Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption

Philipp Jovanovic

SPEED-B

October 20, 2016

ACKs

Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption

Robert Granger[1], Philipp Jovanovic[2], Bart Mennink[3], and Samuel Neves[4]

[1] Laboratory for Cryptologic Algorithms, École polytechnique fédérale de Lausanne, Switzerland,
robert.granger@epfl.ch
[2] Decentralized and Distributed Systems Lab, École polytechnique fédérale de Lausanne, Switzerland,
philipp.jovanovic@epfl.ch
[3] Dept. Electrical Engineering, ESAT/COSIC, KU Leuven, and iMinds, Belgium,
bart.mennink@esat.kuleuven.be
[4] CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal,
sneves@dei.uc.pt

(published at Eurocrypt'16)

https://eprint.iacr.org/2015/999

Outline

Tweakable Blockciphers & Authenticated Encryption

Masked Even-Mansour

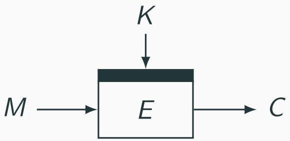
Applications to Authenticated Encryption

Implementation & Evaluation

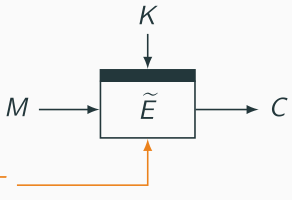
Conclusion

Tweakable Blockciphers & Authenticated Encryption

Tweakable Blockciphers (TBC)

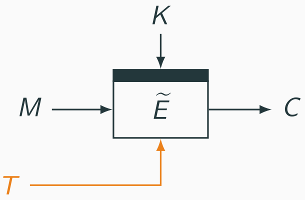


Tweakable Blockciphers (TBC)



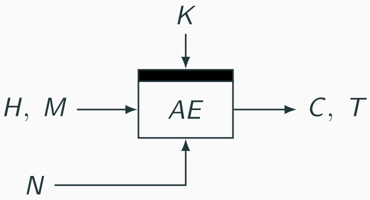
- Tweak T:
  - Public parameter
  - Adds flexibility to the cipher

Tweakable Blockciphers (TBC)



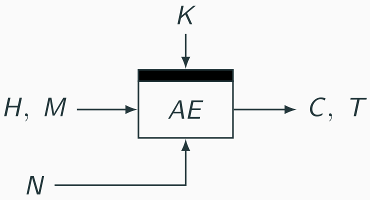
- Tweak T:
  - Public parameter
  - Adds flexibility to the cipher
- Different tweak $\Rightarrow$ different permutation

Authenticated Encryption (AEAD)



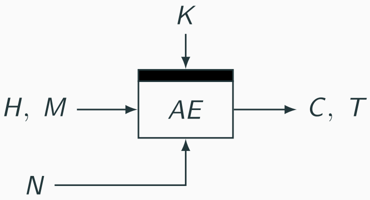
- Input: Key K, nonce N, associated data H, message M

Authenticated Encryption (AEAD)



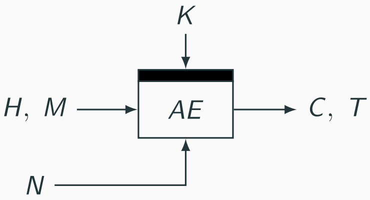
- Input: Key K, nonce N, associated data H, message M
- Output: Ciphertext C, authentication tag T

Authenticated Encryption (AEAD)



- Input: Key K, nonce N, associated data H, message M
- Output: Ciphertext C, authentication tag T
- Protects:
    - Confidentiality and integrity/authenticity of M
    - Integrity/authenticity of N and H

Authenticated Encryption (AEAD)



- Input: Key K, nonce N, associated data H, message M
- Output: Ciphertext C, authentication tag T
- Protects:
  - Confidentiality and integrity/authenticity of M
  - Integrity/authenticity of N and H

Nonce N randomizes the scheme (similar to a tweak)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, “first tweakable cipher”, tweak: spice)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, “first tweakable cipher”, tweak: spice)

2001: Mercy [Cro01] (disk encryption)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, “first tweakable cipher”, tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, “first tweakable cipher”, tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

2004: XE / XEX [Rog04] (OCB)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, “first tweakable cipher”, tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

2004: XE / XEX [Rog04] (OCB)

2007: Threefish [FLS+07] (in SHA-3 submission Skein)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, "first tweakable cipher", tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

2004: XE / XEX [Rog04] (OCB)

2007: Threefish [FLS+07] (in SHA-3 submission Skein)

2014: TWEAKEY [JNP14] (in CAESAR submissions Deoxys, Joltik, and KIASU)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, "first tweakable cipher", tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

2004: XE / XEX [Rog04] (OCB)

2007: Threefish [FLS+07] (in SHA-3 submission Skein)

2014: TWEAKEY [JNP14] (in CAESAR submissions Deoxys, Joltik, and KIASU)

TBC History

1998: Hasty Pudding Cipher [Sch98] (AES submission, "first tweakable cipher", tweak: spice)

2001: Mercy [Cro01] (disk encryption)

2002: Formalization of Tweakable Blockciphers [LRW02]

2004: XE / XEX [Rog04] (OCB)

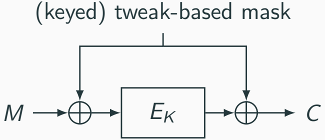
2007: Threefish [FLS+07] (in SHA-3 submission Skein)

2014: TWEAKEY [JNP14] (in CAESAR submissions Deoxys, Joltik, and KIASU)

What is the simplest way to build a tweakable blockcipher?

