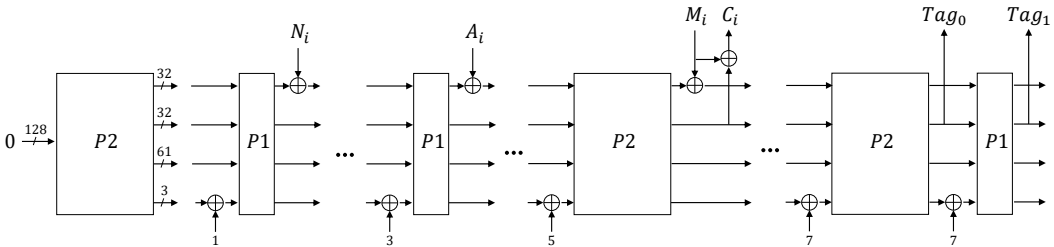
Birthday-Bound Slide Attacks on TinyJAMBU's Keyed-Permutations for All Key Sizes

Ferdinand Sibleyras Yu Sasaki Yosuke Todo
Akinori Hosoyamada Kan Yasuda

NTT Social Informatics Laboratories, Tokyo, Japan

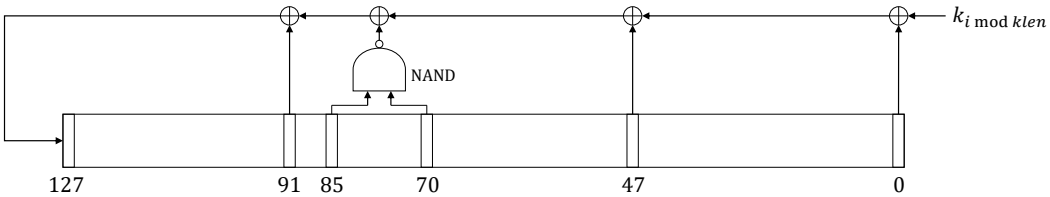
May 10, 2022

TinyJAMBU AEAD mode



Finalist of NIST LWC. Small state (128bits).
Sponge-like AEAD mode but with a keyed-permutation (blockcipher?).
Proven secure given **ideal underlying permutation**.

Internal keyed Permutation



$$a_{i+128} = k_i \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91}$$

$P1$ and $P2$ only differs by the number of rounds.

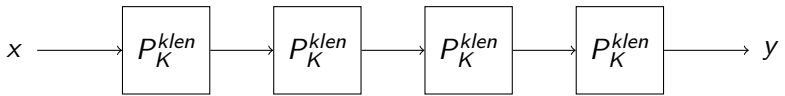
Summary

Key size	Setting	Data	Time	Memory
128	KP	2^{65}	2^{65}	2^{64}
	KP	2^{64}	2^{65}	2^{64}
	ACP	$2^{72.5}$	$2^{72.5}$	negl.
192	ACP	2^{65}	2^{66}	2^{65}
	CP	2^{67}	2^{69}	2^{66}
256	ACP	$2^{67.5}$	$2^{69.5}$	$2^{67.5}$

Requires at least 2^{64} data $> 2^{50}$ data limit.
Attack not applicable to the AEAD mode.

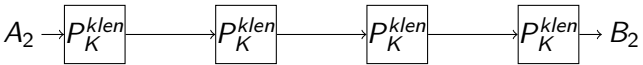
Security of the underlying permutation doesn't increase much with the key length.

Structure

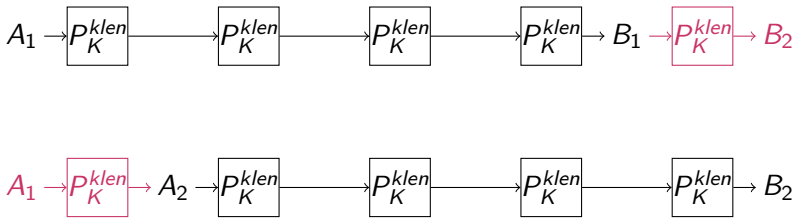


No key schedule implies the same permutation is applied every *klen*.
As a blockcipher, it makes it subject to **slide attacks**.

Slide Attacks



Slide Attacks



Slid Pairs

$$A_2 = P_K^{klen}(A_1) \iff B_2 = P_K^{klen}(B_1)$$

In such case, (A_1, B_1) and (A_2, B_2) form a slid pair.

