdot \sigma_e$ combinations. If the primitive query starts from the tag, the adversary sees τ unmasked bits. Assuming $\overline{\mathsf{bad}}_1$, there are at most w equal tags over all encryption queries. So, the probability for a collision is $2^{-(n-\tau)}$, for $w \cdot q_p$ combinations.

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_4 | \overline{\mathsf{bad}}_1\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{w \cdot q_p}{2^{n-\tau}} \leq \frac{\sigma_e \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}}.$$

bad_5: Collision of permutation outputs between an encryption construction query and a primitive query. Again, $U_{j+a^i}^i$ can be derived from $M_j^i \oplus C_j^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j)}^i)$ and the values C_j^i and $V_{\rho(i,j)}^i$ are sampled randomly. So, the values $U_{j+a^i}^i \,\|\, V_{j+a^i}^i$ take any value with probability at most 2^{-n}. If the primitive query starts at any other block, $\mathbf{A}$ can see $r-s$ bits. Hence, the probability is at most $2^{-(c+s)}$ for $q_p \cdot \sigma_e$ combinations. Following a similar argument as for bounding bad_4 and excluding bad_1, we obtain over all combinations of indices that

$$\Pr\left[\mathsf{bad}_5 | \overline{\mathsf{bad}}_1\right] \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}}.$$

bad_6: Initial-state collision with a primitive query. Here, we know that the key is chosen uniformly at random. We distinguish between collisions depending on whether the primitive query was a forward query or a backward query.
If the primitive query was a forward query, it must hit the correct value of $K \oplus_d d_N$. So, the probability is at most $q_p/2^k$ to collide with encryption construction queries. Considering also decryption queries, a nonce can repeat but change d_N. Since there exist at most three distinct values for d_N, the probability is at most $3q_p/2^k$ to collide.
If the primitive query was in backward direction, its response must hit any initial state of a construction query. If the construction query was asked before the primitive, $\mathbf{A}$ sees at best $r-s$ bits of C_1. Then, the probability is at most $q_c \cdot q_p/2^{c+s}$.

If the primitive query was asked before the construction query, $\mathbf{A}$ can use the nonce part of the primitive query's result as nonce. Though, a collision needs the key part to be correct, which holds with probability at most $3q_p/2^k$. Over all possible options, we obtain

$$\Pr\left[\mathsf{bad}_6\right] \leq \frac{3q_p}{2^k} + \frac{q_c \cdot q_p}{2^{c+s}}.$$

bad_7: Multi-collision in the rate part of w outputs of forward primitive queries. Since π is chosen randomly from the set of all permutations, the outputs are chosen randomly from a set of size $2^n - (i-1)$ for the i-th primitive query. So, the probability for w distinct queries to collide in their rate part is at most $1/2^{r(w-1)}$ as for bad_7 in the nAE proof. Over all queries, the probability is upper bounded by

$$\Pr\left[\mathsf{bad}_7\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}}.$$

bad_8: Multi-collision in the rate part of w outputs of backward primitive queries. Following a similar argumentation as for bad_7, we obtain

$$\Pr\left[\mathsf{bad}_8\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}}.$$

Our bound in Lemma 1 follows from summing up all probabilities.

D.2 Proof of Lemma 2

Proof. It remains to lower bound the ratio of real and ideal probability of obtaining a good transcript τ. Let $\tau = (\tau_e, \tau_d, \tau_p)$ be an attainable transcript, where $\tau_d = \bot_{\text{all}}$ contains only $\bot$ for all responses. Since all ciphertext-block outputs and all internal chaining values in encryption queries are sampled independently and uniformly at random, their probability is $1/2$ per bit. We define $\sigma_{\mathsf{distinct}}$ for the number of distinct calls to the permutation over all encryption and decryption queries. In the ideal world, it holds that

$$\begin{aligned}\Pr\left[\Theta_{\text{ideal}} = \tau\right] &= \Pr\left[K\right] \cdot \Pr\left[\tau_e \wedge \tau_p \wedge \tau_d\right] \\ &= \Pr\left[K\right] \cdot \Pr\left[\tau_e\right] \cdot \Pr[\tau_p] \cdot \Pr[\tau_d] = \frac{1}{2^k} \cdot \frac{1}{(2^n)^{\sigma_{\mathsf{distinct}}}} \cdot \frac{1}{(2^n)_{q_p}} \cdot 1\end{aligned}$$