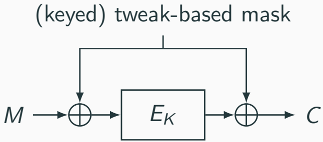
Masking-Based TBC

Blockcipher-Based

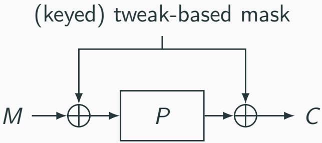


Masking-Based TBC

Blockcipher-Based

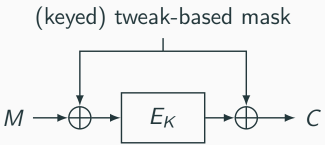


Permutation-Based



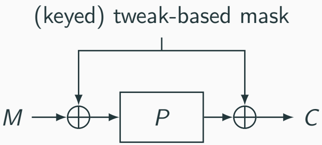
Masking-Based TBC

Blockcipher-Based



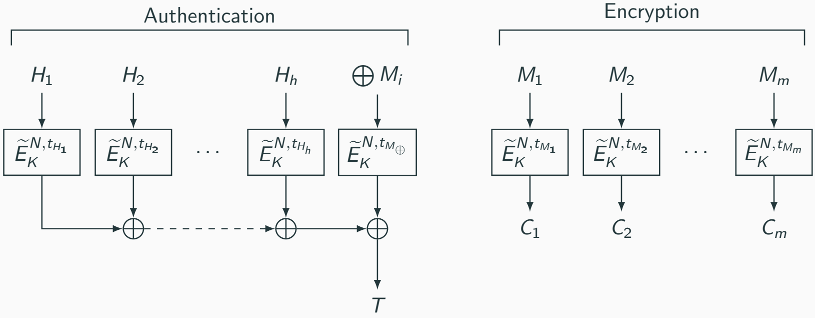
typically 128 bits

Permutation-Based



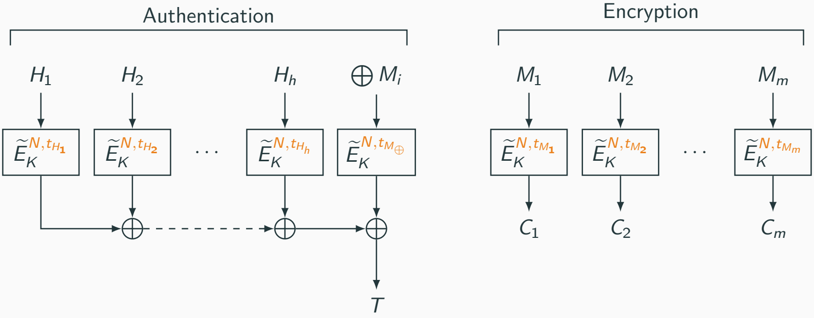
typically 256 – 1600 bits

TBC and AEAD



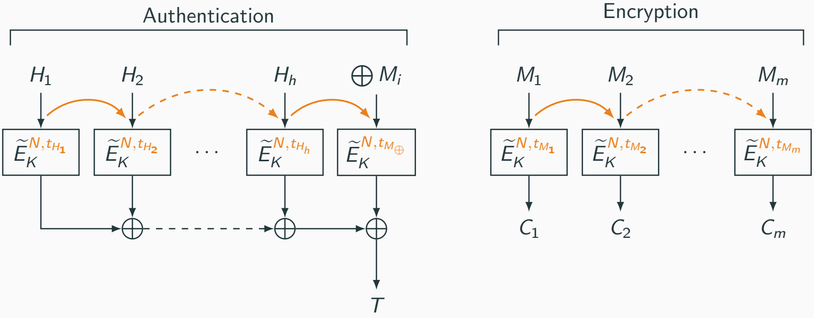
- OCBx: generalized OCB [RBBK01] [Rog04] [KR11]
- Based on tweakable blockcipher $\widetilde{E}$

TBC and AEAD



- OCBx: generalized OCB [RBBK01] [Rog04] [KR11]
- Based on tweakable blockcipher $\widetilde{E}$
- Tweak (N, t):
  - Unique for every evaluation
  - Different tweaks for different blocks

TBC and AEAD

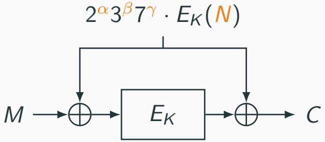


- OCBx: generalized OCB [RBBK01] [Rog04] [KR11]
- Based on tweakable blockcipher $\widetilde{E}$
- Tweak (N, t):
  - Unique for every evaluation
  - Different tweaks for different blocks
  - Change should be efficient

Powering-Up Masking

XEX

[Rog04]

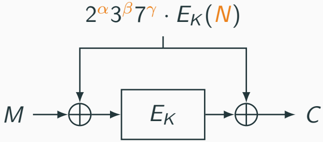


- Tweak (simplified): $(\alpha, \beta, \gamma, N)$

Powering-Up Masking

XEX

[Rog04]

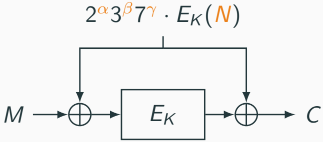


- Tweak (simplified): $(\alpha, \beta, \gamma, N)$
- Used in OCB2 and various CAESAR candidates

Powering-Up Masking

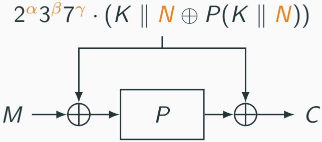
XEX

[Rog04]



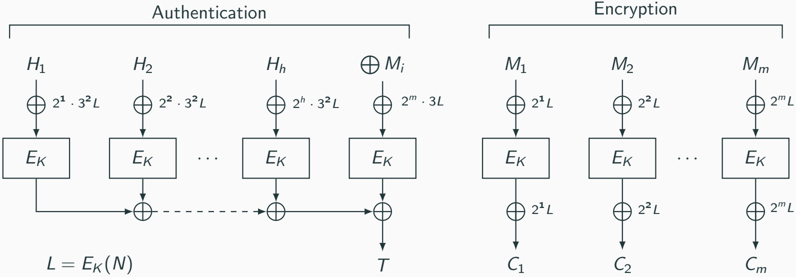
TEM

[STA+14]

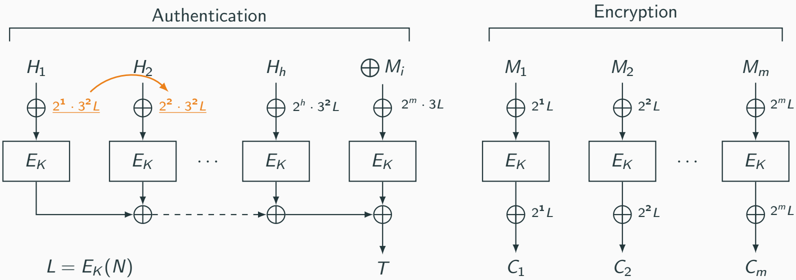


- Tweak (simplified): $(\alpha, \beta, \gamma, N)$
- Used in OCB2 and various CAESAR candidates
- Permutation-based variants: Minalpher and Prøst

