Authenticated Encryption

Applied Cryptography – Spring 2024

Bart Mennink

February 19, 2024

Institute for Computing and Information Sciences
Radboud University

Encryption

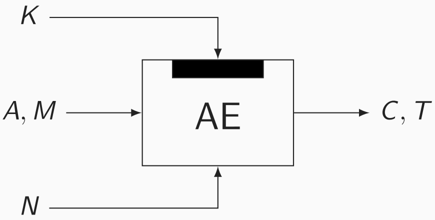
- Security goal: confidentiality
- Examples: ECB, counter mode

Authentication

- Security goal: data integrity
- Examples: CBC-MAC, Poly1305

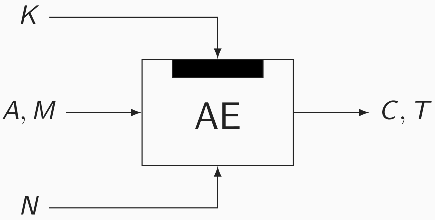
Authenticated encryption combines both

Authenticated Encryption



- Using key K:
  - Message M is encrypted in ciphertext C
  - Associated data A and message M are authenticated using T
- Nonce N randomizes the scheme

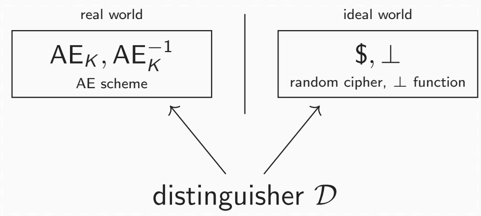
Authenticated Encryption



- Using key K:
  - Message M is encrypted in ciphertext C
  - Associated data A and message M are authenticated using T
- Nonce N randomizes the scheme

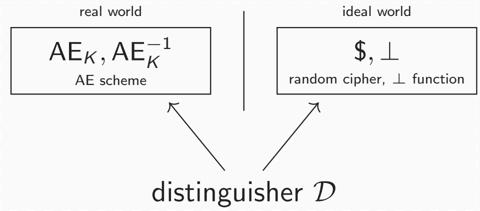
- Authenticated decryption discloses M if and only if T is correct

Authenticated Encryption Security



- Two oracles: $(\mathrm{AE}_K, \mathrm{AE}_K^{-1})$ (for secret key K) and $(\$, \perp)$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ unique nonce for each encryption query, and no trivial queries
- $\mathcal{D}$ tries to determine which oracle it communicates with

Authenticated Encryption Security



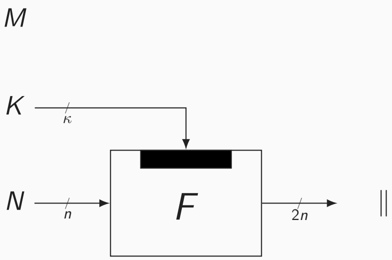
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\$, \perp)$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ unique nonce for each encryption query, and no trivial queries
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_{\mathsf{AE}}^{\mathrm{ae}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{AE}_K, \mathsf{AE}_K^{-1} \,;\, \$, \perp\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{AE}_K, \mathsf{AE}_K^{-1}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\$, \perp} = 1\right)\right|$$

- $\mathbf{Adv}_{\mathsf{AE}}^{\mathrm{ae}}(q_e, q_v)$: supremal advantage over any $\mathcal{D}$ with query complexity q_e, q_v

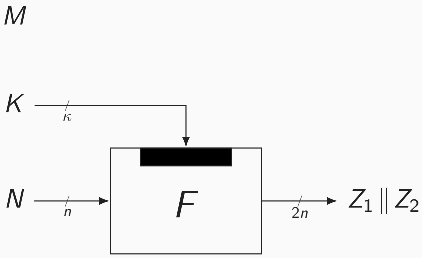
Outline

Authenticated Encryption Design



Encryption

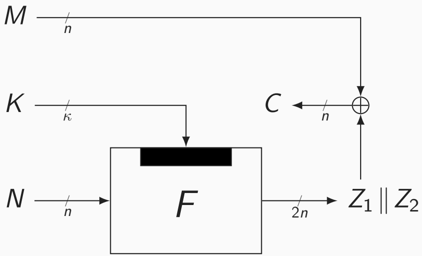
- Input: (N, M)



Encryption

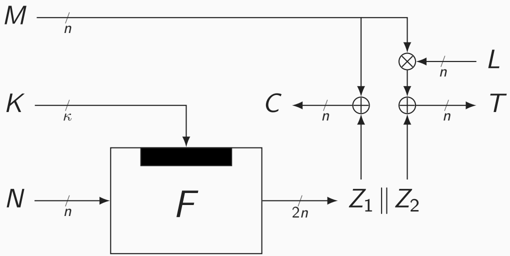
- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$

Simple Example



Encryption

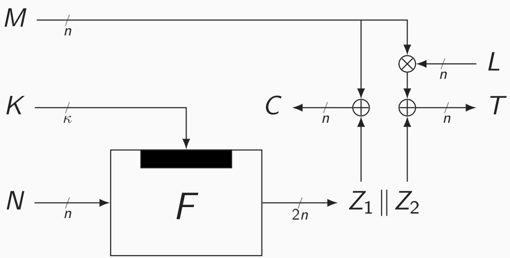
- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$



Encryption

- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$
  - $T = Z_2 \oplus (M \otimes L)$

Simple Example

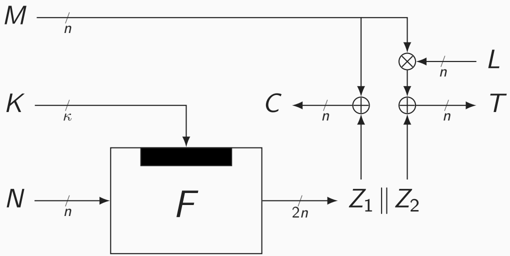


Encryption

- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$
  - $T = Z_2 \oplus (M \otimes L)$

Decryption

- Input: (N, C, T)

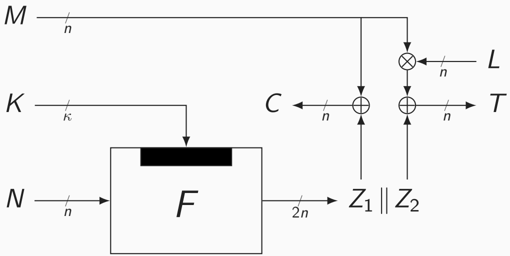


Encryption

- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$
  - $T = Z_2 \oplus (M \otimes L)$

Decryption

- Input: (N, C, T)
- Compute keystream $Z_1 \parallel Z_2$
- Compute $M = Z_1 \oplus C$



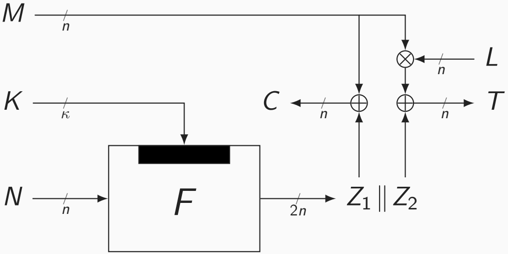
Encryption

- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$
  - $T = Z_2 \oplus (M \otimes L)$

Decryption

- Input: (N, C, T)
- Compute keystream $Z_1 \parallel Z_2$
- Compute $M = Z_1 \oplus C$
- Compute $T^\star = Z_2 \oplus (M \otimes L)$

Simple Example



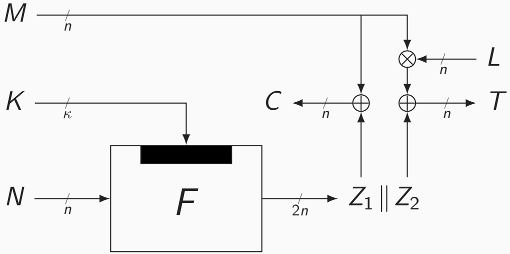
Encryption

- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$
- Output:
  - $C = Z_1 \oplus M$
  - $T = Z_2 \oplus (M \otimes L)$

Decryption

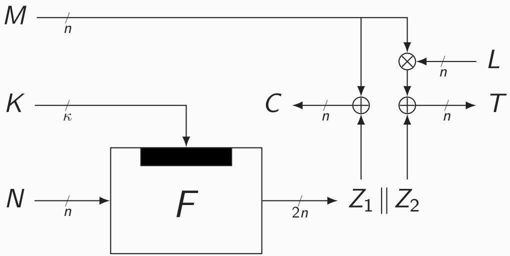
- Input: (N, C, T)
- Compute keystream $Z_1 \parallel Z_2$
- Compute $M = Z_1 \oplus C$
- Compute $T^\star = Z_2 \oplus (M \otimes L)$
- Output: $\begin{cases} M \text{ if } T = T^\star \\ \perp \text{ otherwise} \end{cases}$

Simple Example: Confidentiality



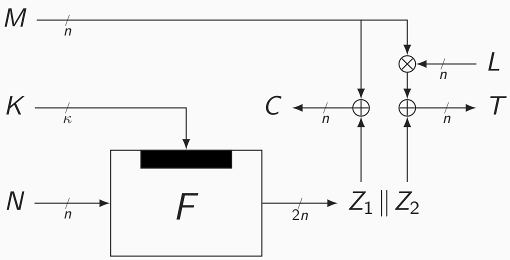
Confidentiality

- Consider new query (N, M)
- N should be fresh



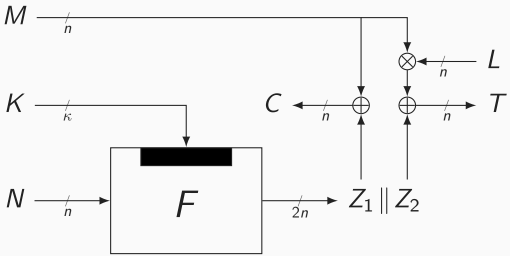
Confidentiality

- Consider new query (N, M)
- N should be fresh
- Random $Z_1 \parallel Z_2$
  (if F is a good stream cipher)