since the outputs from encryption queries are sampled uniformly at random; so, the encryption and decryption transcripts τ_e and τ_d are independent from τ_p.
In the real world, the probabilities for choosing K as key and π as permutation are equal to those of the ideal world. We can separate the probability into

$$\Pr\left[\Theta_{\text{real}} = \tau\right] = \Pr\left[K\right] \cdot \Pr\left[\tau_e \wedge \tau_p \wedge \tau_d\right] = \frac{1}{2^k} \cdot \Pr\left[\tau_e, \tau_d | \tau_p\right] \cdot \Pr\left[\tau_p\right]$$

since the encryption and the decryption transcript depend on the choice of the permutation π. Let $\top_i$ denote that the i-th decryption query was a valid forgery. We can upper bound

$$\begin{aligned}\Pr\left[\tau_e, \tau_d | \tau_p\right] \cdot \Pr\left[\tau_p\right] &\leq \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] - \left(\sum_{i=1}^{q_d} \Pr\left[\tau_e \wedge \top_i | \tau_p\right] \cdot \Pr\left[\tau_p\right]\right) \\ &= \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] - \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] \cdot \left(\sum_{i=1}^{q_d} \Pr\left[\top_i | \tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right]\right) \\ &= \left(\Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right]\right) \cdot (1 - \epsilon)\,, \qquad (1)\end{aligned}$$

where we define

$$\epsilon \stackrel{\text{def}}{=} \sum_{i=1}^{q_d} \epsilon_i \qquad \text{and} \qquad \epsilon_i \stackrel{\text{def}}{=} \Pr\left[\top_i | \tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right].$$

The probability of primitive queries is given by the fraction of all permutations π that would produce τ_p, which is

$$\Pr[\tau_p] = \frac{1}{(2^n)_{q_p}},$$

as in the ideal world. The ciphertext blocks C_j^i from encryption queries as well as the chaining values V_j^i are results from the permutation π and hence, depend on the permutation. Since τ is a good transcript, there are no undesired collisions, e.g., between primitive and construction queries. Hence, all internally computed values $(U_j^i \,\|\, V_j^i)$ – note that U_j^i can be derived from $C_{j-a^i}^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j-a^i)}^i) \oplus M_{j-a^i}^i$ by the adversary – are results of fresh values or predefined in decryption queries from the result of previous encryption queries. Then, the probabilities for the outputs of π in construction queries are given by $1/\left(2^n\right)_{\sigma_{\mathsf{distinct}}}$. It is not difficult to see that for positive $\sigma_{\mathsf{distinct}}$, the ratio of the interpolation probabilities from Equation (1) can be bounded by

$$\frac{\left(\Pr\left[\tau_e | \tau_p | \Theta_{\text{real}}\right] \cdot \Pr\left[\tau_p | \Theta_{\text{real}}\right]\right)}{\left(\Pr\left[\tau_e | \Theta_{\text{ideal}}\right]\right)} = \frac{\frac{1}{(2^n)_{\sigma_{\mathsf{distinct}}}}}{\frac{1}{(2^n)^{\sigma_{\mathsf{distinct}}}}} = \frac{(2^n)^{\sigma_{\mathsf{distinct}}}}{(2^n)_{\sigma_{\mathsf{distinct}}}} \geq 1\,.$$

It remains to upper bound ϵ. For this purpose, we upper bound the values ϵ_i for transcripts that contain forgeries. Since τ is a good transcript, we assume that bad events do not hold. Hence, either $\top_i$ does not hold, which yields $\epsilon_i = 0$; in the opposite case, we have to consider a few mutually exclusive cases in the following. We assume that there exists a decryption query (N^i, A^i, C^i, T^i) s. t. T^i is valid. In all cases, the tag can simply be guessed correctly if the block $(X_{a^i+m^i}^i \,\|\, Y_{a^i+m^i}^i)$ is fresh. Then, the probability for the tag to be correct is $2^{-\tau}$. So, we can concentrate on the cases where it is non-fresh in the following. Prior, we define $(X_{i_1}, X_{i_2}, \ldots, X_{i_{w+1}})$ as a w-chain if there exist $(Y_{i_1}, Y_{i_2}, \ldots, Y_{i_{w+1}})$ s. t. the following chain has been obtained from primitive queries:

$$\begin{aligned}
\pi\left(X_{i_1} \,\|\, Y_{i_1}\right) &= \left(U_{i_2} \,\|\, V_{i_2}\right) = \left(U_{i_2} \,\|\, Y_{i_2}\right), \\
\pi\left(X_{i_2} \,\|\, Y_{i_2}\right) &= \left(U_{i_3} \,\|\, V_{i_3}\right) = \left(U_{i_3} \,\|\, Y_{i_3}\right), \\
&\vdots \\
\pi\left(X_{i_w} \,\|\, Y_{i_w}\right) &= \left(U_{i_{w+1}} \,\|\, V_{i_{w+1}}\right) = \left(U_{i_{w+1}} \,\|\, Y_{i_{w+1}}\right).
\end{aligned}$$

