IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–23. DOI:XXXXXXXX

On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis

Dhiman Saha[1], Yu Sasaki[2], Danping Shi[3,4], Ferdinand Sibleyras[5], Siwei Sun[3,4] and Yingjie Zhang[3,4]

[1] de.ci.phe.red Lab, Department of Electrical Engineering and Computer Science, IIT Bhilai, Chhattisgarh, India, dhiman@iitbhilai.ac.in
[2] NTT Secure Platform Laboratories, Tokyo, Japan, yu.sasaki.sk@hco.ntt.co.jp
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, {shidanping,zhangyingjie}@iie.ac.cn,siweisun.isaac@gmail.com
[4] University of Chinese Academy of Sciences, Beijing, China
[5] Inria, Paris, France ferdinand.sibleyras@inria.fr

Abstract. This paper presents the first third-party security analysis of TinyJAMBU, which is one of 32 second-round candidates in NIST's lightweight cryptography standardization process. TinyJAMBU adopts an NLFSR based keyed-permutation that computes only a single NAND gate as a non-linear component per round. The designers evaluated the minimum number of active AND gates, however such a counting method neglects the dependency between multiple AND gates. There also exist previous works considering such dependencies with stricter models, however those are known to be too slow. In this paper, we present a new model that provides a good balance of efficiency and accuracy by only taking into account the first-order correlation of AND gates that frequently occurs in TinyJAMBU. With the refined model, we show a 338-round differential with probability $2^{-62.68}$ that leads to a forgery attack breaking 64-bit security. This implies that the security margin of TinyJAMBU with respect to the number of unattacked rounds is approximately 12%. We also show a differential on full 384 rounds with probability $2^{-70.64}$, thus the security margin of full rounds with respect to the data complexity, namely the gap between the claimed security bits and the attack complexity, is less than 8 bits. Our attacks also point out structural weaknesses of the mode that essentially come from the minimal state size to be lightweight.

Keywords: TinyJAMBU · NIST lightweight cryptography · AEAD · differential · linear · MILP · model

1 Introduction

National Institute of Standards and Technology (NIST) initiated a public competition-like process to solicit, evaluate, and standardize authenticated encryption and hashing schemes suitable for highly constrained computing environments like RFID tags, lightweight industrial controllers and sensor nodes [Nat19a]. In 2019, 57 submissions were received, among which 56 were accepted as the first-round candidates. On August 30, 2019, NIST announced the 32 second-round candidates, which were selected based on public feedbacks and internal reviews [Nat19b]. The evaluation process of the second round is expected to last twelve months. Therefore, there is a need to analyze the security of the second round candidates within this one-year time window for a better selection process.

Received: 20XX-XX-XX, Accepted: 20XX-XX-XX, Published: 20XX-XX-XX

TinyJAMBU is a family of lightweight authenticated encryption algorithms designed by Wu and Huang [WH19]. TinyJAMBU adopts a small variant of the JAMBU mode [WH15], which is the smallest authenticated encryption mode in the CAESAR competition [CAE] successfully making its way to the third round of the competition. TinyJAMBU further reduces the implementation cost and is thus an attractive design for a small implementation. Several remarkable features of TinyJAMBU are listed below.

Minimum State Size. In high-level, TinyJAMBU adopts a duplex construction [BDPA11] with a keyed-permutation, not a public permutation. In other words, it resembles the SAEB mode [NMSS18]. TinyJAMBU ensures 64-bit security for authenticity by using the state of 128 bits. The use of the 128-bit state for 64-bit security is very minimal because collisions of the state are inevitable.

No Key Schedule Algorithm. TinyJAMBU uses keyed permutations without key schedule algorithms, which saves the combinatorial logic and control circuit for the key schedule part. Moreover, the alternate use of the key bits without any key state update leaves open the possibility that the key XORs can be implemented by "direct wiring" without using key registers in certain implementations.

NLFSR Based Keyed-Permutation. The underlying keyed-permutation is very hardware-friendly. It is basically a 128-bit nonlinear feedback shift register. It computes only a single NAND gate in each clock, and this is the only non-linear component of the keyed-permutation. Besides, it computes only four XOR gates in each clock. We invite the readers to take a glance at Figure 2 in Section 2 to have a concrete feeling for the lightweightness of the design.

Provable Security. Privacy of the TinyJAMBU mode is proven against nonce-respecting adversaries in the chosen plaintext attack and authenticity is proven against nonce-misuse adversaries by assuming ideal behaviors of the underlying keyed-permutation.

Because of the highly optimized structure of TinyJAMBU both in the mode and primitive levels, cryptanalysis is important to have better understanding of its security.

Previous Cryptanalysis and Research Challenges. At the time of writing, the only existing security evaluation of TinyJAMBU is the one provided in the design document [WH19], which counts the number of active AND gates to find differential and linear trails with the minimum of such active gates by using Mixed Integer Linear Programming (MILP) [MWGP]. However, this kind of analysis is insufficient due to the following reasons.

- To count the number of active AND gates is insufficient to capture the correlation between multiple AND gates. It has been observed that such correlations have a significant impact on the actual probabilities of the differential or linear trails [KLT15, AEL+18]. Hence the provided evaluation may underestimate the probabilities of the trails or give impossible trails. As evidence, while the designers suggested the 384-round differential trail with probability 2^{-80} by regarding each AND gate independent, we confirmed that there is no such trail by taking into account the dependency.

- It is known that the effect of differentials, a cluster of multiple differential characteristics having the same input and output differential masks, can be strong [AK18], particularly for the simple computation structure of TinyJAMBU.

- Regarding the linear cryptanalysis, the designers evaluated the bias of the keyed permutation only in the setting that the attacker has access to all the input bits, which does not correspond to the actual attack setting.

Regarding the first issue, it is possible to evaluate the exact probability by using the techniques [SHW+15a, SHW+15b] developed for SIMON [BSS+13], which limits the search space to only valid trails. However, such models involve too many variables and constraints, and thus cannot be solved in practical time. As reported in [SHW+15a], such a precise trail search is useful to verify the validity of a given trail but not so efficient to find optimal ones.

To conclude, the main difficulty of the analysis of `TinyJAMBU` is to achieve both accuracy and efficiency simultaneously. The designers' approach is relatively fast but inaccurate, while the known accurate methods are too slow.

Our Contributions. In this paper, we present the first third-party security analysis of `TinyJAMBU` by applying differential and linear cryptanalysis. Our attacks in the AEAD setting are valid for all members of `TinyJAMBU`, because those exploit the computations (nonce-setup, associated data processing, or tag generation) in which the number of rounds of the underlying keyed-permutation is fixed to 384 for all members of `TinyJAMBU`.

We first present a refined MILP model that takes into account the first-order correlation that naturally occurs in `TinyJAMBU`. We observe that `TinyJAMBU` computes an AND operation $a \cdot b$ for two bits a, b in some round, and computes $b \cdot c$ for another bit c in another round. Those two AND operations have a strong correlation. When $(\Delta a, \Delta b, \Delta c) = (1, 0, 1)$, the output differences of those two AND gates are either both 0 with probability 2^{-1} or both 1 with probability 2^{-1}. Namely, two AND gates are controlled with probability 2^{-1}.[1] We present a model to take into account these correlated AND gates. The refined model hits a balance between accuracy and efficiency. The resulting models can be solved in reasonable time and most of the trails identified are valid.

We then apply the refined model to `TinyJAMBU` such that the input difference to the underlying keyed-permutation is injected from the input data blocks specified in the `TinyJAMBU` mode, thus the identified trails immediately lead to the forgery attacks. We evaluate the number of rounds that breaks the claimed 64-bit security against forgery attacks. We find a differential trail with probability 2^{-64} for 338 (out of 384) rounds, which involves 76 active AND gates but contains 12 correlated paired AND gates. After evaluating multiple trails with the same input and output differential masks, the differential probability is $2^{-62.68}$. Hence, the security margin with respect to the number of unattacked rounds is about 12%. Regarding the full 384 rounds, we find a differential trail with probability 2^{-74}, which involves 88 active AND gates but contains 14 correlated pairs. The differential probability turns out to be $2^{-70.68}$. Hence, the security margin with respect to the data complexity for the full rounds is less than 8 bits.

We also evaluate `TinyJAMBU` by removing the constraints on the active-bit positions of the input and output. Namely, we evaluate the underlying primitive as a standalone keyed-permutation. The designers only evaluated the minimum number of active AND gates for reduced-round versions: 4 for 192 rounds and 13 for 320 rounds. We find a differential trail for full rounds with probability 2^{-19} which involves 20 active AND gates but contains 1 correlated pair. Due to the practical probability, we verify this differential trail by using the implementation provided by the designers, which allows us to generate conforming pairs with probability close to the theoretical estimation.

Finally, we evaluate the linear cryptanalysis by applying the refined model. The designers evaluated in the setting that the input mask can be active in any bit, while the output mask can be active only in the 32 bits corresponding to the outer part of the mode. In the same setting, we show that the linear bias can be larger. For example, the bias of 2^{-12} for 256 rounds in [WH19] is improved to 2^{-10}, which is experimentally verified.

