IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2023, No. 2, pp. 176–188. DOI:10.46586/tosc.v2023.i2.176-188

Practical Related-Key Forgery Attacks on Full-Round TinyJAMBU-192/256

Orr Dunkelman, Shibam Ghosh and Eran Lambooij

Department of Computer Science, University Of Haifa, Haifa, Israel
orrd@cs.haifa.ac.il, sghosh03@campus.haifa.ac.il, eran@hideinplainsight.io

Abstract. TinyJAMBU is one of the finalists in the NIST lightweight cryptography competition. It is considered to be one of the more efficient ciphers in the competition and has undergone extensive analysis in recent years as both the keyed permutation as well as the mode are new designs. In this paper we present a related-key forgery attack on the updated TinyJAMBU-v2 scheme with 256- and 192-bit keys. We introduce a high probability related-key differential attack where the differences are only introduced into the key state. Therefore, the characteristic is applicable to the TinyJAMBU mode and can be used to mount a forgery attack. The time and data complexity of the forgery are 2^{33} using 2^{14} related-keys for the 256-bit key version, and 2^{43} using 2^{16} related-keys for the 192-bit key version.
For the 128-bit key we construct a related-key differential characteristic on the full keyed permutation of TinyJAMBU with a probability of 2^{-16}. We extend the related-key differential characteristics on TinyJAMBU to practical-time key-recovery attacks that extract the full key from the keyed permutation with a time and data complexity of 2^{24}, 2^{21}, and 2^{19} for respectively the 128-, 192-, and 256-bit key variants.
All characteristics are experimentally verified and we provide key nonce pairs that produce the same tag to show the feasibility of the forgery attack. We note that the designers do not claim related-key security, however, the attacks proposed in this paper suggest that the scheme is not key-commiting, which has been recently identified as a favorable property for AEAD schemes.

Keywords: TinyJAMBU · Differential cryptanalysis · Related-Key · Forgery · NIST-LWC

1 Introduction

In recent years it has been understood that, although confidentiality is an important feature of cryptographic primitives, authenticity of the data is, often overlooked, but equally valuable. Therefore, when NIST launched the lightweight cryptography competition [Tec17], they included authenticity as one of the primary design goals.

When looking at authenticated encryption, the scheme is considered to be broken when either the confidentiality or the authenticity of the ciphertext are diminished. This both improves and worsens the job of a cryptanalyst, as now, for a full break of the scheme, two things must happen in parallel: First the cryptographer has to find a favorable property in the primitive; Second, it must be possible to observe this property through the (often restrictive) mode.

In this paper we look at the authenticated encryption scheme TinyJAMBU [WH19, WH21], which is based on the duplex construction [BDPA08, BDPA11] and a lightweight keyed permutation. The keyed permutation used in TinyJAMBU has a 128-bit state and a 128-, 192-, or 256-bit key. The permutation is constructed from a Non-Linear Feedback

Received: 2023-03-01 Accepted: 2023-05-01 Published: 2023-06-16

Shift Register (NLFSR), with a single NAND gate as the only non-linear component. Each round the feedback bit is XOR-ed with the next key bit using a cyclic key schedule.

In the TinyJAMBU mode two versions of the keyed permutation are used, which only differ in the number of rounds used in the NLFSR. We denote these as: $\mathcal{P}^a$, and $\mathcal{P}^b$. $\mathcal{P}^a$ is used in phases where no output is observed, and consists of 640 rounds for all the key sizes. $\mathcal{P}^b$ is used in the first initialization step and when part of the state can be observed. It consists of 1024, 1152, and 1280 rounds for respectively the 128-, 192-, and 256-bit key variants. We note that the designers of TinyJAMBU have recently changed the number of rounds of $\mathcal{P}^a$ from 384 to 640 [WH21] to counter a differential attack by Saha *et al.* [SSS+20]. Therefore, most of the results in the literature are on 384 rounds.

We first look at the current results on TinyJAMBU before we turn our attention to our contributions. In their specification of TinyJAMBU [WH19, WH21] the designers provide a security analysis of the design against various attacks. Using MILP modelling they show that there exists a differential characteristic through 384 rounds of the permutation with a probability of less than 2^{-78}. In [SSS+20], Saha *et al.* look at the differential characteristics through the TinyJAMBU permutation using a model that captures (first-order) relations between multiple NANDs. This improves the probability of the best found characteristics compared to the previous work. The authors propose a differential characteristic through 338 and 384 rounds of the permutation with a probability of respectively $2^{-62.68}$ and $2^{-70.68}$.

Most attacks consider the case where the attacker only has access to 32 in- and output bits as is dictated by the mode. When there are no constraints on the bits that the attacker can access, Saha *et al.* report a characteristic through 384 rounds of the keyed permutation with probability 2^{-19} [SSS+20]. To mitigate these attacks, the designers of TinyJAMBU increased the number of rounds from 384 to 640 in the second version [WH21] of the specification.

In the updated specification the designers of TinyJAMBU improved the differential characteristics through the keyed permutation in the case that the attacker can only affect 32 of the in- and output bits. They found differential characteristics with probabilities 2^{-41}, 2^{-64}, and 2^{-88} covering respectively 384, 512, and 640 rounds.