The cases are:

- **Case (A):** N^i is fresh; so, there is no earlier construction query $i' \neq i$ s. t. $N^i = N^{i'}$.
- **Case (B):** N^i is old, but (N^i, A^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i) = (N^{i'}, A^{i'})$.
- **Case (C):** (N^i, A^i) is old, but (N^i, A^i, C^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i, C^i) = (N^{i'}, A^{i'}, C^{i'})$, and no w-chain of primitive queries is hit.
- **Case (D):** (N^i, A^i, C^i) is old; (N^i, A^i, C^i) a prefix of another construction query.
- **Case (E):** (N^i, A^i) is old and there exists a w-chain of primitive queries that is hit.

Clearly, the cases cover all possible options. We assume that no previous bad events occur, in particular, no w-multi-collisions or collisions with primitive queries occurred.

Case (A). We excluded bad_6 in this case. The probability that $(N^i \,\|\, K) \oplus_d d_N$ hits any block $(X_j^{i'} \,\|\, Y_j^{i'})$ from another construction query so that the final block is old is at most

$$\Pr\left[\left((N^i \,\|\, K) \oplus_d d_N\right) = \left(X_j^{i'} \,\|\, Y_j^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}}.$$

Case (B). Let $p \leq a^i + m^i$ denote the length of the longest common prefix of the i-th query with all other queries. In Case (B), the probability that any block $(X_j^i \,\|\, Y_j^i)$ with $j \geq p+1$ matches the permutation input of any other encryption-query block or primitive query can be upper bounded by

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}}.$$

Case (C). A similar argument as for Case (B) can be applied in Case (C). The probability that there exists $i' \neq i$, s. t. for some block indices, it holds that (j, j'): $(X_j^i \,\|\, Y_j^i) = (X_{j'}^{i'} \,\|\, Y_{j'}^{i'})$ is at most $1/2^{c+s}$. So, it holds that

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}}.$$

Case (D). This case needs that $(X_{a^i+m^i+1}^i \,\|\, Y_{a^i+m^i+1}^i)$ matches the permutation input of any other encryption-query block or primitive query. The probability can be upper bounded by

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}}.$$

Case (E). Assume that $(X_{p+1}^i \,\|\, Y_{p+1}^i)$ hits a w-chain of primitive queries. Under the assumption that no other bad events occurred, the probability is at most

$$\Pr\left[\left(X_{p+1}^i \,\|\, Y_{p+1}^i\right) \,\middle|\, \bigwedge_{i=1}^{8} \overline{\mathsf{bad}_i}\right] \leq \frac{(m^i+1)\cdot w}{2^{c+s}}.$$

Over all decryption queries, we obtain

$$\epsilon \leq \sum_{i=1}^{q_d} \left(\frac{1}{2^\tau} + \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}} + \frac{(m^i+1)\cdot w}{2^{c+s}}\right) \leq \frac{q_d}{2^\tau} + \frac{q_d(q_p+\sigma_e)}{2^{c+s}} + \frac{(\sigma_d+q_d)\cdot w}{2^{c+s}}.$$

Our claim in Lemma 2 follows.

E Details of the Int-RUP Analysis of Oribatida

E.1 Proof of Lemma 3

Proof. In the following, we upper bound the probabilities of the individual bad events. For the most of them, we differentiate between encryption and decryption queries.

bad_1: Collision of two permutation inputs in construction queries.

a) Among encryption queries only: here, it holds that

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq 2^{-n}.$$

Since there exist $\binom{\sigma_e}{2}$ block combinations, we obtain $\binom{\sigma_e}{2}/2^n$.

b) Dec-then-Enc: If we consider an encryption query block to collide with a block from a previous decryption query, the probability is at most $2^{-(c+s)}$ since $\mathbf{A}$ can see $r-s$ bits that it can use as nonce. We have $q_e\sigma_d$ combinations of such blocks. For the remaining $\sigma_e\sigma_d$ blocks, the probability is 2^{-n}.

