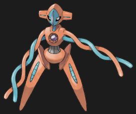


Deoxys - Joltik

Jérémy Jean - Ivica Nikolić - **Thomas Peyrin**

NTU - Singapore

DIAC 2015

Singapore - September 28, 2015

http://www1.spms.ntu.edu.sg/~syllab/Deoxys
http://www1.spms.ntu.edu.sg/~syllab/Joltik

NANYANG TECHNOLOGICAL UNIVERSITY

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - ▷ TWEAKEY and the STK construction
   - ▷ Deoxys-BC
   - ▷ Joltik-BC
3. The operating mode(s)
   - ▷ Nonce-respecting mode: Deoxys$\neq$ and Joltik$\neq$
   - ▷ Nonce-misuse resistant mode: Deoxys$=$ and Joltik$=$
   - ▷ Security claims
   - ▷ Other features
4. Conclusion

Structure and update

For 2nd round, we have changed one of the operating modes

	Mode		Internal primitive	
	$\neq$ mode (TAE-like)	$=$ mode (COPA-like)	Deoxys-BC	Joltik-BC
Deoxys≠	✓		✓	
Deoxys=		✓	✓	
Joltik≠	✓			✓
Joltik=		✓		✓

Structure and update

For 2nd round, we have changed one of the operating modes

	Mode		Internal primitive	
	$\neq$ mode (TAE-like)	$=$ mode (SCT)	Deoxys-BC	Joltik-BC
Deoxys$\neq$	✓		✓	
Deoxys$=$		✓	✓	
Joltik$\neq$	✓			✓
Joltik$=$		✓		✓

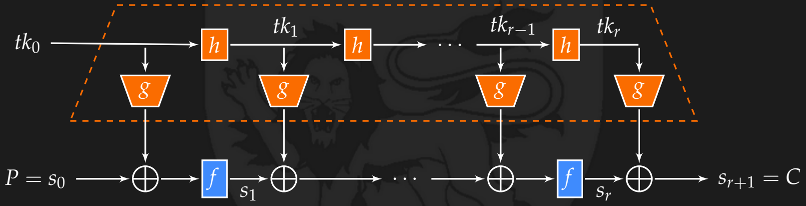
Outline

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - ▷ TWEAKEY and the STK construction
   - ▷ Deoxys-BC
   - ▷ Joltik-BC
3. The operating mode(s)
   - ▷ Nonce-respecting mode: Deoxys≠ and Joltik≠
   - ▷ Nonce-misuse resistant mode: Deoxys= and Joltik=
   - ▷ Security claims
   - ▷ Other features
4. Conclusion

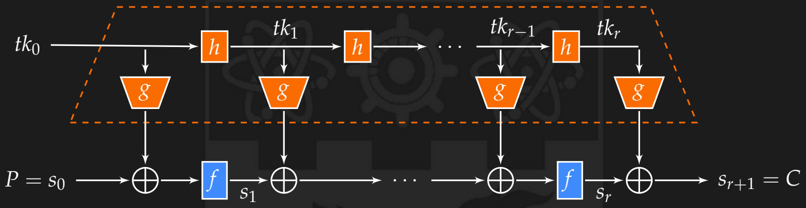
The TWEAKEY framework

The TWEAKEY framework rationale [ASIACRYPT'14]:
tweak and key should be treated the same way ⟶ tweakey



TWEAKEY generalizes the class of key-alternating ciphers

The TWEAKEY framework

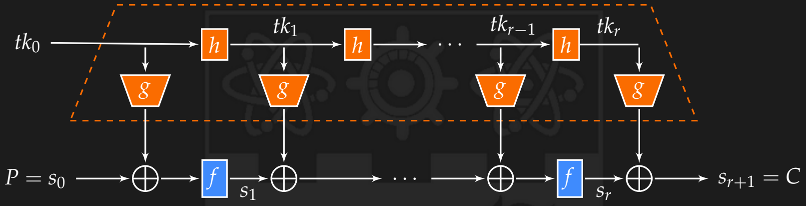


The TWEAKEY framework

The regular key schedule is replaced by a TWEAKEY schedule that generates subtweakeys. An n-bit key n-bit tweak TBC has $2n$-bit tweakey and g compresses $2n$ to n bits:

- such a primitive would be a TK-2 primitive (TWEAKEY of order 2).
- the same primitive can be seen as a $2n$-bit key cipher with no tweak (or $1.5n$-bit key and $0.5n$-bit tweak, etc).

