Introduction to Tweakable Blockciphers

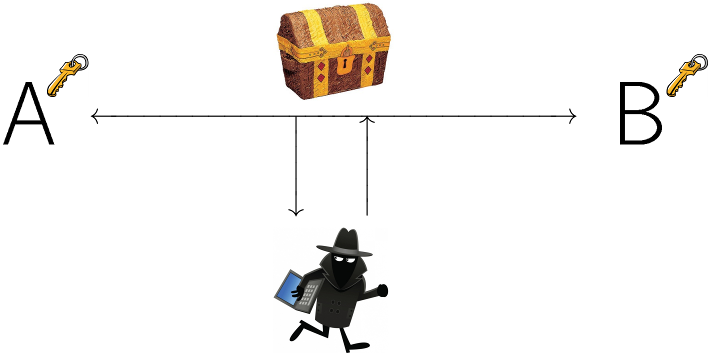
Bart Mennink
Radboud University (The Netherlands)

Summer school on real-world crypto and privacy

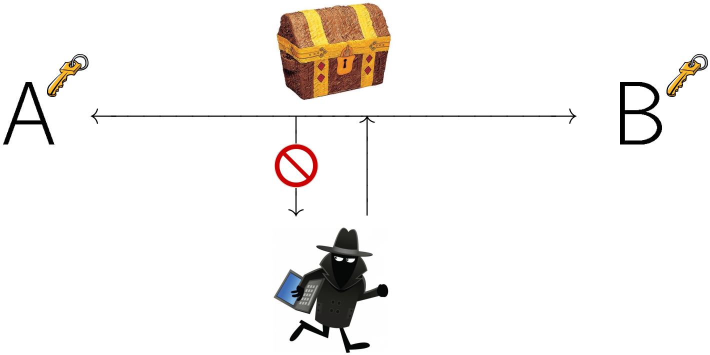
June 5, 2017

Authenticated Encryption

Authenticated Encryption



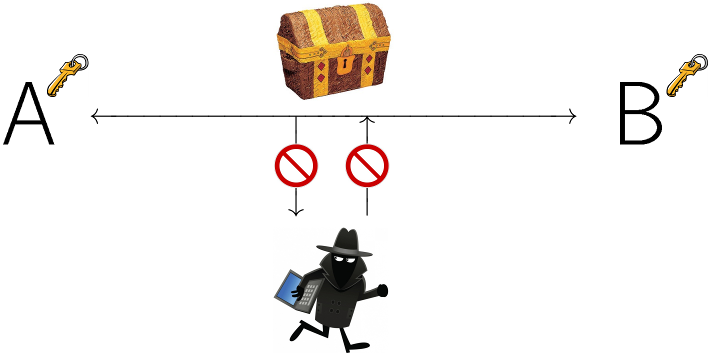
Authenticated Encryption



Encryption

- No outsider can learn anything about data

Authenticated Encryption



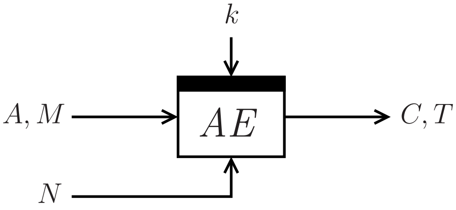
Encryption

- No outsider can learn anything about data

Authentication

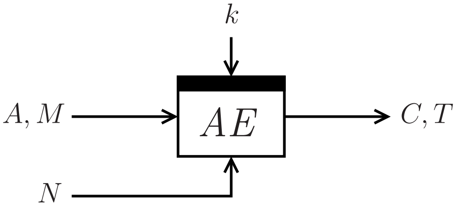
- No outsider can manipulate data

Authenticated Encryption



- Ciphertext C encryption of message M
- Tag T authenticates associated data A and message M

Authenticated Encryption



- Ciphertext C encryption of message M
- Tag T authenticates associated data A and message M
- Nonce N randomizes the scheme

CAESAR Competition

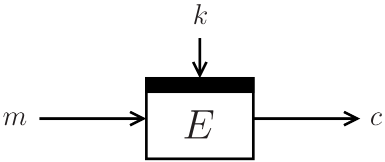
Competition for Authenticated Encryption:
Security, Applicability, and Robustness

Goal: portfolio of authenticated encryption schemes

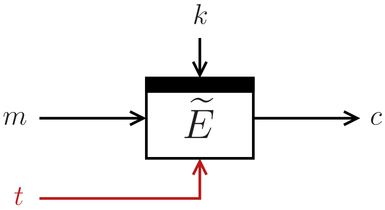
Mar 15, 2014: 57 first round candidates
Jul 7, 2015: 29.5 second round candidates
Aug 15, 2016: 16 third round candidates
??: announcement of finalists
Dec 15, 2017: announcement of final portfolio (?)

CAESAR Competition, Not To Be Confused With:

Tweakable Blockciphers

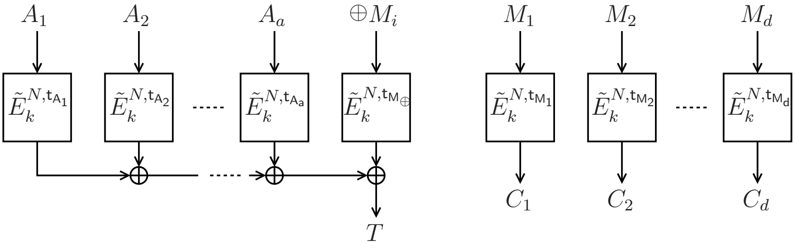


Tweakable Blockciphers



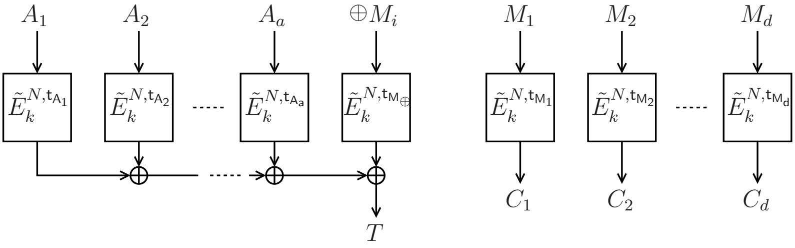
- Tweak: flexibility to the cipher
- Each tweak gives different permutation

Tweakable Blockciphers in OCBx



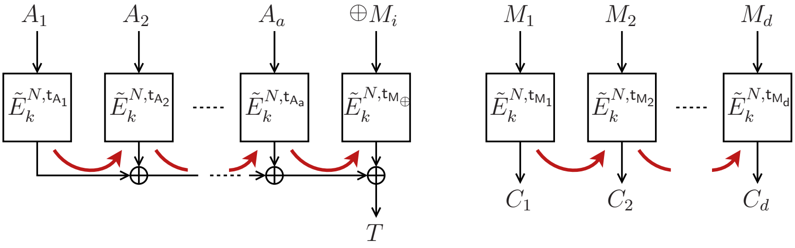
- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]

Tweakable Blockciphers in OCBx



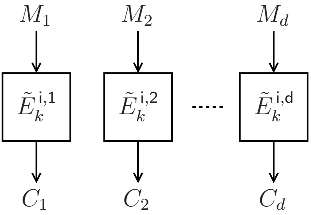
- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable blockcipher $\widetilde{E}$
    - Tweak (N, tweak) is unique for every evaluation
    - Different blocks always transformed under different tweak

Tweakable Blockciphers in OCBx



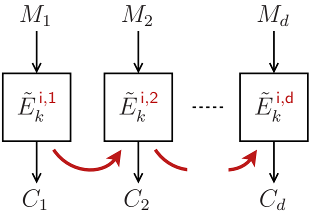
- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable blockcipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation
  - Different blocks always transformed under different tweak
- Change of tweak should be efficient

Tweakable Blockciphers in XTS



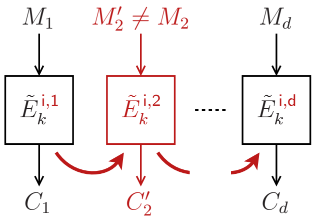
- XTS mode for disk encryption
- Tweak $(\mathsf{i}, \mathsf{j}) = (\mathsf{sector}, \mathsf{block})$ unique for every block

Tweakable Blockciphers in XTS



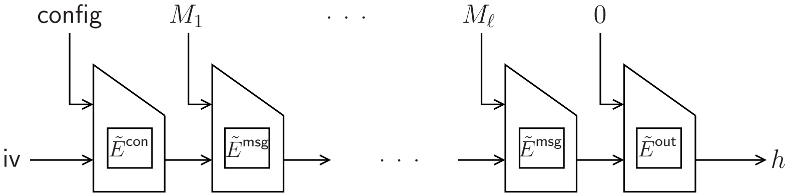
- XTS mode for disk encryption
- Tweak $(i, j) = (\text{sector}, \text{block})$ unique for every block
- Change of tweak should be efficient (as before)

Tweakable Blockciphers in XTS



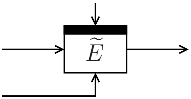
- XTS mode for disk encryption
- Tweak $(\mathsf{i},\mathsf{j}) = (\mathsf{sector},\mathsf{block})$ unique for every block
- Change of tweak should be efficient (as before)
- Incrementality: change in one (or few) blocks

Tweakable Blockciphers in Skein

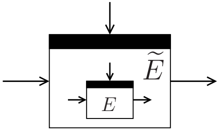


- Skein hash function by Ferguson et al. [FLS+07]
- Based on Threefish tweakable blockcipher
- Tweaks used for domain separation

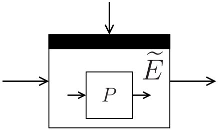
Tweakable Blockcipher Designs



Dedicated

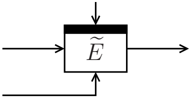


Blockcipher-Based



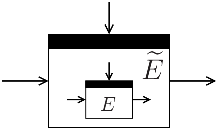
Permutation-Based

Tweakable Blockcipher Designs in CAESAR



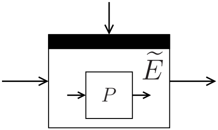
Dedicated

KIASU,
Joltik,
SCREAM,
Deoxys



Blockcipher-Based

CBA, COBRA, iFeed,
Marble, **OMD**, **POET**,
SHELL, **AEZ**, **COPA/ELmD**, **OCB**, **OTR**