c) Among decryption queries only: w.l.o.g., we consider the first such collision. If $\mathbf{A}$ modifies the nonce in the later query, the bound is the same as for encryption-only queries. So, we assume in the remainder of that the later query is a decryption query. Let $j-1$ be the first modified block and assume it is in the message-processing part. If the block indices differ $j \neq j'$, the probability is $2^{-(c+s)}$. Otherwise, assume $j = j'$ and $A^i = A^{i'}$. Then, the permutation output $(U_j^{i'} \,\|\, V_j^{i'})$ is sampled randomly in the ideal world. If $\mathbf{A}$ leaves $C^i_{j-a^i} = C^{i'}_{j-a^i}$, it automatically holds that

$$X_j^i = C^i_{j-a^i} \oplus \mathsf{lsb}_s(V^i_{\rho(i,j-a^i)}) = X_j^{i'} = C^i_{j-a^i} \oplus \mathsf{lsb}_s(V^i_{\rho(i,j-a^i)}).$$

So, $X_j^i = X_j^{i'}$. With probability 2^{-c}, it also holds for the capacity part $V^i_{j+1} = V^{i'}_{j+1}$ and thus $Y^i_{j+1} = Y^{i'}_{j+1}$. Note that this approach holds only for the first differing block, for which which yields a term of $\binom{q_d}{2}/2^c$. if the collision does not hold, the masks beginning for the $(j+2)$-th block will differ and the probability decreases to $2^{-(c+s)}$, which produces a term of $\binom{\sigma_d}{2}/2^{c+s}$.

d) Enc-then-Dec: It remains to consider collisions between an encryption query, followed by a decryption query. If the block indices $j \neq j'$ differ, the probability is again $2^{-(c+s)}$, for at most $\sigma_e \cdot \sigma_d$ combinations. Otherwise, if $j = j'$, $\mathbf{A}$ can apply the strategy above for a collision. Then, the probability is 2^{-c}; though, the q_d queries can collide at best with most encryption query each since we consider the first collision, producing a term of $q_d/2^c$.

Over all cases, we obtain

$$\Pr\left[\mathsf{bad}_1\right] \leq \frac{\binom{\sigma_e}{2}}{2^n} + \max\left(\frac{q_e\sigma_d}{2^{c+s}} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{q_d}{2^c}\right) + \frac{\binom{\sigma_d}{2}}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} \leq \frac{\binom{\sigma_e}{2}}{2^n} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c}.$$

bad_2: Collision of two permutation outputs in encryption construction queries. This case is analogous to bad_1. Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_2\right] \leq \frac{\binom{\sigma_e}{2}}{2^n} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c}.$$

bad_3: Multi-collision on w tags from encryption queries. Since the tags are sampled uniformly and independently at random in the ideal world, it holds that

$$\Pr\left[T^{i_1}_{j_1} = T^{i_2}_{j_2} = \ldots = T^{i_w}_{j_w}\right] \leq \frac{\binom{q_e}{w}}{2^{\tau(w-1)}}.$$

bad_4: Collision of permutation inputs between a construction and a primitive query. Again, we consider $X_j^i \leftarrow C^i_{j-a^i} \oplus_s \mathsf{lsb}_s(V^i_{\rho(i,j-a^i)})$ and $Y_j^i \leftarrow V_j^i \oplus \mathrm{const}$, where const is a public constant. The values $C^i_{j-a^i}$ and $V^i_{\rho(i,j-a^i)}$ are sampled randomly, the values $(X_j^i \,\|\, Y_j^i)$ take any value with probability at most 2^{-n}.

a) Assume, the primitive query was asked before the construction query. If the construction query was in encryption direction, the collision probability for fixed queries is at most 2^{-n}, for $q_p \cdot q_c$ combinations.

b) Otherwise, if the construction query was a decryption query, **A** can see $r - s$ bits. Hence, the probability is at most $2^{-(c+s)}$, for $q_p \cdot q_c$ combinations.

c) The same argument can be applied in the case when the primitive query was asked after a decryption query. Then, the adversary can see $r-s$ unmasked bits of the rate from C^i_j. Again, the probability is at most $2^{-(c+s)}$ and we have $q_p \cdot q_c$ combinations.

d) If the primitive query was asked after an encryption query, then, the latter produced a tag. If the primitive query targets any other block, the argument is the same as in Case c). If the primitive query starts from the tag, the adversary sees τ unmasked bits. Assuming $\overline{\mathsf{bad}}_3$, there are at most w equal tags over all encryption queries. So, the probability for a collision is $2^{-(n-\tau)}$, for $w \cdot q_p$ combinations.