Slide Attacks (2)

Attack Procedure

Collect many plaintext/ciphertext pairs.
Filter the pairs to find a slid pair.
Recover the key from the slid pair.

Slide Attacks (2)

Attack Procedure

Collect many plaintext/ciphertext pairs.
Filter the pairs to find a slid pair.
Recover the key from the slid pair.

This is a full state collision and so happens at the birthday bound.
For TinyJAMBU, the state is 128 bits so about 2^{64} **data** is required.

Filter for TinyJAMBU-128

Input: $A_1 \to \{a_0, a_1, \ldots, a_{127}\}$ Output: $A_2 \to \{a_{128}, a_{129}, \ldots, a_{255}\}$

For $0 \leq i \leq 36$:

$$\begin{aligned} k_i &= a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91} \\ &= b_{i+128} \oplus b_i \oplus b_{i+47} \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus b_{i+91} \end{aligned}$$

Filter for (A_1, B_1) and (A_2, B_2):

$$\begin{aligned} G_1[i] &= a_i \oplus b_i \oplus a_{i+47} \oplus b_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus a_{i+91} \oplus b_{i+91} \\ G_2[i] &= a_{i+128} \oplus b_{i+128} \end{aligned}$$

Filter for TinyJAMBU-128

Input: $A_1 \to \{a_0, a_1, \ldots, a_{127}\}$ Output: $A_2 \to \{a_{128}, a_{129}, \ldots, a_{255}\}$

For $37 \leq i \leq 42$:

$$\begin{aligned} k_i &= a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91} \\ &= b_{i+128} \oplus b_i \oplus b_{i+47} \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus b_{i+91} \end{aligned}$$

Filter for (A_1, B_1) and (A_2, B_2):

$$\begin{aligned} G_1[i] =& a_i \oplus b_i \oplus a_{i+47} \oplus b_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus (\neg(b_{i+70} \wedge b_{i+85})) \\ G_2[i] =& a_{i+128} \oplus b_{i+128} \oplus a_{i+91} \oplus b_{i+91} \end{aligned}$$

Filter for TinyJAMBU-128

Input: $A_1 \rightarrow \{a_0, a_1, \ldots, a_{127}\}$ Output: $A_2 \rightarrow \{a_{128}, a_{129}, \ldots, a_{255}\}$

For $43 \leq i \leq 57$:

$$\begin{aligned} k_i &= a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91} \\ &= b_{i+128} \oplus b_i \oplus b_{i+47} \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus b_{i+91} \end{aligned}$$

Filter for (A_1, B_1) and (A_2, B_2):

$$\begin{aligned} G_1[i] &= N.A. \\ G_2[i] &= N.A. \end{aligned}$$

Filter for TinyJAMBU-128

Input: $A_1 \rightarrow \{a_0, a_1, \ldots, a_{127}\}$ Output: $A_2 \rightarrow \{a_{128}, a_{129}, \ldots, a_{255}\}$

For $58 \leq i \leq 80$:

$$\begin{aligned} k_i &= a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91} \\ &= b_{i+128} \oplus b_i \oplus b_{i+47} \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus b_{i+91} \end{aligned}$$

Filter for (A_1, B_1) and (A_2, B_2):

$$\begin{aligned} G_1[i] =& a_i \oplus b_i \oplus a_{i+47} \oplus b_{i+47} \\ G_2[i] =& a_{i+128} \oplus b_{i+128} \oplus a_{i+91} \oplus b_{i+91} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \\ & \oplus (\neg(b_{i+70} \wedge b_{i+85})) \end{aligned}$$

Filter for TinyJAMBU-128

Input: $A_1 \rightarrow \{a_0, a_1, \ldots, a_{127}\}$ Output: $A_2 \rightarrow \{a_{128}, a_{129}, \ldots, a_{255}\}$

For $81 \leq i \leq 127$:

$$\begin{aligned} k_i &= a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91} \\ &= b_{i+128} \oplus b_i \oplus b_{i+47} \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus b_{i+91} \end{aligned}$$

Filter for (A_1, B_1) and (A_2, B_2):

$$\begin{aligned} G_1[i] =& a_i \oplus b_i \\ G_2[i] =& a_{i+128} \oplus b_{i+128} \oplus a_{i+91} \oplus b_{i+91} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \\ & \oplus (\neg(b_{i+70} \wedge b_{i+85})) \oplus a_{i+47} \oplus b_{i+47} \end{aligned}$$

Attack on TinyJAMBU-128

- Gather 2^{64} plaintext/ciphertext pairs (A, B).
- Find a collision $G_1(A_1, B_1) = G_2(A_2, B_2)$; 113-bit filter.
- Compute the key from A_1 and A_2.

$$k_i = a_{i+128} \oplus a_i \oplus a_{i+47} \oplus (\neg(a_{i+70} \wedge a_{i+85})) \oplus a_{i+91}$$

- If wrong key, go to next collision.