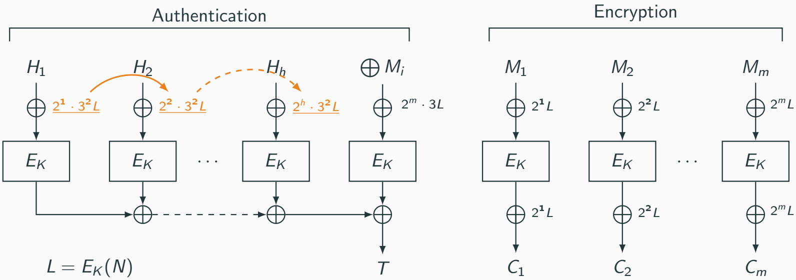
Powering-Up Masking in OCB2



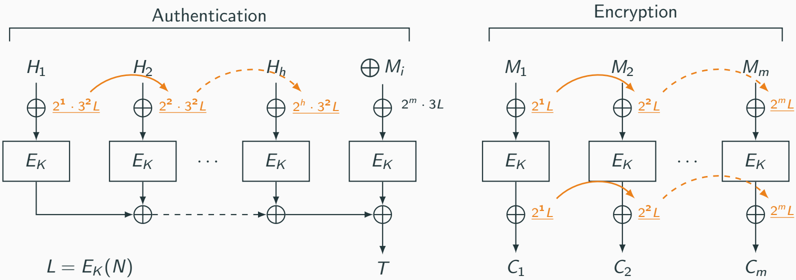
Powering-Up Masking in OCB2



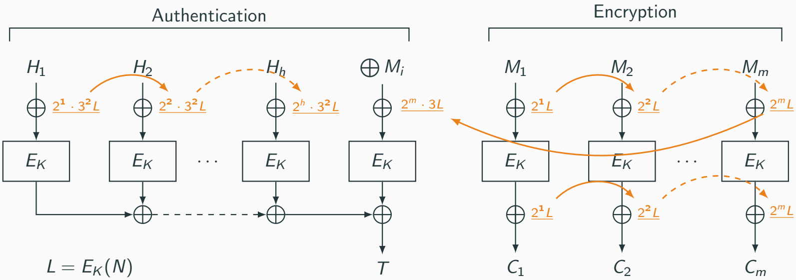
Powering-Up Masking in OCB2



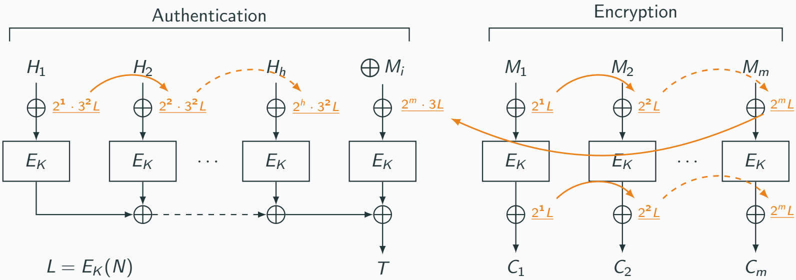
Powering-Up Masking in OCB2



Powering-Up Masking in OCB2



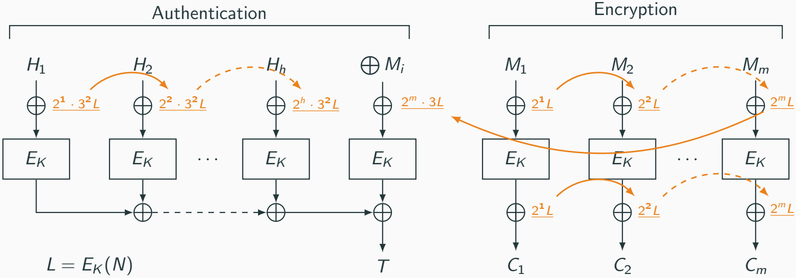
Powering-Up Masking in OCB2



- Update of mask: shift and conditional XOR

$$2^1 L = \begin{cases} L \ll 1 & \text{if } \mathtt{msb}(L) = 0 \\ (L \ll 1) \oplus 0^{120}10^4 1^3 & \text{else} \end{cases}$$

Powering-Up Masking in OCB2

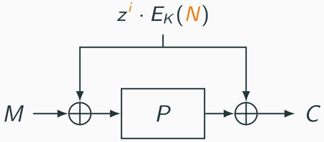


- Update of mask: shift and conditional XOR

$$2^1 L = \begin{cases} L \ll 1 & \text{if } \mathtt{msb}(\mathtt{L}) = 0 \\ (L \ll 1) \oplus 0^{120}10^41^3 & \text{else} \end{cases}$$

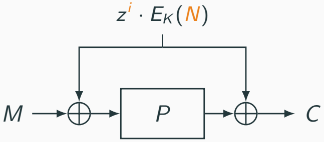
- Variable time computation
- Expensive on certain platforms

Word-based Powering-Up Masking



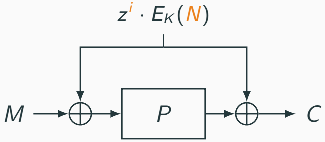
- By Chakraborty and Sarkar [CS06]
- Tweak: (i, N)

Word-based Powering-Up Masking



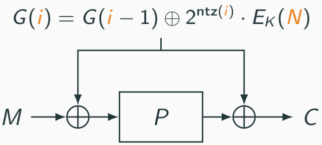
- By Chakraborty and Sarkar [CS06]
- Tweak: (i, N)
- Tower of fields:
  - $z^i \in \mathbb{F}_{2^w}[z]/g$ for $z \in \{0,1\}^w$ ...
  - ... instead of $x^i \in \mathbb{F}_2[x]/f$

Word-based Powering-Up Masking



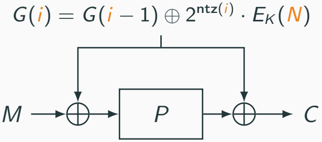
- By Chakraborty and Sarkar [CS06]
- Tweak: (i, N)
- Tower of fields:
  - $z^i \in \mathbb{F}_{2^w}[z]/g$ for $z \in \{0,1\}^w$ ...
  - ... instead of $x^i \in \mathbb{F}_2[x]/f$
- More software-friendly
- Similar drawbacks as regular variant

Gray Code Masking



- Used in OCB1 and OCB3
- Tweak: (i, N)
- Update:

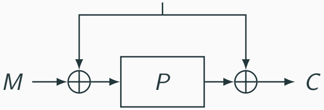
Gray Code Masking



- Used in OCB1 and OCB3
- Tweak: (i, N)
- Update:
    - Single XOR
    - $\log_2 i$ field doublings (precomputation possible)

Gray Code Masking

$$G(i) = G(i-1) \oplus 2^{\mathsf{ntz}(i)} \cdot E_K(N)$$



- Used in OCB1 and OCB3
- Tweak: (i, N)
- Update:
  - Single XOR
  - $\log_2 i$ field doublings (precomputation possible)
- More efficient than powering-up [KR11]

Masked Even-Mansour

High-Level Contributions

Masked Even-Mansour

- Improved masking for tweakable blockciphers
- Very efficient

High-Level Contributions

Masked Even-Mansour

- Improved masking for tweakable blockciphers
- Very efficient
- Simple to implement
- Constant time (by default)
- Relies on breakthroughs in discrete log computation

High-Level Contributions

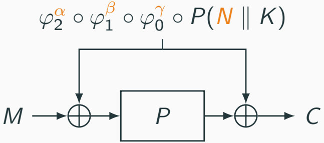
Masked Even-Mansour

- Improved masking for tweakable blockciphers
- Very efficient
- Simple to implement
- Constant time (by default)
- Relies on breakthroughs in discrete log computation

Application to Authenticated Encryption

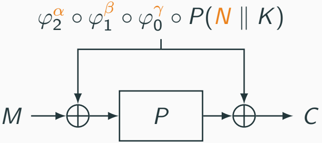
- Nonce-respecting AE in 0.55 cpb
- Misuse-resistant AE in 1.06 cpb

Masked Even-Mansour (MEM)



- LFSRs: φ_i
- Tweak (simplified): $(\alpha, \beta, \gamma, N)$

Masked Even-Mansour (MEM)



- LFSRs: φ_i
- Tweak (simplified): $(\alpha, \beta, \gamma, N)$
- Combines advantages of:
  - Powering-up masking
  - Word-based LFSRs