Due to the cyclic nature of the permutation slide attacks are a natural direction of analysis. Sibleyras *et al.* [SST+22] discuss full round slide attacks on TinyJAMBU in the single-key setting. The designers of TinyJAMBU already mention in the specification that there exists a simple related-key slide attack. But, due to the frame bits used in the mode these slide attacks cannot be used to attack the full scheme. The complexities of these attacks are beyond the birthday bound.

1.1 Our Contributions

We propose an iterative related-key differential characteristic that can be applied to the keyed permutation inside the mode. The attack works for the 256-bit as well as the 192-bit key variants of the permutation. Combining this characteristic with the proper nonce differences we can reach a zero difference in the state just before the first message addition with a probability of 2^{-32} and 2^{-42} for respectively the 256- and 192-bit key variants. For a given initial state and a given 256-bit key-pair the attack succeeds with probability 2^{-14} and for a 192-bit key it succeeds with probability 2^{-18}. If we allow the attacker to request 2^5 related-keys, the success probability increases to 2^{-9} and 2^{-13} for the 256- and 192-bit variants. We note that the designers of TinyJAMBU do not claim security in the related-key model, however the existence of a (high probability) related-key differential characteritic shows that the scheme is not key-committing [ADG+22].

We extend this attack to a nonce respecting forgery attack which has a data and time complexity of 2^{32} and 2^{42} for respectively the 256- and 192-bit variants of TinyJAMBU.

Table 1: A summary of distinguishers on TinyJAMBU. The results only include distinguishers where the attacker has access to 32 in-, and output bits.

Key size	#Rounds	Data/#Weak Keys	Setting	Type	Source
128	1024	$2^{17}/-$	RK CP	Differential	Sec. 3.3
192	1152	$2^{13}/2^{179}$	RK CP	Differential	Sec. 3.3
256	1280	$2^{11}/2^{247}$	RK CP	Differential	Sec. 3.3
any	338	$2^{62.68}$	CP	Differential	[SSS+20]
any	384	$2^{70.68}$	CP	Differential	[SSS+20]
any	384	2^{41}	CP	Differential	[WH21]
any	512	2^{64}	CP	Differential	[WH21]
any	512	2^{64}	KP	Linear	[WH21]
any	640	2^{88}	CP	Differential	[WH21]
128	∞	2^{64}	KP	Slide	[SST+22]
192	∞	2^{65}	ACP	Slide	[SST+22]
256	∞	$2^{67.5}$	ACP	Slide	[SST+22]

CP = Chosen Plaintext
ACP = Adaptive Chosen Plaintext
KP = Known Plaintext
RK = Related-key

The results are summarised in Table 1 and we provide inputs that lead to a forgery for both versions in Table 4.

The 128-bit key variant of TinyJAMBU cannot be attacked using the same characteristic. But, we show that if we allow for differences to be inserted into state bits, we can get a similar iterative characteristic for the 128-bit variant with a probability of 2^{-2} per 128 rounds.

We note that our results are also applicable to the first version of the TinyJAMBU scheme, but with slightly better complexities.

1.2 Paper Structure

We give a short overview of the mode and the keyed permutation used in TinyJAMBU in Section 2. Next we describe the related-key differential characteristic and the resulting forgery on the full mode in Section 3 and Section 4. We conclude the paper in Section 5.

2 The Specification of TinyJAMBU

TinyJAMBU is one of the finalists of the NIST lightweight competition. The design principle of TinyJAMBU is based on the sponge duplex mode using a keyed permutation. The keyed permutation is derived from a NLFSR with a 128-bit state using a NAND gate as the non-linear operation. The key bits are added in a cyclic fashion.

In the upcoming section we give a quick overview of the important parts of TinyJAMBU with respect to understanding the attack. For a complete specification of TinyJAMBU we refer the reader to the full specification [WH21].

2.1 The keyed permutation

The TinyJAMBU keyed permutation uses a 128-bit state and is defined for 128-, 192-, or 256-bit keys. Given an n-bit register $x \in \{0,1\}^n$, x_i denotes the i-th bit of the register. Where x_0 is the least significant bit, and x_{n-1} is the most significant bit of the register.

Table 2: Parameters for the different variants of TinyJAMBU.

Variant	a	b	State	Key	Nonce	Tag
TinyJAMBU-128	640	1024	128	128	96	64
TinyJAMBU-192	640	1152	128	192	96	64
TinyJAMBU-256	640	1280	128	256	96	64

Using this notation we can denote the n-bit register x as $(x_{n-1}, x_{n-2}, \ldots, x_0)$. The size of a register x is denoted as $|x|$ and the concatenation of two registers x, y is denoted as $x\|y$.