Complexities

2^{64} Data (Known Plaintext).
2^{65} Time (Compute G_1 and G_2).
2^{64} Memory.

Filter for TinyJAMBU-192

Consider the 113 bits of key k_i whose non-linear term is either in the input or output.

	128-bit Input					64-bit Unobservable				128-bit Output		
	a_0	a_1	a_2	...	a_{127}	a_{128}	...	a_{190}	a_{191}	a_{192}	...	a_{319}
k_0	1	0	0	...	0	1	...	0	0	0	...	0
k_1	0	1	0	...	0	0	...	0	0	0	...	0
k_2	0	0	1	...	0	0	...	0	0	0	...	0
⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋱	⋮	⋮	⋮	⋱	⋮
k_{190}	0	0	0	...	0	0	...	1	0	0	...	0
k_{191}	0	0	0	...	0	0	...	0	1	0	...	1

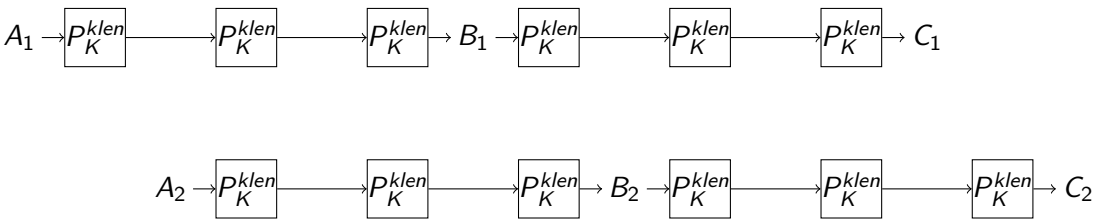
Filter for TinyJAMBU-192

128-bit Input				64-bit Unobservable					128-bit Output		
a_0	a_1	...	a_{127}	a_{128}	a_{129}	...	a_{190}	a_{191}	a_{192}	...	a_{319}
$E^1_{64\times128}$				$I_{64\times64}$					$S^1_{64\times128}$		
$E^2_{49\times128}$				$0_{49\times49}$					$S^2_{49\times128}$		

Perform Gaussian Elimination and Deduce a $113 - 64 = 49$-bit filter.

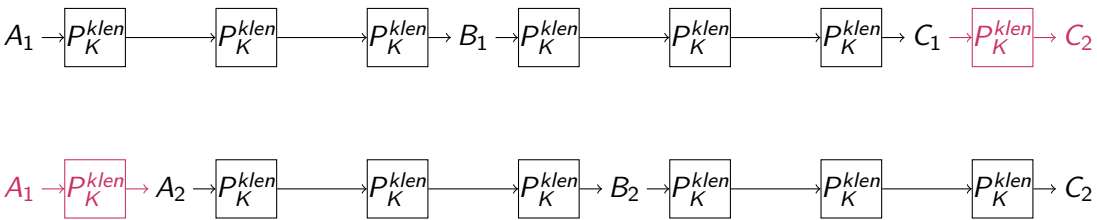
Multiply the Filter

Chaining queries can help multiply the strength of our filter.



Multiply the Filter

Chaining queries can help multiply the strength of our filter.



Related Slid Pairs

$$A_2 = P_K^{klen}(A_1) \iff B_2 = P_K^{klen}(B_1) \iff C_2 = P_K^{klen}(C_1)$$

In such case, (A_1, B_1) / (A_2, B_2) and (B_1, C_1) / (B_2, C_2) form two slid pairs.

Key Recovery TinyJAMBU-192

128-bit Input				64-bit Unobservable					128-bit Output		
a_0	a_1	...	a_{127}	a_{128}	a_{129}	...	a_{190}	a_{191}	a_{192}	...	a_{319}
$E^1_{64\times128}$				$I_{64\times64}$					$S^1_{64\times128}$		
$E^2_{49\times128}$				$0_{49\times49}$					$S^2_{49\times128}$		

- Guess the bit of key corresponding to the first line and deduce a_{128}.
- Add to the binary matrix the new key bit whose non-linear term can now be computed (with a_{128}).