The TWEAKEY framework



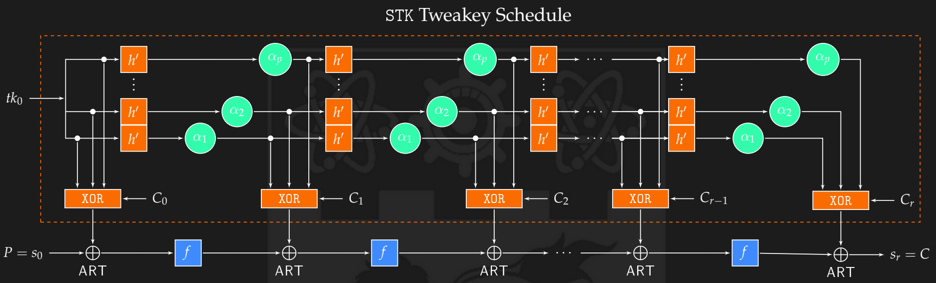
The main issue:

adding more tweakey state makes the security drop, or renders security hard to study, even for automated tools

Idea:

separate the tweakey material in several words, design a secure tweakey schedule for one word and then superpose them in a secure way

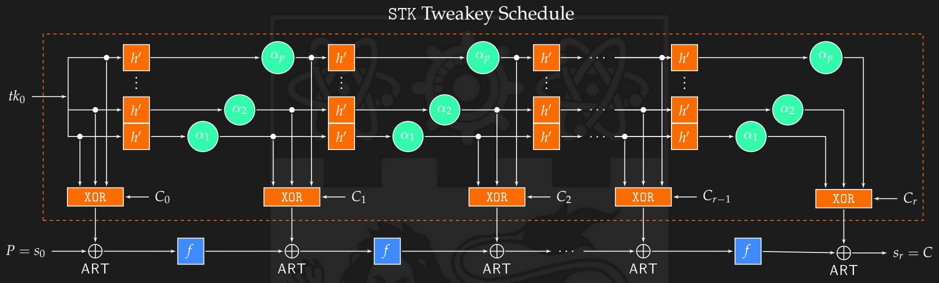
The STK construction (Superposition-TWEAKEY)



From the TWEAKEY framework to the STK construction:

- the tweakey state update function h consists in the same subfunction h' applied to each tweakey word
- the subtweakey extraction function g consists in XORing all the words together
  - reduce the implementation overhead
  - reduce the area footprint by reusing code
  - simplify the security analysis

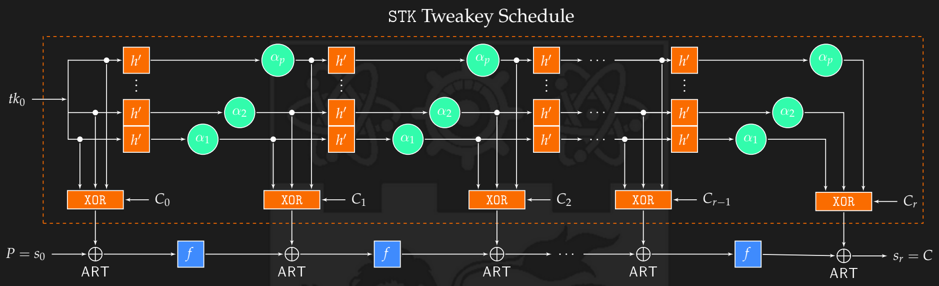
The STK construction (Superposition-TWEAKEY)