Design Considerations

Powering-up / Gray Code Masking

1. Start from mathematical structure
2. Try to find an efficient map

Design Considerations

Powering-up / Gray Code Masking

1. Start from mathematical structure
2. Try to find an efficient map

Problems

- Galois field operations
- Conditional shifts and XORs
- Hard to implement in constant time
- Expensive on certain platforms

Design Considerations

MEM Masking

1. Start from efficient linear map φ on b-bit state
2. Prove that φ is an isomorphism on $\mathbb{F}_{2^b}$

Design Considerations

MEM Masking

1. Start from efficient linear map φ on b-bit state
2. Prove that φ is an isomorphism on $\mathbb{F}_{2^b}$

Advantages

- Simple operations (XORs, shifts, etc.)
- Very efficient (SIMD-friendly)
- Minimal space usage
- Constant-time by design

Masking Function Search

1. Start with linear mapping φ on n words of w-bits each

$$\varphi : (x_0, \ldots, x_{n-1}) \mapsto (x_1, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}))$$

and feedback function f

Masking Function Search

1. Start with linear mapping φ on n words of w-bits each

$$\varphi : (x_0, \ldots, x_{n-1}) \mapsto (x_1, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}))$$

   and feedback function f

2. Model φ as matrix

$$M = \begin{pmatrix} 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \\ X_0 & X_1 & \cdots & X_{n-1} \end{pmatrix} \in \mathbb{F}_{2^{nw}} \times \mathbb{F}_{2^{nw}}$$

   with $X_i \in \{0, I, \mathrm{SHL}_c,\ \mathrm{SHR}_c,\ \mathrm{ROT}_c,\ \mathrm{AND}_c\}$, $\dim(X_i) = w$

Masking Function Search

1. Start with linear mapping φ on n words of w-bits each

$$\varphi : (x_0, \ldots, x_{n-1}) \mapsto (x_1, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}))$$

   and feedback function f

2. Model φ as matrix

$$M = \begin{pmatrix} 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \\ X_0 & X_1 & \cdots & X_{n-1} \end{pmatrix} \in \mathbb{F}_{2^{nw}} \times \mathbb{F}_{2^{nw}}$$

   with $X_i \in \{0, I, \mathrm{SHL}_c, \mathrm{SHR}_c, \mathrm{ROT}_c, \mathrm{AND}_c\}$, $\dim(X_i) = w$

3. Check: Is minimal polynomial of M primitive of degree $b = nw$?
   - Yes: $\varphi^i(L) = M^i \cdot L$ has maximum period $2^b - 1$; keep φ
   - No: Discard φ; move on to the next candidate

Sample LFSRs

Suitable for	b	w	n	φ
	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \gg 3)$
	128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \gg 25)$
	256	64	4	$(x_1, \ldots, x_3, (x_0 \lll 3) \oplus (x_3 \gg 5))$
	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \gg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Sample LFSRs

Suitable for	b	w	n	φ
AES	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \gg 3)$
	128	32	4	$(x_1, \ldots, x_3, \; (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, \; (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \gg 25)$
	256	64	4	$(x_1, \ldots, x_3, \; (x_0 \lll 3) \oplus (x_3 \gg 5))$
	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, \; (x_0 \lll 29) \oplus (x_1 \ll 9))$
	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \gg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Sample LFSRs

Suitable for	b	w	n	φ
AES	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \gg 3)$
	128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \gg 25)$
	256	64	4	$(x_1, \ldots, x_3, (x_0 \lll 3) \oplus (x_3 \gg 5))$
ChaCha	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \gg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Sample LFSRs

Suitable for	b	w	n	φ
AES	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \ggg 3)$
	128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \ggg 25)$
	256	64	4	$(x_1, \ldots, x_3, (x_0 \lll 3) \oplus (x_3 \gg 5))$
ChaCha	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
Keccak-f[800]	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \ggg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

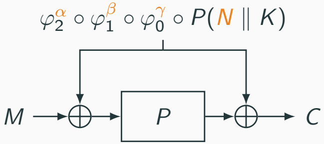
Sample LFSRs

Suitable for	b	w	n	φ
AES	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \gg 3)$
	128	32	4	$(x_1, \ldots, x_3, \ (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, \quad (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \gg 25)$
	256	64	4	$(x_1, \ldots, x_3, \ (x_0 \lll 3) \oplus (x_3 \gg 5))$
ChaCha	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, \ (x_0 \lll 29) \oplus (x_1 \ll 9))$
Keccak-f[800]	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
BLAKE2b	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \gg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Sample LFSRs

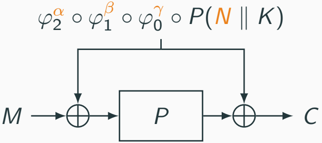
Suitable for	b	w	n	φ
AES	128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 2) \oplus ((x_4 \parallel x_3) \gg 3)$
	128	32	4	$(x_1, \ldots, x_3, \; (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
	128	64	2	$(x_1, \qquad (x_0 \lll 4) \oplus ((x_1 \parallel x_0) \gg 25)$
	256	64	4	$(x_1, \ldots, x_3, \; (x_0 \lll 3) \oplus (x_3 \gg 5))$
ChaCha	512	32	16	$(x_1, \ldots, x_{15}, (x_0 \lll 5) \oplus (x_3 \gg 7))$
	512	64	8	$(x_1, \ldots, x_7, \; (x_0 \lll 29) \oplus (x_1 \ll 9))$
Keccak-f[800]	800	32	25	$(x_1, \ldots, x_{24}, (x_0 \lll 25) \oplus x_{21} \oplus (x_{21} \gg 13))$
BLAKE2b	1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
	1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
Keccak-f[1600]	1600	64	25	$(x_1, \ldots, x_{24}, (x_0 \lll 14) \oplus ((x_1 \parallel x_0) \gg 13)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Back to MEM



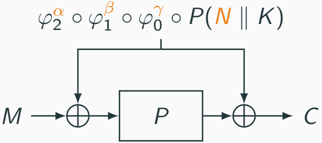
$\varphi_2^{\alpha} \circ \varphi_1^{\beta} \circ \varphi_0^{\gamma} \circ P(N \parallel K)$
M
P
C

Back to MEM



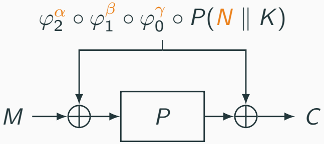
- Not just one but three maskings!

Back to MEM



- Not just one but three maskings!
- From one to three:
  - $\varphi_0 = \varphi$
  - $\varphi_1 = \varphi \oplus id$
  - $\varphi_2 = \varphi^2 \oplus \varphi \oplus id$

Back to MEM



- Not just one but three maskings!
- From one to three:
  - $\varphi_0 = \varphi$
  - $\varphi_1 = \varphi \oplus id$
  - $\varphi_2 = \varphi^2 \oplus \varphi \oplus id$
- For which (α, β, γ) is the above masking unique?

Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$

Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

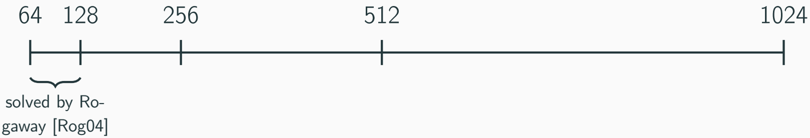
  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)

Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)
- Requires computation of discrete logarithms

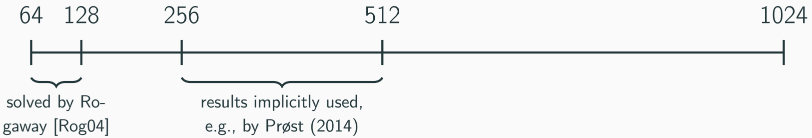


Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)
- Requires computation of discrete logarithms

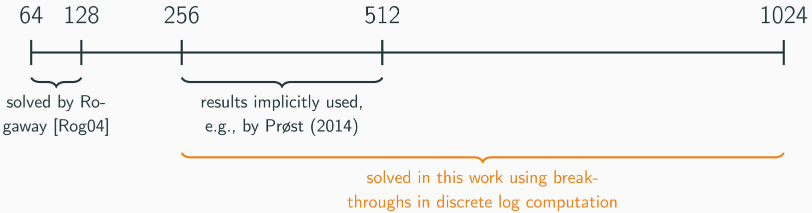


Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)
- Requires computation of discrete logarithms

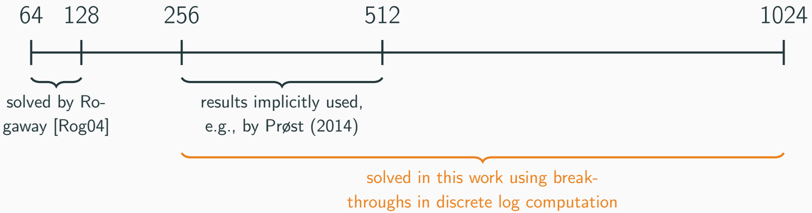


Uniqueness of Masking

- Intuitively, masking goes well as long as

$$\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha} \neq \varphi_2^{\gamma'} \circ \varphi_1^{\beta'} \circ \varphi_0^{\alpha'}$$

  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)
- Requires computation of discrete logarithms



- Logs for $\mathbb{F}_{2^n}$, $n \leq 13$ doable with latest techniques

Tweak Space Domain Separation

Lemma

- *Let* $\varphi : \mathbb{F}_{2^{1024}} \mapsto \mathbb{F}_{2^{1024}}$, *with*

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

and associated transformation matrix M

Tweak Space Domain Separation

Lemma

- *Let* $\varphi : \mathbb{F}_{2^{1024}} \mapsto \mathbb{F}_{2^{1024}}$, *with*

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

  and associated transformation matrix M
- *Let*
  - $\varphi_0^{\alpha}(L) = M^{\alpha} \cdot L$,
  - $\varphi_1^{\beta}(L) = (M + I)^{\beta} \cdot L$
  - $\varphi_2^{\gamma}(L) = (M^2 + M + I)^{\gamma} \cdot L$

Tweak Space Domain Separation

Lemma

- *Let* $\varphi : \mathbb{F}_{2^{1024}} \mapsto \mathbb{F}_{2^{1024}}$, *with*

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

  and associated transformation matrix M
- *Let*
  - $\varphi_0^{\alpha}(L) = M^{\alpha} \cdot L$,
  - $\varphi_1^{\beta}(L) = (M + I)^{\beta} \cdot L$
  - $\varphi_2^{\gamma}(L) = (M^2 + M + I)^{\gamma} \cdot L$

The composition

$$\varphi_0^{\alpha} \circ \varphi_1^{\beta} \circ \varphi_2^{\gamma}$$

specifies an injective map on the tweak space

$$\mathcal{T} = \mathcal{T}_0 \times \mathcal{T}_1 \times \mathcal{T}_2 = \{0, 1, \ldots, 2^{1020} - 1\} \times \{0, 1, 2, 3\} \times \{0, 1\}$$

Tweak Space Domain Separation

Proof Outline

For LFSRs φ_0, φ_1, φ_2 and

$$\mathcal{T} = \mathcal{T}_0 \times \mathcal{T}_1 \times \mathcal{T}_2 = \{0, 1, \ldots, 2^{1020} - 1\} \times \{0, 1, 2, 3\} \times \{0, 1\}$$

show that

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$

Tweak Space Domain Separation

Proof Outline

For LFSRs φ_0, φ_1, φ_2 and

$$\mathcal{T} = \mathcal{T}_0 \times \mathcal{T}_1 \times \mathcal{T}_2 = \{0, 1, \ldots, 2^{1020} - 1\} \times \{0, 1, 2, 3\} \times \{0, 1\}$$

show that

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$

Proof Sketch

1.1 Minimal polynomial of M:

$$p(x) = x^{1024} + x^{901} + x^{695} + x^{572} + x^{409} + x^{366} + x^{203} + x^{163} + 1$$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$ ✓
2. $\varphi_2^\gamma \circ \varphi_1^\beta \circ \varphi_0^\alpha$ is injective on $\mathcal{T}$

Proof Sketch

1.1 Minimal polynomial of M:

$$p(x) = x^{1024} + x^{901} + x^{695} + x^{572} + x^{409} + x^{366} + x^{203} + x^{163} + 1$$

1.2 p is irreducible and primitive:

- M has order $2^{1024} - 1$
- φ_0 has period $2^{1024} - 1$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$ ✓
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$

Proof Sketch

2.1 Compute $l_1 = \log_x(x+1)$ and $l_2 = \log_x(x^2+x+1)$ in $\mathbb{F}_2[x]/p(x)$, then $M^{\alpha}(M+I)^{\beta}(M^2+M+I)^{\gamma} \Leftrightarrow M^{\alpha}M^{l_1\beta}M^{l_2\gamma}$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$ ✓
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$

Proof Sketch

2.1 Compute $l_1 = \log_x(x+1)$ and $l_2 = \log_x(x^2+x+1)$ in $\mathbb{F}_2[x]/p(x)$, then $M^{\alpha}(M+I)^{\beta}(M^2+M+I)^{\gamma} \Leftrightarrow M^{\alpha} M^{l_1 \beta} M^{l_2 \gamma}$

2.2 Given distinct (α, β, γ), $(\alpha', \beta', \gamma') \in \mathcal{T}$:

$$M^{\alpha} M^{l_1 \beta} M^{l_2 \gamma} = M^{\alpha'} M^{l_1 \beta'} M^{l_2 \gamma'}$$

$$\Leftrightarrow$$

$$\alpha - \alpha' = (\beta - \beta') l_1 + (\gamma - \gamma') l_2 \bmod 2^{1024} - 1$$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$ ✓
2. $\varphi_2^\gamma \circ \varphi_1^\beta \circ \varphi_0^\alpha$ is injective on $\mathcal{T}$

Proof Sketch

2.1 Compute $l_1 = \log_x(x+1)$ and $l_2 = \log_x(x^2+x+1)$ in $\mathbb{F}_2[x]/p(x)$, then $M^\alpha(M+I)^\beta(M^2+M+I)^\gamma \Leftrightarrow M^\alpha M^{l_1\beta} M^{l_2\gamma}$