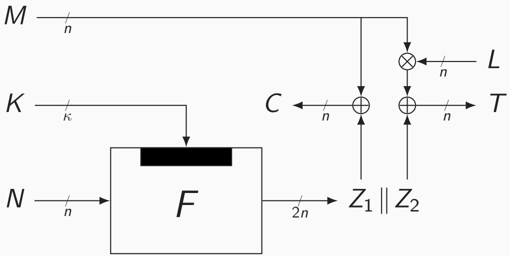
Confidentiality

- Consider new query (N, M)
- N should be fresh
- Random $Z_1 \| Z_2$
  (if F is a good stream cipher)
- Random (C, T)



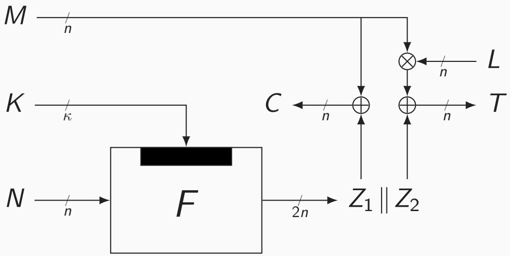
Authenticity

- Consider forgery attempt (N, C, T)



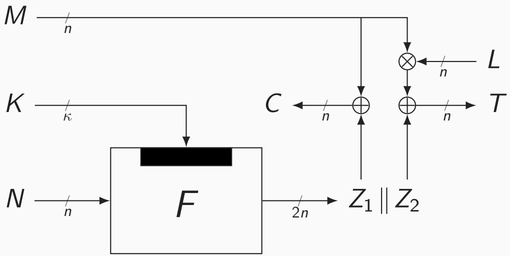
Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce



Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce
- N fresh:
    - $T^\star$ is random, unpredictable

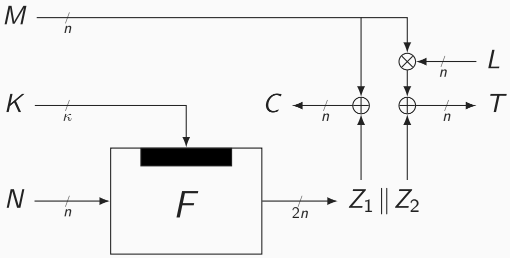


Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce

- N fresh:
  - $T^\star$ is random, unpredictable
- N repeated:
  - Let (N, M', C', T') be old

Simple Example: Authenticity

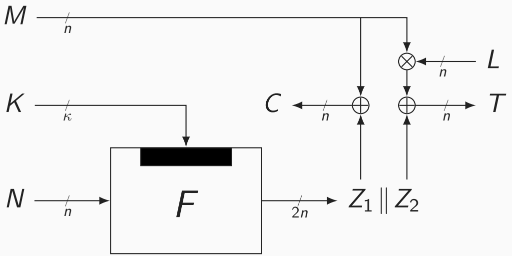


Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce

- N fresh:
  - $T^\star$ is random, unpredictable
- N repeated:
  - Let (N, M', C', T') be old
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$

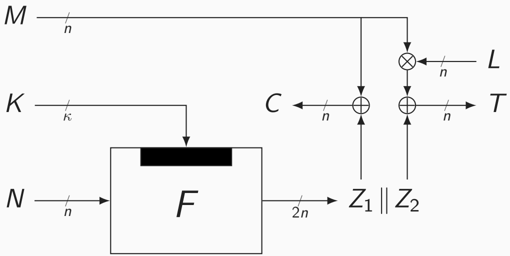
Simple Example: Authenticity



Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce
- N fresh:
  - $T^\star$ is random, unpredictable
- N repeated:
  - Let (N, M', C', T') be old
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$
  - $T^\star = Z_2 \oplus (M \otimes L)$
    $= T' \oplus ((M \oplus M') \otimes L)$
    $= T' \oplus ((C \oplus C') \otimes L)$

Simple Example: Authenticity

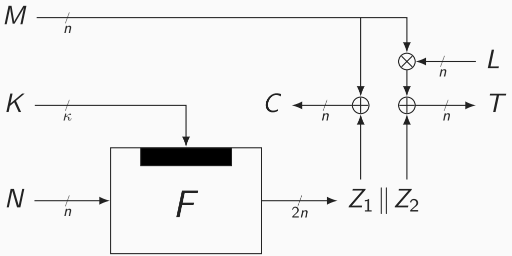


Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce

- N fresh:
  - $T^\star$ is random, unpredictable
- N repeated:
  - Let (N, M', C', T') be old
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$
  - $T^\star = Z_2 \oplus (M \otimes L)$
    $T^\star = T' \oplus ((M \oplus M') \otimes L)$
    $T^\star = T' \oplus ((C \oplus C') \otimes L)$
  - Forgery successful if
    $T \oplus T' = (C \oplus C') \otimes L$

Simple Example: Authenticity

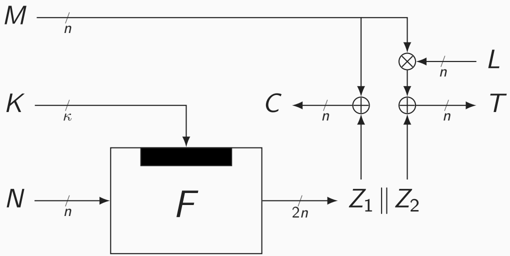


Authenticity

- Consider forgery attempt (N, C, T)
- N could be repeated nonce

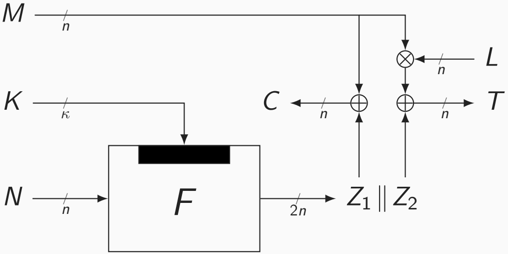
- N fresh:
  - $T^\star$ is random, unpredictable
- N repeated:
  - Let (N, M', C', T') be old
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$
  - $T^\star = Z_2 \oplus (M \otimes L)$
    $= T' \oplus ((M \oplus M') \otimes L)$
    $= T' \oplus ((C \oplus C') \otimes L)$
  - Forgery successful if
    $T \oplus T' = (C \oplus C') \otimes L$
  - Requires guessing L

Simple Example: How to Support Variable Length?



Suppose M is Variable-Length?

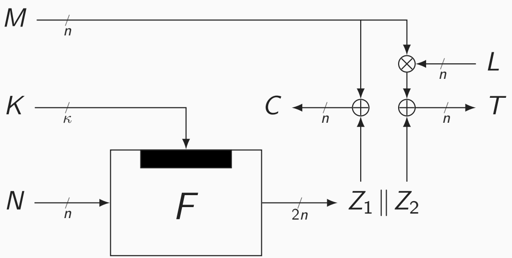
Simple Example: How to Support Variable Length?



Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary $\#$ of bits?

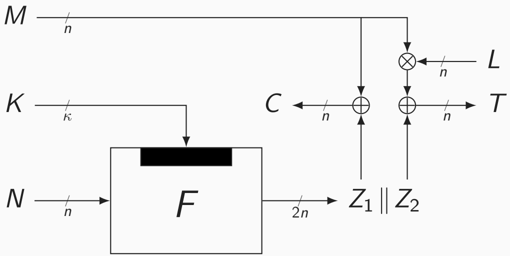
Simple Example: How to Support Variable Length?



Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary $\#$ of bits?
- $|M| + n$ bits turns out to suffice:
    - Use streaming mode for F
    - Replace $M \otimes L$ by $H_L(M)$

Simple Example: How to Support Variable Length?

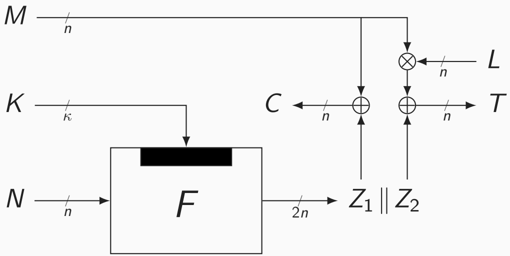


Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary # of bits?
- $|M| + n$ bits turns out to suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

What about AD A?

Simple Example: How to Support Variable Length?



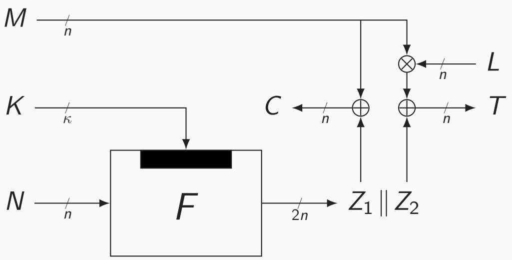
Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary # of bits?
- $|M| + n$ bits turns out to suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

What about AD A?

- Can be processed by H_L as well:
  - $H_L(A, M)$

Simple Example: How to Support Variable Length?