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_4 | \overline{\mathsf{bad}}_3\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{\sigma_d \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}} \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}}.$$

bad_5: Collision of permutation outputs between a construction and a primitive query. Again, $U^i_{j+a^i}$ can be derived from $M^i_j \oplus C^i_j \oplus_s \mathsf{lsb}_s(V^i_{\rho(i,j)})$; the values C^i_j and $V^i_{\rho(i,j)}$ are sampled randomly. This case is similar as bad_4. Over all index combinations

$$\Pr\left[\mathsf{bad}_5 | \overline{\mathsf{bad}}_3\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{\sigma_d \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}} \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}}.$$

bad_6: Initial-state collision with a primitive query. Here, we distinguish between the cases whether the construction query was asked before or after the primitive query, and whether the primitive query was in forward or backward direction.

a) Assume, the primitive query was asked after the construction query. If the primitive query was a forward query, it must hit the correct value of $K \oplus_d d_N$. This probability is at most $q_p/2^k$ to collide when considering encryption construction queries. Considering also decryption queries, a nonce can repeat often; though, the initial state can take three different values for the same nonce, namely if the decryption query changes the length of associated data and message, affecting d_N. Since there exist at most three distinct values for d_N, the probability to collide is at most $3q_p/2^k$.

b) If the primitive query was in backward direction, its response must hit any initial state of a construction query. If the construction query was asked before the primitive, **A** sees at best $r - s$ bits of C_1. Then, the probability is at most $q_c \cdot q_p/2^{c+s}$.

c) If the primitive query was asked before the construction query, **A** can use the nonce part of the primitive query's result as nonce. However, a collision must hit the key part, which holds with probability at most $3q_p/2^k$.

d) If the primitive query was in backward direction, **A** sees at best $r - s$ bits of C_1. Then, there is at most one starting state, assuming $\overline{\mathsf{bad}}_1$, which yields $q_p/2^{c+s}$.

Over all possible options, we obtain

$$\Pr\left[\mathsf{bad}_6 | \overline{\mathsf{bad}}_1\right] \leq \frac{3q_p}{2^k} + \max\left(\frac{q_c \cdot q_p}{2^{c+s}}, \frac{q_p}{2^{c+s}}\right) \leq \frac{3q_p}{2^k} + \frac{q_c \cdot q_p}{2^{c+s}} + \frac{q_p}{2^{c+s}}\,.$$

bad_7: Multi-collision in the rate part of w outputs of forward primitive queries. Since π is chosen randomly from the set of all permutations, the outputs are sampled uniformly at random from a set of size at least $2^n - (i-1)$ for the i-th query. So, the probability for w distinct queries to collide in their rate part is upper bounded by

$$\frac{2^c-1}{2^n-1} \cdot \frac{2^c-2}{2^n-2} \cdot \dots \cdot \frac{2^c-(w-1)}{2^n-(w-1)} = \prod_{i=1}^{w-1} \frac{2^c-i}{2^n-i} \leq \left(\frac{2^c}{2^n}\right)^{w-1} = 2^{-r(w-1)}.$$

Over all primitive query indices, it holds that

$$\Pr\left[\mathsf{bad}_7\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}}.$$

bad_8: Multi-collision in the rate part of w outputs of backward primitive queries. Using a similar argumentation as for bad_7, we obtain

$$\Pr\left[\mathsf{bad}_8\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}}.$$

bad_9: Forgeries if all blocks are old. It remains to bound the probability of a successful forgery of a verification query (N^i, A^i, C^i, T^i) s. t. T^i is valid and where each block is old. We consider the same five mutually exclusive cases as in the nAE proof. In all cases, the tag can simply be guessed correctly if the block $(X^{a^i+m^i} \,\|\, Y^{a^i+m^i})$ is fresh. Then, the probability for the tag to be correct is upper bounded by $2^{-\tau}$. We adopt the cases and the notions from the nAE proof and assume that no previous bad events occur, in particular no w-multi-collisions described earlier or collisions with primitive queries.