[1] With respect to controlling the difference propagation of two AND gates, one may feel some resemblance in collision attacks on SHA-1/MD5 [WYY05, WY05] and conditional cube attacks on Keccak [LDB+19], Ascon [LDW17], etc.

Our attacks also point out the structural weakness of the TinyJAMBU mode that processes a nonce and a tag in multiple blocks. Because of its small size, TinyJAMBU absorbs the input and squeezes the output up to 32 bits per block. Hence, it absorbs a 96-bit nonce in three blocks and squeezes a 64-bit tag in two blocks. Differential cryptanalysis can exploit three blocks of the nonce. The input and output differences of the primitive can be injected in the first two blocks, and we still have a space to choose a new value in the third block. Namely after finding two colliding nonces in the first two blocks, the attacker can reuse them in the nonce-respect manner by renewing the third nonce block. Linear cryptanalysis can exploit two blocks for the tag. Note that linear cryptanalysis requires the knowledge of the state value but TinyJAMBU protects the encryption part from linear cryptanalysis by computing at least 1,024 rounds of the underlying primitive. Because linear cryptanalysis does not need to choose the input value to the primitive, observing two blocks of the tag is sufficient. We believe that those will be good lessons for AEAD designers who process the nonce and the tag in multiple blocks.

Paper Outline. The rest of the paper is organized as follows. Basic notations and the design of TinyJAMBU will be introduced in Section 2. Previous and new MILP models to efficiently search for differential and linear trails will be explained in Section 3. Our results on differential cryptanalysis and linear cryptanalysis will be explained in Section 4 and Section 5, respectively. Finally, we will conclude this paper in Section 6.

2 Notations and the Specification of TinyJAMBU

Let $\mathbb{F}_2 = \{0, 1\}$ be the finite field with two elements. Given a vector or bit string $v \in \mathbb{F}_2^n$, $|v| = n$ denotes the size of v in bits. For a and $b \in \mathbb{F}_2$, $\bar{a}$, $a \oplus b$, ab denote the logical negation, exclusive-or, and logical AND, respectively. Also, a NAND gate sends its input bits a and b to $\overline{ab} = ab \oplus 1$. Given two vectors $u, v \in \mathbb{F}_2^n$, their inner product is denoted by $u \cdot v$.

TinyJAMBU [WH19], a small variant of JAMBU [WH15], is a family of authenticated encryption with associated data (AEAD) schemes submitted to the NIST Lightweight Cryptography (LWC) Standardization project, and it was selected as one of the 32 round-2 candidates in August, 2019 [Nat19a]. Under the control of a secret key $K = (k_{|K|-1}, k_{|K|-2}, \cdots, k_1, k_0) \in \mathbb{F}_2^{|K|}$, TinyJAMBU maps a message M, a nonce N, and an associated data A to a ciphertext C and an authentication tag T as shown in Figure 1.

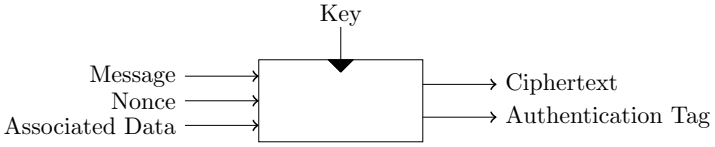

Figure 1: The high-level structure of the encryption algorithm of an AEAD scheme

The round function of TinyJAMBU computes a permutation $\mathcal{P}_b : \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ with $b \in \mathbb{F}_2$, which transforms a 128-bit state $(s_{127}, s_{126}, \cdots, s_1, s_0)$ to $(z, s_{127}, s_{126}, \cdots, s_2, s_1)$ with $z = s_0 \oplus s_{47} \oplus \overline{s_{70}s_{85}} \oplus s_{91} \oplus b$. This permutation is depicted in Figure 2. Let l be the number of rounds. TinyJAMBU computes the l-round transformation with a given key $K = (k_{|K|-1}, k_{|K|-2}, \cdots, k_1, k_0)$ as follows.

$$\prod_{i=0}^{l-1} \mathcal{P}_{k_i} = \mathcal{P}_{k_{l-1}} \circ \mathcal{P}_{k_{l-2}} \circ \cdots \circ \mathcal{P}_{k_1} \circ \mathcal{P}_{k_0}, \tag{1}$$

Table 1: The three variants of TinyJAMBU

AEAD	Sizes in bits				# of rounds	
	State	Key	Nonce	Tag	$\mathcal{P}_K$	$\hat{\mathcal{P}}_K$
TinyJAMBU-128	128	128	96	64	384	1024
TinyJAMBU-192	128	192	96	64	384	1152
TinyJAMBU-256	128	256	96	64	384	1280

where the subscripts of k are always computed modulo $|K|$, namely $k_i = k_{i \bmod |K|}$. We are now ready to describe the encryption algorithm of TinyJAMBU. Note that TinyJAMBU comes with three variants, whose parameters are listed in Table 1.

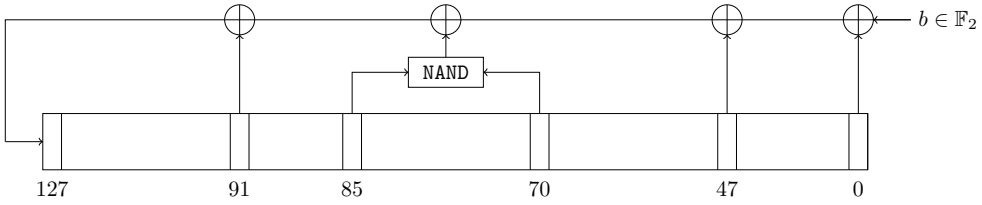

Figure 2: The permutation $\mathcal{P}_b$

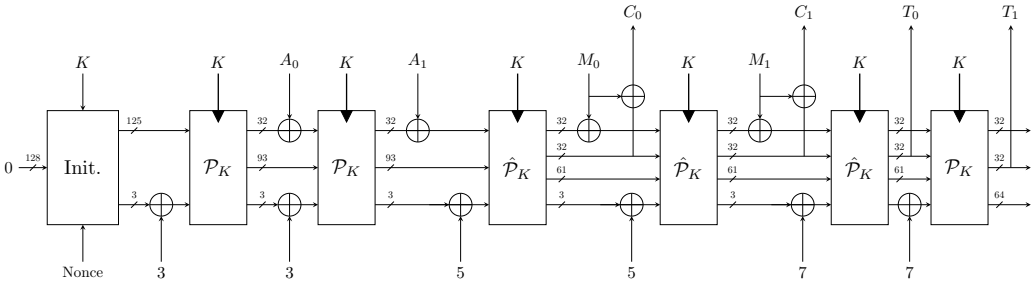

Figure 3: The high-level structure of TinyJAMBU

The encryption algorithm of TinyJAMBU can be divided into four phases. A visualized description of the encryption algorithm of TinyJAMBU for two associated data blocks and two message blocks can be found in Figure 3, where the permutations $\mathcal{P}_K$ and $\hat{\mathcal{P}}_K$ are specified in Table 1.

Initialization. Apply $\hat{\mathcal{P}}$ to the initial state $(0, \cdots, 0) \in \mathbb{F}_2^{128}$, and then the three nonce blocks Nonce_0, Nonce_1, and Nonce_2 are processed as depicted in Figure 4.

Associated Data Processing. Assuming we have two associated data blocks A_0 and A_1, they are processed step by step. In each step, the so-called 3-bit Framebits (the value is 3 for the associated data) are XORed with the state, then we update the state using the permutation $\mathcal{P}_K$, and finally the associated data A_0 is XORed with the state. A_1 is processed in a similar approach.

Encryption. After processing the associated data, the plaintext blocks M_0 and M_1 are encrypted one by one. In each step, the Framebits (the value is 5 for the plaintext) are

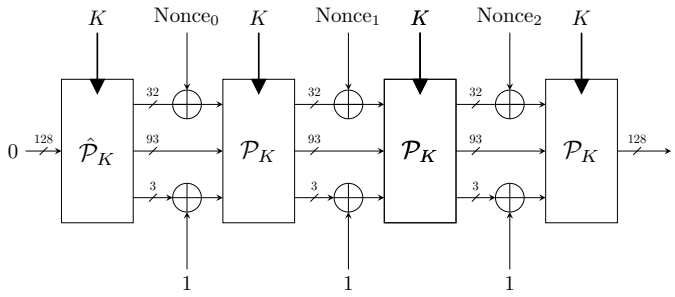

Figure 4: The initialization process of `TinyJAMBU`

XORed with the state, then the state is updated by the permutation $\hat{\mathcal{P}}_K$, then the 32-bit message block M_0 is XORed with the state, and finally we obtain the 32-bit ciphertext block C_0 by XORing M_0 with another part of the state. The next message block M_1 is encrypted in a similar way.

Note that the number of rounds of $\hat{\mathcal{P}}_K$ is much larger than that of $\mathcal{P}_K$. Namely, the encryption part is protected by a large number of rounds.