We define the permutation $\mathcal{P}$ on the state $v \in \{0,1\}^{128}$, given a key $k \in \{0,1\}^{\kappa}$ (where $\kappa \in [128, 192, 256]$), and round i as:

$$\mathcal{P}_{(k,i)}(v) = (v_{91} \oplus \overline{v_{85} v_{70}} \oplus v_{47} \oplus v_0 \oplus k_{(i \mod |k|)}, v_{127}, \ldots, v_1)$$

Now we define the r-round permutation $\mathcal{P}$ for some key k as:

$$\mathcal{P}^r = \mathcal{P}_{(k,r-1)} \circ \mathcal{P}_{(k,r-2)} \circ \ldots \circ \mathcal{P}_{(k,0)}$$

The round function $\mathcal{P}$ of the TinyJAMBU permutation is depicted in Figure 1.

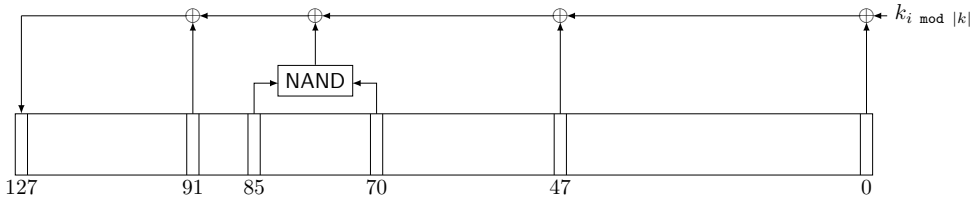

Figure 1: Round function of the TinyJAMBU permutation.

2.2 The TinyJAMBU mode

The TinyJAMBU mode uses the permutation described in Section 2.1 in a duplex construction with a 32-bit message injection part, a 32-bit squeezing part. The mode consists of four separate phases: the key initialization, the associated data, the encryption, and the finalization part. After every permutation call a constant depending on the current phase const_ℓ is added to the state. One thing to note is that unlike most duplex constructions the squeezing and the injection of data occurs in different parts of the state.

TinyJAMBU uses keyed permutations using the same key k, but a different number of rounds in different phases of the encryption. We denote them as $\mathcal{P}^a$ and $\mathcal{P}^b$ where the values of a and b for the different key sizes are given in Table 2.

2.2.1 Initialization.

In the initialization the key k is mixed into the state $s \in \{0,1\}^{128}$ by applying $\mathcal{P}^b$ to the initial state $s = (0, 0, ..., 0)$. After that, in the nonce setup phase, a 96-bits nonce N is split up into three 32-bit nonce parts $N_0\|N_1\|N_2$ and for each part of the nonce the state is updated with $\mathcal{P}^a$ after which the nonce is added to the most significant bits of the state. A depiction of the initialization is given in Figure 2.

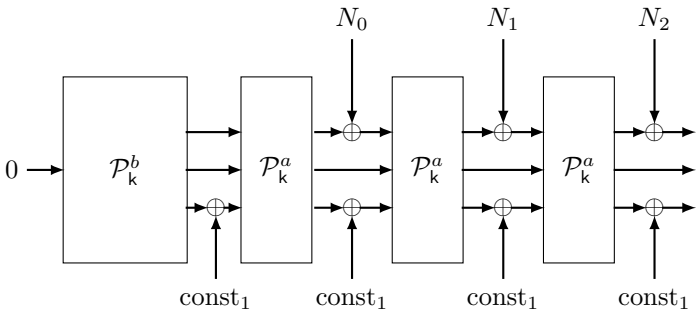

Figure 2: TinyJAMBU initialization and nonce addition.

2.2.2 Associated Data Processing.

During the associated data processing the associated data is added to the state. The associated data is divided into 32-bit blocks. For each block the state is updated with $\mathcal{P}^a$, after which the associated data block is XOR-ed into the state. When the associated data is empty we skip this part of the construction and continue with the encryption. We depict the associated data processing in Figure 3.

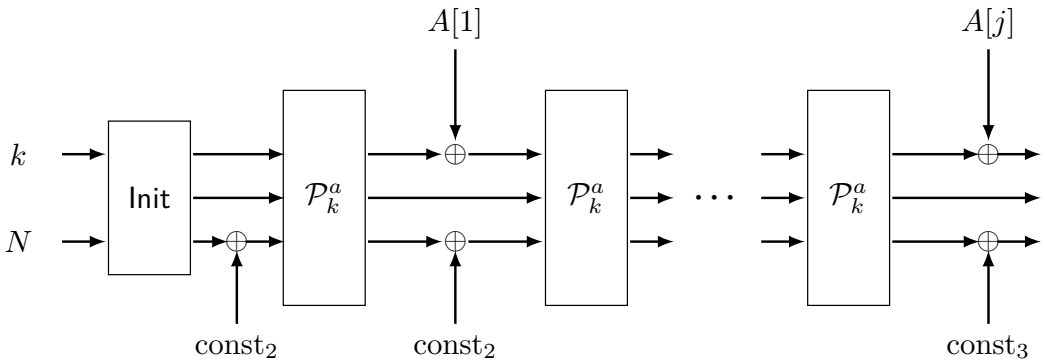

Figure 3: Processing the associated data.