Permutation-Based

Prøst,
Minalpher

first round, **second round**, **third round**

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

Beyond Masking-Based Tweakable Blockciphers

Conclusion

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

Beyond Masking-Based Tweakable Blockciphers

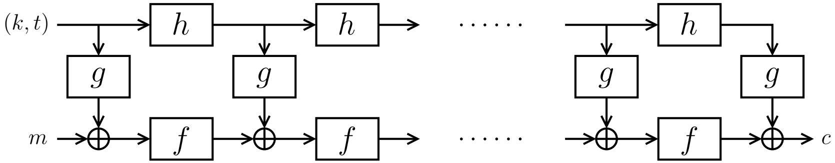
Conclusion

Dedicated Tweakable Blockciphers

- Hasty Pudding Cipher [Sch98]
    - AES submission, "first tweakable cipher"
- Mercy [Cro01]
    - Disk encryption
- Threefish [FLS+07]
    - SHA-3 submission Skein
- TWEAKEY framework [JNP14]
    - Four CAESAR submissions
    - SKINNY & MANTIS

TWEAKEY Framework

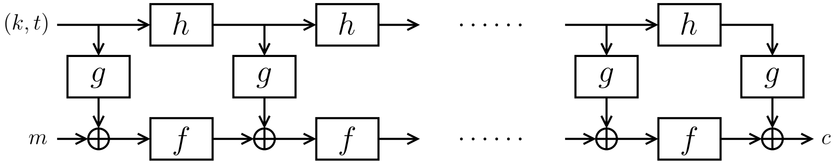
- TWEAKEY by Jean et al. [JNP14]:



- f: round function
- g: subkey computation
- h: transformation of (k, t)

TWEAKEY Framework

- TWEAKEY by Jean et al. [JNP14]:



- f: round function
- g: subkey computation
- h: transformation of (k, t)

- Security measured through cryptanalysis
- Our focus: modular design

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

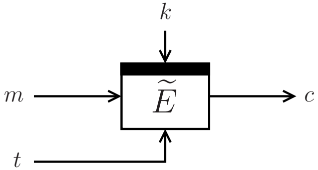
Beyond Masking-Based Tweakable Blockciphers

Conclusion

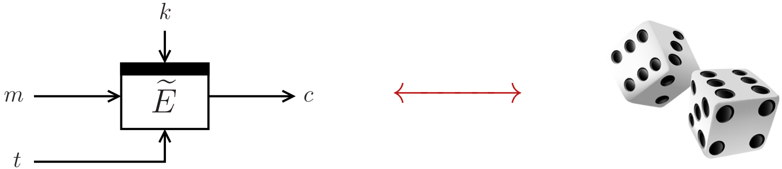
Basic Generic Recipe

1. Determine appropriate security model
2. Design the scheme
3. Perform security analysis

Basic Generic Recipe Step 1: Security Model



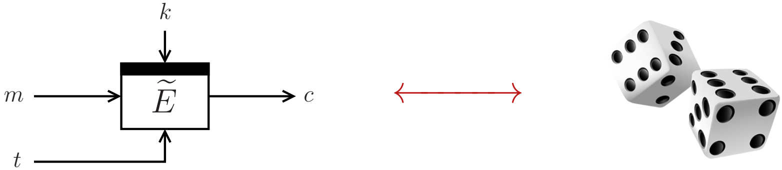
Basic Generic Recipe Step 1: Security Model



Tweakable Pseudorandom Permutation Security

- $\widetilde{E}_k$ should look like random permutation for every t
- Different tweaks $\longrightarrow$ pseudo-independent permutations

Basic Generic Recipe Step 1: Security Model



Tweakable Pseudorandom Permutation Security

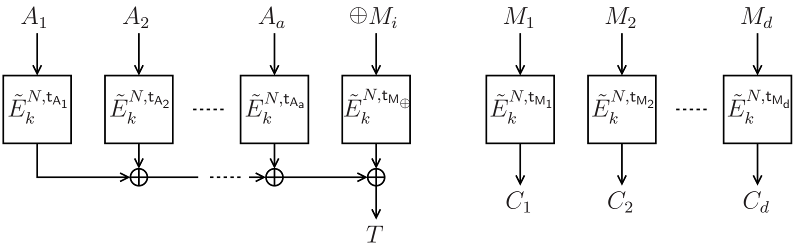
- $\widetilde{E}_k$ should look like random permutation for every t
- Different tweaks $\longrightarrow$ pseudo-independent permutations

Strong Tweakable Pseudorandom Permutation Security

- Adversary may have encryption and decryption access to $\widetilde{E}$

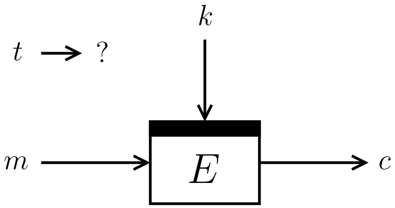
Basic Generic Recipe Step 1: Security Model

Example



- Tag generation: $\widetilde{E}_k$ evaluated in forward direction only
- Encryption/decryption: $\widetilde{E}_k$ evaluated in both directions

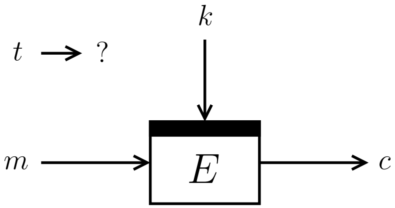
Basic Generic Recipe Step 2: Playground



- Consider a blockcipher E with κ-bit key and n-bit state

How to mingle the tweak into the evaluation?

Basic Generic Recipe Step 2: Playground



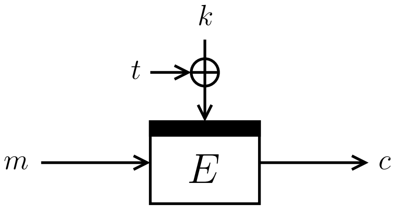
- Consider a blockcipher E with κ-bit key and n-bit state

How to mingle the tweak into the evaluation?

blend it with the key

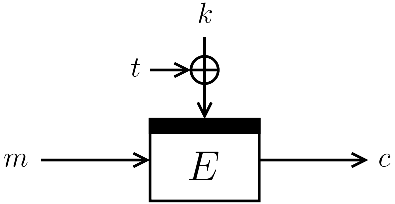
blend it with the state

Basic Generic Recipe Step 2: Playground



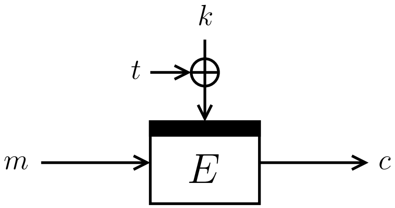
- Blending tweak and key works. . .
- . . . but: careful with related-key attacks!

Basic Generic Recipe Step 2: Playground



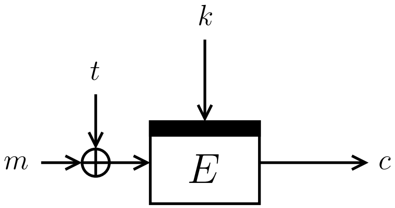
- Blending tweak and key works...
- ... but: careful with related-key attacks!
- For $\oplus$-mixing, key can be recovered in $2^{\kappa/2}$ evaluations
- Scheme is insecure if E is Even-Mansour

Basic Generic Recipe Step 2: Playground



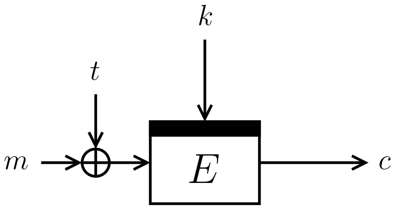
- Blending tweak and key works...
- ... but: careful with related-key attacks!
- For $\oplus$-mixing, key can be recovered in $2^{\kappa/2}$ evaluations
- Scheme is insecure if E is Even-Mansour
- TWEAKEY blending is more advanced

Basic Generic Recipe Step 2: Playground



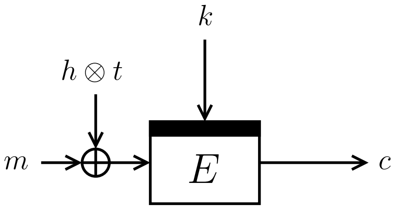
- Simple blending of tweak and state does not work

Basic Generic Recipe Step 2: Playground



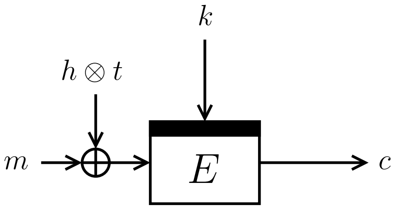
- Simple blending of tweak and state does not work
  - $\widetilde{E}_k(t, m) = \widetilde{E}_k(t \oplus C, m \oplus C)$

Basic Generic Recipe Step 2: Playground



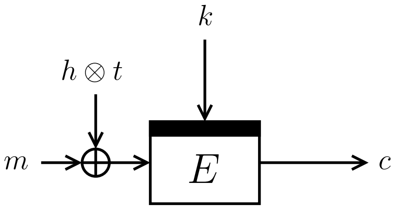
- Simple blending of tweak and state does not work
    - $\widetilde{E}_k(t, m) = \widetilde{E}_k(t \oplus C, m \oplus C)$
- Some secrecy required: h

Basic Generic Recipe Step 2: Playground



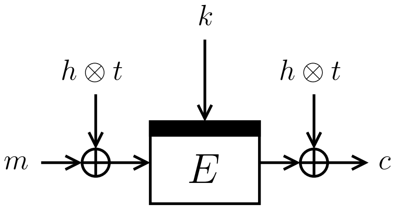
- Simple blending of tweak and state does not work
  - $\widetilde{E}_k(t, m) = \widetilde{E}_k(t \oplus C, m \oplus C)$
- Some secrecy required: h
- Still does not work if adversary has access to $\widetilde{E}_k^{-1}$

Basic Generic Recipe Step 2: Playground



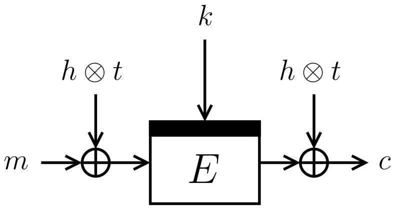
- Simple blending of tweak and state does not work
  - $\widetilde{E}_k(t, m) = \widetilde{E}_k(t \oplus C, m \oplus C)$
- Some secrecy required: h
- Still does not work if adversary has access to $\widetilde{E}_k^{-1}$
  - $\widetilde{E}_k^{-1}(t, c) \oplus \widetilde{E}_k^{-1}(t \oplus C, c) = h \otimes C$

Basic Generic Recipe Step 2: Playground



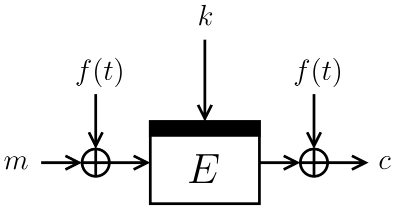
- Simple blending of tweak and state does not work
  - $\widetilde{E}_k(t, m) = \widetilde{E}_k(t \oplus C, m \oplus C)$
- Some secrecy required: h
- Still does not work if adversary has access to $\widetilde{E}_k^{-1}$
  - $\widetilde{E}_k^{-1}(t, c) \oplus \widetilde{E}_k^{-1}(t \oplus C, c) = h \otimes C$
  - Two-sided masking necessary

Basic Generic Recipe Step 2: Playground



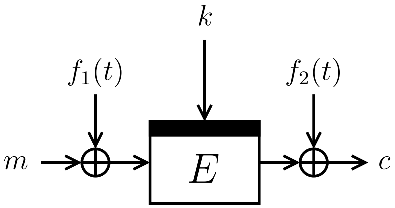
- Two-sided secret masking seems to work
- Can we generalize?