Finalization. After encrypting the plaintext, the authentication tag (T_0, T_1) is generated in two steps. First, the Framebits (the value is 7 for the finalization) are XORed with the state. Then the state is updated by the permutation $\hat{\mathcal{P}}_K$, and the 32-bit T_0 is extracted from the state. Then, the Framebits are XORed with the state again followed by an application of the permutation $\mathcal{P}_K$, and T_2 is extracted from the resulting state.

2.1 Security Claim

The primary security goal of `TinyJAMBU` is the nonce-respect security. The number of bits of security for each version is given in Table 2. The designers assume that each key is used to process at most 2^{50} bytes of messages. In their security analysis, the designers seem to regard that the attack with a success probability of about 2^{-15} after making queries of 2^{50} message bytes is a valid attack.

Table 2: Security goals of `TinyJAMBU` with unique nonce [WH19, Table 4.1].

Version	Encryption	Authentication
`TinyJAMBU-128`	112-bit	64-bit
`TinyJAMBU-192`	168-bit	64-bit
`TinyJAMBU-256`	224-bit	64-bit

2.2 Security Proofs and Assumptions

The designers of `TinyJAMBU` proved the security of the `TinyJAMBU` mode. Privacy is proven against nonce-respecting adversaries in the chosen plaintext attack and authenticity is proven against nonce-reuse adversaries. In both proofs, the designers assume that $\mathcal{P}_K$ and $\hat{\mathcal{P}}_K$ are ideal keyed permutations.

3 MILP Models for TinyJAMBU

After the work by Mouha et al. [MWGP], an automated method to find differential and linear trails by MILP has been studied extensively. Thanks to the simple operations, to build the model just for counting the number of active AND gates in $\mathcal{P}_K$ is not difficult. However, it has been observed that AND gates in the structure like TinyJAMBU may be dependent on each other and such correlations have a significant impact. In this section, we first explain how to build the model to take into account such dependencies for TinyJAMBU in Section 3.1. This model is relatively accurate but slow. Similar dependencies also affects the correlations of the linear trails, and in Section 3.2 we recall how to compute correlations of the linear trails of TinyJAMBU with higher accuracy. We then explain how to make the models for differential and linear trails more efficient by focusing on the effective part particular to TinyJAMBU in Section 3.3 and Section 3.4.

3.1 Stricter Model for Differential Trails

We now consider how the differences propagate through the permutation $\prod_{i=0}^{l-1} \mathcal{P}_{k_i}$, and we emphasize again that in our notation the subscript i of k should be computed modulo the key length $|K|$. A 128-bit state updated by $\mathcal{P}_{k_i}$ is depicted in Figure 2. In the resulting state, only the 127th (the leftmost) bit is *fresh* in the sense that a secret key bit is involved to compute it, and all other state bits are obtained by a shift. Let us introduce two sets of variables $\{x_0, \cdots, x_{l+127}\}$ and $\{\Delta x_0, \cdots, \Delta x_{l+127}\}$ to track down the *values* and *differences* of the bits.

Since $\overline{ab} = ab \oplus 1$, we can safely omit the constant XOR by replacing the NAND gate by an AND gate without affecting the results of our differential analysis. For the AND gate taking place of the NAND gate, we introduce two sets of variables $\{z_0, \cdots, z_{l-1}\}$ and $\{\Delta z_0, \cdots, \Delta z_{l-1}\}$ to capture the *values* and *differences* of the output bits of the AND gate. Then, we can obtain a system of equations

$$z_i = x_{85+i}x_{70+i},\ i \in \{0, 1, \cdots, l-1\}. \tag{2}$$

The value of z_i after the differences are added is $(x_{85+i} \oplus \Delta x_{85+i})(x_{70+i} \oplus \Delta x_{70+i})$. Hence $\Delta z_i = x_{85+i}x_{70+i} \oplus (x_{85+i} \oplus \Delta x_{85+i})(x_{70+i} \oplus \Delta x_{70+i})$. Then, we can obtain a new system of equations from Eq. (2):

$$\Delta x_{85+i}x_{70+i} \oplus \Delta x_{70+i}x_{85+i} = \Delta z_i \oplus \Delta x_{70+i}\Delta x_{85+i} \tag{3}$$

for $i \in \{0, 1, \cdots, l-1\}$. If we treat the Δ variables as known values, Eq. (3) turns into a system of linear equations. Let the matrix representation of the system of linear equations be

$$M(x_{i_0}, x_{i_1}, \cdots, x_{i_{s-1}})^T = \Delta \in \mathbb{F}_2^s \tag{4}$$

for some integer s. Let M and $\hat{M} = (M \mid \Delta)$ be the coefficient matrix and augmented coefficient matrix associated with Eq. (4), respectively. When a differential trail is fixed, the values of the variables representing the differences (Δx_j's) are fixed. The differential trail is valid if and only if there are actual data pairs following the differential, or equivalently, the system of equations given by Eq. (3) has a solution. In the language of the matrix representation of the system of linear equations, we have $\gamma(M) = \gamma(\hat{M})$, where M is the coefficient matrix of the system of equations, $\hat{M}$ is the augmented coefficient matrix, and $\gamma(\cdot)$ computes the rank of a matrix.

For this system of equations, we have $s = |\{x_{70+i}, x_{85+i} : 0 \le i < l\}|$ independent variables, and thus there are 2^s different input values for the AND gates. The probability of the given differential can be computed as the number of solutions of the system of

equations divided by 2^s, that is,

$$\frac{2^{s-\gamma(M)}}{2^s} = 2^{-\gamma(M)}. \tag{5}$$

When we search for differential trails of `TinyJAMBU`, we can add the constraints describing the system of equations [SHW+15a, SHW+15b] into the MILP model and set the objective function to minimize the active AND gates. Therefore, the solution of the model always satisfies the system of equations. However, the solution is not guaranteed to be an optimal differential trail since a trail minimizing the number of active AND gates does not necessarily minimize the rank $\gamma(M)$ of the corresponding coefficient matrix. After we obtain a solution, the probability of the corresponding differential trail needs to be computed (outside the MILP context) using Eq. (5).

3.2 Correlation of Linear Trails

First of all, regardless of the AEAD context, we can analyze the security of the underlying keyed permutations against linear cryptanalysis by treating $\mathcal{P}_K$ and $\hat{\mathcal{P}}_K$ as block ciphers using existing tools for linear cryptanalysis of block ciphers. However, to evaluate the security of `TinyJAMBU` against more meaningful attacks (e.g., distinguishing attacks in the AEAD context), we need a more involved model.

Let us first look back at Figure 3. If we set all the message blocks M_j to zero, we arrive at a key stream generator illustrated in Figure 5 which is compatible with the one presented in [SSS+19]. For `TinyJAMBU`, $\mathcal{G}_i$'s are just some truncation operations extracting 32-bit of data from the 128-bit data bus.

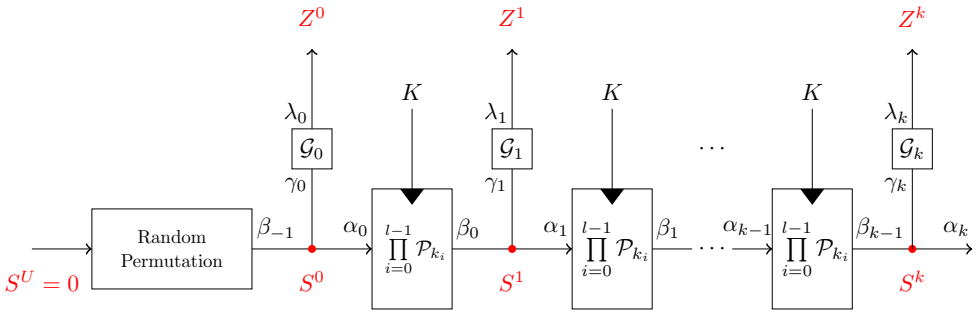

Figure 5: Linear masks for a general key stream generator

Therefore, using the same technique from [SSS+19], we can try to search for a sequence of linear masks $(\lambda_0, \cdots, \lambda_k)$ for the output blocks Z^i such that

$$\bigoplus_{i=0}^{k} \lambda_i \cdot Z^i \tag{6}$$

is biased. If such masks can be found, we can perform a distinguishing attack on the target. It is shown that this kind of linear approximations involving only key stream blocks Z^i can be searched using the same tools for linear cryptanalysis of block ciphers with additional constraints imposed on the linear masks.