2.2 Given distinct (α, β, γ), $(\alpha', \beta', \gamma') \in \mathcal{T}$:

$$M^\alpha M^{l_1\beta} M^{l_2\gamma} = M^{\alpha'} M^{l_1\beta'} M^{l_2\gamma'}$$

$$\Leftrightarrow$$

$$\alpha - \alpha' = (\beta - \beta')l_1 + (\gamma - \gamma')l_2 \bmod 2^{1024} - 1$$

2.3 Then $$2^{1020.58} \leq |(\beta - \beta')l_1 + (\gamma - \gamma')l_2|$$

Tweak Space Domain Separation

Show

1. LFSR φ_0 $(= \varphi)$ has period $2^{1024} - 1$ ✓
2. $\varphi_2^{\gamma} \circ \varphi_1^{\beta} \circ \varphi_0^{\alpha}$ is injective on $\mathcal{T}$ ✓

Proof Sketch

2.1 Compute $l_1 = \log_x(x+1)$ and $l_2 = \log_x(x^2+x+1)$ in $\mathbb{F}_2[x]/p(x)$, then $M^{\alpha}(M+I)^{\beta}(M^2+M+I)^{\gamma} \Leftrightarrow M^{\alpha} M^{l_1\beta} M^{l_2\gamma}$

2.2 Given distinct (α, β, γ), $(\alpha', \beta', \gamma') \in \mathcal{T}$:

$$M^{\alpha} M^{l_1\beta} M^{l_2\gamma} = M^{\alpha'} M^{l_1\beta'} M^{l_2\gamma'}$$

$$\Leftrightarrow$$

$$\alpha - \alpha' = (\beta - \beta')l_1 + (\gamma - \gamma')l_2 \bmod 2^{1024} - 1$$

2.3 Then $|\alpha - \alpha'| \leq 2^{1020} - 1 < 2^{1020.58} \leq |(\beta - \beta')l_1 + (\gamma - \gamma')l_2|$

Tweak Space Domain Separation via Lattices

- Lattice spanned by rows of

$$\begin{pmatrix} K \cdot 1 & w_0 & 0 & 0 \\ K \cdot l_1 & 0 & w_1 & 0 \\ K \cdot l_2 & 0 & 0 & w_2 \\ K \cdot m & 0 & 0 & 0 \end{pmatrix}$$

  for integers K, $m = 2^b - 1$, weights w_i, and dlogs l_1,l_2

Tweak Space Domain Separation via Lattices

- Lattice spanned by rows of

$$\begin{pmatrix} K \cdot 1 & w_0 & 0 & 0 \\ K \cdot l_1 & 0 & w_1 & 0 \\ K \cdot l_2 & 0 & 0 & w_2 \\ K \cdot m & 0 & 0 & 0 \end{pmatrix}$$

  for integers K, $m = 2^b - 1$, weights w_i, and dlogs l_1,l_2
- Then

$$(\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 + km, \Delta\alpha w_0, \Delta\beta w_1, \Delta\gamma w_2)$$

  is shortest vector if

$$\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 \equiv 0 \pmod{2^b - 1}$$

Tweak Space Domain Separation via Lattices

- Lattice spanned by rows of

$$\begin{pmatrix} K \cdot 1 & w_0 & 0 & 0 \\ K \cdot l_1 & 0 & w_1 & 0 \\ K \cdot l_2 & 0 & 0 & w_2 \\ K \cdot m & 0 & 0 & 0 \end{pmatrix}$$

  for integers K, $m = 2^b - 1$, weights w_i, and dlogs l_1, l_2

- Then

$$(\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 + km, \Delta\alpha w_0, \Delta\beta w_1, \Delta\gamma w_2)$$

  is shortest vector if

$$\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 \equiv 0 \pmod{2^b - 1}$$

- Finally, weights

$$(w_0, w_1, w_2) = (1, 2^{1019}, 2^{1022})$$

  give a similar tweak space as in Lemma on last slide

Tweak Space Domain Separation via Lattices

- Lattice spanned by rows of

$$\begin{pmatrix} K \cdot 1 & w_0 & 0 & 0 \\ K \cdot l_1 & 0 & w_1 & 0 \\ K \cdot l_2 & 0 & 0 & w_2 \\ K \cdot m & 0 & 0 & 0 \end{pmatrix}$$

  for integers K, $m = 2^b - 1$, weights w_i, and dlogs l_1, l_2

- Then

$$(\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 + km, \Delta\alpha w_0, \Delta\beta w_1, \Delta\gamma w_2)$$

  is shortest vector if

$$\Delta\alpha + \Delta\beta l_1 + \Delta\gamma l_2 \equiv 0 \pmod{2^b - 1}$$

- Finally, weights

$$(w_0, w_1, w_2) = (1, 2^{1019}, 2^{1022})$$

  give a similar tweak space as in Lemma on last slide

Credits: Antoine Joux

Computation of Discrete Logarithms

Overview

- Recent improvements of Function Field Sieve (FFS) enabled dlog-breakthroughs

Computation of Discrete Logarithms

Overview

- Recent improvements of Function Field Sieve (FFS) enabled dlog-breakthroughs
- Current records:
    - $\mathbb{F}_{2^{9234}}$, Kummer extension, 45 core years [GKZ14]
    - $\mathbb{F}_{3^{6 \cdot 509}}$, Non-Kummer extension, 200 core years [AMCMORR16]

Computation of Discrete Logarithms

Overview

- Recent improvements of Function Field Sieve (FFS) enabled dlog-breakthroughs
- Current records:
  - $\mathbb{F}_{2^{9234}}$, Kummer extension, 45 core years [GKZ14]
  - $\mathbb{F}_{3^{6 \cdot 509}}$, Non-Kummer extension, 200 core years [AMCMORR16]

Our Computations

- Standard desktop computer (2.0 GHz AMD Opteron)
- MAGMA v2.19-1

Computation of Discrete Logarithms

Overview

- Recent improvements of Function Field Sieve (FFS) enabled dlog-breakthroughs
- Current records:
  - $\mathbb{F}_{2^{9234}}$, Kummer extension, 45 core years [GKZ14]
  - $\mathbb{F}_{3^{6\cdot 509}}$, Non-Kummer extension, 200 core years [AMCMORR16]

Our Computations

- Standard desktop computer (2.0 GHz AMD Opteron)
- MAGMA v2.19-1
- Total time for $\log_x(x+1)$ and $\log_x(x^2+x+1)$ over
  - $\mathbb{F}_{2^{512}}$: $\approx$ 14 h
  - $\mathbb{F}_{2^{1024}}$: $\approx$ 57 h

Computation of Discrete Logarithms

Overview

- Recent improvements of Function Field Sieve (FFS) enabled dlog-breakthroughs
- Current records:
  - $\mathbb{F}_{2^{9234}}$, Kummer extension, 45 core years [GKZ14]
  - $\mathbb{F}_{3^{6 \cdot 509}}$, Non-Kummer extension, 200 core years [AMCMORR16]

Our Computations

- Standard desktop computer (2.0 GHz AMD Opteron)
- MAGMA v2.19-1
- Total time for $\log_x(x+1)$ and $\log_x(x^2+x+1)$ over
  - $\mathbb{F}_{2^{512}}$: $\approx$ 14 h
  - $\mathbb{F}_{2^{1024}}$: $\approx$ 57 h
- Note: Logarithms need to be computed only once!