2.2.3 Encryption.

During the encryption stage a keystream is generated to encrypt a message into a ciphertext. The plaintext is divided into 32-bit blocks. For each block, the state is updated with $\mathcal{P}^b$, after which the plaintext block is XORed into the most significant part of the state. Finally, we obtain the 32-bit ciphertext block by XORing bits $95 \ldots 64$ of the state with the plaintext block. Note that, the plaintext and nonce are added to the 32 most significant bits of the state which are $127 \ldots 96$. The keystream used for encryption is obtained from bits $95 \ldots 64$.

2.2.4 Finalization.

After encrypting the plaintext the 64-bit authentication tag $T_0 \| T_1$ is generated in two steps. First, to generate T_0, we apply $\mathcal{P}^b$ and extract bits $95 \ldots 64$. Then we apply $\mathcal{P}^a$ after which we extract the same 32 bits of the state to get T_1. We depict the finalization and encryption process in Figure 4.

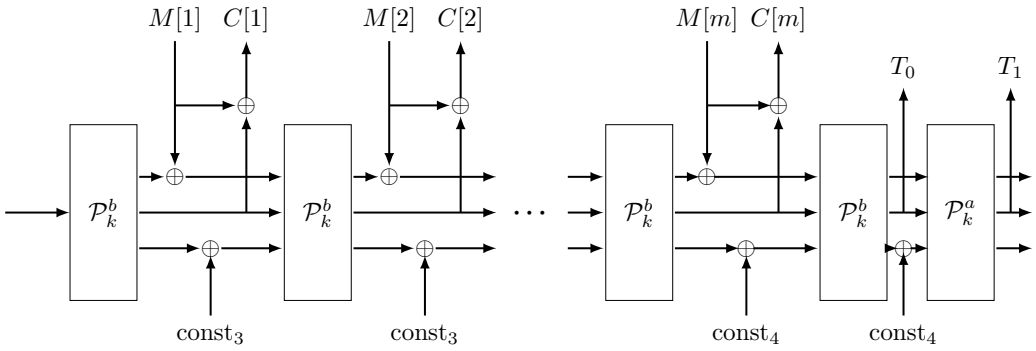

Figure 4: TinyJAMBU encryption and finalization.

3 Related Key Differential Characteristics on $\mathcal{P}$

We describe a high probability related-key differential characteristic on the full-round permutation $\mathcal{P}$ with 256- and 192-bit keys. This characteristic uses the simplicity of the key schedule of the keyed permutation to introduce and cancel differences in the state. Moreover, at all times there is at most one active bit in the state, which gives us a characteristic through the full permutation $\mathcal{P}^{1280}$ with a probability of 2^{-10} for the 256-bit key variant. For the 192-bit key variant the characteristic has a probability of 2^{-12} through $\mathcal{P}^{1152}$. The probability for the characteristic through $\mathcal{P}^{640}$ is only 2^{-4} for the 256-bit variant and 2^{-6} for the 192-bit variant. These characteristics allow an attacker to reach a zero difference state just before the message addition with a probability of 2^{-32} for the 256-bit key case and 2^{-42} for the 192-bit key case. In the following sections we explain how the characteristic is constructed.

3.1 The 256-bit key variant

We start with a difference in key bit k_0 which is inserted into the state in the first round. The propagation of the difference through the linear taps is cancelled by the following key differences: k_{37}, k_{81}, k_{128}. After 128 rounds the difference is cancelled and the state difference is 0 for the next 127 rounds with probability $(\frac{1}{2})^2$. This is because when the introduced state difference reaches a NAND gate it produces a zero difference with probability $\frac{1}{2}$ and the difference enters the NAND gate twice before getting cancelled. At round 256 we return to the original configuration of state and key differences.

The above key difference gives a zero state difference after $\mathcal{P}^{1280}$ with probability 2^{-10}. However, if we use the same key differences for $\mathcal{P}^{640}$ there is a difference in the least significant bit of the state. The problem here is that we cannot cancel this difference using the nonce, since the nonce is added to the most significant bits of the state. The solution to this problem is to shift the key difference by 127 bits. This shifts the difference to the most significant bit of the state after $\mathcal{P}^{640}$, and keeps the 0 difference in the state after $\mathcal{P}^{1280}$. Since we can cancel the 32 most significant bits using the Nonce we actually have 32 possible characteristics. The above gives us the following set of possible useable input differences in the key bits:

$$(k_{164-t}, k_{208-t}, k_{255-t}, k_{127-t}) \quad \text{for } 0 \leq t < 32$$

and a difference in the following nonce bits (where the nonce is denoted by N):