Basic Generic Recipe Step 2: Playground



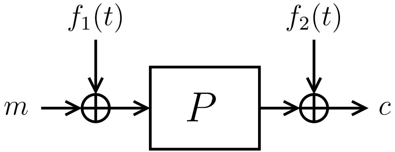
- Two-sided secret masking seems to work
- Can we generalize?
- Generalizing masking? Depends on function f

Basic Generic Recipe Step 2: Playground



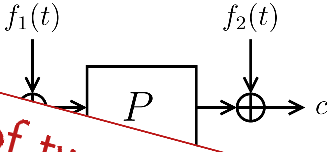
- Two-sided secret masking seems to work
- Can we generalize?
- Generalizing masking? Depends on function f
- Variation in masking? Depends on functions f_1, f_2

Basic Generic Recipe Step 2: Playground



- Two-sided secret masking seems to work
- Can we generalize?
- Generalizing masking? Depends on function f
- Variation in masking? Depends on functions f_1, f_2
- Releasing secrecy in E? Usually no problem

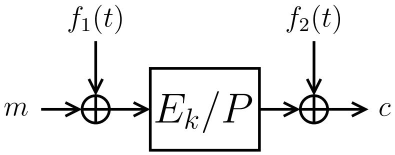
Basic Generic Recipe Step 2: Playground



Majority of tweakable blockciphers follow mask-E_k/P-mask principle

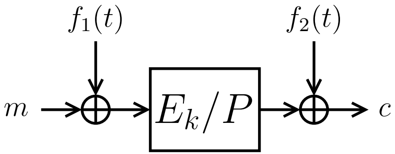
- Two-sided sec
- Can we generalize?
- Generalizing masking? Depends on func
- Variation in masking? Depends on functions f_1, f_2
- Releasing secrecy in E? Usually no problem

Basic Generic Recipe Step 3: Analysis



- $\widetilde{E}_k$ should "look like" random permutation for every t
- Consider adversary $\mathcal{A}$ that makes q evaluations of $\widetilde{E}_k$

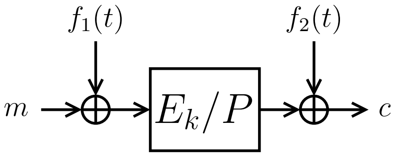
Basic Generic Recipe Step 3: Analysis



- $\widetilde{E}_k$ should "look like" random permutation for every t
- Consider adversary $\mathcal{A}$ that makes q evaluations of $\widetilde{E}_k$

- Step 3a:
  - How many evaluations does $\mathcal{A}$ need at most?
  - Boils down to finding generic attacks

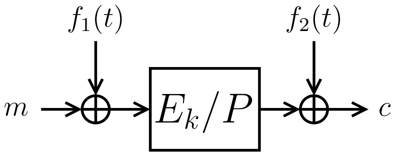
Basic Generic Recipe Step 3: Analysis



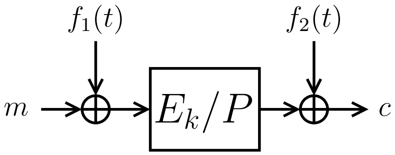
- $\widetilde{E}_k$ should "look like" random permutation for every t
- Consider adversary $\mathcal{A}$ that makes q evaluations of $\widetilde{E}_k$

- Step 3a:
  - How many evaluations does $\mathcal{A}$ need at most?
  - Boils down to finding generic attacks
- Step 3b:
  - How many evaluations does $\mathcal{A}$ need at least?
  - Boils down to provable security

Basic Generic Recipe Step 3a: Generic Attack



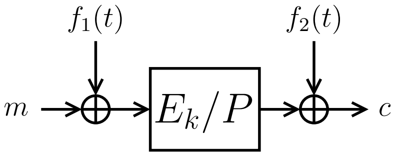
Basic Generic Recipe Step 3a: Generic Attack



- For any two queries (t, m, c), (t', m', c'):

$$m \oplus f_1(t) = m' \oplus f_1(t') \Longrightarrow c \oplus f_2(t) = c' \oplus f_2(t')$$

Basic Generic Recipe Step 3a: Generic Attack

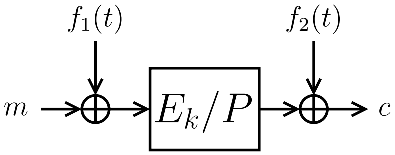


- For any two queries (t, m, c), (t', m', c'):

$$m \oplus f_1(t) = m' \oplus f_1(t') \Longrightarrow c \oplus f_2(t) = c' \oplus f_2(t')$$

- Unlikely to happen for random family of permutations

Basic Generic Recipe Step 3a: Generic Attack

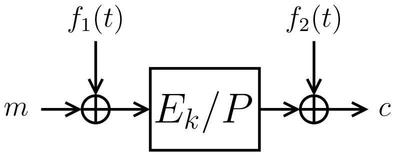


- For any two queries (t, m, c), (t', m', c'):

$$m \oplus f_1(t) = m' \oplus f_1(t') \implies c \oplus f_2(t) = c' \oplus f_2(t')$$

- Unlikely to happen for random family of permutations
- Implication still holds with difference C xored to m, m'

Basic Generic Recipe Step 3a: Generic Attack



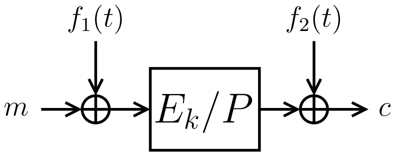
- For any two queries (t, m, c), (t', m', c'):

$$m \oplus f_1(t) = m' \oplus f_1(t') \Longrightarrow c \oplus f_2(t) = c' \oplus f_2(t')$$

- Unlikely to happen for random family of permutations
- Implication still holds with difference C xored to m, m'

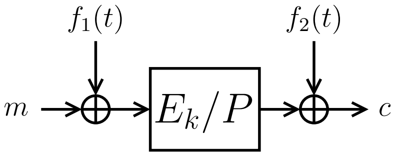
Scheme can be broken in $\approx 2^{n/2}$ evaluations

Basic Generic Recipe Step 3b: Security Proof



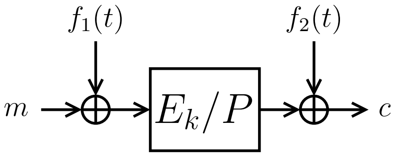
- The fun starts here!
- More technical and often more involved

Basic Generic Recipe Step 3b: Security Proof



- The fun starts here!
- More technical and often more involved
- Typical approach:
  - Consider any transcript τ an adversary may see
  - Most τ's should be equally likely in both worlds
  - Odd ones should happen with very small probability

Basic Generic Recipe Step 3b: Security Proof



- The fun starts here!
- More technical and often more involved
- Typical approach:
  - Consider any transcript τ an adversary may see
  - Most τ's should be equally likely in both worlds
  - Odd ones should happen with very small probability

All constructions in this presentation: secure up to $\approx 2^{n/2}$ evaluations

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

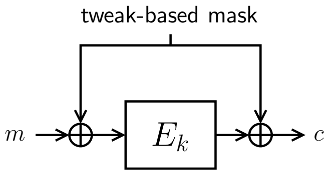
- State of the Art
- Improved Efficiency
- Improved Security

Beyond Masking-Based Tweakable Blockciphers

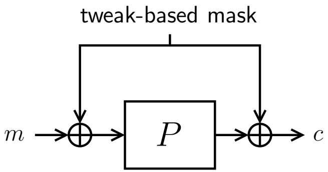
Conclusion

Tweakable Blockciphers Based on Masking

Blockcipher-Based

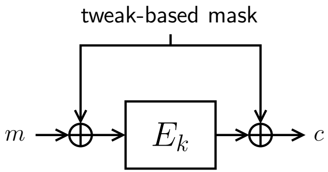


Permutation-Based



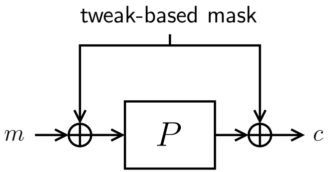
Tweakable Blockciphers Based on Masking

Blockcipher-Based



typically 128 bits

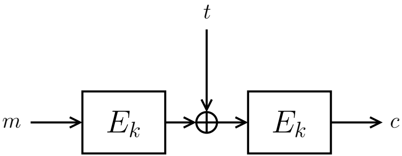
Permutation-Based

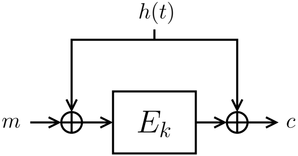


much larger: 256-1600 bits

Original Constructions

- LRW_1 and LRW_2 by Liskov et al. [LRW02]:

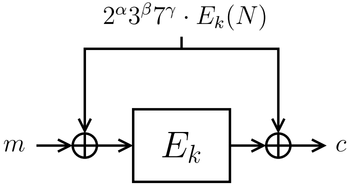




- h is XOR-universal hash
  - E.g., $h(t) = h \otimes t$ for n-bit "key" h

Powering-Up Masking (XEX)

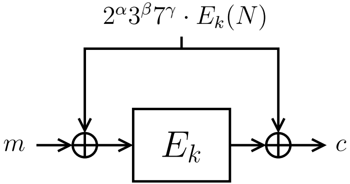
- XEX by Rogaway [Rog04]:



- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)

Powering-Up Masking (XEX)

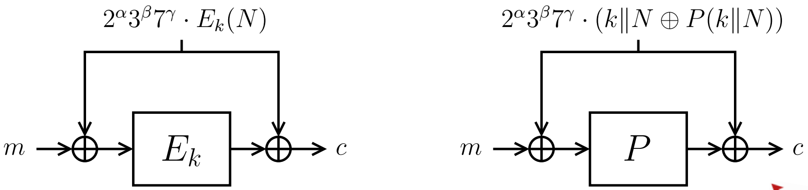
- XEX by Rogaway [Rog04]:



- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Used in OCB2, $\pm$14 CAESAR candidates, and XTS

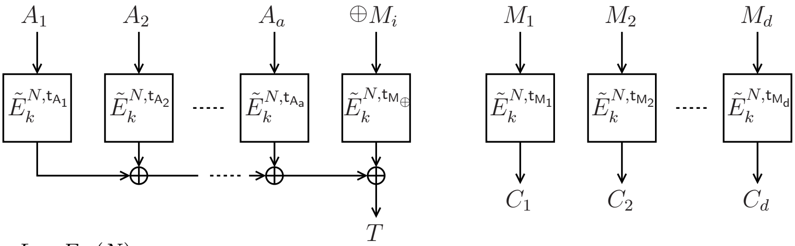
Powering-Up Masking (XEX)

- XEX by Rogaway [Rog04]:



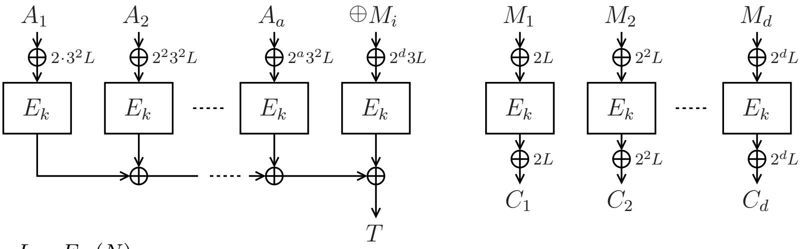
- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Used in OCB2, $\pm$14 CAESAR candidates, and XTS
- Permutation-based variants in Minalpher and Prøst (generalized by Cogliati et al. [CLS15])

Powering-Up Masking in OCB2



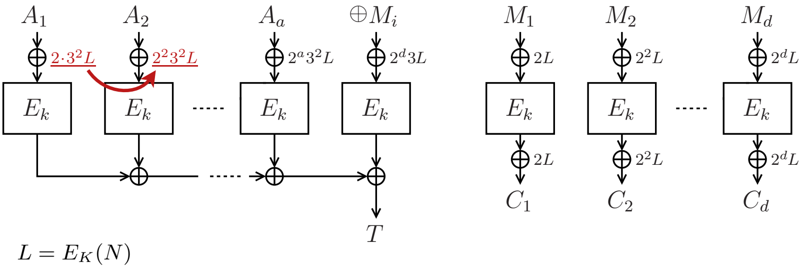
$L = E_K(N)$

Powering-Up Masking in OCB2

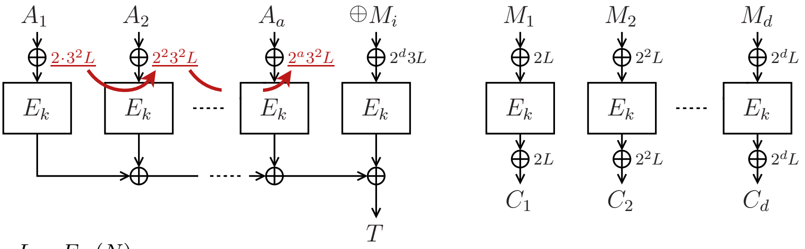


$L = E_K(N)$

Powering-Up Masking in OCB2

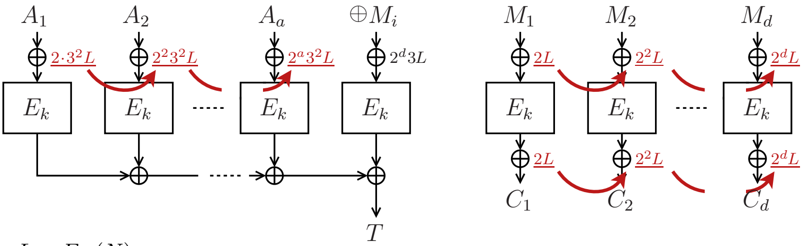


Powering-Up Masking in OCB2



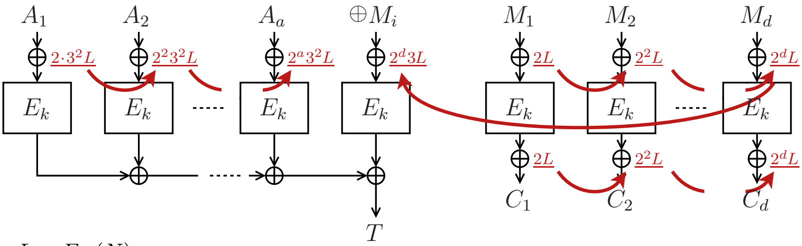
$L = E_K(N)$

Powering-Up Masking in OCB2



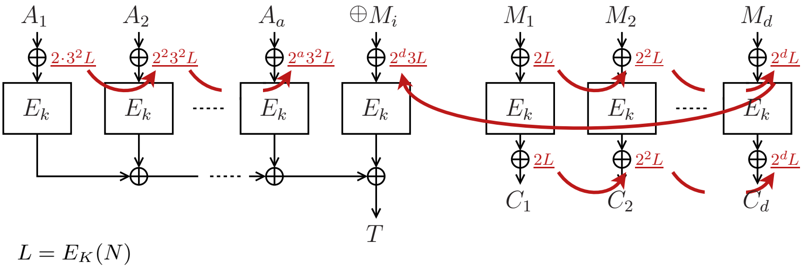
$L = E_K(N)$

Powering-Up Masking in OCB2



$L = E_K(N)$

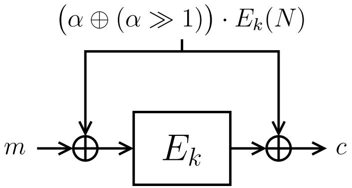
Powering-Up Masking in OCB2



- Update of mask:
  - Shift and conditional XOR
- Variable time computation
- Expensive on certain platforms

Gray Code Masking

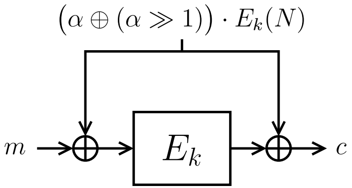
- OCB1 and OCB3 use Gray Codes:



- (α, N) is tweak
- Updating: $G(\alpha) = G(\alpha - 1) \oplus 2^{\mathsf{ntz}(\alpha)}$

Gray Code Masking

- OCB1 and OCB3 use Gray Codes:



- (α, N) is tweak
- Updating: $G(\alpha) = G(\alpha - 1) \oplus 2^{\mathsf{ntz}(\alpha)}$
  - Single XOR
  - Logarithmic amount of field doublings (precomputed)
- More efficient than powering-up [KR11]

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

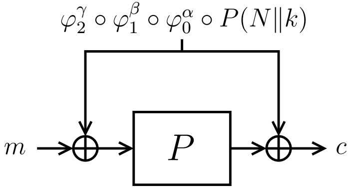
- State of the Art
- Improved Efficiency
- Improved Security

Beyond Masking-Based Tweakable Blockciphers

Conclusion

Masked Even-Mansour (MEM)

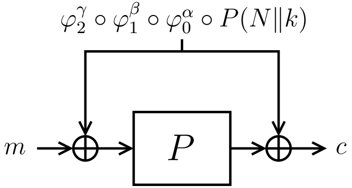
- MEM by Granger et al. [GJMN16]:



- φ_i are fixed LFSRs, $(\alpha, \beta, \gamma, N)$ is tweak (simplified)

Masked Even-Mansour (MEM)

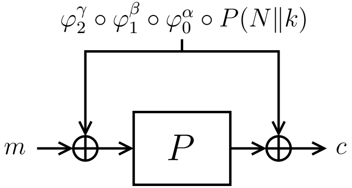
- MEM by Granger et al. [GJMN16]:



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

Masked Even-Mansour (MEM)

- MEM by Granger et al. [GJMN16]:



- φ_i are fixed LFSRs, $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Combines advantages of:
  - Powering-up masking
  - Word-based LFSRs
- Simpler, constant-time (by default), more efficient

MEM: Design Considerations

- Particularly suited for large states (permutations)
- Low operation counts by clever choice of LFSR

MEM: Design Considerations

- Particularly suited for large states (permutations)
- Low operation counts by clever choice of LFSR
- Sample LFSRs (state size b as n words of w bits):

b	w	n	φ
128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 1) \oplus (x_9 \gg 1) \oplus (x_{10} \ll 1))$
128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
128	64	2	$(x_1, (x_0 \lll 11) \oplus x_1 \oplus (x_1 \ll 13))$
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	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

MEM: Design Considerations

- Particularly suited for large states (permutations)
- Low operation counts by clever choice of LFSR
- Sample LFSRs (state size b as n words of w bits):

b	w	n	φ
128	8	16	$(x_1, \ldots, x_{15}, (x_0 \lll 1) \oplus (x_9 \gg 1) \oplus (x_{10} \ll 1))$
128	32	4	$(x_1, \ldots, x_3, (x_0 \lll 5) \oplus x_1 \oplus (x_1 \ll 13))$
128	64	2	$(x_1, (x_0 \lll 11) \oplus x_1 \oplus (x_1 \ll 13))$
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	$(x_1, \ldots, x_7, (x_0 \lll 29) \oplus (x_1 \ll 9))$
1024	64	16	$(x_1, \ldots, x_{15}, (x_0 \lll 53) \oplus (x_5 \ll 13))$
1600	32	50	$(x_1, \ldots, x_{49}, (x_0 \lll 3) \oplus (x_{23} \gg 3))$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Work exceptionally well for ARX primitives

MEM: Uniqueness of Masking

- Intuitively, masking goes well as long as

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

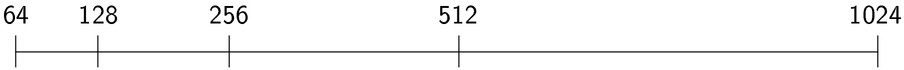
  for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
- Challenge: set proper domain for (α, β, γ)
- Requires computation of discrete logarithms