Lemma 1 ([SSS+19]). *If the sequence of linear masks*

$$(\beta_{-1}, \gamma_0, \lambda_0, \alpha_0, \beta_0, \cdots, \alpha_{k-1}, \beta_{k-1}, \gamma_k, \lambda_k, \alpha_k)$$

for the key stream generator given in Figure 5 satisfies $\beta_{-1} = 0$, $\alpha_k = 0$, $\alpha_i \oplus \gamma_i \oplus \beta_{i-1} = 0$, *for* $0 \le i \le k$, *and the associated linear approximations are independent, then*

$$\mathrm{Corr}\left(\bigoplus_{i=0}^{k} \lambda_i \cdot Z^i\right) = \prod_{i=0}^{k} \mathrm{Corr}\left(\gamma_i \cdot S_i \oplus \lambda_i \cdot Z_i\right) \prod_{i=0}^{k-1} \mathrm{Corr}\left(\alpha_i \cdot S_i \oplus \beta_i \cdot S_{i+1}\right).$$

Sketch of proof. It comes from the fact that if

$$\begin{cases} \beta_{-1} = 0 \\ \alpha_k = 0 \\ \alpha_i \oplus \gamma_i \oplus \beta_{i-1} = 0, \quad 0 \le i \le k \end{cases},$$

we have

$$\begin{aligned}
\bigoplus_{i=0}^{k} \lambda_i \cdot Z^i &= \bigoplus_{i=0}^{k} \lambda_i \cdot Z^i \quad \oplus \quad \bigoplus_{i=0}^{k} \gamma_i \cdot S^i \quad \oplus \quad \bigoplus_{i=0}^{k} \gamma_i \cdot S^i \\
&= \bigoplus_{i=0}^{k} (\gamma_i \cdot S^i \oplus \lambda_i \cdot Z^i) \quad \oplus \quad \bigoplus_{i=0}^{k} \gamma_i \cdot S^i \\
&= \bigoplus_{i=0}^{k} (\gamma_i \cdot S^i \oplus \lambda_i \cdot Z^i) \quad \oplus \quad \beta_{-1} \cdot S^0 \quad \oplus \quad \bigoplus_{i=0}^{k-1} (\alpha_i \cdot S^i \oplus \beta_i \cdot S^{i+1}) \quad \oplus \quad \alpha_k \cdot S^k \\
&= \bigoplus_{i=0}^{k} (\gamma_i \cdot S^i \oplus \lambda_i \cdot Z^i) \quad \oplus \quad \bigoplus_{i=0}^{k-1} (\alpha_i \cdot S^i \oplus \beta_i \cdot S^{i+1}).
\end{aligned}$$

Applying the Piling-up lemma completes the proof. □

Lemma 1 tells us that to search for biased linear approximations $\bigoplus_{i=0}^{k} \lambda_i \cdot Z_i$ involving only output blocks Z_j's, we can set up an MILP model for the linear trails of G_i and $\prod_{i=0}^{l-1} \mathcal{P}_{k_i}$ and add the additional constraints specified Lemma 1, where we call $k+1$ the span of the linear approximation if $\lambda_0 \neq 0$ and $\lambda_k \neq 0$. In our analysis of `TinyJAMBU`, we only find useful trails involving only two consecutive output blocks (i.e., the *span* [SSS+19] or the number of output blocks involved of the trails are two). In particular, if we treat all transformations before the tag generation operations of `TinyJAMBU` as a random permutation, we can search for linear trails carrying the correlation of the tag $T_0 \,\|\, T_1$ as shown in Figure 6.

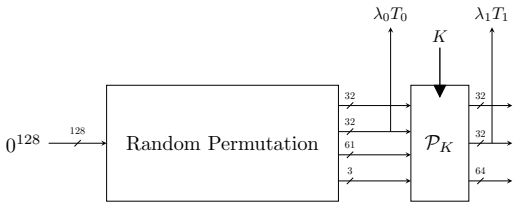

Figure 6: Linear trails of `TinyJAMBU` carrying the correlation of the tag

In practice, we search for such linear masks using MILP-based method with the objective to minimize the number of active AND gates in hope to discover a trail with high bias. However, this approach only considers local constraints of the propagations of the linear masks, where the propagation of the input and output masks of each AND gate is modeled independently and the global dependencies between these gates are ignored, that is, we assume that the local linear approximations of the involved AND gates are independent.

Table 3: Restrictions on the values of a and b in $a \cdot b = z$ when $\Delta z = 1$.

Δa	Δb	$\Delta z = 1$ iff
0	0	Never
0	1	$a = 1$
1	0	$b = 1$
1	1	$a = b$

Consequently, the trails found can be globally inconsistent, meaning that a "good" trail output by an MILP solver with a very small number of active AND gates can be actually of correlation zero. Even if we partly take the dependencies into account as we will do in Section 3.4, we are not guaranteed to eliminate all inconsistencies. Taking the scenario shown in Figure 6 as an example, to get the real value of $\mathrm{Corr}(\lambda_0 \cdot T_0 \oplus \lambda_1 \cdot T_1)$, we can express $\lambda_0 \cdot T_0 \oplus \lambda_1 \cdot T_1$ as a Boolean function in the bits of T_0 and K, and then compute the correlation of this Boolean function. However, there is no polynomial-time algorithm for computing the correlation of a Boolean function with degree higher than two.

In this paper, we make the assumption that each output bit of the AND gate affected by a key bit can be regarded as a freshly new variable, and therefore each application of $\mathcal{P}_{k_i}$ introduces a new variable. In this way, $\lambda_1 \cdot T_1$ shown in Figure 6 can be expressed as a quadratic Boolean function in the bits of T_0 and these new variables, and so does $\lambda_0 \cdot T_0 \oplus \lambda_1 \cdot T_1$. Let the quadratic Boolean function representing $\lambda_0 \cdot T_0 \oplus \lambda_1 \cdot T_1$ be f. In practice, f can be derived as follows. After we get a sequence of linear masks (including the intermediate masks) of $\prod_{i=0}^{l-1} \mathcal{P}_{k_i}$, we can identify h linearly active `AND` gates $\{\mathtt{AND}_0, \cdots, \mathtt{AND}_{h-1}\}$ whose output mask bits are 1. Assuming that w_i is the output bit of the gate $\mathtt{AND}_i$, and x_i and y_i are the input bits whose linear masks are η_i and μ_i respectively. Then we have

$$f = \sum_{i=0}^{h-1} (x_i y_i + \eta_i x_i + \mu_i y_i). \tag{7}$$

Therefore $\mathrm{Corr}\left(\lambda_0 \cdot T_0 \oplus \lambda_1 \cdot T_1\right) = \mathrm{Corr}\left(f\right)$, which can be computed from the *disjoint quadratic form* of f [SSS+19].

3.3 The Refined Model for Differential Trails

Related Gates. In the simple model for counting the number of active AND gates, if there is a difference on at least one of the two input bits, the output of the AND gates has a difference with probability 2^{-1} or does not with probability 2^{-1}. It considers independently every AND gate and treats every AND gate in the same way. However, the differential propagation through the AND gates restricts the value of the related bits in different ways. Let us consider the single AND gates $ab = z$. When the output has a difference, the input value is restricted as shown in Table 3 depending on the difference of the inputs. The simple model ignores those restrictions of the values, and we will see that it especially fails to capture the relation of the AND gates in the computation of `TinyJAMBU`.

The main observation that motivates this refinement of the model is that the same value, as it is shifted, will enter twice in two different AND gates. Indeed S_{85} will pass the AND gate with S_{70} and also, 15 rounds later, with S_{100}. This raises the question about the correlation of $a \cdot b$ and $b \cdot c$ for some values a, b, c. The structure is depicted in Figure 7.

Let Δab and Δbc denote the output difference of the AND gates ab and bc, respectively. If we look jointly at the differential propagation property of Δab and Δbc then only one case does not match the simple approach. It is the case when $\Delta a = \Delta c = 1$ but $\Delta b = 0$. In this particular case, the conditions we derive from Table 3 are the same. Both AND

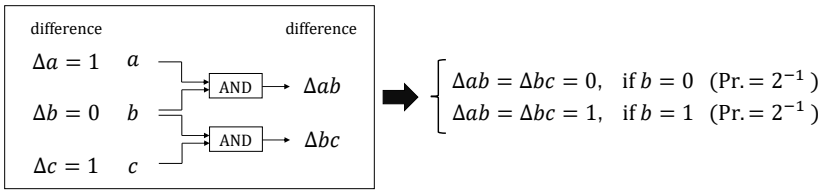

Figure 7: Dependency of two AND gates that appears frequently in `TinyJAMBU`.

gates will produce the difference, or not, depending on the underlying value b. If $b = 0$ then $\Delta ab = \Delta bc = 0$, if $b = 1$ then $\Delta ab = \Delta bc = 1$. Both cases happen with probability 2^{-1} since it depends on a single bit value. In the refined model we simply force, in this scenario, that both differences jointly propagate, or not, and we only count this as a single active gate.

By doing this we avoid modeling the full underlying state and manage to stay at the differential trail level which keeps the model simple enough for in-depth analysis of `TinyJAMBU`.

The Model. Concretely the refined model adds additional constraints on top of the simple model's constraints. The notations for the MILP model variables are:

- d_a modelizes Δa that is the difference on bit a.
- d_{ab} modelizes Δab that is the difference on the output of the AND gate ab.
- γ_{abc} indicates whether there's a correlation between the two AND gates ab and bc.

First we put all constraints of the simple model and record all chained AND gates. For instance, let the recorded chains be of the form $\{(d_{ab}, d_a, d_b), (d_{bc}, d_b, d_c), (d_{cd}, d_c, d_d), ...\}$. Then for all consecutive couples $\big((d_{ab}, d_a, d_b), (d_{bc}, d_b, d_c)\big)$ the following constraint is added:

$$\begin{aligned} \gamma_{abc} &= d_a \overline{d_b} d_c \\ d_{ab} - d_{bc} &\leq 1 - \gamma_{abc} \\ d_{bc} - d_{ab} &\leq 1 - \gamma_{abc} \end{aligned}$$

Namely, γ_{abc} is 1 when $(d_a, d_b, d_c) = (1, 0, 1)$ and $\gamma_{abc} = 0$ otherwise. When $\gamma_{abc} = 0$, the last two inequalities are always true. Thus those three constraints do not have any impact. When $\gamma_{abc} = 1$, the last two inequalities ensure that $d_{ab} = d_{bc}$.