$$(N_{95-t}, N_{63-t}, N_{31-t}) \quad \text{for } 0 \leq t < 32.$$

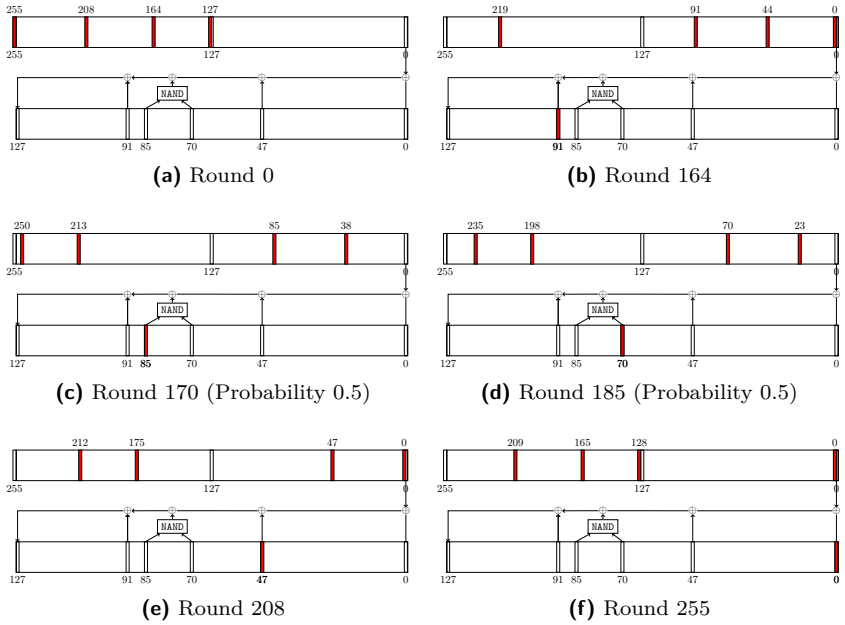

Figure 5: The iterative differential characteristic for the 256-bit key case. The rounds are modulo 256.

This produces a one bit difference in the output in the t-th most significant bit after $\mathcal{P}^{640}$, while still keeping the zero difference after $\mathcal{P}^{1280}$. One added benefit is that for $\mathcal{P}^{640}$ the characteristic passes through two less non-linear taps, reducing the probability of the characteristic from 2^{-6} to 2^{-4}. The characteristic for $t = 0$ is given in Figure 5.

Due to this characteristic, we get, with probability $2^{-10-3\cdot4-10} = 2^{-32}$, a zero difference before the message addition. This is the first point in which we can observe (part of) the state difference through the ciphertext.

An important note to make with regards to this characteristic is that, because of the mode, in the key-initialization the state is only affected by the key and the initial state. We can only add randomness to the state after one $\mathcal{P}^a$ and one $\mathcal{P}^b$. Due to this, the characteristic holds for a certain related key pair with a probability of 2^{-14}. There are 32 characteristics that can lead to a forgery, thus the probability that we can mount a forgery against a specific key is 2^{-9}.

We investigated the possibility of allowing for state differences with a Hamming weight greater than one (*i.e.*, two). However, this does not allow us to improve the distinguisher. The probability of such a characteristic is (roughly) squared when compared to the probabilities of the characteristics described in this section.

3.2 The 192-bit key variant

The characteristic for the 192-bit variant is nearly the same as the characteristic for the 256-bit variant. Between the two, the only difference is that instead of offsetting the characteristic by 127 positions to get the 1 difference in the most significant bit after $\mathcal{P}^{640}$, we offset the difference by 63. This gives us the following set of possible input differences in the key bits:

$$(k_{100-t}, k_{144-t}, k_{191-t}, k_{63-t}) \quad \text{for } 0 \leq t < 32$$

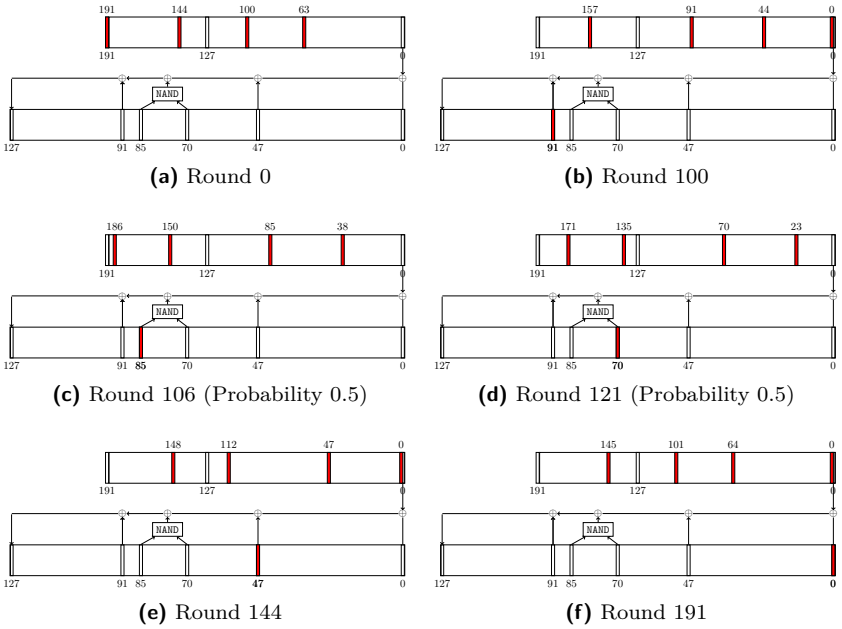

Figure 6: The iterative differential characteristic for TinyJAMBU with a 192-bit key. The rounds are modulo 192.

and the following input differences in the nonce bits (where the nonce is denoted by N):

$$(N_{95-t}, N_{63-t}, N_{31-t}) \quad \text{for } 0 \leq t < 32$$

We note that, compared to the 256-bit variant, the probability is slightly lower. This is due to the fact that instead of 2 non-linear taps per 256 rounds of the cipher we get 2 non-linear taps per 192 rounds of the cipher. This leads to a characteristic that has a probability of 2^{-6} and 2^{-12} for respectively $\mathcal{P}^{640}$, and $\mathcal{P}^{1152}$ and produces a 1 bit difference in the (t-th) most significant bit of the state after $\mathcal{P}^{640}$ and a zero difference after $\mathcal{P}^{1152}$. In Figure 6 we depict the characteristic for $t = 0$.