- **Case (A):** N^i is fresh; so, there is no earlier construction query $i' \neq i$ s. t. $N^i = N^{i'}$.
- **Case (B):** N^i is old, but (N^i, A^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i) = (N^{i'}, A^{i'})$.
- **Case (C):** (N^i, A^i) is old, but (N^i, A^i, C^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i, C^i) = (N^{i'}, A^{i'}, C^{i'})$, and no w-chain of primitive queries is hit.
- **Case (D):** (N^i, A^i, C^i) is old; (N^i, A^i, C^i) is a prefix of another construction query.
- **Case (E):** (N^i, A^i) is old and there exists a w-chain of primitive queries that is hit.

Clearly, the cases cover all possible options. We assume that no previous bad events occur, in particular no w-multi-collisions described earlier or collisions with primitive queries.

Case (A). We excluded bad_4, i.e., collisions of permutation inputs between construction and primitive queries in this case. The probability that $(N^i \,\|\, K) \oplus_d d_N$ hits any block $(X_j^{i'} \,\|\, Y_j^{i'})$ from another construction query so that the final block is old is at most

$$\Pr\left[\left((N^i \,\|\, K) \oplus_d d_N\right) = \left(X_j^{i'} \,\|\, Y_j^{i'}\right)\right] \leq \frac{\sigma + q_p}{2^{c+s}}.$$

Cases (B)–(D). Let $p \leq a^i + m^i$ denote the length of the longest common prefix of the i-th query with all other queries. The probability that any block $(X_j^i \,\|\, Y_j^i)$ with $j \geq p+1$ matches the permutation input of any other query block or primitive query can be upper bounded analogously as bad_1 and bad_4:

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e + q_d}{2^{c+s}} + \frac{\sigma_e \cdot \sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} + \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}}.$$

Case (E). Assume that $(X^i_{p+1} \,\|\, Y^i_{p+1})$ hits a w-chain of primitive queries. Under the assumption that no other bad events occurred, the probability is at most

$$\Pr\left[(X^i_{p+1} \,\|\, Y^i_{p+1})\right] \leq \frac{(m^i+1)\cdot w}{2^{c+s}}.$$

Over all verification queries, we obtain

$$\Pr\left[\mathsf{bad}_9 \,\middle|\, \bigwedge_{i=1}^{8} \overline{\mathsf{bad}}_i\right] \leq \frac{q_v}{2^\tau} + \frac{\sigma_e \cdot \sigma_d}{2^{c+s}} + \frac{\binom{q_d+q_v}{2}}{2^c} + \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}} + \frac{w \cdot (\sigma_d + q_d)}{2^{c+s}}.$$

Our bound in Lemma 3 follows from summing up all probabilities.

E.2 Proof of Lemma 4

Proof. It remains to bound the ratio of real and ideal probability of obtaining a good transcript τ. The bound is similar to that of Lemma 2. The difference to the nAE proof is that the ideal decryption oracle also generates pseudorandom output blocks M^i_j beyond the longest common prefix. The nAE transcript also contained the sampled internal values, as does the transcript τ here. Since we assume that no bad events have occurred, we revisit the following cases for forgeries:

- **Case (A):** The final input to π, $(X^i_{a^i+m^i} \,\|\, Y^i_{a^i+m^i})$ is fresh, i.e., has not occurred before. Then, the probability that the authentication tag τ^i is valid is at most $1/2^\tau$.
- **Case (B):** The final input to π, $(X^i_{a^i+m^i} \,\|\, Y^i_{a^i+m^i})$ is old, but there exists some block index $j \in [1..a^i+m^i]$ s. t. $(X^i_j \,\|\, Y^i_j)$ is fresh. Since the input is old, the probability that the result of any of the next blocks is old is at most $\frac{(\sigma+q_p)}{2^{c+s}}$.
- **Case (C):** There exists no $j \in [1..a^i+m^i]$ s. t. $(X^i_j \,\|\, Y^i_j)$ is fresh. The probability that all of those blocks are old is at most $\frac{m^i(\sigma_e+q_p)}{2^{c+s}}$.

It follows that

$$\epsilon_i \leq \frac{1}{2^\tau} + \frac{\sigma_e + q_p}{2^{c+s}} + \frac{m^i(\sigma_e+q_p)}{2^{c+s}}.$$

Over all indices $i \in [1..q_d]$, it follows that

$$\epsilon \leq \sum_{i=1}^{q_d} \epsilon_i \leq \frac{q_d}{2^\tau} + \frac{\sigma_d(\sigma_e+q_p)}{2^{c+s}}.$$

Our claim in Lemma 4 follows. □