MEM: 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

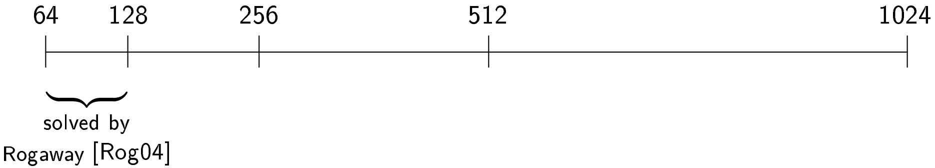


MEM: Uniqueness of Masking

- 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

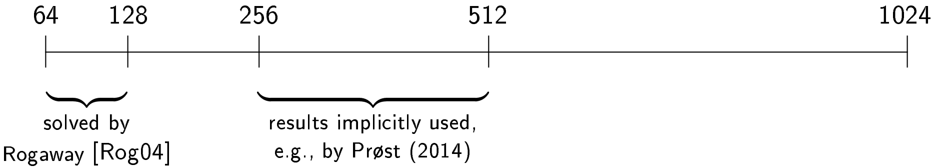


MEM: Uniqueness of Masking

- 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

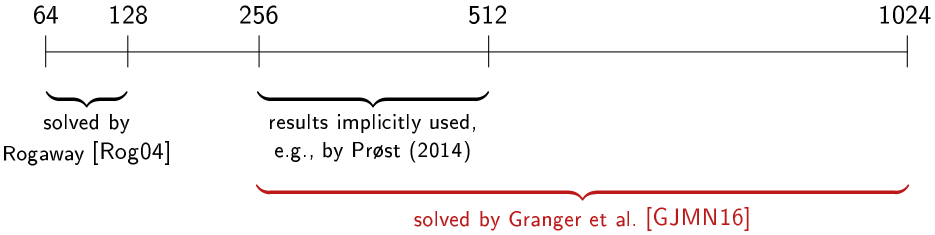


MEM: Uniqueness of Masking

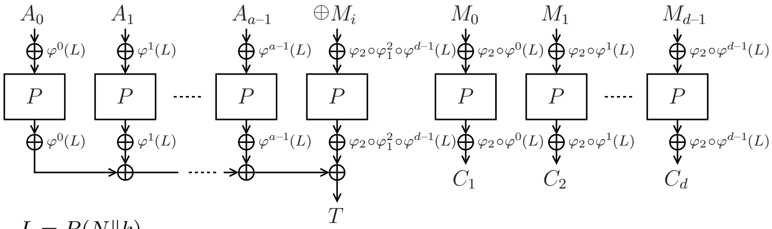
- Intuitively, masking goes well as long as

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

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



Application to AE: OPP

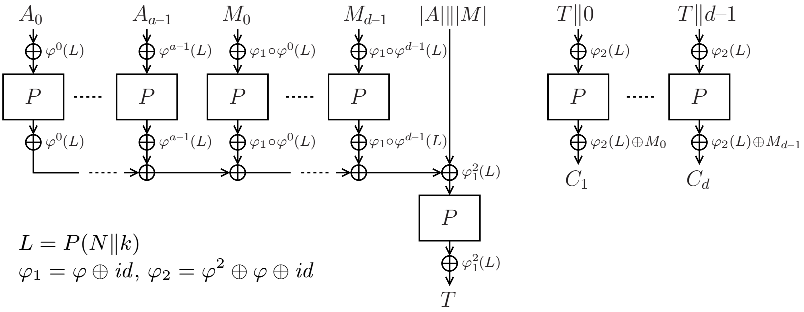


$L = P(N \| k)$

$\varphi_1 = \varphi \oplus id$, $\varphi_2 = \varphi^2 \oplus \varphi \oplus id$

- Offset Public Permutation (OPP)
- Generalization of OCB3:
  - Permutation-based
  - More efficient MEM masking
- Security against nonce-respecting adversaries
- 0.55 cpb with reduced-round BLAKE2b

Application to AE: MRO



- Misuse-Resistant OPP (MRO)
- Fully nonce-misuse resistant version of OPP
- 1.06 cpb with reduced-round BLAKE2b

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

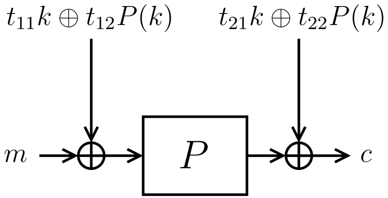
- State of the Art
- Improved Efficiency
- Improved Security

Beyond Masking-Based Tweakable Blockciphers

Conclusion

XPX

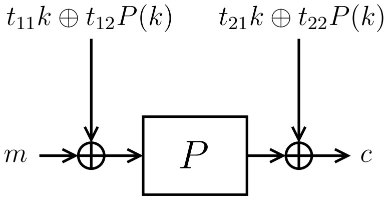
- XPX by Mennink [Men16]:



- $(t_{11}, t_{12}, t_{21}, t_{22})$ from some tweak set $\mathcal{T} \subseteq (\{0,1\}^n)^4$
- $\mathcal{T}$ can (still) be any set

XPX

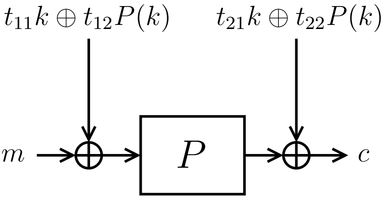
- XPX by Mennink [Men16]:



- $(t_{11}, t_{12}, t_{21}, t_{22})$ from some tweak set $\mathcal{T} \subseteq (\{0,1\}^n)^4$
- $\mathcal{T}$ can (still) be any set
- Security of XPX strongly depends on choice of $\mathcal{T}$

XPX

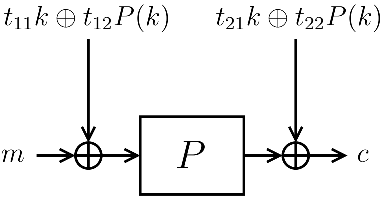
- XPX by Mennink [Men16]:



- $(t_{11}, t_{12}, t_{21}, t_{22})$ from some tweak set $\mathcal{T} \subseteq (\{0,1\}^n)^4$
- $\mathcal{T}$ can (still) be any set
- Security of XPX strongly depends on choice of $\mathcal{T}$
    1. "Weak" $\mathcal{T} \quad \longrightarrow \quad$ insecure

XPX

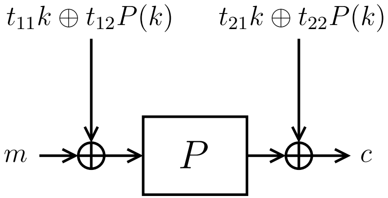
- XPX by Mennink [Men16]:



- $(t_{11}, t_{12}, t_{21}, t_{22})$ from some tweak set $\mathcal{T} \subseteq (\{0,1\}^n)^4$
- $\mathcal{T}$ can (still) be any set
- Security of XPX strongly depends on choice of $\mathcal{T}$
  1. "Weak" $\mathcal{T}$ $\longrightarrow$ insecure
  2. "Normal" $\mathcal{T}$ $\longrightarrow$ single-key secure

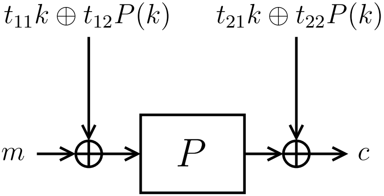
XPX

- XPX by Mennink [Men16]:

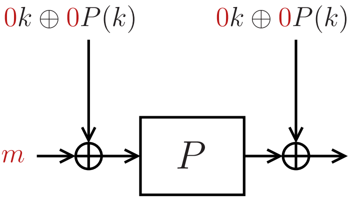


- $(t_{11}, t_{12}, t_{21}, t_{22})$ from some tweak set $\mathcal{T} \subseteq (\{0,1\}^n)^4$
- $\mathcal{T}$ can (still) be any set
- Security of XPX strongly depends on choice of $\mathcal{T}$
  1. "Weak" $\mathcal{T}$ $\longrightarrow$ insecure
  2. "Normal" $\mathcal{T}$ $\longrightarrow$ single-key secure
  3. "Strong" $\mathcal{T}$ $\longrightarrow$ related-key secure

XPX: Weak Tweaks

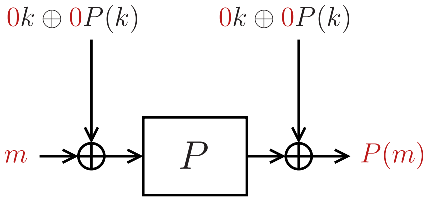


XPX: Weak Tweaks



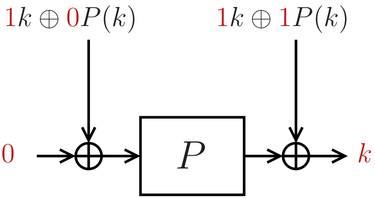
$(0, 0, 0, 0) \in \mathcal{T}$

XPX: Weak Tweaks



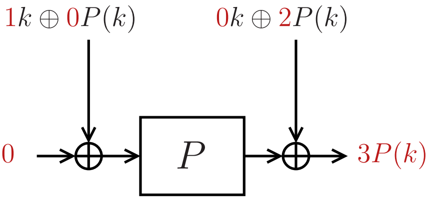
$$(0,0,0,0) \in \mathcal{T} \quad \Longrightarrow \quad \mathsf{XPX}_k((0,0,0,0),m) = P(m)$$

XPX: Weak Tweaks



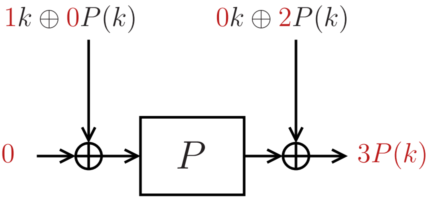
$$
\begin{aligned}
(0,0,0,0) \in \mathcal{T} \quad &\Longrightarrow \quad \mathsf{XPX}_k((0,0,0,0),m) = P(m) \\
(1,0,1,1) \in \mathcal{T} \quad &\Longrightarrow \quad \mathsf{XPX}_k((1,0,1,1),0) = k
\end{aligned}
$$