Then we can subtract all values γ_{abc} in the objective function to only count this once, whereas the simple model would count two active gates. Notice that the convex hull computation technique can easily translate the first equation into a set of linear inequalities.

Also, the first equation ensures that two consecutive γ_{abc} and γ_{bcd} will never be true at the same time and thus their respective correlation cannot happen simultaneously avoiding any clashes.

Detecting Related Couples. In `TinyJAMBU`, by construction it is easy to see that all concerned gate couples are $\big((d_{S_i S_{i+15}}, d_{S_i}, d_{S_{i+15}}), (d_{S_{i+15} S_{i+30}}, d_{S_{i+15}}, d_{S_{i+30}})\big)$ for all $70 \leq i \leq 84 + r - 30$ with r the number of rounds as S_{70} and S_{84+r} are the first and last values respectively to enter an AND gate. There will be 15 different chains since all values from S_{70} to S_{84} will only enter the AND gate once and thus form the start of the chain.

3.4 The Refined Model for Linear Trails

Related Approximations. We can adapt the same idea to refine our model to look for better linear approximations. Indeed, the simple model allows ab to be approximated by $0, a, b$ or $a \oplus b$ with a $1/2$ correlation (and so a bias of $1/4$). So the simple model allows $ab \oplus bc$ to be approximated by any linear combination of a, b and c with a $1/4$ correlation. However, $ab \oplus bc$ can actually be linearly approximated only by $0, b, a \oplus c$ and $b \oplus a \oplus c$ with a $1/2$ correlation. It is very easy to verify that as $ab \oplus bc = b(a \oplus c)$.

Taking into account those related approximations will eliminate wrong approximations while strengthening others. This is very similar to what happens for differential trails in Section 3.3.

Chained Related Approximations. There is a subtlety when considering related linear approximations that was not presented for differential trails. `TinyJAMBU` is built in such a manner that the values entering the AND gate forms a kind of chain. In the linear case the correlation happens when we need to linearize ab and bc but it may be that we also need to linearize cd.

Thus we must be careful not to count too many relations. Luckily the linearization of $ab \oplus bc \oplus cd$ can be done by independently linearizing $ab \oplus bc$ and cd. That means that we should ignore the relation between bc and cd when the former is already related to another AND gate.

The Model. Concretely we add some constraints on top of the simple model. For all a, b that enters an AND gate, the simple model expresses $ab \simeq \ell_a^{ab} a \oplus \ell_b^{ab} b$ for unconstrained boolean values ℓ_a^{ab} and ℓ_b^{ab} so that effectively ab can be linearized by any combination of a and b.

So we use the following boolean notations:

- ℓ_a^{ab} controls whether a is used to linearly approximate ab.
- α^{ab} indicates whether ab needs to be linearized (in `TinyJAMBU` it is the LSB bit of the linear mask that gets shifted out in the next round).
- α^{abc} indicates whether $ab \oplus bc$ needs to be jointly linearized, that is we take into account their relation. This variable does not exist in the simple model.

First we record the chained values of related AND gates in the order of which they are chained: $\{(\alpha^{ab}, \ell_a^{ab}, \ell_b^{ab}), (\alpha^{bc}, \ell_b^{bc}, \ell_c^{bc}), (\alpha^{cd}, \ell_c^{cd}, \ell_d^{cd}), ...\}$. Then we start by the constraints on the head of the chain that is $(\alpha^{ab}, \ell_a^{ab}, \ell_b^{ab})$ and $(\alpha^{bc}, \ell_b^{bc}, \ell_c^{bc})$ (meaning a enters only once the AND gate like S_{70} in `TinyJAMBU`):

$$\begin{aligned} \alpha^{abc} &= \alpha^{ab}\alpha^{bc} \\ \ell_a^{ab} - \ell_c^{bc} &\leq 1 - \alpha^{abc} \\ \ell_c^{bc} - \ell_a^{ab} &\leq 1 - \alpha^{abc} \end{aligned}$$

At last we iterate the constraints for $(\alpha^{bc}, \ell_b^{bc}, \ell_c^{bc}), (\alpha^{cd}, \ell_c^{cd}, \ell_d^{cd})$ using in addition the previous indicator α^{abc} (a may not be the start of the chain now):

$$\begin{aligned} \alpha^{bcd} &= \overline{\alpha^{abc}}\alpha^{bc}\alpha^{cd} \\ \ell_b^{bc} - \ell_d^{cd} &\leq 1 - \alpha^{bcd} \\ \ell_d^{cd} - \ell_b^{bc} &\leq 1 - \alpha^{bcd} \end{aligned}$$

As α^{bcd} captures whether we take into account the relations between bc and cd the first equation ensures that it cannot be 1 if we already restricted the previous chained AND

gates. The last two equations force $\ell_b^{bc} = \ell_d^{cd}$ when the relation is active effectively forcing the linearization to prevent wrong approximations. To properly estimate the resulting correlation we simply subtract all α^{abc} values in the objective function.

4 Differential Cryptanalysis

In this section, we show several results which we obtained with the refined model for differential trails. We first summarize the existing security analysis by the designers in Section 4.1. We then show our results for the AEAD setting in Section 4.2. Finally, we show our results for the underlying keyed-permutation in Section 4.3.

4.1 Summary of the Security Analysis by the Designers

In the submission document of `TinyJAMBU` [WH19], the designers searched for the differential trail that has the minimum number of active AND gates in the simple model, i.e., by regarding every AND gate independent. The designers searched for the differential trails under four different constraints about the active-bit positions of the input and output of $\mathcal{P}_K$.

Type 1: Input differences only exist in the 32 MSBs. No constraint on the output.

Type 2: No constraint on the input. Output differences only exist in the 32 MSBs.

Type 3: Both of the input and output differences only exist in the 32 MSBs.

Type 4: No constraint.

Type 3 is most relevant to the security as AEAD, while Type 4 is purely for the underlying keyed-permutation $\mathcal{P}_K$. The details will be explained later, but most notably, the designers claim that the maximum probability of the 384-round trail of Type 3 is 2^{-80} and the maximum probability of the 320-round characteristic of Type 4 is 2^{-13}, where 384 is the number of rounds of $\mathcal{P}_K$ in all members of `TinyJAMBU`.

4.2 Attacks for the AEAD Setting

4.2.1 Forgery for `TinyJAMBU` Mode

Overall, our strategy is to attack the nonce setup or the associated data processing because the number of rounds of the keyed permutation is 384, which is significantly smaller than that of the encryption. We inject the difference to 32 bits of the ith input block and cancel all the state differences by injecting another difference to 32 bits of the $i+1$th input block. The differential trails of Type 3 correspond to that situation. Suppose that there exists a differential trail that maps $(\Delta_i \,\|\, 0^{96})$ to $(\Delta_{i+1} \,\|\, 0^{96})$ over $\mathcal{P}_K$ with probability p, where Δ_i and Δ_{i+1} are two independent 32-bit differences. There are two ways to exploit such a trail to mount a forgery attack.

- The simple scenario is a probabilistic nonce-respect almost universal forgery, where 'almost' denotes that the length of AD must be at least two blocks, or 64 bits. Note that the designers also mention this strategy in [WH19]. Let N, $A_0 \,\|\, A_1$, and M be the nonce, AD, and message to be forged, respectively. Then, an attacker makes a query of $(N, (A_0 \oplus \Delta_i \,\|\, A_1 \oplus \Delta_{i+1}), M)$ to the encryption oracle to observe the tag T. Then, T is a valid tag for $(N, A_0 \,\|\, A_1, M)$ with probability p. As long as $p \geq 2^{-64}$, this breaks the 64-bit security.

  Because the nonce is processed similarly as the AD, the same attack can be applied by injecting the differences into the nonce. Note that the nonce size is 96 bits for all

members of TinyJAMBU, and nonces are processed over 3 blocks with $\mathcal{P}_K$ (Figure 4). This approach works for arbitrary AD, and this can be advantageous because it is not always true that the attacker has full control of the AD value.

- Let us consider the MAC reforgeability for multiple targets [BC09]: whether or not the success of forging the first target helps to attack the second target. The above scenario does not help to attack the second target. The scenario here is a nonce-respect almost universal forgery with the reforgeability, where 'almost' means that (A, M) can be arbitrary, and the first 64 bits of N can be chosen by the attacker. This restriction is weak because implementations accept any N as long as it is new, and the attacker still can choose a new N by modifying the last 32 bits of N. Saying differently, for any (A, M), the attacker can find N and T such that T is a valid tag for (N, A, M) in the nonce-respect manner with a tradeoff between the data complexity and the success probability. Once such $N_0 \parallel N_1 \parallel N_2$ is derived, the attacker can forge the tag for any (A, M) immediately.