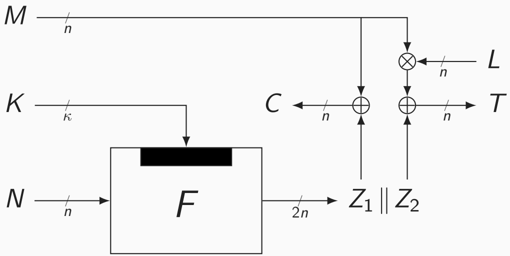
Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary $\#$ of bits?
- $|M| + n$ bits turns out to suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

What about AD A?

- Can be processed by H_L as well:
  - $H_L(A, M)$

This is almost exactly GCM!



Suppose M is Variable-Length?

- Output F should be twice as large?
- $\otimes$ over arbitrary # of bits?
- $|M| + n$ bits turns out to suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

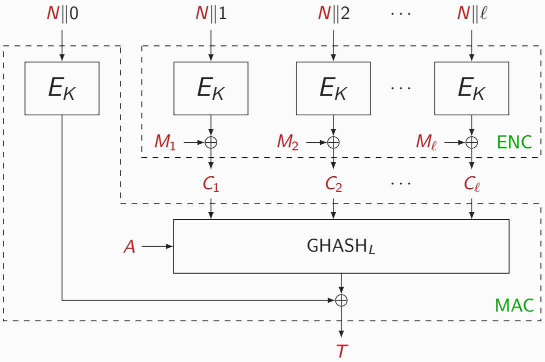
What about AD A?

- Can be processed by H_L as well:
  - $H_L(A, M)$

This is almost exactly GCM!

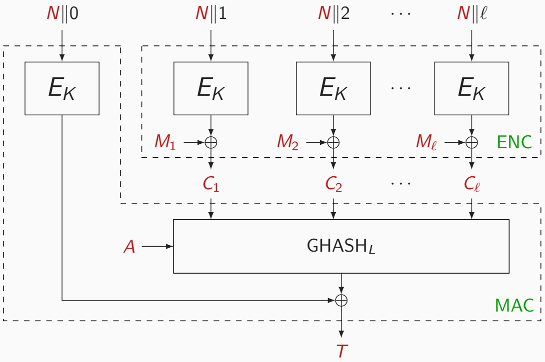
- Encrypt-*then*-MAC: $H_L(A, C)$
- Take CTR mode for F

GCM for 96-bit Nonce N



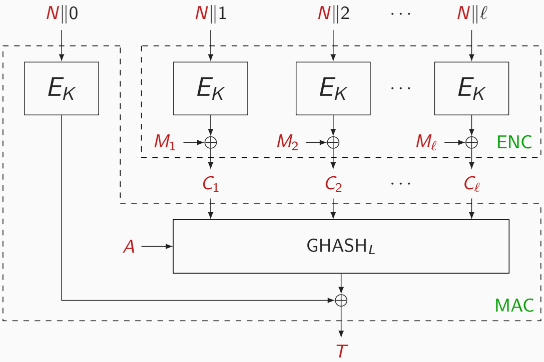
- McGrew and Viega (2004)
- EtM design
- Widely used (TLS!)
- Patent-free

GCM for 96-bit Nonce N



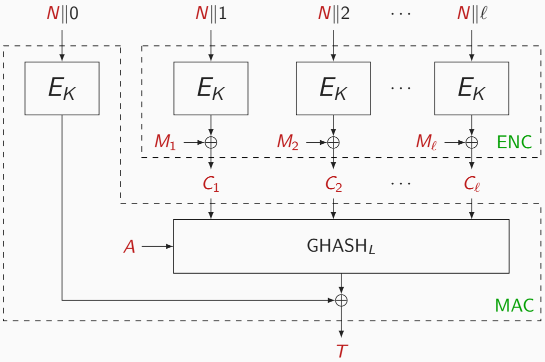
- McGrew and Viega (2004)
- EtM design
- Widely used (TLS!)
- Patent-free
- Parallelizable
- Evaluates E only (no E^{-1})
- Provably secure (if E is PRP)
- Very efficient in HW
- Reasonably efficient in SW

GCM for 96-bit Nonce N



- McGrew and Viega (2004)
- EtM design
- Widely used (TLS!)
- Patent-free
- Parallelizable
- Evaluates E only (no E^{-1})
- Provably secure (if E is PRP)
- Very efficient in HW
- Reasonably efficient in SW
- Note: equally popular is ChaCha20-Poly1305!

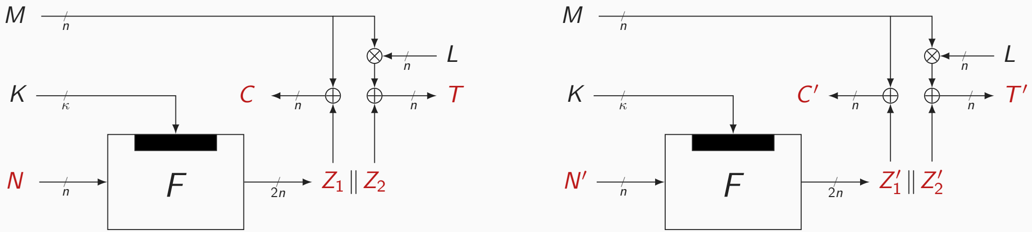
GCM for 96-bit Nonce N



- McGrew and Viega (2004)
- EtM design
- Widely used (TLS!)
- Patent-free
- Parallelizable
- Evaluates E only (no E^{-1})
- Provably secure (if E is PRP)
- Very efficient in HW
- Reasonably efficient in SW
- Note: equally popular is ChaCha20-Poly1305!

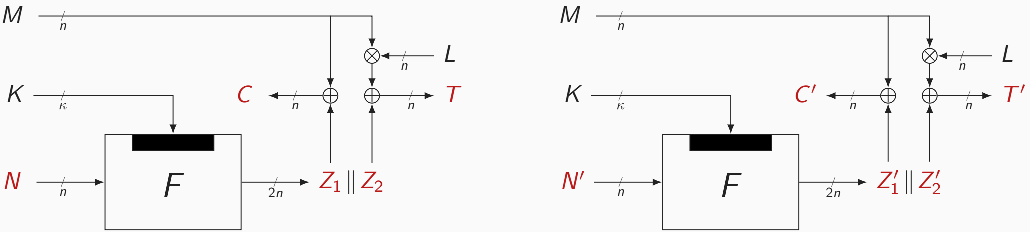
What happens if nonce is re-used?

Nonce = “Number Used Once”



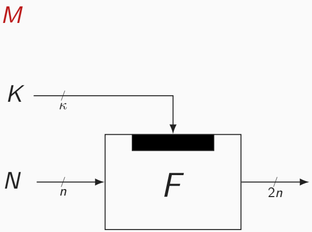
- Nonces N and N' should be distinct for two different evaluations

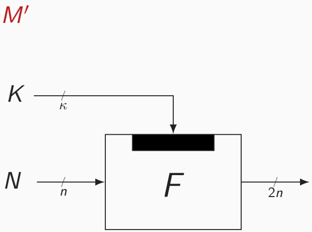
Nonce = “Number Used Once”



- Nonces N and N' should be distinct for two different evaluations
- What happens if a nonce would be repeated?

What if a Nonce is Repeated?

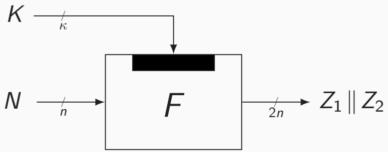




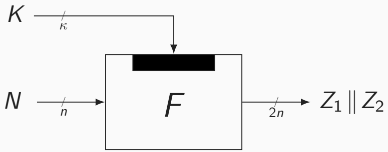
- Consider evaluations for identical nonce but distinct M, M'

What if a Nonce is Repeated?

M

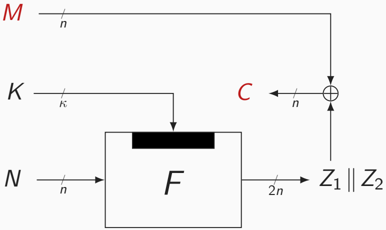


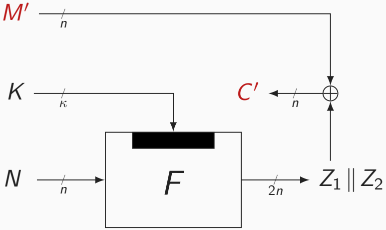
M'



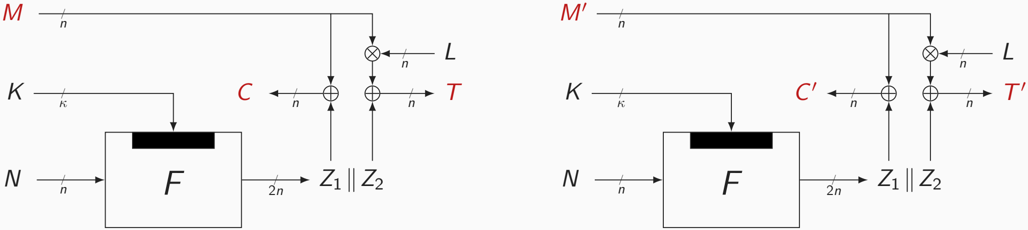
- Consider evaluations for identical nonce but distinct M, M'
- Key streams will be identical

What if a Nonce is Repeated?