XPX: Weak Tweaks



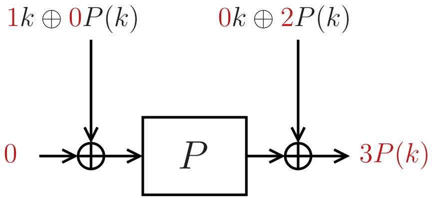
$$
\begin{aligned}
(0,0,0,0) \in \mathcal{T} \quad &\Longrightarrow \quad \mathsf{XPX}_k((0,0,0,0),m) = P(m) \\
(1,0,1,1) \in \mathcal{T} \quad &\Longrightarrow \quad \mathsf{XPX}_k((1,0,1,1),0) = k \\
(1,0,0,2) \in \mathcal{T} \quad &\Longrightarrow \quad \mathsf{XPX}_k((1,0,0,2),0) = 3P(k)
\end{aligned}
$$

XPX: Weak Tweaks



$$
\begin{array}{ccl}
(0,0,0,0) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((0,0,0,0),m) = P(m) \\
(1,0,1,1) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((1,0,1,1),0) = k \\
(1,0,0,2) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((1,0,0,2),0) = 3P(k) \\
\cdots & \cdots & \cdots
\end{array}
$$

XPX: Weak Tweaks

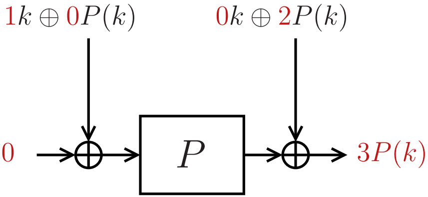


$$
\begin{array}{lcl}
(0,0,0,0) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((0,0,0,0), m) = P(m) \\
(1,0,1,1) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((1,0,1,1), 0) = k \\
(1,0,0,2) \in \mathcal{T} & \Longrightarrow & \mathsf{XPX}_k((1,0,0,2), 0) = 3P(k) \\
\ldots & \ldots & \ldots
\end{array}
$$

"Valid" Tweak Sets

- Technical definition to eliminate weak cases

XPX: Weak Tweaks



$$
\begin{array}{ccc}
(0,0,0,0) \in \mathcal{T} & \implies & \mathsf{XPX}_k((0,0,0,0),m) = P(m) \\
(1,0,1,1) \in \mathcal{T} & \implies & \mathsf{XPX}_k((1,0,1,1),0) = k \\
(1,0,0,2) \in \mathcal{T} & \implies & \mathsf{XPX}_k((1,0,0,2),0) = 3P(k) \\
\dots & \dots & \dots
\end{array}
$$

"Valid" Tweak Sets

- Technical definition to eliminate weak cases
- $\mathcal{T}$ invalid $\iff$ XPX insecure
- $\mathcal{T}$ valid $\iff$ XPX single- or related-key secure

XPX Covers Even-Mansour

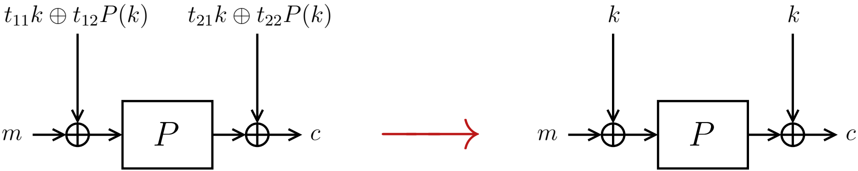
for $\mathcal{T} = \{(1, 0, 1, 0)\}$

XPX Covers Even-Mansour

for $\mathcal{T} = \{(1,0,1,0)\}$

- Single-key STPRP secure (surprise?)

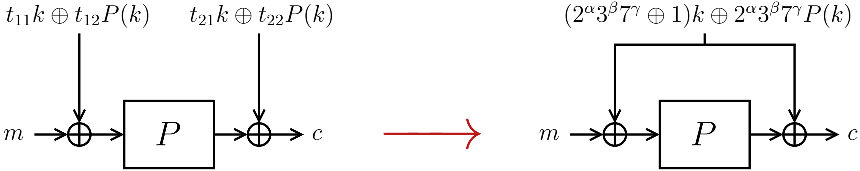
XPX Covers Even-Mansour



for $\mathcal{T} = \{(1,0,1,0)\}$

- Single-key STPRP secure (surprise?)
- Generally, if $|\mathcal{T}| = 1$, XPX is a normal blockcipher

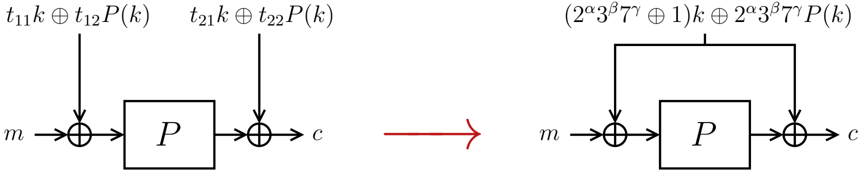
XPX Covers XEX With Even-Mansour



for $\mathcal{T} = \left\{ \begin{array}{l} (2^\alpha 3^\beta 7^\gamma \oplus 1, 2^\alpha 3^\beta 7^\gamma, \\ \;\; 2^\alpha 3^\beta 7^\gamma \oplus 1, 2^\alpha 3^\beta 7^\gamma) \end{array} \;\middle|\; (\alpha, \beta, \gamma) \in \{\text{XEX-tweaks}\} \right\}$

- (α, β, γ) is in fact the "real" tweak

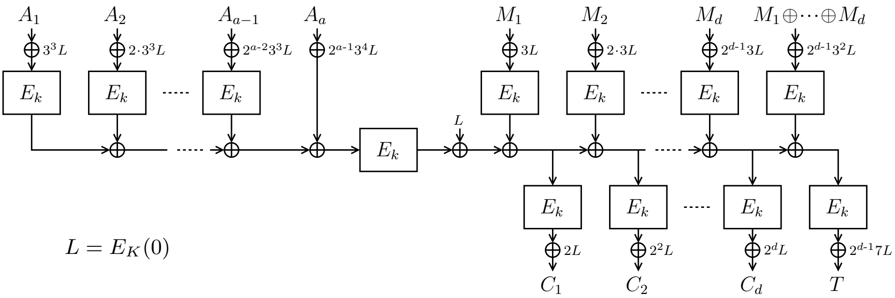
XPX Covers XEX With Even-Mansour



$$\text{for } \mathcal{T} = \left\{ \begin{array}{l} (\, 2^\alpha 3^\beta 7^\gamma \oplus 1 \,,\, 2^\alpha 3^\beta 7^\gamma \,, \\ \;\; 2^\alpha 3^\beta 7^\gamma \oplus 1 \,,\, 2^\alpha 3^\beta 7^\gamma \,) \end{array} \;\middle|\; (\alpha, \beta, \gamma) \in \{\text{XEX-tweaks}\} \right\}$$

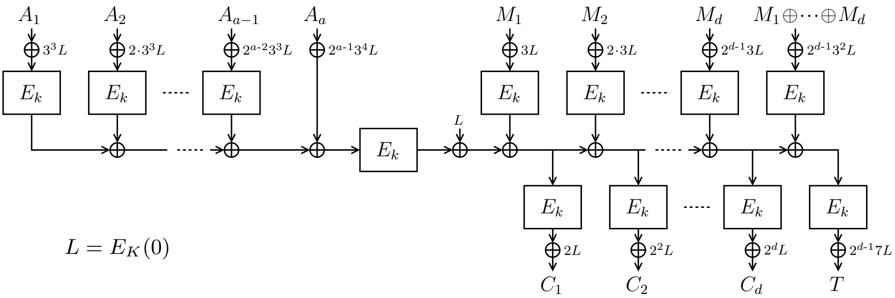
- (α, β, γ) is in fact the "real" tweak
- Related-key STPRP secure (if $2^\alpha 3^\beta 7^\gamma \neq 1$)

Application to AE: COPA and Prøst-COPA



- By Andreeva et al. (2014)
- Implicitly based on XEX based on AES

Application to AE: COPA and Prøst-COPA



- By Andreeva et al. (2014)
- Implicitly based on XEX based on AES

- Prøst-COPA by Kavun et al. (2014):
  COPA based on XEX based on Even-Mansour

Application to AE: COPA and Prøst-COPA

Single-Key Security of COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \qquad \boxed{P}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Related-Key Security of COPA

- Existing proof generalizes

$$\boxed{\text{COPA}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Related-Key Security of Prøst-COPA

- Existing proof generalizes

$$\boxed{\text{COPA}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \qquad \boxed{P}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

$$\boxed{\text{COPA}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Related-Key Security of Prøst-COPA

- Existing proof generalizes

$$\boxed{\text{COPA}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XEX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{E} \xrightarrow[\text{rk}]{\Omega\left(1\right)} \boxed{P}$$

Application to AE: COPA and Prøst-COPA

Single-Key Security of Prøst-COPA

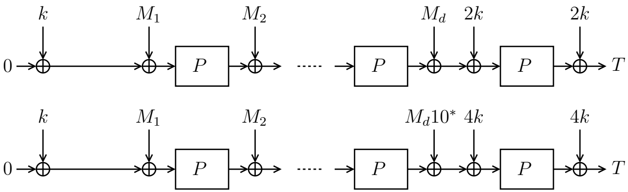
$$\begin{tikzcd} \boxed{\text{COPA}} \arrow[r, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)", "\text{sk}"'] & \boxed{\text{XEX}} \arrow[r, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)", "\text{sk}"'] & \boxed{E} \arrow[r, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)", "\text{sk}"'] & \boxed{P} \end{tikzcd}$$

Related-Key Security of Prøst-COPA

- Existing proof generalizes

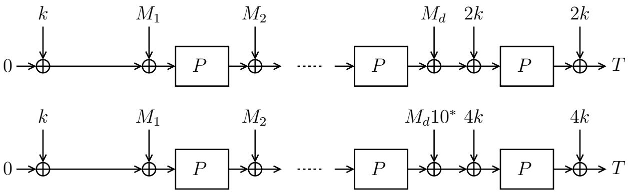
$$\begin{tikzcd} \boxed{\text{COPA}} \arrow[r, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)", "\text{rk}"'] & \boxed{\text{XEX}} \arrow[r, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)", "\text{rk}"'] \arrow[rr, bend right, "\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)" description, "\text{rk}"'] & \boxed{E} \arrow[r, "\Omega\left(1\right)", "\text{rk}"'] & \boxed{P} \end{tikzcd}$$