  Let $N_0 \parallel N_1 \parallel N_2$ be the block-wise description of the nonce N. The attacker first finds a collision between $(N_0 \parallel N_1 \parallel N_2, A^*, M^*)$ and $(N_0 \oplus \Delta_i \parallel N_1 \oplus \Delta_{i+1} \parallel N_2, A^*, M^*)$. By examining D distinct nonces, the attacker finds a collision after the first two blocks with a probability $D \times p$. Let $N_0 \parallel N_1$ and $N_0' \parallel N_1'$ be two 64-bits of nonces that make a state collision. Then for any target A, M, the attacker chooses the last 32 bits of nonce N_2^* arbitrary, and obtains a tag T for $(N_0 \parallel N_1 \parallel N_2^*, A, M)$ by making an encryption query. Then, T is also valid for $(N_0' \parallel N_1' \parallel N_2^*, A, M)$.

Note that the reforgeability and the amplification of the success probability for any (A, M) in the nonce-respect manner are not general for the sponge-like constructions. In general, the attacker needs to shift to the nonce-misuse attack or needs to compromise the goal to the existential forgery to enjoy the tradeoff. In fact, if N is embedded inside the initial state value or if the entire 96 bits of N is processed in one block, the same attack goal as the second scenario cannot be achieved (however the security proof seems to be lost with these modifications). Hence the second scenario that exploits three blocks of the nonce is a security issue particular to TinyJAMBU that needs to process 96-bit nonces with the tightly minimized state size for being very lightweight.

In the following, we focus on p for particular rounds. While the designers only considered the single trail, it is known that the construction like TinyJAMBU allows a differential, or a cluster of multiple trails sharing the same input and output difference, of a much high probability. Because we can fix the target input and output differences, we can apply the techniques of [SHW+15a] to look for many trails while ensuring they are right. Then we can sum all of the probabilities of those trails to better approach the probability of the targeted differential going through independently of the actual trail taken.

4.2.2 Observations on Full 384 Rounds

As briefly mentioned above, the designers claim that the maximum probability of the full-round characteristic of this type is 2^{-80}, which is sufficient to claim 64-bit security.

We first point out that there is no differential trail for 384 rounds that can be satisfied with probability 2^{-80}. To reach this conclusion, we first searched for differential trails with the simple model, and we could actually find several trails with 80 active AND gates. However, all of such trails include some contradiction when the dependency of different AND gates is taken into account.

Secondly, by using the refined model, the best differential trail we found for 384 rounds is given in Table 4. (Details of the trail in the table format will be explained later for 338 rounds in Table 7.) It consists of 88 active AND gates among which 14 couples are correlated as explained in Section 3.3. Therefore this differential trail propagates with

probability $2^{-74}(=2^{-88+14})$. We then evaluated its differential probability by identifying multiple differential trails with the same input and output differential masks. We found 103 distinct differential trails with probability 2^{-80} or more, whose distribution is listed in Table 5. Summing everything gives a differential probability of $2^{-70.68}$, which is much higher than originally evaluated by the designers.

The targeted security level against forgery being of 64 bits, the security margin against differential cryptanalysis, namely the gap between the claimed security bits and the attack complexity, thus stands at less than 8 bits.

Table 4: Probability 2^{-74} Type 3 differential trail for 384 rounds found with the refined model. The differences are restricted to the 32 MSBs in the input and output. Those are stressed by red. Information of intermediate differences is provided to recover all the differential propagations in the trail.

Input:	$\Delta S_{127..0}$	01004800	00000000	00000000	00000000
	$\Delta S_{255..128}$	81044c80	24080304	d9200000	22090000
	$\Delta S_{383..256}$	81004082	00010200	83000010	26090240
Output:	$\Delta S_{511..384}$	81004082	00000000	00000000	00000000

Table 5: Differential effect for 384 rounds

Probability	2^{-74}	2^{-75}	2^{-76}	2^{-77}	2^{-78}	2^{-79}	2^{-80}
# Trails	1	5	9	14	20	24	30

4.2.3 Differential Cryptanalysis of 338 Rounds

Another way of computing the security margin is to ask ourselves how many rounds can we break using differential cryptanalysis. Therefore we have run the same tools on a growing number of rounds looking for the largest number of rounds with security less than 64 bits. The results of the evaluation is listed in Table 6. Here, the 'score' means the number of AND gates minus the number of couples that are correlated as explained in Section 3.3. Hence, the trail with a score of X can be satisfied with probability 2^{-X}.

Because active-bit positions are restricted both in the input and the output, the probability does not monotonically decreases as the number of rounds increases. This requires us to evaluate every round numbers, which takes long particularly when the number of rounds becomes large. However, such an approach is necessary to find a spot that allows a high probability trail. As shown in Table 6, the probability of the single trail reaches 2^{-67} for 302 rounds, but it suddenly jumps up to 2^{-64} for 329 - 338 rounds. In addition, from 348 rounds, we only check if the score can be lower bounded by 67.

In the end, the maximum number of rounds we can attack is 338, which allowed a trail with a score of 64. The details of the differential trail is given in Table 7.

Here we explain all the details of the differential propagation in Table 7. Let $S_{127}, S_{126}, \ldots, S_0$ be the 128 state bits of the input to $\mathcal{P}_K$ and ΔS_i be its difference. In round r, one bit is computed by $\mathcal{P}_{k_r}$ and its difference is denoted by ΔS_{127+r} for $r = 1, 2, \ldots, 338$. As shown in Table 7, there are 42 intermediate state bits with the non-zero difference.

$$S_{97}, S_{100}, S_{104}, S_{107}, S_{110}, S_{116}, S_{127}, S_{134}, S_{137}, S_{141}, S_{144}, S_{147}, S_{158}, S_{164}, S_{171}, S_{184},$$
$$S_{188}, S_{191}, S_{195}, S_{197}, S_{201}, S_{202}, S_{215}, S_{218}, S_{221}, S_{225}, S_{228}, S_{234}, S_{235}, S_{238}, S_{244}, S_{265},$$
$$S_{272}, S_{282}, S_{306}, S_{312}, S_{319}, S_{329}, S_{434}, S_{440}, S_{447}, S_{457}.$$

The difference in S_i makes the AND gates active in round $i-85$ and round $i-70$. Let $n_r = 1$ and $n_r = 0$ denote that the AND gates in round r is active and inactive, respectively.

Table 6: Score of the best trail found for various number of rounds in the refined model. 'weight' is defined as a number of active AND gates and 'score' is defined as weight minus the number of correlated AND gates. Question mark (?) signifies that the solver did not finish but the score is lower bounded by 67. Hyphen (–) signifies that no valid trail exists.

Rounds	Score									
191-200	–	–	–	–	–	–	87	91	82	82
201-210	82	76	74	71	71	71	71	71	71	71
211-220	71	70	70	70	70	71	71	72	70	70
221-230	70	70	73	73	67	63	60	60	60	60
231-240	60	66	66	71	71	62	64	64	64	64
241-250	65	65	65	66	66	66	66	66	66	66
251-260	66	64	64	64	64	64	64	64	64	64
261-270	66	59	59	59	59	59	59	59	59	59
271-280	59	59	59	59	65	67	67	67	55	55
281-290	55	55	55	68	68	67	67	67	67	67
291-300	67	61	61	61	65	65	65	65	65	65
301-310	65	67	67	67	68	68	68	68	68	68
311-320	68	71	71	71	71	70	70	70	70	70
321-330	75	75	72	72	72	72	72	72	64	64
331-340	64	64	64	64	64	64	64	64	71	71
341-350	71	71	71	71	71	71	71	?	?	?
351-360	?	?	?	?	?	?	?	?		

Table 7: Probability 2^{-64} Type 3 differential trail for 338 rounds found with the refined model.

Input:	$\Delta S_{127..0}$	80104912	00000000	00000000	00000000
	$\Delta S_{255..128}$	00104c12	24800628	91000810	40092240
	$\Delta S_{383..256}$	00000000	00000200	81040000	04010200
Output:	$\Delta S_{465..338}$	00802041	00000000	00000000	00000000

Then we have the following 76 n_r with 1. Hence by simply counting the number of active AND gates, this differential trail is regarded to be satisfied with probability 2^{-76}.

$$n_{12}, n_{15}, n_{19}, n_{22}, n_{25}, n_{27}, n_{30}, n_{31}, n_{34}, n_{37}, n_{40}, n_{42}, n_{46}, n_{49}, n_{52}, n_{56},$$
$$n_{57}, n_{59}, n_{62}, n_{64}, n_{67}, n_{71}, n_{73}, n_{74}, n_{77}, n_{79}, n_{86}, n_{88}, n_{94}, n_{99}, n_{101}, n_{103},$$
$$n_{106}, n_{110}, n_{112}, n_{114}, n_{116}, n_{117}, n_{118}, n_{121}, n_{125}, n_{127}, n_{130}, n_{131}, n_{132}, n_{133}, n_{136}, n_{140},$$
$$n_{143}, n_{145}, n_{148}, n_{149}, n_{150}, n_{151}, n_{153}, n_{155}, n_{158}, n_{159}, n_{164}, n_{165}, n_{168}, n_{174}, n_{180}, n_{187},$$
$$n_{195}, n_{197}, n_{202}, n_{212}, n_{221}, n_{227}, n_{234}, n_{236}, n_{242}, n_{244}, n_{249}, n_{259}.$$

