Radbound University

Security of Encryption Modes

Bart Mennink

Radboud University (The Netherlands)

Spring School on Symmetric Cryptography

March 13, 2025

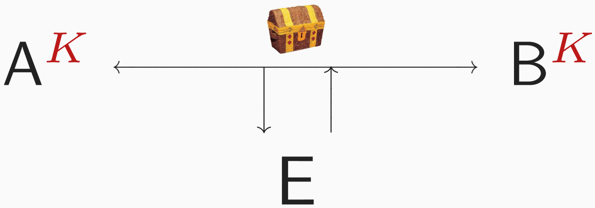
Keyed Symmetric Cryptography

General Setting

$$A^{K} \longleftrightarrow B^{K}$$

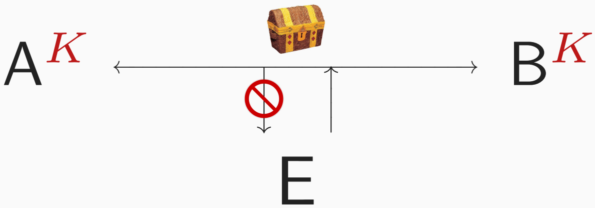
- Two parties, Alice and Bob, communicate over a public channel
    - They have agreed on a joint key K and use it to transmit data

General Setting



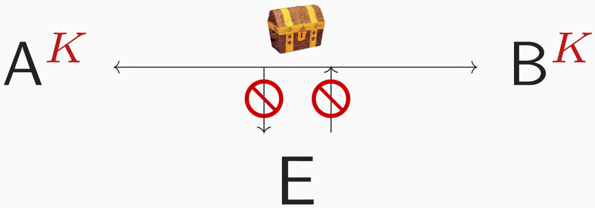
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- In symmetric cryptography, we are concerned with two main security properties:

General Setting



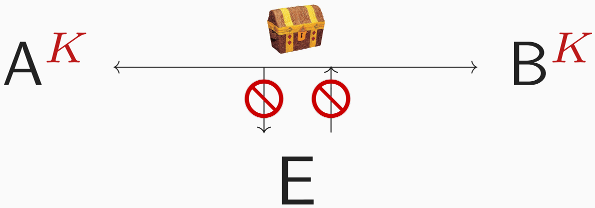
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- In symmetric cryptography, we are concerned with two main security properties:
  - **Confidentiality (or data privacy):** Eve cannot learn anything about data

General Setting



- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- In symmetric cryptography, we are concerned with two main security properties:
  - **Confidentiality (or data privacy):** Eve cannot learn anything about data
  - **Authenticity:** Eve cannot manipulate the data

General Setting



- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- In symmetric cryptography, we are concerned with two main security properties:
  - **Confidentiality (or data privacy):** Eve cannot learn anything about data
  - **Authenticity:** Eve cannot manipulate the data

In this presentation I will mainly focus on confidentiality

One-Time Pad Encryption

Encryption:

$$
\begin{array}{rccccccccccccccc}
M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus
\end{array}
$$

One-Time Pad Encryption

Encryption:

$$
\begin{array}{rccccccccccccccc}
M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \quad \bigoplus \\
\hline
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

One-Time Pad Encryption

Encryption:

$$\begin{array}{rccccccccccccccc} M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus \\ \hline C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Decryption:

$$\begin{array}{rccccccccccccccc} C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus \end{array}$$

One-Time Pad Encryption

Encryption:

$$\begin{array}{rccccccccccccccccl} M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus \\ \hline C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \end{array}$$

Decryption:

$$\begin{array}{rccccccccccccccccl} C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \\ K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus \\ \hline M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \end{array}$$

Properties of One-Time Pad

- One-time pad is a type of stream encryption

Properties of One-Time Pad

- One-time pad is a type of stream encryption
- Perfect secrecy (against an attacker that has no knowledge about the key)
  - Given C, an attacker correctly guesses M with probability $1/2^{|K|}$

Properties of One-Time Pad

- One-time pad is a type of stream encryption
- Perfect secrecy (against an attacker that has no knowledge about the key)
  - Given C, an attacker correctly guesses M with probability $1/2^{|K|}$
- Key must be as long as the plaintext!

Properties of One-Time Pad

- One-time pad is a type of stream encryption
- Perfect secrecy (against an attacker that has no knowledge about the key)
  - Given C, an attacker correctly guesses M with probability $1/2^{|K|}$
- Key must be as long as the plaintext!

Stream Ciphers

- Generate long keystream Z from short key K

Properties of One-Time Pad

- One-time pad is a type of stream encryption
- Perfect secrecy (against an attacker that has no knowledge about the key)
  - Given C, an attacker correctly guesses M with probability $1/2^{|K|}$
- Key must be as long as the plaintext!

Stream Ciphers

- Generate long keystream Z from short key K
- Much more practical!