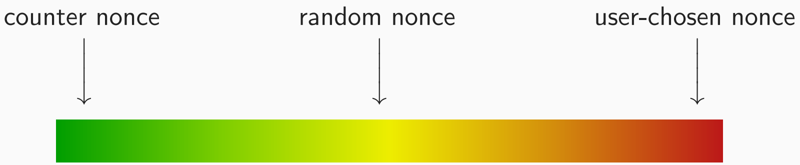


- Consider evaluations for identical nonce but distinct M, M'
- Key streams will be identical
- Ciphertexts satisfy $C \oplus C' = M \oplus M' \longrightarrow$ attacker knew C' in advance

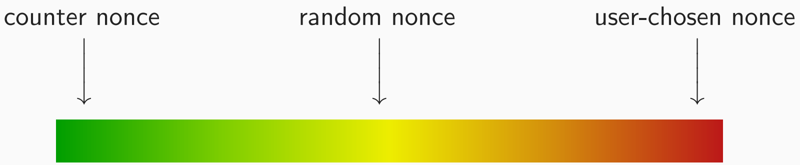


- Consider evaluations for identical nonce but distinct M, M'
- Key streams will be identical
- Ciphertexts satisfy $C \oplus C' = M \oplus M'$ $\longrightarrow$ attacker knew C' in advance
- Tags satisfy $T \oplus T' = M \otimes L \oplus M' \otimes L = (M \oplus M') \otimes L$ $\longrightarrow$ key recovery

Guaranteeing Uniqueness of Nonce

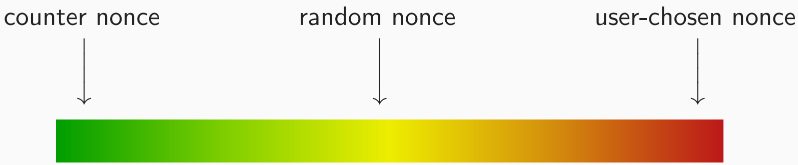


Guaranteeing Uniqueness of Nonce



- Issues with nonce generation:
  - Counter needs storage
  - Need synchronization or transmission
  - Efficiency cost
  - Laziness or mistake of implementor
  - ...

Guaranteeing Uniqueness of Nonce



- Issues with nonce generation:
  - Counter needs storage
  - Need synchronization or transmission
  - Efficiency cost
  - Laziness or mistake of implementor
  - ...
- Sometimes, attacker can use same nonce multiple times

Nonce-Reuse in Practice

Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS

Böck et al., USENIX WOOT 2016

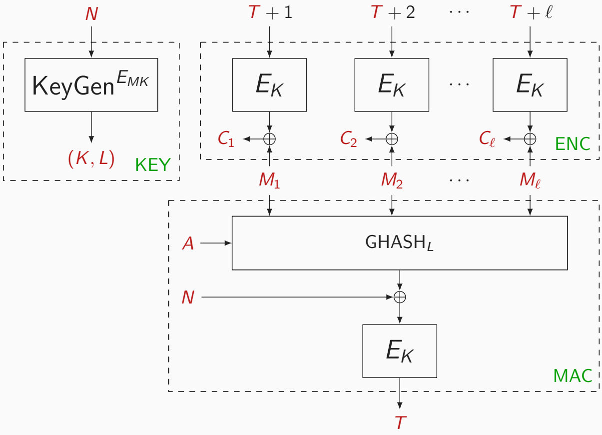
- GCM is widely used authenticated encryption scheme
- Used in TLS ("https")

- Internet-wide scan for GCM implementations
- 184 devices with duplicated nonces
    - VISA, Polish bank, German stock exchange, ...
- $\approx$ 70.000 devices with random nonce

Intuition

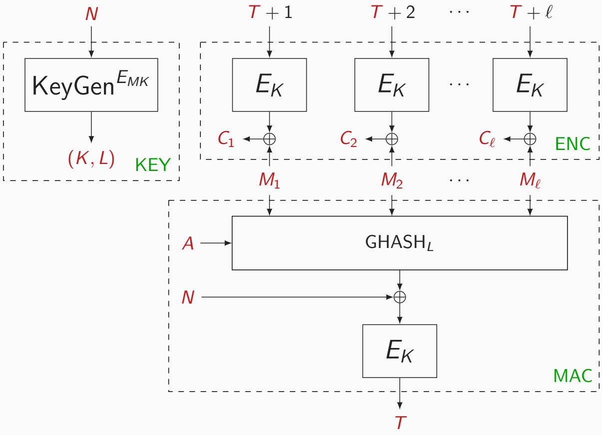
- All input should be cryptographically transformed
- Any change in $(N, A, M) \longrightarrow$ unpredictable (C, T)
- Often comes at a price:
  - Efficiency
  - Security
  - Parallelizability
  - ...

GCM-SIV



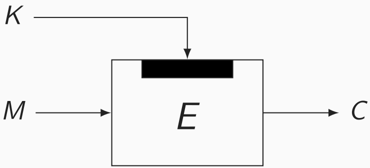
- Gueron and Lindell (2015)
- MtE design
- Ongoing standardization (IETF RFC)
- Patent-free

GCM-SIV



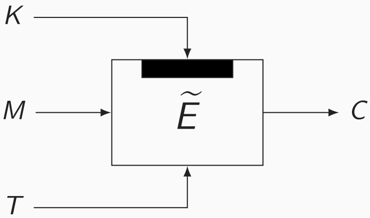
- Gueron and Lindell (2015)
- MtE design
- Ongoing standardization (IETF RFC)
- Patent-free
- Inherits GCM features
- Secure against nonce-reuse
- Proof: Iwata and Seurin (2017)

Tweakable Block Ciphers



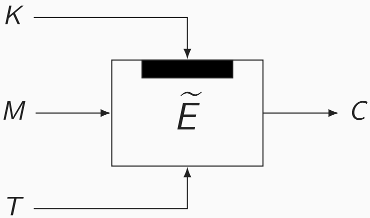
- Using key K, message M is bijectively transformed to ciphertext C

Tweakable Block Ciphers



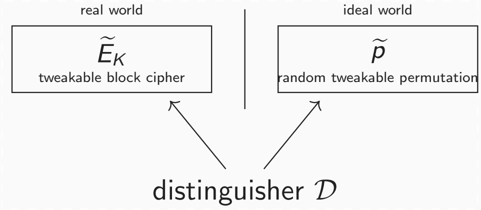
- Using key K, message M is bijectively transformed to ciphertext C
- Tweak T: flexibility to the cipher
- Each tweak gives different permutation

Tweakable Block Ciphers



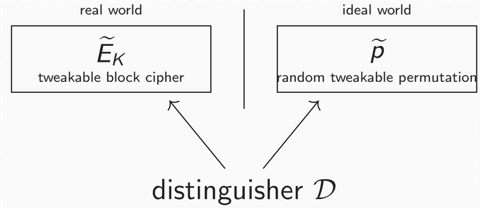
- Using key K, message M is bijectively transformed to ciphertext C
- Tweak T: flexibility to the cipher
- Each tweak gives different permutation
- A good tweakable block cipher should behave like a random tweakable permutation

Tweakable Block Cipher Security



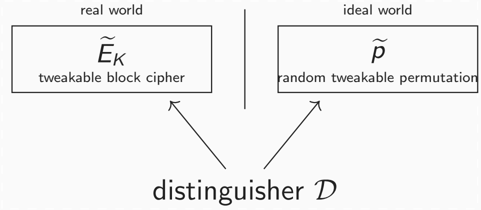
- Two oracles: $\widetilde{E}_K$ (for secret key K) and $\widetilde{p}$ (secret)

Tweakable Block Cipher Security

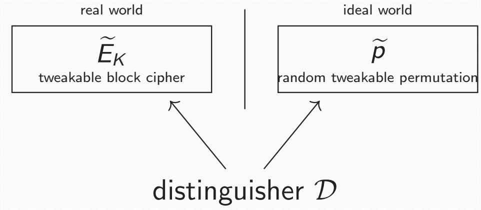


- Two oracles: $\widetilde{E}_K$ (for secret key K) and $\widetilde{p}$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Tweakable Block Cipher Security



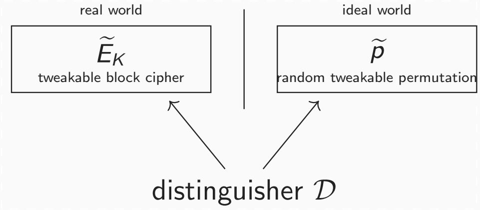
- Two oracles: $\widetilde{E}_K$ (for secret key K) and $\widetilde{p}$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with



- Two oracles: $\widetilde{E}_K$ (for secret key K) and $\widetilde{p}$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_{\widetilde{E}}^{\mathrm{tprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\widetilde{E}_K \,;\, \widetilde{p}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\widetilde{E}_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\widetilde{p}} = 1\right)\right|$$

Tweakable Block Cipher Security

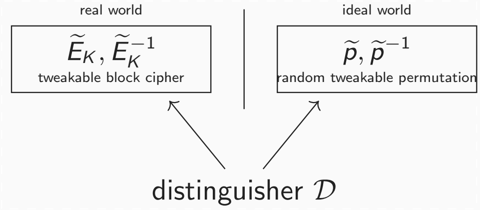


- Two oracles: $\widetilde{E}_K$ (for secret key K) and $\widetilde{p}$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

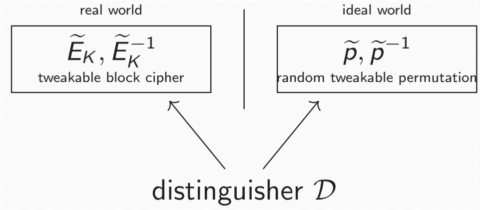
$$\mathbf{Adv}_{\widetilde{E}}^{\mathrm{tprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\widetilde{E}_K \,;\, \widetilde{p}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\widetilde{E}_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\widetilde{p}} = 1\right)\right|$$