From the TWEAKEY framework to the STK construction:

- **problem**: strong interaction between the parallel branches of tweakey state
- **solution**: differentiate the parallel branches by simply using distinct multiplications in a small field

The STK construction (Superposition-TWEAKEY)

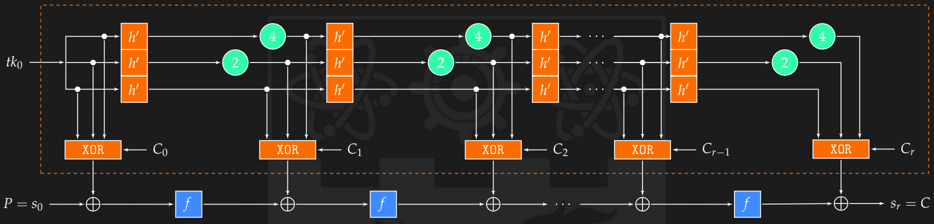


In details:

- assume the n-bit internal state of the cipher is divided into p nibbles of c bits: we divide the tweakey material into n-bit words, and then c-bit nibbles
- h' will simply be a permutation of the nibbles positions
- each nibble of the k-th tweakey word is multiplied by a value $\alpha_k \in GF(2^c)$

STK with a 4 × 4 internal state matrix

STK construction (for TK-3) with a 4 × 4 internal state matrix



- multiplication factors are 1, 2 and 4 in $GF(2^c)$
- h' is a simple permutation of the 16 nibbles:

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix} \xrightarrow{h'} \begin{pmatrix} 1 & 5 & 9 & 13 \\ 6 & 10 & 14 & 2 \\ 11 & 15 & 3 & 7 \\ 12 & 0 & 4 & 8 \end{pmatrix}$$

The STK construction: rationale

Design choices

- multiplication in $GF(2^c)$ controls the number of cancellations in g, when the subtweakeys are XORed to the internal state
- rely on a linear code to bound the number of cancellations

Implementation

- very simple transformations: linear and lightweight
- multiplications constants chosen as $1, 2, 4, \ldots$ for efficiency

Security analysis

A security analysis is now possible with STK:

- when considering one tweakey word, we ensure that function h' is itself a good tweakey schedule
- when considering several tweakey words, we reuse existing tools searching for good differential paths: for these tools it is easy to add the cancellation bound

Security of the STK construction

Related-key related-tweak attacks (4×4 AES-like design)

We prove that no good related-key related-tweak attacks differential path exist (even boomerang), with a computer-aided search tool.

rounds	active SBoxes	upper bound on probability	method used
1-2	0	2^{0}	trivial
3	1	$2^{-6}/2^{-2}$	Matsui's
4	5	$2^{-30}/2^{-8}$	Matsui's
5	9	$2^{-54}/2^{-18}$	Matsui's
6	12	$2^{-72}/2^{-24}$	Matsui's
8	≥ 17	$2^{-108}/2^{-34}$	ex. split (4R+4R)
10	≥ 22	$2^{-132}/2^{-44}$	ex. split (5R+5R)

Security of the STK construction

Related-key related-tweak attacks (4×4 AES-like design)

We prove that no good related-key related-tweak attacks differential path exist (even boomerang), with a computer-aided search tool.

rounds	active SBoxes	upper bound on probability	method used
8	≥ 17	$2^{-108}/2^{-34}$	ex. split (4R+4R)
10	≥ 22	$2^{-132}/2^{-44}$	ex. split (5R+5R)

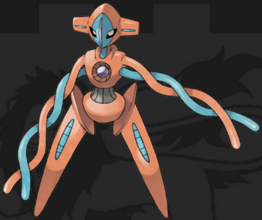
Meet-in-the-middle attacks

Using a computer-aided search tool, we checked that Demirci-Selćuk MitM attack and its improvements cannot apply, even when using the tweak input as extra leverage.