Similarly to the 256-bit key case the probability that the characteristic can be used in a forgery attack with a specific key is 2^{-13}.

3.3 Other Related Key Characteristics on $\mathcal{P}^{1024}$

The characteristics in Section 3 do not work for the 128-bit key variant as it requires cancelling the state difference with the key difference after 129 rounds. However, if we allow the attacker to insert a difference into the state as well as the key we can attack the 128-bit key variant. Or in other words, if we analyse the permutation as a stand alone primitive, we can use the same idea to construct a differential characteristic for the 128-bit key variant. This idea could be used to construct a fault-attack on the scheme.

We insert a difference in the state on bit v_{127}, and a difference in the key state in k_{36} and k_{80}. This difference is chosen such that the difference does not diffuse through the linear taps of the NLSFR. Since there is one active bit the bit reaches a NAND gate twice every 128 rounds. This leads to a probability of 2^{-2} per 128 rounds and a probability of 2^{-16} for $\mathcal{P}^{1024}$.

We can improve the probability of the distinguisher by a factor of 2^{-2} by moving the difference bit in the state to behind the second non-linear tap, i.e., v_{69}.

3.4 Key recovery attack

In the case that we allow for the attacker to observe the full input and output states we can do a key recovery attack on the keyed permutation using the characteristics discussed in Section 3.3. Such an attack is not applicable to the TinyJAMBU construction, but is given as a warning against using the permutation as a stand alone primitive. To construct the attack we use the observation that, if the output difference of the NAND is zero while one of the input differences is one, the values of the non-difference bits of the NAND are 0. Thus, observing the output difference immediately leaks the value of one bit of the state in 16 rounds of the cipher. By looking at the first two state bits that are leaked we can recover two of the key bits. We know that the value of these bits need to be zero and since the input of the first NAND only depends linearly on the key bits, we can compute the value of the key bits involved. The two first state bits that are leaked give us the following equation:

$$v_{i+91} \oplus v_{i+85}v_{i+70} \oplus v_{i+47} \oplus v_i \oplus k_i \oplus 1 = 0$$

Thus, we can recover key bits k_0 and k_{15} in the chosen plaintext model.

By using multiple characteristics we can recover $k_0 \dots k_{37}$. The rest of the key bits can be recovered by first recovering $k_0 \dots k_{37}$ and using the recovered key bits to simplify the expressions for the subsequent bits. Thus, to recover the full key we need to run the distinguisher at most 2^8 times. This allows us to recover the full key in $2^{16+7} = 2^{23}$, $2^{12+8} = 2^{20}$, and $2^{10+8} = 2^{18}$ time and data for respectively the 128-, 192-, and 256-bit primitives. For the 128-bit variant we use the characteristic described in Section 3.3, while for the 192-, and 256-bit variants we use the characteristic described in Section 3.1 and Section 3.2

3.5 Experimental Verification

To verify the existence and theoretical probability of the differential described in the paper we ran experiments. For TinyJAMBU-192 we conducted 50 experiments with 2^{32} nonce-key pairs and checked for the difference after the key-setup and nonce initialization. The probability of the distinguisher for the key and nonce initialization is: $2^{-30.0} \pm 2^{-30.0}$.[1] Due to the nature of the characteristic, where only a part of the keys leads to a forgery, we also tested the probability that the required difference exists after the key initialization as well as between the key initialization and nonce addition. For these we did 50 experiments, each with respectively 2^{14} and 2^{20} keys. The experimental probability that a key difference produces the correct output difference after the key initialization is: $2^{-12.0} \pm 2^{-13.1}$ and the experimental probability that after the characteristic passed through the key initialization it also passes through the nonce addition is $2^{18} \pm 2^{19.3}$.

For TinyJAMBU-256 we conducted 50 experiments with 2^{24} nonce-key pairs to check the existence and estimate the probability of the distinguisher after the key-setup and nonce initialization. The probability of the distinguisher for the key and nonce initialization is: $2^{-22.0} \pm 2^{-23.2}$. We also tested the probability that the required difference exists after the key initialization. For this we did 50 experiments each with 2^{20} keys. The probability that a key difference produces the correct output difference after the key initialization is: $2^{-10.0} \pm 2^{-15.0}$. See Table 3 for an overview of the experiments.

We also verified the probabilities for the differential distinguisher we described on the permutation with a 128-bit key. The probability that the described differential occurs is: $2^{-15.9} \pm 2^{-18.0}$ (with 50 experiments each testing 2^{20} keys.