- $\mathbf{Adv}_{\widetilde{E}}^{\mathrm{tprp}}(q)$: supremal advantage over any $\mathcal{D}$ with query complexity q

Strong Tweakable Block Cipher Security

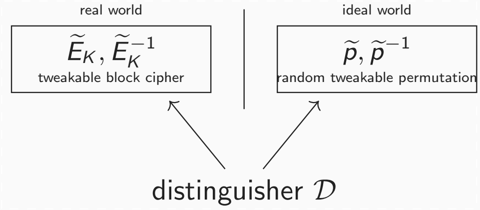


- Two oracles: $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ (for secret key K) and $(\widetilde{p}, \widetilde{p}^{-1})$ (secret)

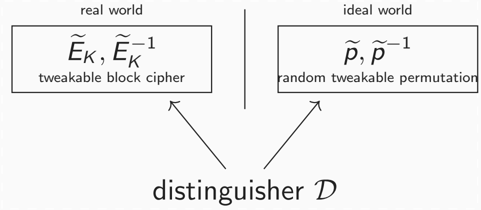


- Two oracles: $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ (for secret key K) and $(\widetilde{p}, \widetilde{p}^{-1})$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Strong Tweakable Block Cipher Security



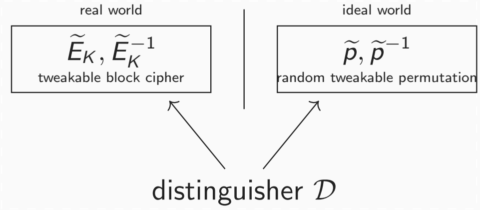
- Two oracles: $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ (for secret key K) and $(\widetilde{p}, \widetilde{p}^{-1})$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with



- Two oracles: $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ (for secret key K) and $(\widetilde{p}, \widetilde{p}^{-1})$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_{\widetilde{E}}^{\mathrm{stprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\widetilde{E}_K, \widetilde{E}_K^{-1} \;;\; \widetilde{p}, \widetilde{p}^{-1}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\widetilde{E}_K, \widetilde{E}_K^{-1}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\widetilde{p}, \widetilde{p}^{-1}} = 1\right)\right|$$

Strong Tweakable Block Cipher Security

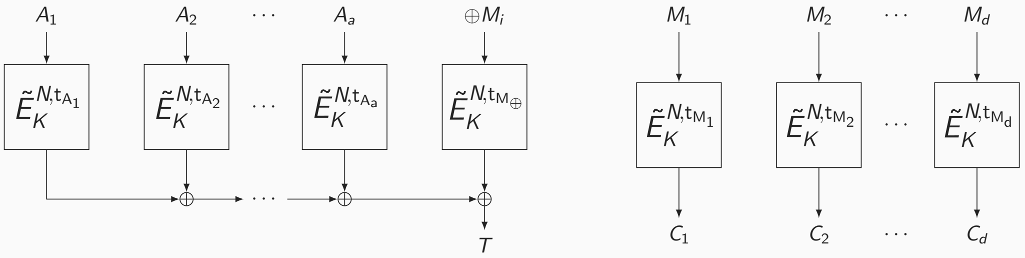


- Two oracles: $(\widetilde{E}_K, \widetilde{E}_K^{-1})$ (for secret key K) and $(\widetilde{p}, \widetilde{p}^{-1})$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_{\widetilde{E}}^{\mathrm{stprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\widetilde{E}_K, \widetilde{E}_K^{-1}\ ;\ \widetilde{p}, \widetilde{p}^{-1}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\widetilde{E}_K, \widetilde{E}_K^{-1}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\widetilde{p}, \widetilde{p}^{-1}} = 1\right)\right|$$

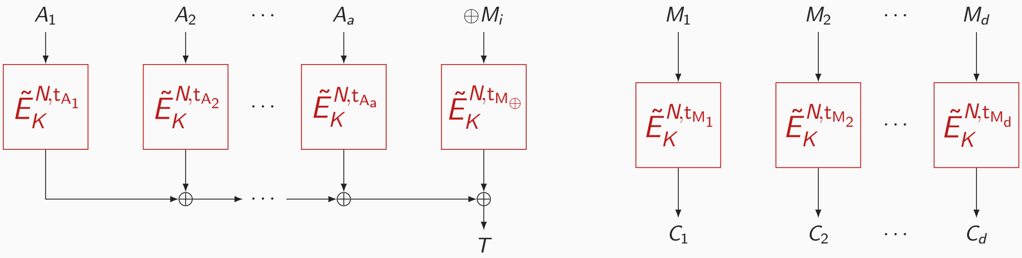
- $\mathbf{Adv}_{\widetilde{E}}^{\mathrm{stprp}}(q)$: supremal advantage over any $\mathcal{D}$ with query complexity q

Example Use in ΘCB (1/2)

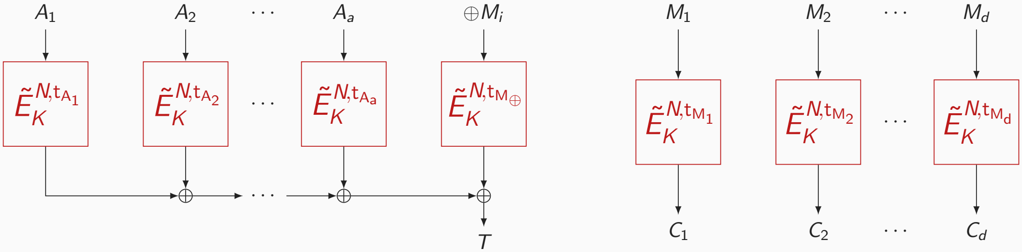


- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]

Example Use in ΘCB (1/2)



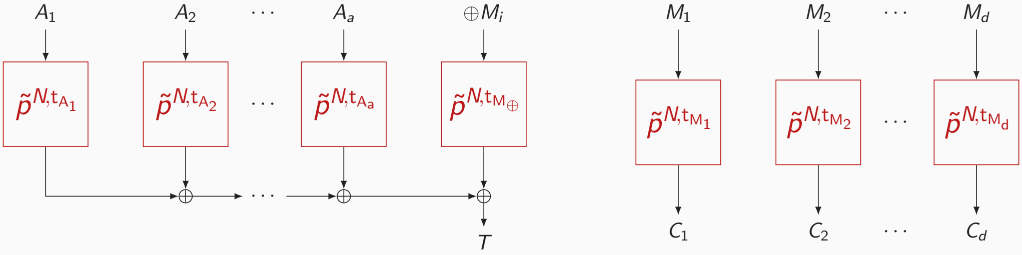
- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable block cipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation
  - Different blocks always transformed under different tweak



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

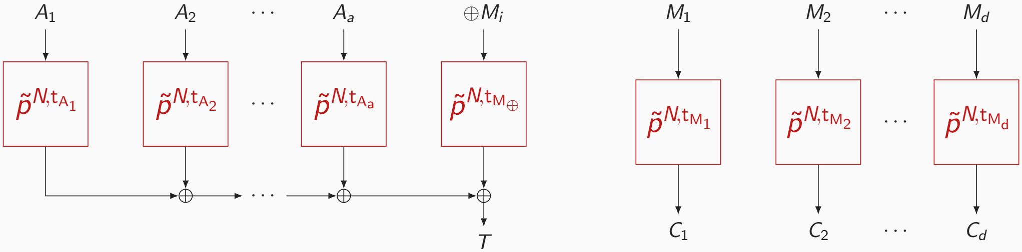
$$\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}[\widetilde{E}_k]}(q)$$

Example Use in ΘCB (1/2)

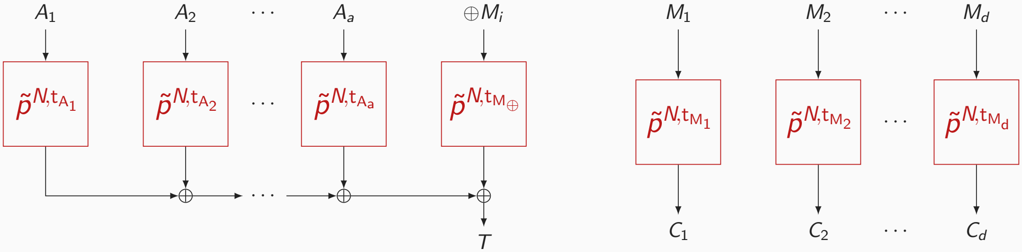


- Generalized OCB by Rogaway et al. [RBBK01,Rog04,KR11]
- Internally based on tweakable block cipher $\widetilde{E}$
  - Tweak (N, tweak) is unique for every evaluation
  - Different blocks always transformed under different tweak
- Triangle inequality:

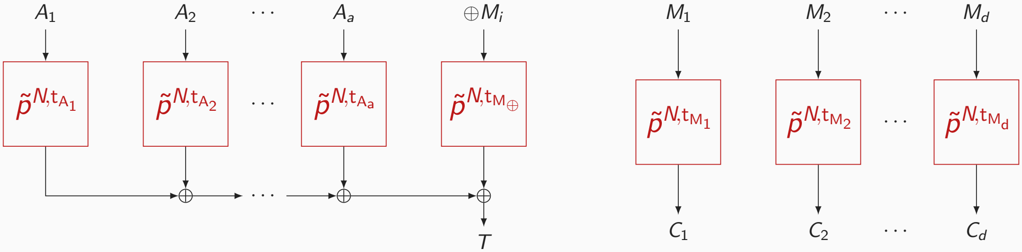
$$\mathbf{Adv}^{\mathrm{ae}}_{\mathrm{AE}[\widetilde{E}_k]}(q) \leq \mathbf{Adv}^{\mathrm{ae}}_{\mathrm{AE}[\widetilde{p}]}(q) + \mathbf{Adv}^{\mathrm{stprp}}_{\widetilde{E}}(q)$$