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

Deoxys-BC



Fast and software-oriented candidate to CAESAR

The Deoxys-BC tweakable block cipher

128-bit tweakable block ciphers

Two members: Deoxys-BC-256 and Deoxys-BC-384:

- 256 bits for TK-2: $|key| + |tweak| = 256$ (2 tweakey words)
- 384 bits for TK-3: $|key| + |tweak| = 384$ (3 tweakey words)
- Both are instances of the STK construction

The round function:

- The round function is **exactly the AES round function** (AES-NI)
- Deoxys-BC-256 has 14 rounds (TK-2)
- Deoxys-BC-384 has 16 rounds (TK-3)

The TWEAKEY schedule:

- h' is a simple permutation of the 16 nibbles
- Multiplications factors are: 1, 2 and 4 in the AES field
- Constant additions to break symmetries (RCON from AES KS)

Implementations of Deoxys-BC

Software implementations of Deoxys-BC

- AES-NI implementation (on Sandy Bridge):
  - **1.13 c/B for Deoxys-BC-256**
  - **1.32 c/B for Deoxys-BC-384**
  - **Deoxys in the top 10% of AES-NI implementations on SUPERCOP**

Hardware implementations of Deoxys-BC

- ASIC:
  - 2860 GE for Deoxys-BC-256
  - 3575 GE for Deoxys-BC-384
- FPGA:
  - not yet ... but likely to be close to AES

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

Joltik-BC

Lightweight and hardware-oriented candidate to CAESAR

The Joltik-BC tweakable block cipher

64-bit tweakable block ciphers

Two members: Joltik-BC-128 and Joltik-BC-192:

- 128 bits for TK-2: $|key| + |tweak| = 128$ (2 tweakey words)
- 192 bits for TK-3: $|key| + |tweak| = 192$ (3 tweakey words)
- Both are instances of the STK construction

AES-based round function:

- Involutive MDS matrix $\Longrightarrow$ low decryption overhead
- S-Box from the Piccolo block cipher (compact in hardware)
- Joltik-BC-128 has 24 rounds (TK-2)
- Joltik-BC-192 has 32 rounds (TK-3)

The TWEAKEY schedule:

- h' is a simple permutation of the 16 nibbles
- Multiplications factors are: 1, 2 and 4 in $GF(16)/0x13$
- Constant additions to break symmetries (from LED cipher)

Implementations of Joltik-BC

Software implementations of Joltik-BC

- vperm implementation (SSSE3 and avx2): about the same (expected) speed as LED-64
- Projection for bitslice for Joltik-BC-128: about 9 c/B for long messages

Hardware implementations of Joltik-BC

- ASIC: (LED-128: about 1300 GE)
  - **1442 GE for Joltik-BC-128**
  - **1805 GE for Joltik-BC-192**
- FPGA (ATHENa website):
  - **about 500/600 slices for Joltik-BC-128**
  - so far the smallest candidate on ATHENa website

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

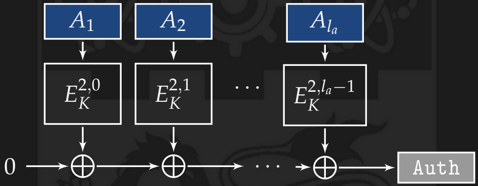
Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys$\neq$ and Joltik$\neq$
   - Nonce-misuse resistant mode: Deoxys$=$ and Joltik$=$
   - Security claims
   - Other features
4. Conclusion

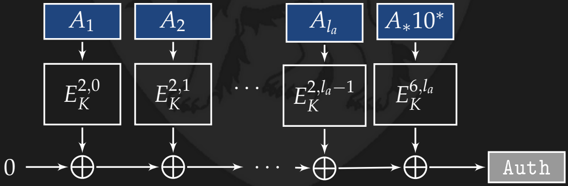
Nonce-respecting mode: Deoxys≠ and Joltik≠