Application to MAC: Chaskey



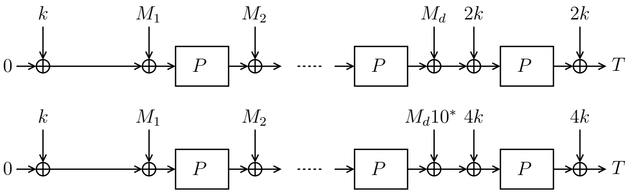
- By Mouha et al. (2014)
- Original proof based on 3 EM's: $\begin{cases} E_k(m) = P(m \oplus k) \oplus k \\ E_k(m) = P(m \oplus 3k) \oplus 2k \\ E_k(m) = P(m \oplus 5k) \oplus 4k \end{cases}$

Application to MAC: Chaskey



- By Mouha et al. (2014)
- Original proof based on 3 EM's: $\begin{cases} E_k(m) = P(m \oplus k) \oplus k \\ E_k(m) = P(m \oplus 3k) \oplus 2k \\ E_k(m) = P(m \oplus 5k) \oplus 4k \end{cases}$
- Equivalent to XPX with $\mathcal{T} = \{(1,0,1,0), (3,0,2,0), (5,0,4,0)\}$

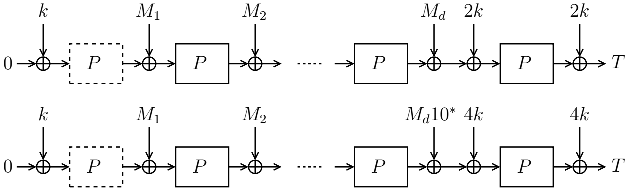
Application to MAC: Chaskey



- By Mouha et al. (2014)
- Original proof based on 3 EM's: $\begin{cases} E_k(m) = P(m \oplus k) \oplus k \\ E_k(m) = P(m \oplus 3k) \oplus 2k \\ E_k(m) = P(m \oplus 5k) \oplus 4k \end{cases}$
- Equivalent to XPX with $\mathcal{T} = \{(1,0,1,0),(3,0,2,0),(5,0,4,0)\}$

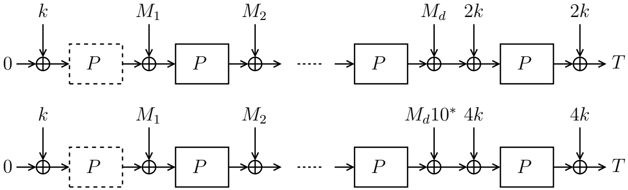
$$\boxed{\text{Chaskey}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XPX}} \xrightarrow[\text{sk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Application to MAC: Adjusted Chaskey



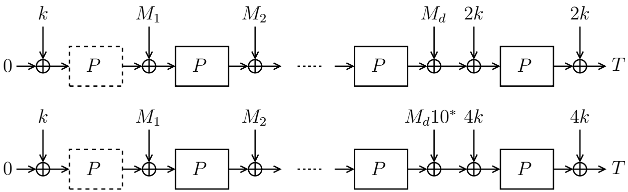
- Extra P-call

Application to MAC: Adjusted Chaskey



- Extra P-call
- Based on XPX with $\mathcal{T}' = \{(0,1,0,1),(2,1,2,0),(4,1,4,0)\}$

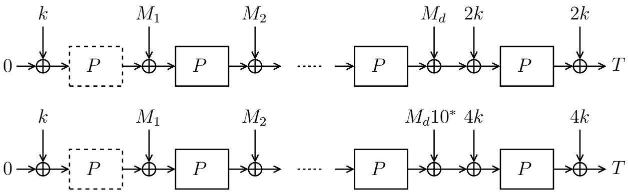
Application to MAC: Adjusted Chaskey



- Extra P-call
- Based on XPX with $\mathcal{T}' = \{(0,1,0,1),(2,1,2,0),(4,1,4,0)\}$

$$\boxed{\text{Chaskey}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XPX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

Application to MAC: Adjusted Chaskey



- Extra P-call
- Based on XPX with $\mathcal{T}' = \{(0,1,0,1),(2,1,2,0),(4,1,4,0)\}$

$$\boxed{\text{Chaskey}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XPX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$

- Approach can also be applied to:
  - Keyed Sponge and Duplex
  - 10 Sponge-inspired CAESAR candidates

Outline

Dedicated Design

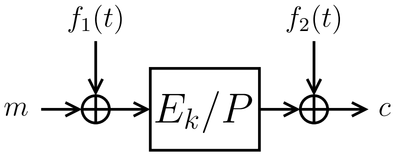
Basic Generic Recipe

Tweakable Blockciphers Based on Masking

Beyond Masking-Based Tweakable Blockciphers

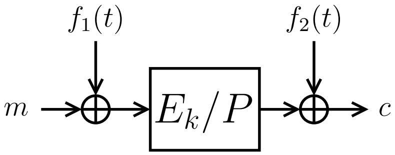
Conclusion

Beyond Masking-Based Tweakable Blockciphers



- "Birthday-bound" $2^{n/2}$ security at best
- Overlying modes inherit security bound

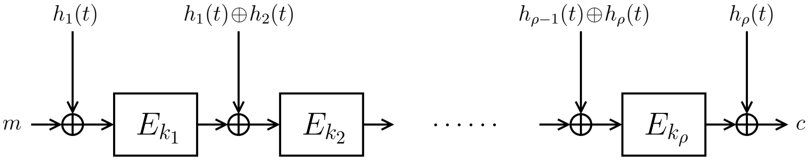
Beyond Masking-Based Tweakable Blockciphers



- "Birthday-bound" $2^{n/2}$ security at best
- Overlying modes inherit security bound

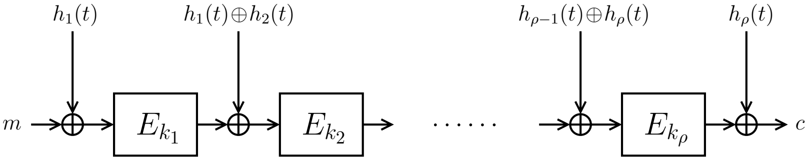
- If n is large enough $\longrightarrow$ no problem
- If n is small $\longrightarrow$ "beyond birthday-bound" solutions
  - Cascading
  - Tweak-rekeying

Cascading LRW's



- $\mathsf{LRW}_2[\rho]$: concatenation of ρ LRW_2's
- $k_1, \ldots, k_\rho$ and $h_1, \ldots, h_\rho$ independent

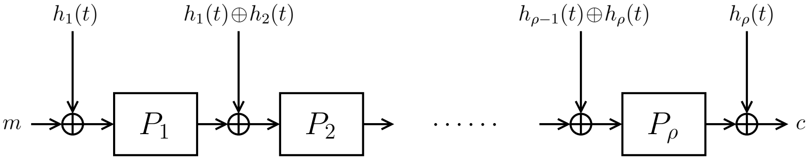
Cascading LRW's



- LRW$_2[\rho]$: concatenation of ρ LRW$_2$'s
- $k_1, \ldots, k_\rho$ and $h_1, \ldots, h_\rho$ independent

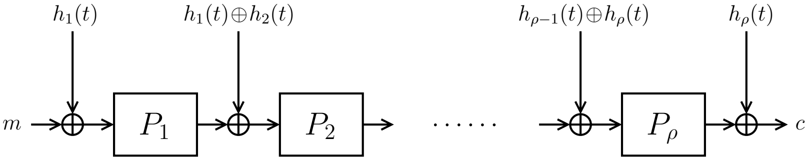
- $\rho = 2$: secure up to $2^{2n/3}$ queries [LST12,Pro14]
- $\rho \geq 2$ even: secure up to $2^{\rho n/(\rho+2)}$ queries [LS13]
- Conjecture: optimal $2^{\rho n/(\rho+1)}$ security

Cascading TEM's



- TEM$[\rho]$: concatenation of ρ TEM's
- $P_1, \ldots, P_\rho$ and $h_1, \ldots, h_\rho$ independent

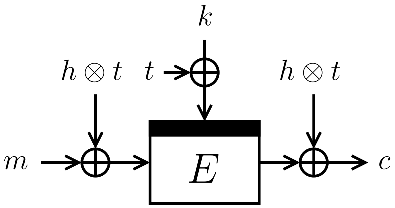
Cascading TEM's



- TEM$[\rho]$: concatenation of ρ TEM's
- $P_1, \ldots, P_\rho$ and $h_1, \ldots, h_\rho$ independent

- $\rho = 2$: secure up to $2^{2n/3}$ queries [CLS15]
- $\rho \geq 2$ even: secure up to $2^{\rho n/(\rho+2)}$ queries [CLS15]
- Conjecture: optimal $2^{\rho n/(\rho+1)}$ security

Tweak-Rekeying



- Mingling tweak into both key and state works
- Secure up to 2^n queries (in ICM!)
- Alternative constructions exist [Min09,Men15,WGZ+16]

More on “beyond birthday-bound security” on Thursday

Outline

Dedicated Design

Basic Generic Recipe

Tweakable Blockciphers Based on Masking

Beyond Masking-Based Tweakable Blockciphers

Conclusion

Conclusion

Tweakable Blockciphers: Simple and Powerful

- Myriad applications to AE, MAC, encryption, ...
- Choice of masking influences efficiency and security

Conclusion

Tweakable Blockciphers: Simple and Powerful

- Myriad applications to AE, MAC, encryption, ...
- Choice of masking influences efficiency and security

Security Level

- Birthday-bound security: okay if n is large enough
  $\longrightarrow$ Permutation-based tweakable blockciphers
- Beyond birthday-bound security possible
  $\longrightarrow$ More on Thursday

Thank you for your attention!