A_1
A_2
A_a
$\oplus M_i$
$\tilde{p}^{N,t_{A_1}}$
$\tilde{p}^{N,t_{A_2}}$
$\tilde{p}^{N,t_{A_a}}$
$\tilde{p}^{N,t_{M_\oplus}}$
T
M_1
M_2
M_d
$\tilde{p}^{N,t_{M_1}}$
$\tilde{p}^{N,t_{M_2}}$
$\tilde{p}^{N,t_{M_d}}$
C_1
C_2
C_d

Example Use in ΘCB (2/2)

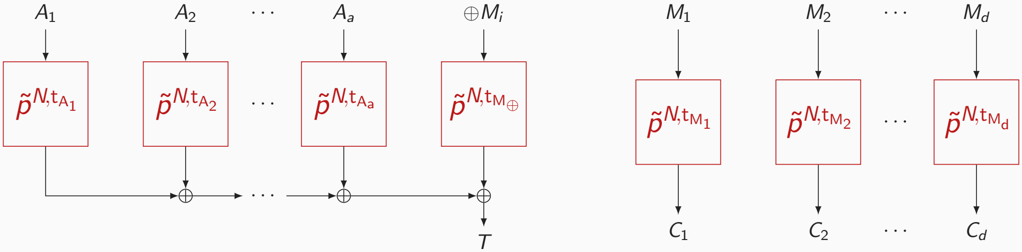


- Nonce uniqueness $\Rightarrow$ tweak uniqueness



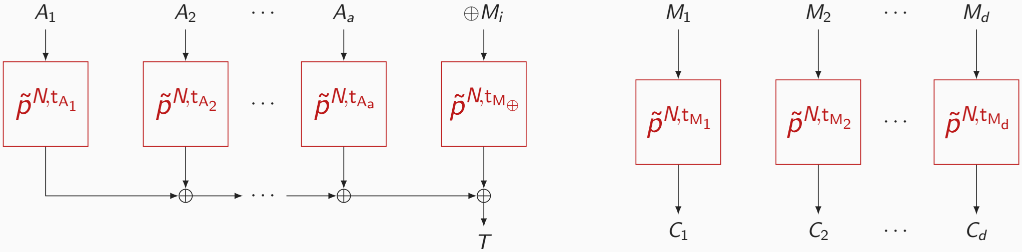
- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $\mathrm{AE}[\tilde{p}] = \$$

Example Use in ΘCB (2/2)



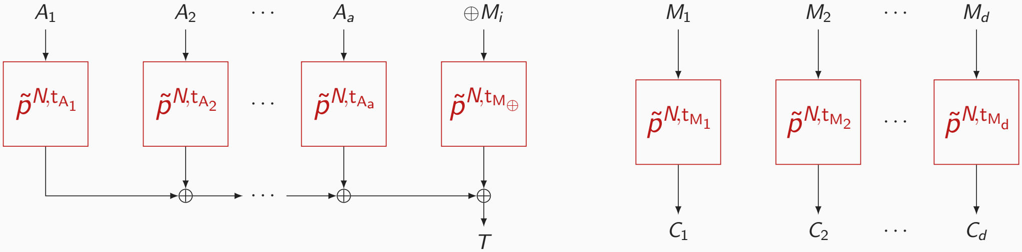
- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $\mathrm{AE}[\tilde{p}] = \$$
- Authentication almost behaves like random function (but nonces may repeat)

Example Use in ΘCB (2/2)



- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $\mathrm{AE}[\tilde{p}] = \$$
- Authentication almost behaves like random function (but nonces may repeat)
  - Tag forged with probability at most $1/(2^n - 1)$

Example Use in ΘCB (2/2)

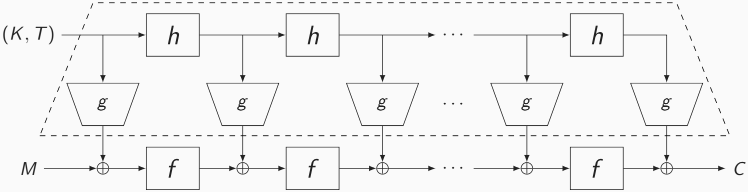


- Nonce uniqueness $\Rightarrow$ tweak uniqueness
- Encryption calls behave like random functions: $\mathrm{AE}[\tilde{p}] = \$$
- Authentication almost behaves like random function (but nonces may repeat)
  - Tag forged with probability at most $1/(2^n - 1)$

$$\mathbf{Adv}^{\mathrm{ae}}_{\mathrm{AE}[\tilde{p}]}(q) \leq 1/(2^n - 1)$$

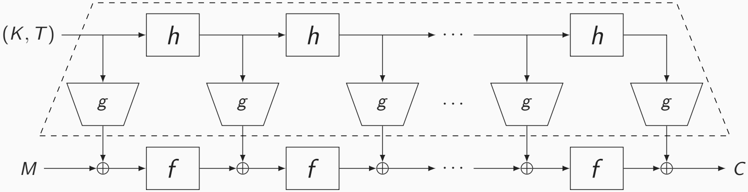
Building Tweakable Block Ciphers

- TWEAKEY by Jean et al. [JNP14]:



- f: round function
- g: subkey computation
- h: transformation of (K, T)

- TWEAKEY by Jean et al. [JNP14]:

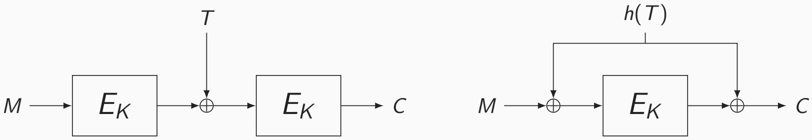


- f: round function
- g: subkey computation
- h: transformation of (K, T)

- Security measured through cryptanalysis
- Our focus: modular design

Original Constructions

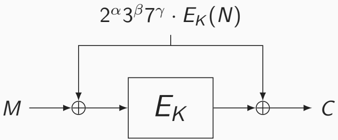
- LRW_1 and LRW_2 by Liskov et al. [LRW02]:



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

Powering-Up Masking (XEX)

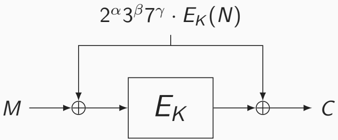
- XEX by Rogaway [Rog04]:



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

Powering-Up Masking (XEX)

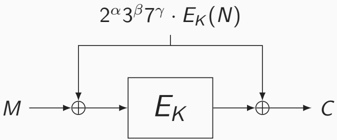
- XEX by Rogaway [Rog04]:

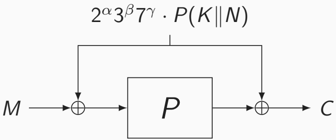


- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Used in OCB2 and ± 14 CAESAR candidates

Powering-Up Masking (XEX)

- XEX by Rogaway [Rog04]:





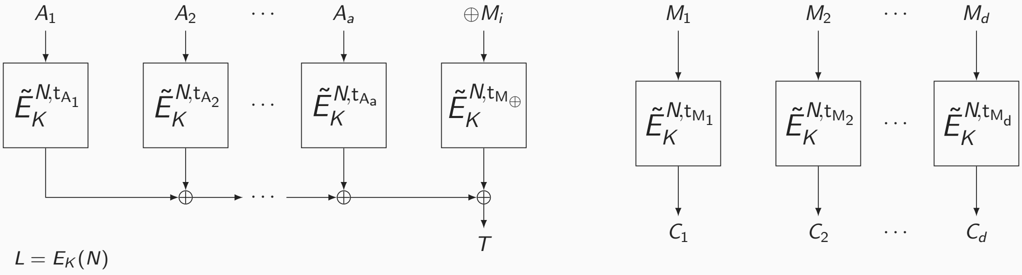
- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Used in OCB2 and ± 14 CAESAR candidates
- Permutation-based variants in Minalpher and Prøst (generalized by Cogliati et al. [CLS15])

Powering-Up Masking (XEX)

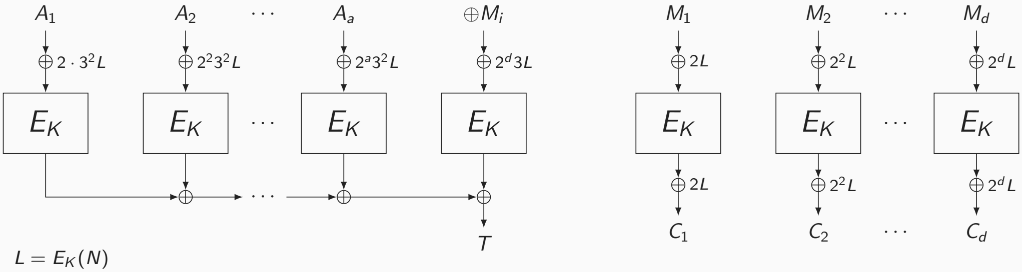
- XEX by Rogaway [Rog04]:

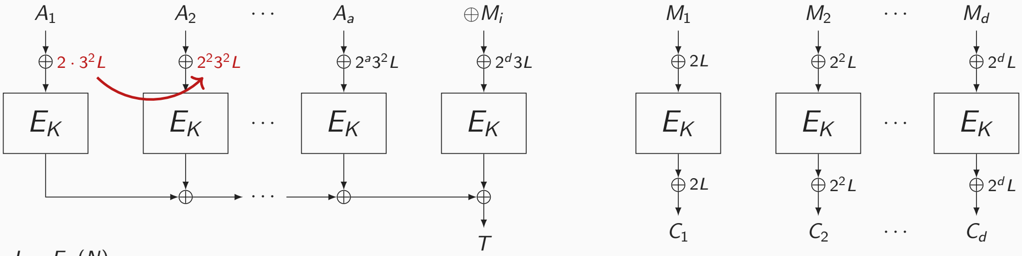
- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Used in OCB2 and ± 14 CAESAR candidates
- Permutation-based variants in Minalpher and Prøst (generalized by Cogliati et al. [CLS15])
- STPRP up to $2^{n/2}$ queries provided masks are all distinct

Powering-Up Masking in OCB2-Like Construction



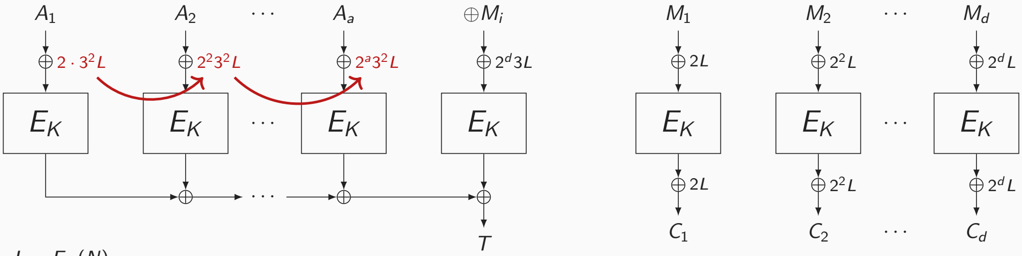
Powering-Up Masking in OCB2-Like Construction



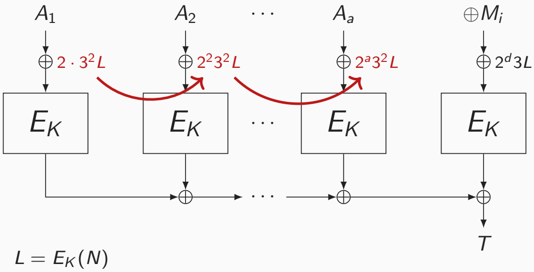

A_1
A_2
A_a
$\oplus M_i$
M_1
M_2
M_d
$2 \cdot 3^2 L$
$2^2 3^2 L$
$2^a 3^2 L$
$2^d 3L$
$2L$
$2^2 L$
$2^d L$
E_K
$2L$
$2^2 L$
$2^d L$
T
C_1
C_2
C_d

$L = E_K(N)$

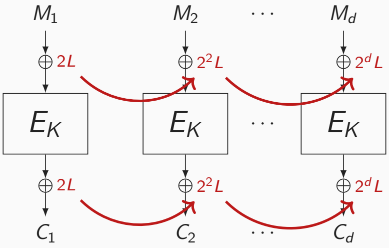
Powering-Up Masking in OCB2-Like Construction



$L = E_K(N)$

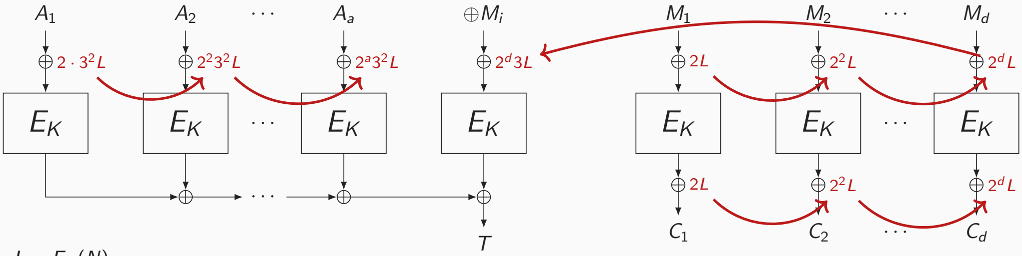


A_1
A_2
$\cdots$
A_a
$\oplus M_i$
$2 \cdot 3^2 L$
$2^2 3^2 L$
$2^a 3^2 L$
$2^d 3L$
E_K
E_K
$\cdots$
E_K
E_K
$\cdots$
T
$L = E_K(N)$



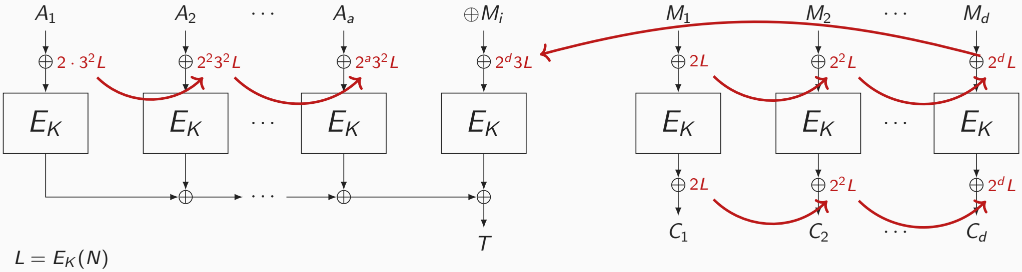
M_1
M_2
$\cdots$
M_d
$2L$
$2^2 L$
$2^d L$
E_K
E_K
$\cdots$
E_K
$2L$
$2^2 L$
$2^d L$
C_1
C_2
$\cdots$
C_d

Powering-Up Masking in OCB2-Like Construction



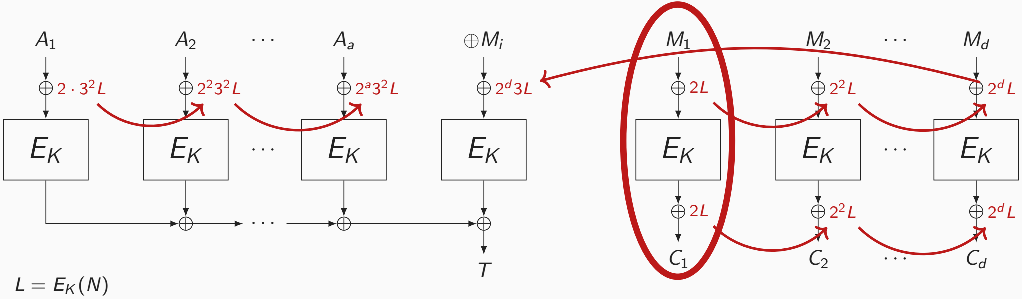
$L = E_K(N)$

Powering-Up Masking in OCB2-Like Construction



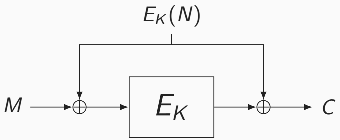
- Update of mask:
  - Shift and conditional XOR
- Variable time computation
- Expensive on certain platforms

Intermezzo: Why Start at $2 \cdot E_K(N)$? (1/2)



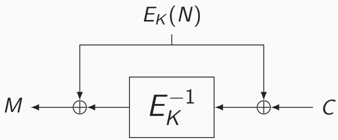
- Update of mask:
    - Shift and conditional XOR
- Variable time computation
- Expensive on certain platforms

- Suppose we would mask with $E_K(N)$:



Intermezzo: Why Start at $2 \cdot E_K(N)$? (2/2)

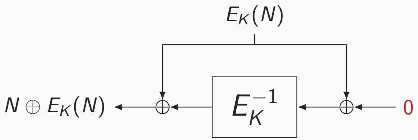
- Suppose we would mask with $E_K(N)$:



- Distinguisher can make inverse queries

Intermezzo: Why Start at $2 \cdot E_K(N)$? (2/2)

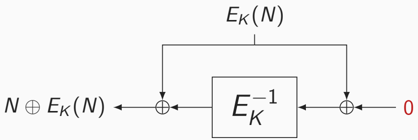
- Suppose we would mask with $E_K(N)$:



- Distinguisher can make inverse queries
- Putting $C = 0$ gives $M = N \oplus E_K(N)$

Intermezzo: Why Start at $2 \cdot E_K(N)$? (2/2)

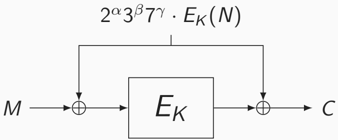
- Suppose we would mask with $E_K(N)$:



- Distinguisher can make inverse queries
- Putting $C = 0$ gives $M = N \oplus E_K(N)$
- Distinguisher knows N so learns "subkey" $E_K(N)$

Powering-Up Masking (XEX): Setting Admissible Domain

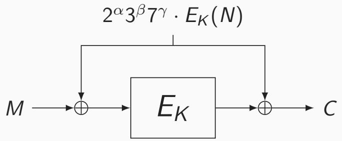
- XEX by Rogaway [Rog04]:



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

Powering-Up Masking (XEX): Setting Admissible Domain

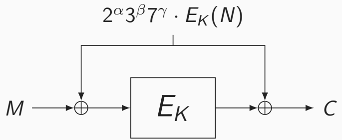
- XEX by Rogaway [Rog04]:



- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- (α, β, γ) must be from a certain admissible domain
- We need that $2^\alpha 3^\beta 7^\gamma \neq 2^{\alpha'} 3^{\beta'} 7^{\gamma'}$ for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
    - Otherwise, attacker can obviously break the scheme

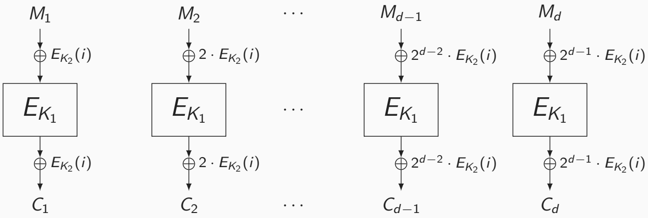
Powering-Up Masking (XEX): Setting Admissible Domain