Deoxys≠ and Joltik≠ are similar to TAE or OCB3

For associated data authentication (full block):



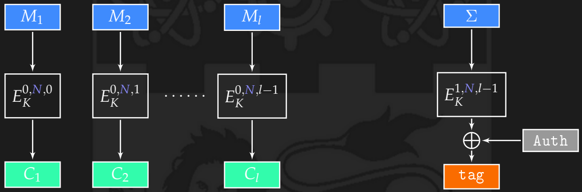
For associated data authentication (partial block):



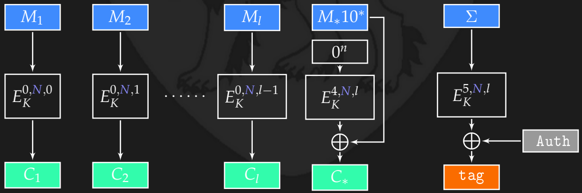
Nonce-respecting mode: Deoxys≠ and Joltik≠

Deoxys≠ and Joltik≠ are similar to TAE or OCB3

For plaintext (full block):



For plaintext (partial block):



Analysis of the ≠ mode

As the nonce is never reused, it is ensured that every call to the TBC during the encryption will have distinct tweak input value

We can directly reuse the TAE or OCB3 security proofs:

- but ensuring full security instead of birthday bound
- the proofs are simpler (see ΘCB3 and OCB3 proofs)
- no long initialization required: fast for short inputs

Universal hash based tweakable block ciphers won't provide full n-bit security (or with bad efficiency), due to the possibility of collisions between the inputs/outputs of the internal block cipher

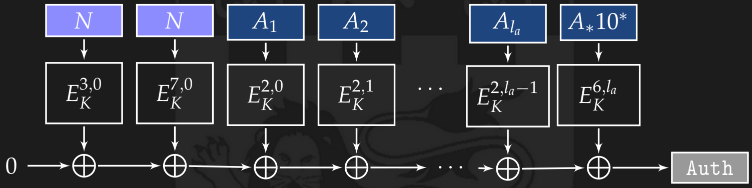
Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

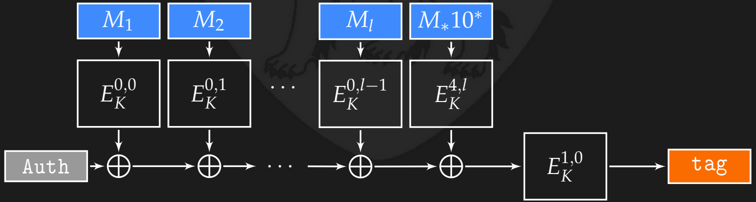
Nonce-misuse resistant mode: Deoxys= and Joltik=

Deoxys= and Joltik= are based on new SCT mode (joint work with Y. Seurin)

For associated data authentication (partial block):



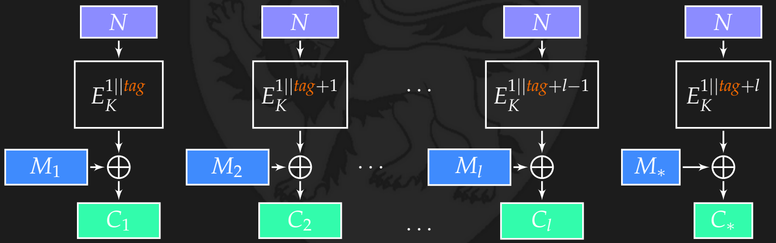
For plaintext authentication (partial block):



Nonce-misuse resistant mode: Deoxys= and Joltik=

Deoxys= and Joltik= are based on new SCT mode (joint work with Y. Seurin)

For plaintext encryption (partial block):



Analysis of the = mode (SCT mode)

When the nonce is not reused, we ensure that every call to the TBC during encryption will have distinct input values

When the nonce is reused, the attacker has to either attack the PMAC part or to collide on the tweak input during encryption