Discrete Logarithms

SageMath verification script for $\mathbb{F}_{2^{512}}$:

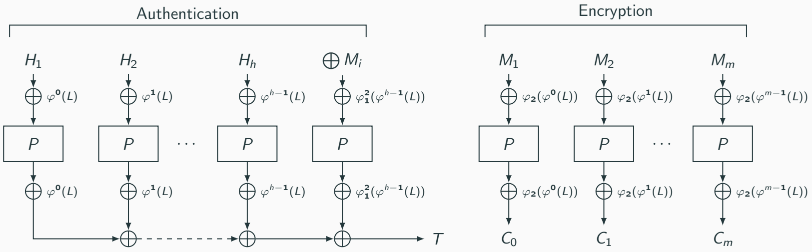
```
p = x^512+x^335+x^201+x^67+1
K.<x> = GF(2^512, name='x', modulus=p)

l1 = 50163230286657067056366097095502896190369019796688 73\
     48726437885165144058824116111559205826863092667238 54\
     5122357792870542653280226105514939849018182092 9802

l2 = 77897950545970351229609335026530822098657247807843 81\
     21666265130193338780341425004779419500813036756334 01\
     1185966465812007766565485320190254829936577378 9462

x^l1 == x+1
x^l2 == x^2+x+1
```

SageMath verification script for $\mathbb{F}_{2^{1024}}$:

```
p = x^1024+x^901+x^695+x^572+x^409+x^366+x^203+x^163+1
K.<x> = GF(2^1024, name='x', modulus=p)

l1 = 3560313810702380168941895068061768846768652879916524\
     2796753456565509842707655755413753100620979021885720\
     1966785351480307697311709456831372018598499174441196\
     1470332602216161583378362583657570756631024935927984\
     2498272238699528576230685242805763938951155448126495\
     51247501486738714968190387640606750264547115219 3

l2 = 1610056439189028793452144461315558447020117376432642\
     5524859486238161374654279717800300706136749607630601\
     4967362673777547140089938700144112424081388711871290\
     7973319251629628361398267351880948069161459793052257\
     1907117948291164323355528169854354396482029507781947\
     25341713130769377757979091597888793618760998888 34

x^l1 == x+1
x^l2 == x^2+x+1
```

Applications to Authenticated Encryption

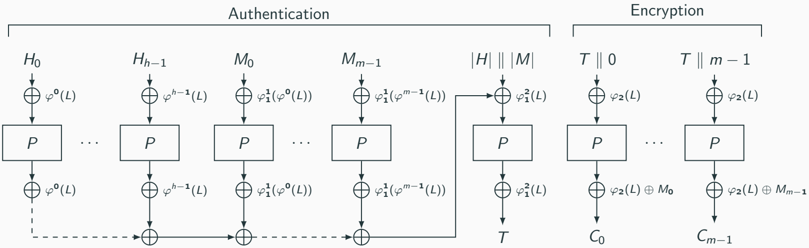
Offset Public Permutation (OPP)



$L = P(N \parallel K)$
$\varphi_1 = \varphi \oplus id$
$\varphi_2 = \varphi^2 \oplus \varphi \oplus id$

- Security against nonce-respecting adversaries
- 1-pass
- Fully parallelizable

Misuse-Resistant OPP (MRO)



$L = P(N \parallel K)$
$\varphi_1 = \varphi \oplus id$
$\varphi_2 = \varphi^2 \oplus \varphi \oplus id$

- Fully nonce-misuse resistant version of OPP
- 2-pass
- Fully parallelizable

Implementation & Evaluation

Setup

Parameterization for OPP/MRO

- State size $b = 1024$ bits

Setup

Parameterization for OPP/MRO

- State size $b = 1024$ bits
- Permutation P from BLAKE2b with 4 and 6 rounds

Setup

Parameterization for OPP/MRO

- State size $b = 1024$ bits
- Permutation P from BLAKE2b with 4 and 6 rounds
- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

Setup

Parameterization for OPP/MRO

- State size $b = 1024$ bits
- Permutation P from BLAKE2b with 4 and 6 rounds
- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

Implementations in plain C, NEON, AVX, AVX2

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{array}$$

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & & & \end{array}$$

- $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & & \end{array}$$

  - $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$
  - $x_{17} = (x_1 \lll 53) \oplus (x_6 \ll 13)$

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & \end{array}$$

- $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$
- $x_{17} = (x_1 \lll 53) \oplus (x_6 \ll 13)$
- $x_{18} = (x_2 \lll 53) \oplus (x_7 \ll 13)$

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

- $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$
- $x_{17} = (x_1 \lll 53) \oplus (x_6 \ll 13)$
- $x_{18} = (x_2 \lll 53) \oplus (x_7 \ll 13)$
- $x_{19} = (x_3 \lll 53) \oplus (x_8 \ll 13)$

Mask Implementation

- LFSR on 16 words of 64 bits:

$$\varphi(x_0, \ldots, x_{15}) = (x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$$

- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

  - $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$
  - $x_{17} = (x_1 \lll 53) \oplus (x_6 \ll 13)$
  - $x_{18} = (x_2 \lll 53) \oplus (x_7 \ll 13)$
  - $x_{19} = (x_3 \lll 53) \oplus (x_8 \ll 13)$

- Computable in parallel (e.g. AVX2)

Mask Implementation

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+1}

Mask Implementation

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+1}

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+2}

Mask Implementation

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

L_{i+1}

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

L_{i+2}

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

L_{i+3}

Mask Implementation

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+1}

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+2}

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+3}

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

L_{i+4}

Mask Implementation – Wordslicing

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[0],\ L_{i+\mathbf{2}}[0],\ L_{i+\mathbf{3}}[0],\ L_{i+\mathbf{4}}[0]$

Mask Implementation – Wordslicing

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[0],\ L_{i+\mathbf{2}}[0],\ L_{i+\mathbf{3}}[0],\ L_{i+\mathbf{4}}[0]$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[1],\ L_{i+\mathbf{2}}[1],\ L_{i+\mathbf{3}}[1],\ L_{i+\mathbf{4}}[1]$

Mask Implementation – Wordslicing

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[0],\ L_{i+\mathbf{2}}[0],\ L_{i+\mathbf{3}}[0],\ L_{i+\mathbf{4}}[0]$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[1],\ L_{i+\mathbf{2}}[1],\ L_{i+\mathbf{3}}[1],\ L_{i+\mathbf{4}}[1]$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+\mathbf{1}}[2],\ L_{i+\mathbf{2}}[2],\ L_{i+\mathbf{3}}[2],\ L_{i+\mathbf{4}}[2]$