The source code for the experiments can be found in https://github.com/ShibamCrS/TinyAtttacksOnTinyJambu.git.

[1] We use $\mu \pm \sigma$, where μ is the (sample) mean and σ is the standard deviation.

Table 3: The results of the experiments to show the existence and probability of the differential after the key initialization, the nonce intialization with a random state, and the combined key and nonce initialization.

Key Size	Stage	Theoretical Prob.	Experimental Prob.	# Keys/Nonces tested
192	Key	2^{-12}	$2^{-12.0} \pm 2^{-13.1}$	$50 \cdot 2^{14}$
	Nonce	2^{-18}	$2^{-18.0} \pm 2^{-19.3}$	$50 \cdot 2^{20}$
	Key + Nonce	2^{-30}	$2^{-30.0} \pm 2^{-30.0}$	$50 \cdot 2^{32}$
256	Key	2^{-10}	$2^{-10.0} \pm 2^{-15.2}$	$50 \cdot 2^{20}$
	Nonce	2^{-12}	$2^{-11.9} \pm 2^{-13.2}$	$50 \cdot 2^{14}$
	Key + Nonce	2^{-22}	$2^{-22.0} \pm 2^{-23.2}$	$50 \cdot 2^{24}$
128	Key	2^{-16}	$2^{-15.9} \pm 2^{-18.0}$	$50 \cdot 2^{20}$

4 Forgery Attack on TinyJAMBU

Using the related-key differential characteristic from Section 3 we can create a forgery attack on TinyJAMBU. We generate a key, nonce, message pair that produces the same tag for different nonces and keys. We first discuss the attack on the 256-bit key version of TinyJAMBU, but is also applicable to the 192-bit key version, although with a slightly higher complexity.

The main hurdle in creating a forgery is to find a related-key pair that can be used for the attack. This is mainly because during the key initialization the only input we have access to is the key, and the fixed initial state. It is only after the key initialization, in the nonce setup, that we can insert more data to randomize the state. To reach the first non-key input we need to go through one $\mathcal{P}^b$ and one $\mathcal{P}^a$, so the probability that a 256-bit key pair is susceptible to a forgery attack is 2^{-14}. Since we can use 32 different characteristics we can ask for data encrypted under 32 related keys, *i.e.*, the key pairs $(k, k \oplus \Delta_0), (k, k \oplus \Delta_1), \ldots, (k, k \oplus \Delta_{31})$. The probability that any pair in this group is susceptible to a forgery is 2^{-9}. We note that this does not increase the success probability in the related-key model, but it does increase the weak-key class, as for any key there are 32 keys that can be used to mount the attack.

To find a forgery we start with 2^{14} key pairs. For each pair we encrypt the same message block with 2^{18} nonce pairs. The difference for the nonce is such that it cancels the difference after $\mathcal{P}^{640}$. We can observe the output difference through the first ciphertext. We expect to see one 'golden' key pair that survives this initial filtering step, which with high probability, follows the characteristic through the key initialization and the nonce addition.

Using this key and nonce pair we search for a forgery by changing the last nonce. Since we have to pass once through $\mathcal{P}^{640}$ and twice through $\mathcal{P}^{1280}$ to produce both tags, the probability of finding a colliding tag is 2^{-24} after we found a golden key and nonce pair. One thing to consider is that the last finalization step is using $\mathcal{P}^{640}$ as the permutation, which as discussed in Section 3 produces a difference in the most significant bit of the state. Nevertheless, since we extract bits 64 to 95 to use as the tag, the tag difference remains zero.

The total cost of this attack is $2^{33} + 2^{25}$ data and time, where we ask for encryptions under 2^{14} related key pairs in the nonce respecting setting. If we move to the nonce misuse setting we get a forgery in $2^{33} + 2^{15}$ time since we can add the randomness in the message instead of the last nonce part. We note that finding more forgeries after finding a 'golden' key nonce pair has a complexity of 2^{25} data and time in the nonce respecting setting and 2^{14} data and time for the nonce misuse setting. The advantage of the attacker in the single-key model after 2^{33} queries is 2^{-30} according to the proof given in the design document of TinyJAMBU.

Table 4: Key and nonce pairs that produce the same tag (and ciphertext). Nibbles that have a difference are marked. The leftmost byte is the least significant byte.

Key size	Key	Nonce	Message	Ciphertext	Tag
192	9AE19248 8B102E07 AB0F2C02 9EDB377D 090EF19C 66F4AAEB	19A2492E DF81AB70 923635DC	11129DA1	C9211BA2	1734A489 1229B9F6
	9AE19248 8B102E87 AB0F2C02 8EDB377D 090EF09C 66F4AA6B	19A249AE DF81ABF0 9236355C	11129DA1	C9211BA2	1734A489 1229B9F6
256	B429DBD1 14F8B269 7D83ABD0 3893F974 79626DF1 B3A3D867 A415E2BB D5A2A68A	BF8A51BD B71DC3C6 8443C018	29594AD7	E015A04A	1E8CA308 95CBD1F7
	B429DBD1 14F8B269 7D83ABD0 3893F9F4 79626DF1 A3A3D867 A415E3BB D5A2A60A	BF8A513D B71DC346 8443C098	29594AD7	E015A04A	1E8CA308 95CBD1F7

The complexities for the forgery are low enough that we could compute a forgery on our own desktop. In Table 4 we provide key and nonce inputs that produce the same tag for both the 256- and 192-bit key cases. The code to generate these forgeries can be found in the following repository: https://github.com/ShibamCrS/TinyAtttacksOnTinyJambu.git.