Key Recovery TinyJAMBU-192

128+1-bit Input					63-bit Unobservable				128-bit Output			
a_0	a_1	...	a_{127}	a_{128}	a_{129}	...	a_{190}	a_{191}	a_{192}	...	a_{319}	
$E^1_{63\times128}$					$I_{63\times63}$				$S^1_{63\times128}$			
$E^2_{51\times128}$					$0_{51\times51}$				$S^2_{51\times128}$			

- Guess the bit of key corresponding to the first line and deduce a_{128}.
- Add to the binary matrix the new key bit whose non-linear term can now be computed (with a_{128}).
- Use the additionnal filter to verify the guess.

Attack on TinyJAMBU-192

- Gather 2^{64} plaintext/ciphertext pairs (A, B) and (B, C).
- Find a collision $G_1(A_1, B_1) = G_2(A_2, B_2)$ and $G_1(B_1, C_1) = G_2(B_2, C_2)$; $49 * 2 = 98$-bit filter.
- Compute the key from the slid pairs.
- If wrong key, go to next collision.

Complexities

2^{65} Data (Adaptative Chosen Plaintext).
2^{66} Time.
2^{65} Memory.

Filter for TinyJAMBU-256

Guess k_0 and k_{15} to compute a_{128} and a_{143}.
We have the **1-bit filter**:

$$\begin{aligned} k_{21} \oplus k_{58} \oplus k_{186} \oplus k_{233} = & a_{21} \oplus a_{68} \oplus (\neg(a_{91} \wedge a_{106})) \oplus a_{112} \oplus a_{58} \oplus a_{105} \\ & \oplus (\neg(a_{128} \wedge a_{143})) \oplus a_{314} \oplus (\neg(a_{256} \wedge a_{271})) \oplus a_{277} \\ & \oplus a_{361} \oplus a_{280} \oplus (\neg(a_{303} \wedge a_{318})) \oplus a_{324}. \end{aligned}$$

Filter for TinyJAMBU-256

Guess k_0 and k_{15} to compute a_{128} and a_{143}.
We have the **1-bit filter**:

$$\begin{aligned} k_{21} \oplus k_{58} \oplus k_{186} \oplus k_{233} = & a_{21} \oplus a_{68} \oplus (\neg(a_{91} \wedge a_{106})) \oplus a_{112} \oplus a_{58} \oplus a_{105} \\ & \oplus (\neg(a_{128} \wedge a_{143})) \oplus a_{314} \oplus (\neg(a_{256} \wedge a_{271})) \oplus a_{277} \\ & \oplus a_{361} \oplus a_{280} \oplus (\neg(a_{303} \wedge a_{318})) \oplus a_{324}. \end{aligned}$$

Increase the filter to **128 bits** with chained queries.

Key Recovery TinyJAMBU-256

Recover some bits with tricks and then fall back to previous key recovery algorithm.

Key Recovery TinyJAMBU-256

Recover some bits with tricks and then fall back to previous key recovery algorithm.

Computing non-linear terms

Even if a_{128} is unobservable, we can compute $a_{113} \wedge a_{128}$ if $a_{113} = 0$.
Among many slid pairs, one will surely have the required bits to 0 allowing us to compute some key bits.

Attack on TinyJAMBU-256

- Query 2^{64} chains of 128 plaintext/ciphertext.
- Guess k_0 and k_{15} to filter the chains of slid pairs.
- Compute the key from the slid pairs.
- If wrong key, guess other k_0 and k_{15}.

Complexities

$2^{67.5}$ Data (Adaptative Chosen Plaintext).
$2^{69.5}$ Time.
$2^{67.5}$ Memory.

Summary

Key size	Setting	Data	Time	Memory
128	KP	2^{65}	2^{65}	2^{64}
	KP	2^{64}	2^{65}	2^{64}
	ACP	$2^{72.5}$	$2^{72.5}$	negl.
192	ACP	2^{65}	2^{66}	2^{65}
	CP	2^{67}	2^{69}	2^{66}
256	ACP	$2^{67.5}$	$2^{69.5}$	$2^{67.5}$

Requires at least 2^{64} data $> 2^{50}$ data limit.
Attack not applicable to the AEAD mode.

Security of the underlying permutation doesn't increase much with the key length.