Mask Implementation – Wordslicing

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

$L_{i+1}[0],\ L_{i+2}[0],\ L_{i+3}[0],\ L_{i+4}[0]$

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

$L_{i+1}[1],\ L_{i+2}[1],\ L_{i+3}[1],\ L_{i+4}[1]$

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

$L_{i+1}[2],\ L_{i+2}[2],\ L_{i+3}[2],\ L_{i+4}[2]$

$\cdots$

x_0	x_1	x_2	x_3
x_4	x_5	x_6	x_7
x_8	x_9	x_{10}	x_{11}
x_{12}	x_{13}	x_{14}	x_{15}
x_{16}	x_{17}	x_{18}	x_{19}

$L_{i+1}[15],\ L_{i+2}[15],\ L_{i+3}[15],\ L_{i+4}[15]$

Mask Implementation – Wordslicing

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+1}[0],\ L_{i+2}[0],\ L_{i+3}[0],\ L_{i+4}[0]$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+1}[1],\ L_{i+2}[1],\ L_{i+3}[1],\ L_{i+4}[1]$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+1}[2],\ L_{i+2}[2],\ L_{i+3}[2],\ L_{i+4}[2]$

$\cdots$

$$\begin{array}{cccc} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \\ x_{16} & x_{17} & x_{18} & x_{19} \end{array}$$

$L_{i+1}[15],\ L_{i+2}[15],\ L_{i+3}[15],\ L_{i+4}[15]$

No data layout changes needed!

Mask Implementation

$$\varphi_0(x) = \varphi(x) = (x_1, \ldots, x_{15}, t_0)$$

$$t_0 = (x_0 \lll 53) \oplus (x_5 \ll 13)$$

Mask Implementation

$$\varphi_0(x) = \varphi(x) = (x_1, \ldots, x_{15}, t_0)$$

$$\begin{aligned}\varphi_1(x) &= \varphi(x) \oplus id(x) \\ &= (x_0 \oplus x_1, \ldots, x_{14} \oplus x_{15}, x_{15} \oplus t_0)\end{aligned}$$

$$t_0 = (x_0 \lll 53) \oplus (x_5 \ll 13)$$

Mask Implementation

$$\varphi_0(x) = \varphi(x) = (x_1, \ldots, x_{15}, t_0)$$

$$\begin{aligned}\varphi_1(x) &= \varphi(x) \oplus id(x) \\ &= (x_0 \oplus x_1, \ldots, x_{14} \oplus x_{15}, x_{15} \oplus t_0)\end{aligned}$$

$$\begin{aligned}\varphi_2(x) &= \varphi^2(x) \oplus \varphi(x) \oplus id(x) \\ &= (x_0 \oplus x_1 \oplus x_2, \ldots, x_{14} \oplus x_{15} \oplus t_0, x_{15} \oplus t_0 \oplus t_1)\end{aligned}$$

$$\begin{aligned}t_0 &= (x_0 \lll 53) \oplus (x_5 \ll 13) \\ t_1 &= (x_1 \lll 53) \oplus (x_6 \ll 13)\end{aligned}$$

Performance of Bare Masking

- Cycles per update in most pessimistic scenario (for ours):

Masking	Sandy Bridge	Haswell
Powering-up	13.108	10.382
Gray code	6.303	3.666
Ours	2.850	2.752

Performance of Bare Masking

- Cycles per update in most pessimistic scenario (for ours):

Masking	Sandy Bridge	Haswell
Powering-up	13.108	10.382
Gray code	6.303	3.666
Ours	2.850	2.752

- Differences may amplify/diminish in a mode

Performance of Bare Masking

- Cycles per update in most pessimistic scenario (for ours):

Masking	Sandy Bridge	Haswell
Powering-up	13.108	10.382
Gray code	6.303	3.666
Ours	2.850	2.752

- Differences may amplify/diminish in a mode
- Speeds closer to each other for smaller states

Performance of OPP/MRO

- Results for long messages (≥ 4 KiB) in cycles per byte:

	nonce-respecting				
Platform	OPP_4	OPP_6	AES-GCM	$Deoxys^{\neq}$	OCB3
Cortex-A8	4.26	5.91	38.6	-	28.9
Sandy Bridge	1.24	1.91	2.55	1.29	0.98
Haswell	0.55	0.75	1.03	0.96	0.69

	misuse-resistant			
Platform	MRO_4	MRO_6	GCM-SIV	$Deoxys^{=}$
Cortex-A8	8.07	11.32	-	-
Sandy Bridge	2.41	3.58	-	2.58
Haswell	1.06	1.39	1.17	1.92

Performance of OPP/MRO

- Results for long messages (≥ 4 KiB) in cycles per byte:

	nonce-respecting				
Platform	OPP_4	OPP_6	AES-GCM	$Deoxys^{\neq}$	OCB3
Cortex-A8	4.26	5.91	38.6	-	28.9
Sandy Bridge	1.24	1.91	2.55	1.29	0.98
Haswell	0.55	0.75	1.03	0.96	0.69

	misuse-resistant			
Platform	MRO_4	MRO_6	GCM-SIV	$Deoxys^{=}$
Cortex-A8	8.07	11.32	-	-
Sandy Bridge	2.41	3.58	-	2.58
Haswell	1.06	1.39	1.17	1.92

Performance of OPP/MRO

- Results for long messages ($\geq$ 4 KiB) in cycles per byte:

	nonce-respecting				
Platform	OPP_4	OPP_6	AES-GCM	$Deoxys^{\neq}$	OCB3
Cortex-A8	4.26	5.91	38.6	-	28.9
Sandy Bridge	1.24	1.91	2.55	1.29	0.98
Haswell	0.55	0.75	1.03	0.96	0.69

	misuse-resistant			
Platform	MRO_4	MRO_6	GCM-SIV	$Deoxys^{=}$
Cortex-A8	8.07	11.32	-	-
Sandy Bridge	2.41	3.58	-	2.58
Haswell	1.06	1.39	1.17	1.92

- Haswell throughput:
  - OPP_4: $\approx$ 6.36 GiBps
  - MRO_4: $\approx$ 3.30 GiBps

Conclusion

Conclusion

Masked Even-Mansour

- Tweakable blockcipher
- Simple, efficient, constant-time (by default) masking

Conclusion

Masked Even-Mansour

- Tweakable blockcipher
- Simple, efficient, constant-time (by default) masking
- Constructive usage of breakthroughs in discrete log computation
- MEM-based AE is able to outperform its closest competitors

Conclusion

Masked Even-Mansour

- Tweakable blockcipher
- Simple, efficient, constant-time (by default) masking
- Constructive usage of breakthroughs in discrete log computation
- MEM-based AE is able to outperform its closest competitors

More Info

- `https://eprint.iacr.org/2015/999`
- `https://github.com/MEM-AEAD`

Conclusion

Masked Even-Mansour

- Tweakable blockcipher
- Simple, efficient, constant-time (by default) masking
- Constructive usage of breakthroughs in discrete log computation
- MEM-based AE is able to outperform its closest competitors

More Info

- `https://eprint.iacr.org/2015/999`
- `https://github.com/MEM-AEAD`

`// Thank You`

`@Daeinar`