Nonce-misuse resistance in the **strong MRAE sense** (not the weaker online misuse-resistance notion)

SCT is the first AEAD mode that provides both:

- full n-bit security when the nonce is not reused
- some ($n/2$-bit) security when the nonce is reused

Conjecture: the security of the SCT mode is close to the full n-bit when the nonce is reused only a few times (which is exactly what will happen in the nonce-misuse scenario)

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

Security claims (in $\log_2$)

nonce-respecting user	Security (bits) $\neq$ mode	Security (bits) $=$ mode
Key recovery	k	k
Confidentiality	n	$n-1$
Integrity	n	$n-1$

nonce-misuse user	Security (bits) $\neq$ mode	Security (bits) $=$ mode
Key recovery	k	k
Confidentiality	none	$n/2$
Integrity	none	$n/2$

Security claims (in $\log_2$) - conjecture

nonce-respecting user	**Security (bits)** $\neq$ mode	**Security (bits)** $=$ mode
Key recovery	k	k
Confidentiality	n	$n-1$
Integrity	n	$n-1$

nonce-misuse user (proven/conj.)	**Security (bits)** $\neq$ mode	**Security (bits)** $=$ mode
Key recovery	k	k
Confidentiality	none	$n - log(m)$
Integrity	none	$n - log(m)$

Security claims (in $\log_2$) - Deoxys

nonce-respecting user	Security (bits) Deoxys≠	Deoxys=
Key recovery	k	k
Confidentiality	128	127
Integrity	128	127

nonce-misuse user (proven/conj.)	Security (bits) Deoxys≠	Deoxys=
Key recovery	k	k
Confidentiality	none	$64/\simeq 128$
Integrity	none	$64/\simeq 128$

Security claims (in $\log_2$) - Joltik

nonce-respecting user	Security (bits) Joltik≠	Joltik=
Key recovery	k	k
Confidentiality	64	63
Integrity	64	63

nonce-misuse user (proven/conj.)	Security (bits) Joltik≠	Joltik=
Key recovery	k	k
Confidentiality	none	$32/\simeq 64$
Integrity	none	$32/\simeq 64$

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys≠ and Joltik≠
   - Nonce-misuse resistant mode: Deoxys= and Joltik=
   - Security claims
   - Other features
4. Conclusion

Other features

Parallelization:

Both our modes are parallelizable

Small messages:

Both our modes are particularly efficient for small messages as almost no initialisation is required

- unlike for sponge-based (long init process), AES-GCM-like or OCB3-like candidates (precomputation tables)
- small messages is a typical use-case of hardware applications
- small messages is a typical use-case of software applications:
  "simple IMIX" is a weighted average simulating sizes of typical IP packages: 7 parts of 40B, 4 parts of 576B, 1 part of 1500B

Memory overhead:

Both our modes have little memory overhead (no precomp. tables)

Side channel resistance:

A part of the tweak inputs of the internal ciphers can be devoted to leakage resilient cryptography

Outline

1. Introduction
2. The Joltik-BC and Deoxys-BC tweakable BC
   - TWEAKEY and the STK construction
   - Deoxys-BC
   - Joltik-BC
3. The operating mode(s)
   - Nonce-respecting mode: Deoxys$\neq$ and Joltik$\neq$
   - Nonce-misuse resistant mode: Deoxys$=$ and Joltik$=$
   - Security claims
   - Other features
4. Conclusion

Summary

For 2nd round, we have changed one of the operating mode

- we now have two simple parallelizable modes, both providing full n-bit security - not birthday security ! - in nonce-respecting scenario, and fast for short messages (no initialization needed):
  - $\neq$: one-pass online mode
  - $=$ or SCT: two-pass mode that also provides MRAE security up to birthday bound (full security when the nonce is not repeated too often)
- these two modes are instantiated with two types of TBC:
  - Deoxys: 128-bit block - very fast in software
  - Joltik: 64-bit block - very small in hardware

Thank you !