4.1 TinyJAMBU-192

The forgery attack on TinyJAMBU-192 is the same as the forgery attack on TinyJAMBU-256, although the probabilities (as is discussed in Section 3) are slightly higher. The probability for the characteristic to pass through $\mathcal{P}^{640}$ and $\mathcal{P}^{1152}$ is respectively 2^{-6} and 2^{-12}. This leads to a forgery with a data and time complexity of $2^{43} + 2^{31}$ in the nonce respecting setting and a data and time complexity of $2^{43} + 2^{19}$ in the nonce misuse setting. The advantage of the attacker in the single-key model after 2^{43} queries is 2^{-20} according to the proof given in the design document of TinyJAMBU.

4.2 Weak-Related-Key Model

Due to the fact that we can only influence the state bits after some applications of the internal permutation, not every key-pair with the correct input difference follows the characteristic. Due to this, the forgery only works on a (large) subset of weak keys. A key is in this weak key class, if there is at least one related key for which the characteristic is followed through the key initialization.

As we have shown in the previous sections, the fraction of keys that is susceptible to the distinguisher is rather high. This results in the probability of hitting such a key by chance quite high, especially if we re-key frequently.

5 Conclusion

We discussed a full round related-key differential characteristic that can be applied to the mode. Using this characteristic we show how to create a tag forgery for TinyJAMBU-192 and TinyJAMBU-256 with a complexity of respectively 2^{32} and 2^{42} time and data using

respectively 2^{10} and 2^{12} related keys. We looked at the keyed permutation as a primitive and showed how to create a related-key distinguisher for the permutation with a data and time complexity of 2^{-17} for the 128-bit key variant. We also show how to recover the key using this distinguisher in 2^{24} data and time for the 128-bit variant.

One important note to make is that TinyJAMBU-128 is the main contribution of the TinyJAMBU submission to the NIST lightweight competition. The forgery discussed in the paper does not apply to this variant. Nevertheless, as we have shown, the keyed permutation used in TinyJAMBU-128 is easily distinguished and can therefore be considered weak in the related-key model. We also note that there exist full-round slide attack [SST+22] on the permutation in the single-key model. The designers have also proposed a trivial related-key slide attack against the permutation. Combining these weaknesses with the fact that (as we have shown) the other variants of the authenticated encryption scheme are broken, the 128-bit version should be used with care.

These attacks were mainly possible due to the fact that the key schedule of the primitives are cyclic. The protection offered against related key attacks by doing the key initialization with a constant state were not enough to protect against these attacks. To circumvent this attack the number of rounds in $\mathcal{P}^a$ and $\mathcal{P}^b$ should almost be doubled. Another easy fix for the problem would be to employ some sort of a key schedule to the permutation or to add a few additional NAND gates. We notified the designers of TinyJAMBU about the attacks which they verified. The designers do not intend to update the design to protect against related-key attacks. If related-key security is needed (which is outside the security model for TinyJAMBU), the designers suggested to use $\mathcal{P}^a$ to process the key after the initialization in the same fashion as the associated data is processed. Thus, for the 256-bit version, eight extra calls to $\mathcal{P}^a$ are made, and, in the 192-bit version, six extra calls to the permutation are made, where each of the calls processes 32 bits of the key.

Acknowledgments

The second author was supported in part by the Center for Cyber, Law, and Policy in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office and by the Israeli Science Foundation through grants No. 880/18 and 3380/19. The third author was supported in part by the IDIT - PhD Fellowship.

References

[ADG+22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In *USENIX Security Symposium*, pages 3291–3308. USENIX Association, 2022.

[BDPA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology – Proceedings of EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

[SSS+20] Dhiman Saha, Yu Sasaki, Danping Shi, Ferdinand Sibleyras, Siwei Sun, and Yingjie Zhang. On the security margin of TinyJAMBU with refined differential

and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.*, 2020(3):152–174, 2020.

[SST+22] Ferdinand Sibleyras, Yu Sasaki, Yosuke Todo, Akinori Hosoyamada, and Kan Yasuda. Birthday-bound slide attacks on TinyJAMBU's keyed permutation for all key sizes. In *Fifth Lightweight Cryptography Workshop*, 2022.

[Tec17] N.I.N.I.S. Technology. *Report on Lightweight Cryptography: NiSTIR 8114*. CreateSpace Independent Publishing Platform, 2017.

[WH19] Hongjun Wu and Tao Huang. TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms: Submission to NIST LwC, 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/tinyjambu-spec-final.pdf.

[WH21] Hongjun Wu and Tao Huang. TinyJAMBU : A family of lightweight authenticated encryption algorithms (version 2), 2021. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/tinyjambu-spec-final.pdf.