Among those 76 active AND gates, only 23 of them propagate the difference. Let $y_r = 1$ and $y_r = 0$ denote that the output of the AND gates in round r is active and inactive, respectively. Then we have the following 23 y_r with 1.

$$y_{25}, y_{30}, y_{46}, y_{57}, y_{74}, y_{94}, y_{110}, y_{117}, y_{130}, y_{143}, y_{148}, y_{150}, y_{153}, y_{155}, y_{158}, y_{164},$$
$$y_{168}, y_{187}, y_{195}, y_{202}, y_{234}, y_{244}, y_{259}.$$

Our analysis of the correlation of two AND gates occurs 12 times in the following situations.

$$(S_{97}, S_{112}, S_{127}), (S_{104}, S_{119}, S_{134}), (S_{107}, S_{122}, S_{137}), (S_{134}, S_{149}, S_{164}),$$
$$(S_{141}, S_{156}, S_{171}), (S_{158}, S_{173}, S_{188}), (S_{171}, S_{186}, S_{201}), (S_{188}, S_{203}, S_{218}),$$
$$(S_{191}, S_{206}, S_{221}), (S_{195}, S_{210}, S_{225}), (S_{235}, S_{250}, S_{265}), (S_{282}, S_{297}, S_{312}).$$

For example, for the first case, $(\Delta S_{97}, \Delta S_{112}, \Delta S_{127}) = (1, 0, 1)$ and this ensures $y_{27} = y_{42}$. Indeed, in our trail, $y_{27} = y_{42} = 0$. In the simple model, two events of $y_{27} = 0$ and $y_{42} = 0$ are counted independently. In fact, both of n_{27} and n_{42} are 1. However, as shown in Figure 7, $y_{27} = 0$ and $y_{42} = 0$ are jointly controlled by a single-bit condition. Hence, the actual probability is higher by a factor of 2. By considering 12 pairs of correlated AND gates, the probability of the trail is $2^{-76+12} = 2^{-64}$. Note that all of ΔS_r, n_r, y_r and the correlated paired AND gates can be uniquely reproduced from the information in Table 7.

We then evaluated its differential probability by identifying multiple differential trails with the same input and output differential masks. Eventually, we found 24 distinct differential trails with probability 2^{-72} or more, whose distribution is listed in Table 8. Summing everything gives a differential probability of $2^{-62.68}$, which is much higher than

Table 8: Differential effect for 338 rounds

Probability	2^{-64}	2^{-66}	2^{-67}	2^{-68}	2^{-69}	2^{-70}	2^{-71}	2^{-72}
# Trails	1	2	4	4	4	5	4	4

originally evaluated by the designers. The security margin in terms of the number of rounds thus stands at 46 rounds, which is only about 12% of the entire construction.

Remarks. One may be interested in converting Table 6 into a graph to see whether we can read some tendency about the relationship between the number of rounds and the score. The graph is shown in Fig. 8. When the number of rounds is small, differential

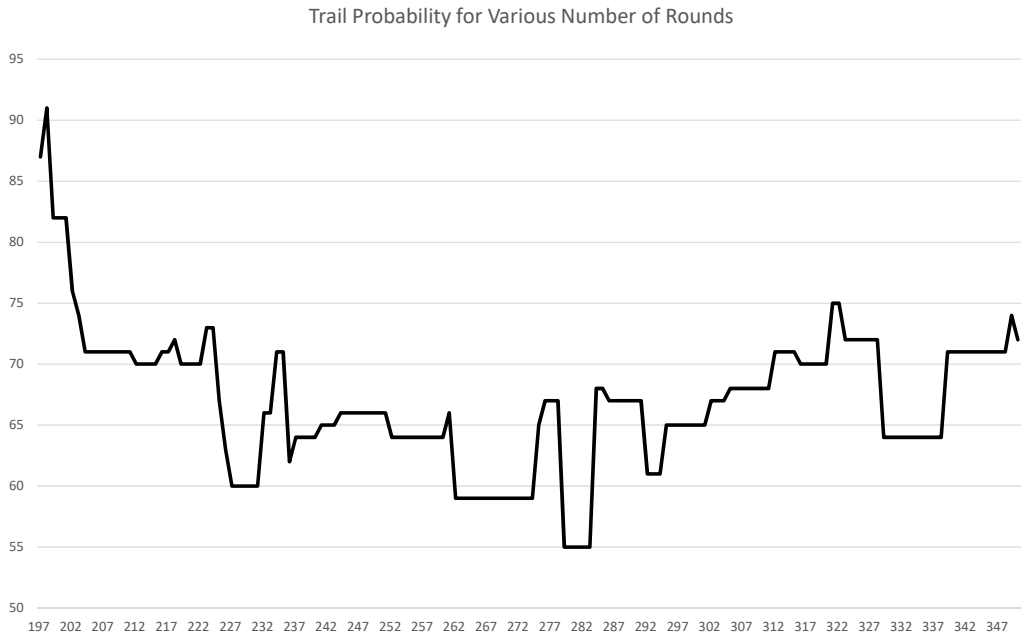

Figure 8: Graph of Table 6. Vertical axis denotes the score and horizontal axis denotes the number of rounds.

propagation is not flexible enough to satisfy the conditions on the input and the output. Solutions appear from 197 rounds. At the beginning, the trail probability gets higher as the number of rounds increases. We believe this is because various trails can be chosen and some of them are efficient. There may exist a periodic property in every 50 rounds (around

230, 280, 330 rounds), while it may be too weak as a ground to predict the score 380 rounds even roughly. It would be an interesting open problem to identify the relationship between the score and the number of rounds.

4.3 Attacks for the Underlying Permutation

Security of the `TinyJAMBU` mode was proven by assuming the ideal behavior of the underling keyed permutations $\mathcal{P}_K$ and $\hat{\mathcal{P}}_K$. This motivates us to evaluate the security of $\mathcal{P}_K$ against differential cryptanalysis as a stand-alone primitive. Note that the differential for 384 rounds with probability $2^{-70.64}$ already suggests a non-ideal behavior of the full-round $\mathcal{P}_K$ as an ideal keyed-permutation. The goal here is to show that the security is even lower.

4.3.1 Unrestricted Differentials

Here we look at the best differential trails with no restriction on the input or output. This is referred as Type 4 differentials in the submission document of `TinyJAMBU` [WH19]. The designers showed that the minimum number of active AND gates is 4 for 192 rounds and 13 for 320 rounds. No result was given for 384 rounds or more.

First of all, we applied the simple model to 384 rounds to find that the minimum number of active AND gates is 18. We then applied the refined model for 192, 320 and 384 rounds. For 384 rounds, the best differential trail we found has a score of 19: 20 actives gates minus 1 correlation between two AND gates. The differential masks for this trail is given in Table 9. In Table 10 we compare our results with the one given by the designers of `TinyJAMBU`.

Experimental Verification. We tried to verify the Type 4 differential trail for 384 rounds using the implementation of the permutation $\mathcal{P}_K$ provided by the designers and it succeeded[2]. We used any fixed key and random state pairs with input difference $\Delta S_{127..0}$ and verified the output difference against $\Delta S_{511..384}$ as per Table 9. We were able to generate conforming state pairs with a probability close to the theoretical estimate.

Table 9: Probability 2^{-19} Type 4 differential trail for 384 rounds found with the refined model.

Input:	$\Delta S_{127..0}$	80000000	20010000	00000092	00000000
	$\Delta S_{255..128}$	00000000	20000000	00004000	00000004
	$\Delta S_{383..256}$	00000000	20000000	00000000	00000000
Output:	$\Delta S_{511..384}$	81020000	20001000	00004080	00000004

Table 10: Score of the best Type 4 trail found for number of rounds evaluated by the original submission document.

Rounds	192	320	384
[WH19] (Simple)	4	13	-
Ours (Refined)	4	12	19

4.3.2 Partly Restricted Differentials

Here we analyse the property of trails where either only the input is restricted, Type 1 trails, or only the output is restricted, Type 2 trails. The designers searched for Type 1

[2]The source code for finding conforming pairs and the MILP trails search can be found here https://github.com/c-i-p-h-e-r/refinedTrailsTinyJambu.

differentials for 256, 320, 384, 448 and 512 rounds and Type 2 differentials for 384 and 512 rounds. We then applied the refined model for all of the evaluated rounds and found that the scores of Type 1 differentials are smaller than originally evaluated in all rounds. For Type 2 differentials, we did not find any gap between ours and the designers' evaluation. We compare the results we got with the previous ones in Tables 11 and 12.

Table 11: Score of the best Type 1 trail found for number of rounds evaluated by the original submission document. Question mark (?) signifies that the solver didn't finish.

Rounds	256	320	384	448	512
[WH19] (Simple)	22	33	45	55	68
Ours (Refined)	20	29	41	51	64?

Table 12: Score of the best Type 2 trail found for number of rounds evaluated by the original submission document.