- XEX by Rogaway [Rog04]:



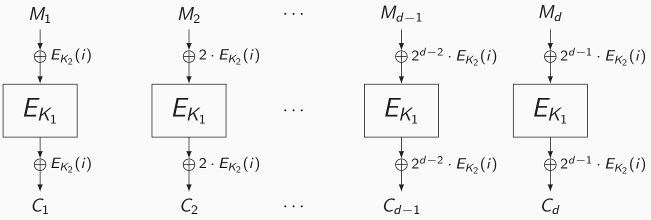
- $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- (α, β, γ) must be from a certain admissible domain
- We need that $2^\alpha 3^\beta 7^\gamma \neq 2^{\alpha'} 3^{\beta'} 7^{\gamma'}$ for any $(\alpha, \beta, \gamma) \neq (\alpha', \beta', \gamma')$
  - Otherwise, attacker can obviously break the scheme
- Typical: $\alpha \in \{1, \ldots, \text{large}\}$, and $\beta, \gamma \in \{0, 1, 2\}$

Intermezzo: XTS Disk Encryption Mode (1/2)



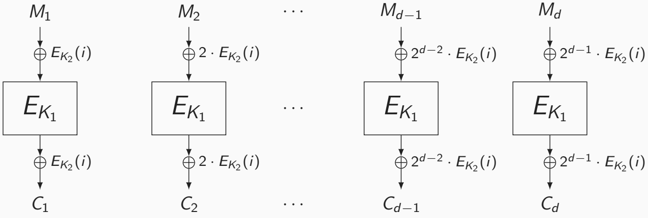
- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing

Intermezzo: XTS Disk Encryption Mode (1/2)



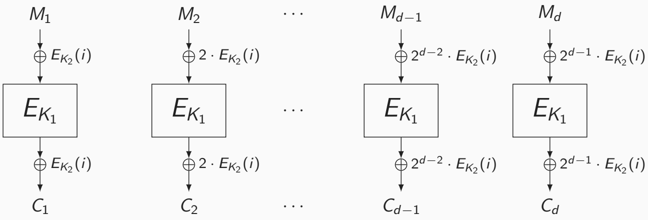
- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing
  - Electronic CodeBook (ECB) ...

Intermezzo: XTS Disk Encryption Mode (1/2)

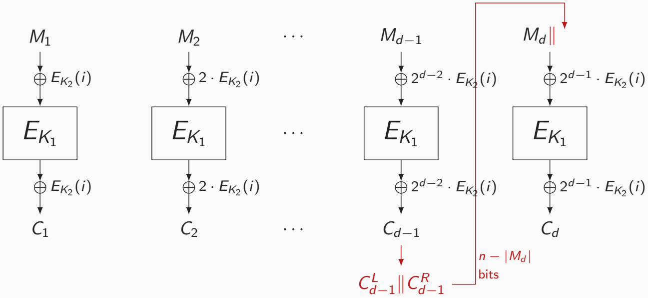


- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing
  - Electronic CodeBook (ECB) ...
  - ...with XEX as primitive (i is sector, j is block within sector) ...

Intermezzo: XTS Disk Encryption Mode (1/2)

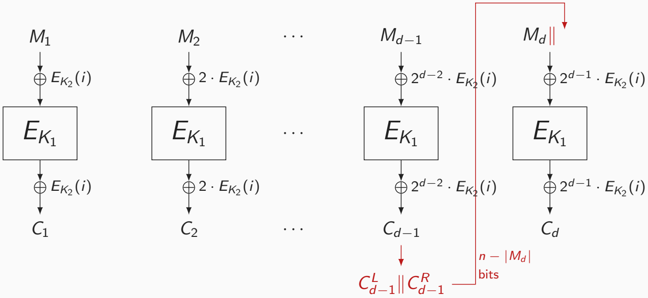


- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing
  - Electronic CodeBook (ECB) ...
  - ...with XEX as primitive (i is sector, j is block within sector) ...
  - ...and doing a fancy thing called "ciphertext stealing"



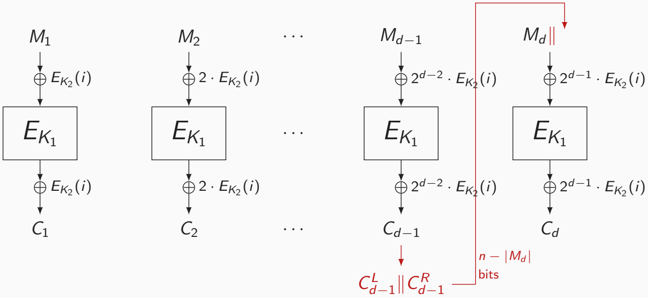
- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing
  - Electronic CodeBook (ECB) ...
  - ...with XEX as primitive (i is sector, j is block within sector) ...
  - ...and doing a fancy thing called “ciphertext stealing”

Intermezzo: XTS Disk Encryption Mode (1/2)



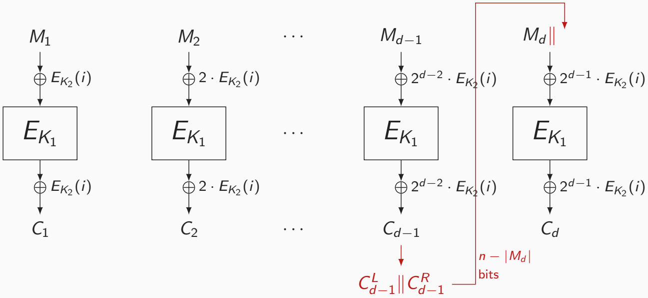
- XTS = XEX-based Tweaked-codebook mode with ciphertext Stealing
  - Electronic CodeBook (ECB) ...
  - ...with XEX as primitive (i is sector, j is block within sector) ...
  - ...and doing a fancy thing called "ciphertext stealing"
- One sector consists of 512 bytes, or 32 blocks

Intermezzo: XTS Disk Encryption Mode (2/2)



- Features:
  - Tweak unique for every block, changing tweak is efficient
  - Incrementality: change in one (or few) blocks

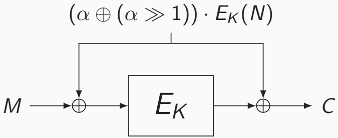
Intermezzo: XTS Disk Encryption Mode (2/2)



- Features:
  - Tweak unique for every block, changing tweak is efficient
  - Incrementality: change in one (or few) blocks
- XTS-AES is standardized as IEEE P1619
- Supported by myriad disk encryption tools: BestCrypt, dm-crypt, TrueCrypt, VeraCrypt, DiskCryptor, FileVault 2 (MacOS), BitLocker (Windows 10)

Gray Code Masking

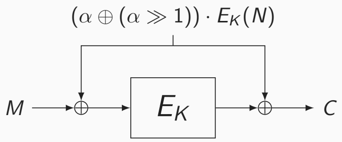
- OCB1 and OCB3 use Gray Codes:



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

Gray Code Masking

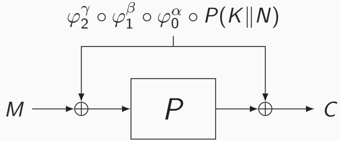
- OCB1 and OCB3 use Gray Codes:



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

Masked Even-Mansour (MEM)

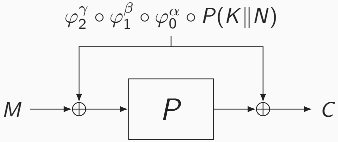
- MEM by Granger et al. [GJMN16]:



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

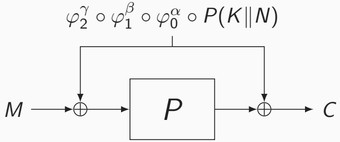
Masked Even-Mansour (MEM)

- MEM by Granger et al. [GJMN16]:



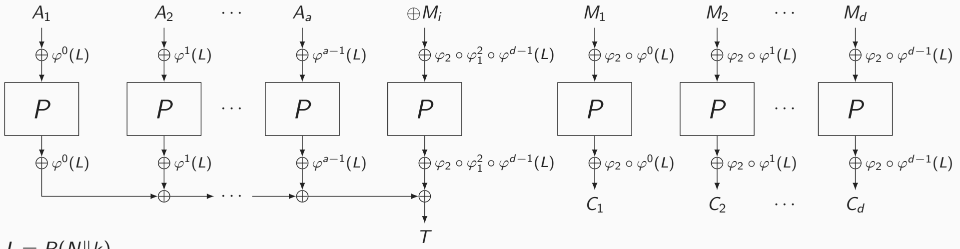
- φ_i are fixed LFSRs, $(\alpha, \beta, \gamma, N)$ is tweak (simplified)
- Combines advantages of:
  - Powering-up masking
  - Word-based LFSRs

- 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

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

NIST Competition

- US NIST recently currently ran competition for lightweight cryptography
- Round 1: 56 submissions in February 2019
- Round 2: 32 submissions in August 2019
- Final round: 10 submissions in March 2021
- Winner Ascon announced February 2023

NIST Competition

- US NIST recently currently ran competition for lightweight cryptography
- Round 1: 56 submissions in February 2019
- Round 2: 32 submissions in August 2019
- Final round: 10 submissions in March 2021
- Winner Ascon announced February 2023

- Some submissions were sponge-based (like Ascon)
- Some submissions used techniques from this lecture