SUPPORTING SLIDES

MEM: Implementation

- State size $b = 1024$
- LFSR on 16 words of 64 bits:

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

- P: BLAKE2b permutation with 4 or 6 rounds

MEM: Implementation

- State size $b = 1024$
- LFSR on 16 words of 64 bits:

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

- P: BLAKE2b permutation with 4 or 6 rounds

- Main implementation results:

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

MEM: Implementation

- State size $b = 1024$
- LFSR on 16 words of 64 bits:

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

- P: BLAKE2b permutation with 4 or 6 rounds

- Main implementation results:

	nonce-respecting					misuse-resistant			
Platform	AES-GCM	OCB3	Deoxys$^{\neq}$	OPP$_4$	OPP$_6$	GCM-SIV	Deoxys$^{=}$	MRO$_4$	MRO$_6$
Cortex-A8	38.6	28.9	-	4.26	5.91	-	-	8.07	11.32
Sandy Bridge	2.55	0.98	1.29	1.24	1.91	-	$\approx$ 2.58	2.41	3.58
Haswell	1.03	0.69	0.96	0.55	0.75	1.17	$\approx$ 1.92	1.06	1.39

MEM: Parallelizability

- LFSR on 16 words of 64 bits:

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

MEM: Parallelizability

- 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

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

MEM: Parallelizability

- 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

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

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

MEM: Parallelizability

- 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

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

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

MEM: Parallelizability

- 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

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

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

MEM: Parallelizability

- 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

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

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

MEM: Parallelizability

- LFSR on 16 words of 64 bits:

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

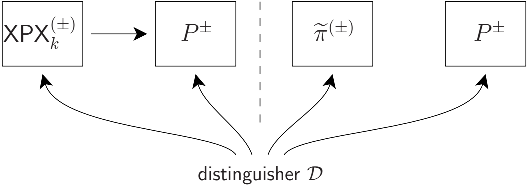
- Begin with state $L_i = [x_0, \ldots, x_{15}]$ of 64-bit words

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

- $x_{16} = (x_0 \lll 53) \oplus (x_5 \ll 13)$
- $x_{17} = (x_1 \lll 53) \oplus (x_6 \ll 13)$
- $x_{18} = (x_2 \lll 53) \oplus (x_7 \ll 13)$
- $x_{19} = (x_3 \lll 53) \oplus (x_8 \ll 13)$
- Parallelizable (AVX2) and word-sliceable

XPX: Single-Key Security

(Strong) Tweakable PRP

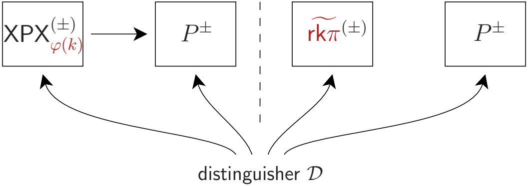


- Information-theoretic indistinguishability
  - $\widetilde{\pi}$ ideal tweakable permutation
  - P ideal permutation
  - k secret key

$$\mathcal{T} \text{ is valid} \quad \Longrightarrow \quad \text{XPX is (S)TPRP up to } \mathcal{O}\left(\frac{q^2 + qr}{2^n}\right)$$

XPX: Related-Key Security

Related-Key (Strong) Tweakable PRP



- Information-theoretic indistinguishability
  - $\widetilde{\mathsf{rk}\pi}$ ideal tweakable related-key permutation
  - P ideal permutation
  - k secret key

- $\mathcal{D}$ restricted to some set of key-deriving functions Φ

XPX: Related-Key Security

Key-Deriving Functions

- $\Phi_\oplus$: all functions $k \mapsto k \oplus \delta$

XPX: Related-Key Security

Key-Deriving Functions

- $\Phi_{\oplus}$: all functions $k \mapsto k \oplus \delta$
- $\Phi_{P\oplus}$: all functions $k \mapsto k \oplus \delta$ or $P(k) \mapsto P(k) \oplus \epsilon$

XPX: Related-Key Security

Key-Deriving Functions

- $\Phi_{\oplus}$: all functions $k \mapsto k \oplus \delta$
- $\Phi_{P\oplus}$: all functions $k \mapsto k \oplus \delta$ or $P(k) \mapsto P(k) \oplus \epsilon$
- Note: maskings in XPX are $t_{i1}k \oplus t_{i2}P(k)$

XPX: Related-Key Security

Key-Deriving Functions

- $\Phi_\oplus$: all functions $k \mapsto k \oplus \delta$
- $\Phi_{P\oplus}$: all functions $k \mapsto k \oplus \delta$ or $P(k) \mapsto P(k) \oplus \epsilon$
- Note: maskings in XPX are $t_{i1}k \oplus t_{i2}P(k)$

Results

if $\mathcal{T}$ is valid, and for all tweaks:	security	Φ
$t_{12} \neq 0$	TPRP	$\Phi_\oplus$
$t_{12}, t_{22} \neq 0$ and $(t_{21}, t_{22}) \neq (0, 1)$	STPRP	$\Phi_\oplus$

XPX: Related-Key Security

Key-Deriving Functions

- $\Phi_{\oplus}$: all functions $k \mapsto k \oplus \delta$
- $\Phi_{P\oplus}$: all functions $k \mapsto k \oplus \delta$ or $P(k) \mapsto P(k) \oplus \epsilon$
- Note: maskings in XPX are $t_{i1}k \oplus t_{i2}P(k)$

Results

if $\mathcal{T}$ is valid, and for all tweaks:	security	Φ
$t_{12} \neq 0$	TPRP	$\Phi_{\oplus}$
$t_{12}, t_{22} \neq 0$ and $(t_{21}, t_{22}) \neq (0, 1)$	STPRP	$\Phi_{\oplus}$
$t_{11}, t_{12} \neq 0$	TPRP	$\Phi_{P\oplus}$
$t_{11}, t_{12}, t_{21}, t_{22} \neq 0$	STPRP	$\Phi_{P\oplus}$

XPX: Security Proof Techniques

Patarin's H-coefficient Technique

- Each conversation defines a transcript
- Define good and bad transcripts

XPX: Security Proof Techniques

Patarin's H-coefficient Technique

- Each conversation defines a transcript
- Define good and bad transcripts

$$\mathbf{Adv}^{\text{rk-(s)prp}}_{\mathsf{XPX}}(\mathcal{D}) \leq \underbrace{\varepsilon}_{\text{prob. ratio for good transcripts}} + \mathbf{Pr}\left[\text{bad transcript for } (\widetilde{\mathsf{rk}\pi}, P)\right]$$

XPX: Security Proof Techniques

Patarin's H-coefficient Technique

- Each conversation defines a transcript
- Define good and bad transcripts

$$\mathbf{Adv}_{\mathsf{XPX}}^{\text{rk-(s)prp}}(\mathcal{D}) \leq \varepsilon + \mathbf{Pr}\left[\text{bad transcript for } (\widetilde{\mathsf{rk}\pi}, P)\right]$$

↰ prob. ratio for good transcripts

- Trade-off: define bad transcripts smartly!

XPX: Security Proof Techniques

Before the Interaction

- Reveal “dedicated” oracle queries

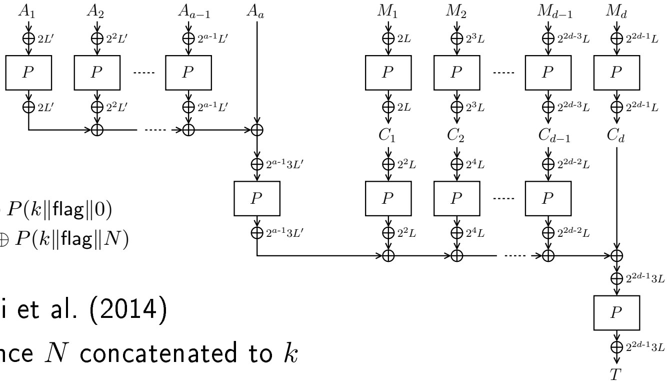
After the Interaction

- Reveal key information
  - Single-key: k and $P(k)$
  - $\Phi_{\oplus}$-related-key: k and $P(k \oplus \delta)$
  - $\Phi_{P\oplus}$-related-key: k and $P(k \oplus \delta)$ and $P^{-1}(P(k) \oplus \varepsilon)$

Bounding the Advantage

- Smart definition of bad transcripts

XPX: Application to AE: Minalpher

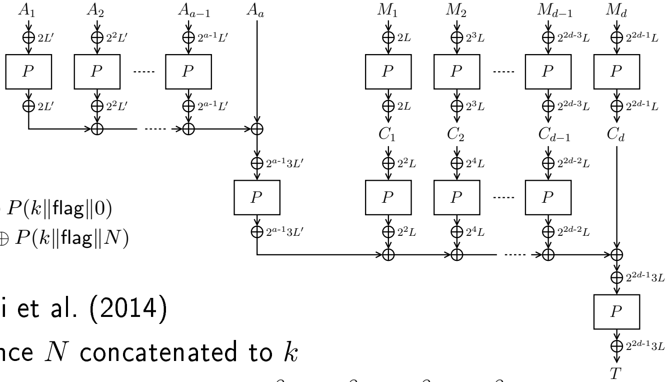


$L' = k\|\text{flag}\|0 \oplus P(k\|\text{flag}\|0)$

$L \ = k\|\text{flag}\|N \oplus P(k\|\text{flag}\|N)$

- By Sasaki et al. (2014)
- Extra nonce N concatenated to k

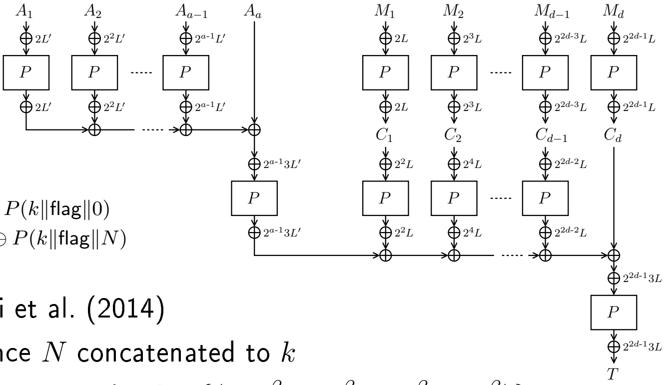
XPX: Application to AE: Minalpher



$$L' = k \| \mathsf{flag} \| 0 \oplus P(k \| \mathsf{flag} \| 0)$$
$$L = k \| \mathsf{flag} \| N \oplus P(k \| \mathsf{flag} \| N)$$

- By Sasaki et al. (2014)
- Extra nonce N concatenated to k
- Based on XPX with $\mathcal{T} = \{(2^\alpha 3^\beta, 2^\alpha 3^\beta, 2^\alpha 3^\beta, 2^\alpha 3^\beta)\}$

XPX: Application to AE: Minalpher



$$L' = k\|\text{flag}\|0 \oplus P(k\|\text{flag}\|0)$$
$$L = k\|\text{flag}\|N \oplus P(k\|\text{flag}\|N)$$

- By Sasaki et al. (2014)
- Extra nonce N concatenated to k
- Based on XPX with $\mathcal{T} = \{(2^\alpha 3^\beta, 2^\alpha 3^\beta, 2^\alpha 3^\beta, 2^\alpha 3^\beta)\}$

$$\boxed{\text{Minalph.}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{\text{XPX}} \xrightarrow[\text{rk}]{\mathcal{O}\left(\frac{\sigma^2}{2^n}\right)} \boxed{P}$$