Rounds	384	512
[WH19] (Simple)	28	47
Ours (Refined)	28	47

As mentioned in the design document [WH19], one can upper bound the probability of the forgery attack in which the differential propagation go through two invocations of $\mathcal{P}_K$ by combining a trail of Types 1 and 2 for 384 rounds. By our analysis such combination for two times 384 rounds will have a probability at most $2^{-41-28} = 2^{-69}$. The former analysis [WH19] derived an upper bound of 2^{-73}, 16 times lower.

5 Linear Cryptanalysis

5.1 Refined Analysis for Partially Restricted Keyed Permutation

As we ran the refined model for linear trails on `TinyJAMBU`, the best linear trails were consistently having no correlated gates. This implies that the optimal solution either coincides or is worse than simple model's one.

The score value of the objective function should be interpreted as such: a score of k (active gates minus joint linearizations) implies a correlation of 2^{-k} and so a bias of 2^{-k-1}. Table 13 compares the different results.

Table 13: Score of the best linear trail with unrestricted input, restricted output found for number of rounds evaluated by the original submission document. Question mark (?) signifies that the solver did not finish.

Rounds	256	320	384	448	512
[WH19]	12	16	22	26	29
Ours (Refined)	10	15	22	27?	46?

Our results for 256 and 320 rounds do not contain any joint linearization but imply a better bias than the one given in [WH19]. This is inconsistent with the fact that the designers used the simple model to evaluate the linear properties. We could experimentally test the linear trail for 256 rounds and found out that for about a quarter of the keys (253 out of 1000 tested) the absolute bias was $2^{-9.66}$ while the other keys gave an absolute bias of $2^{-12.00}$. The bias was computed with 2^{32} trials to be significant. Thus the total expected absolute bias computed experimentally is $2^{-10.99}$ with a clear dependency on the key. The absolute bias of a linear trail does not depend on the key but, in this case, we found 4 trails

with the same input/output mask: the optimal one with a 2^{-11} bias and three others with a 2^{-12} bias. Their sign and, therefore, the way they sum up together (either constructively or destructively) entirely depends on the key value. Experimentally, we only observed two regimes: for 1/4th of the keys the trails sum up for a $3 \times 2^{-11} + 2^{-12} = 2^{-9.68}$ bias and for the rest the trails cancel each other to a 2^{-12} bias. The conditions on the key were not extracted and we leave the existence of weak keys with strong linear bias as an open question.

5.2 Linear Bias of the Tag in the AEAD Setting

Another interesting linear trail to look for is the best linear relation we can find between the two 32-bit words that form the tag in `TinyJAMBU`. Indeed the tag in `TinyJAMBU` is formed by outputting the second most significant 32-bit word then applying 384 rounds of the cipher before outputting again the second most significant 32-bit word. The tag is simply formed by concatenation. With the corresponding restriction the best linear trail found is shown in Table 14. This trail has 40 active gates and no correlation so it has an expected bias of 2^{-41}. It has been validated by the stricter model of Section 3.2.

Table 14: Bias 2^{-41} optimal linear trail for 384 rounds found with the refined model. The mask on the input and output data block can only be active on the second most significant 32-bit word. Those are stressed by red. Information of the intermediate masks is provided to recover all linear approximation steps.

Input:	$mS_{127..0}$	00000000	41100081	00000000	00000000
	$mS_{255..128}$	00408000	41120491	02008024	08000088
	$mS_{383..256}$	30c80024	41804890	00449144	80000089
Output:	$mS_{511..384}$	00000000	00022890	00000000	00000000

A bias of 2^{-41} on the tag's bits would be detectable roughly after 2^{82} collected values which do not contradict the authors' claims.

Moreover the stricter model found 4 linear trails with 40 active gates and 63 with 41 active gates with the input/output linear characteristic of Table 14. The unknown sign of the bias forbids us to simply sum up all the biases like we did for differential paths. However, we can compute the expected absolute value of the correlation by taking the square root of the sum of the square of the correlation of each trails as shown in [AES01, Theorem 7.9.1]. Therefore, taking only the 40 and 41 active gates trails and ignoring the less correlated ones, we can expect an absolute value of the correlation of $\sqrt{4 \cdot 2^{40 \cdot 2} + 63 \cdot 2^{41 \cdot 2}} = 2^{-37.85}$ and thus an expected value of the bias of $2^{-38.85}$ requiring $2^{77.7}$ data to detect.

Notice that [AES01, Theorem 7.9.1] supposes independent random subkeys which is not the case in `TinyJAMBU`. This theorem notably fails to apply for the 256 rounds linear trail in Section 5.1. Analysis of the key dependencies and their relations might give further insight on the actual absolute bias. Nevertheless, this provides for a good approximation of what we could expect.

6 Conclusion

We presented the refined model to efficiently find highly accurate differential and linear trails of `TinyJAMBU`. With the refined model, we found a forgery attack with complexity $2^{62.68}$ on 338 rounds and a differential trail with probability $2^{-70.68}$ for the full 384 rounds. Those imply that the security margin of `TinyJAMBU` is smaller than originally expected, which is now 12% with respect to the number of unattacked rounds or less than 8 bits in

the data complexity for the full rounds. We also applied the refined model to the linear cryptanalysis and found the better bias for some number of rounds.

Though our attacks do not contradict the original security claims, the attacks are generic and the margins are already quite small. One simple solution would be to increase the number of rounds of the small version, $\mathcal{P}_K$, from 384 to 512 rounds. As another suggestion, we noted that the design rationale may have been influenced by the simple model. We cite in [WH19]: *We choose the tap positions that gives excellent differential and linear characteristics.* Therefore using the refined model instead may lead to a better choice of tap positions with respect to differential and linear cryptanalysis.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. We also would like to thank the attendees of ASK 2019 for having helpful discussions. Siwei Sun, Danping Shi, and Yingjie Zhang are supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704704), the Chinese Major Program of National Cryptography Development Foundation (Grant No. MMJJ20180102), the National Natural Science Foundation of China (61772519, 61802400), and the Youth Innovation Promotion Association of Chinese Academy of Sciences. Ferdinand Sibleyras is partly supported by the French DGA.

References

[AEL+18] Tomer Ashur, Maria Eichlseder, Martin M. Lauridsen, Gaëtan Leurent, Brice Minaud, Yann Rotella, Yu Sasaki, and Benoît Viguier. Cryptanalysis of MORUS. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 35–64. Springer, 2018.

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[AK18] Ralph Ankele and Stefan Kölbl. Mind the Gap - A Closer Look at the Security of Block Ciphers against Differential Cryptanalysis. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography, 2018*, volume 11349 of *Lecture Notes in Computer Science*, pages 163–190. Springer, 2018.

[BC09] John Black and Martin Cochran. MAC reforgeability. In Orr Dunkelman, editor, *FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 345–362. Springer, 2009.

[BDPA11] 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, *Selected Areas in Cryptography, 2011*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[CAE] CAESAR: Call for Submission. http://competitions.cr.yp.to/.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015.

[LDB+19] Zheng Li, Xiaoyang Dong, Wenquan Bi, Keting Jia, Xiaoyun Wang, and Willi Meier. New conditional cube attack on keccak keyed modes. *IACR Trans. Symmetric Cryptol.*, 2019(2):94–124, 2019.

[LDW17] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. Conditional cube attack on round-reduced ASCON. *IACR Trans. Symmetric Cryptol.*, 2017(1):175–202, 2017.

[MWGP] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer.

[Nat19a] National Institute of Standards and Technology. Lightweight Cryptography (LWC) Standardization project, 2019. https://csrc.nist.gov/projects/lightweight-cryptography.

[Nat19b] National Institute of Standards and Technology. Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process, 2019. https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8268.pdf.

[NMSS18] Yusuke Naito, Mitsuru Matsui, Takeshi Sugawara, and Daisuke Suzuki. SAEB: A lightweight blockcipher-based AEAD mode of operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):192–217, 2018.

[SHW+15a] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. Constructing mixed-integer programming models whose feasible region is exactly the set of all valid differential characteristics of SIMON. Cryptology ePrint Archive, Report 2015/122, 2015. http://eprint.iacr.org/2015/122.

[SHW+15b] Siwei Sun, Lei Hu, Meiqin Wang, Qianqian Yang, Kexin Qiao, Xiaoshuang Ma, Ling Song, and Jinyong Shan. Extending the applicability of the mixed-integer programming technique in automatic differential cryptanalysis. In *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings*, pages 141–157, 2015.

[SSS+19] Danping Shi, Siwei Sun, Yu Sasaki, Chaoyun Li, and Lei Hu. Correlation of quadratic boolean functions: Cryptanalysis of all versions of full MORUS. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 180–209, 2019.

[WH15] Hongjun Wu and Tao Huang. JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU. Submission to CAESAR: Competition for Authenticated Encryption. Security, Applicability, and Robustness, 2015. http://competitions.cr.yp.to/round2/aesjambuv2.pdf.

[WH19] Hongjun Wu and Tao Huang. TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms. the NIST Lightweight Cryptography (LWC) Standardization project (A Round-2 Candidate), 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/TinyJAMBU-spec-round2.pdf.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.