Stream Encryption

Properties of One-Time Pad

- One-time pad is a type of stream encryption
- Perfect secrecy (against an attacker that has no knowledge about the key)
  - Given C, an attacker correctly guesses M with probability $1/2^{|K|}$
- Key must be as long as the plaintext!

Stream Ciphers

- Generate long keystream Z from short key K
- Much more practical!
- Security degrades:
  1. Key guessing still succeeds with probability $1/2^{|K|}$ but now with shorter key
  2. The stream cipher mechanism is another focal point of attack

Stream Cipher: Vigenère ($\approx$ 1553, Wikipedia)

$K \longrightarrow$ the Pritt stream cipher $\longrightarrow Z = K \| K \| K \| \cdots$

Stream Cipher: Vigenère ($\approx$ 1553, Wikipedia)

$K \longrightarrow$ the Pritt stream cipher $\longrightarrow Z = K \| K \| K \| \cdots$

- Key guessing:
  - Exhaustive key search succeeds with probability $\mathbf{Pr}\left(\text{success}\right) = 1/2^{|K|}$

Stream Cipher: Vigenère (≈ 1553, Wikipedia)

$K \longrightarrow$ the Pritt stream cipher $\longrightarrow Z = K\|K\|K\|\cdots$

- Key guessing:
  - Exhaustive key search succeeds with probability $\mathbf{Pr}\left(\text{success}\right) = 1/2^{|K|}$
- Ciphertext Only Attack:
  - Long ciphertexts leak info via letter frequencies

Stream Cipher: Vigenère (≈ 1553, Wikipedia)

$K \longrightarrow$ the Pritt stream cipher $\longrightarrow Z = K \| K \| K \| \cdots$

- Key guessing:
  - Exhaustive key search succeeds with probability $\mathbf{Pr}\,(\text{success}) = 1/2^{|K|}$
- Ciphertext Only Attack:
  - Long ciphertexts leak info via letter frequencies
- Known Plaintext Attack:
  - Knowledge of short plaintext sequence reveals full keystream

Stream Cipher: Vigenère ($\approx$ 1553, Wikipedia)

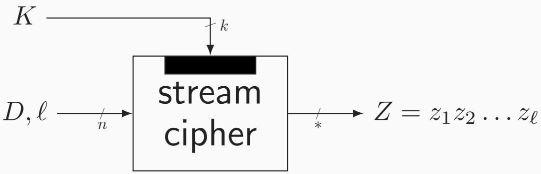
$K \longrightarrow$ the Pritt stream cipher $\longrightarrow Z = K \| K \| K \| \cdots$

- Key guessing:
  - Exhaustive key search succeeds with probability $\mathbf{Pr}\,(\text{success}) = 1/2^{|K|}$
- Ciphertext Only Attack:
  - Long ciphertexts leak info via letter frequencies
- Known Plaintext Attack:
  - Knowledge of short plaintext sequence reveals full keystream

We need something more sophisticated!

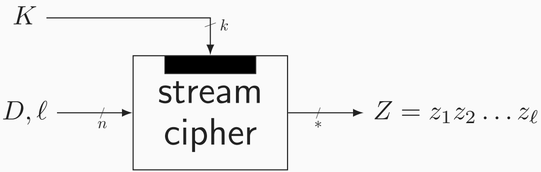
How to Model Security?

Modern Stream Ciphers



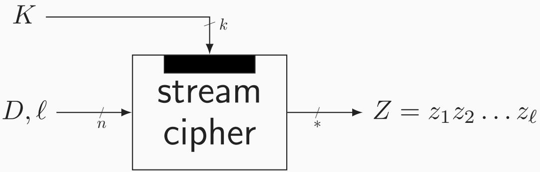
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated

Modern Stream Ciphers



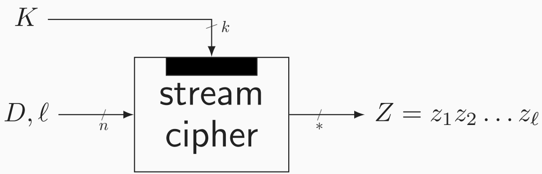
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- The diversifier must be different for each message that is transmitted

Modern Stream Ciphers



- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- The diversifier must be different for each message that is transmitted
- Example: data streams, e.g., pay TV and telephone, often split data in relatively short, numbered, frames. The frame number may serve as diversifier:

$$C_i = M_i \oplus \mathsf{SC}(K, i, |M_i|)$$

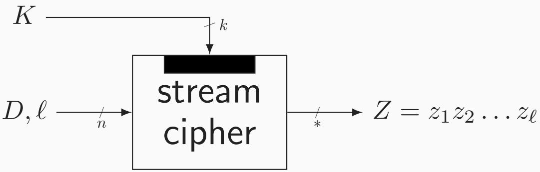


- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- The diversifier must be different for each message that is transmitted
- Example: data streams, e.g., pay TV and telephone, often split data in relatively short, numbered, frames. The frame number may serve as diversifier:

$$C_i = M_i \oplus \mathsf{SC}(K, i, |M_i|)$$

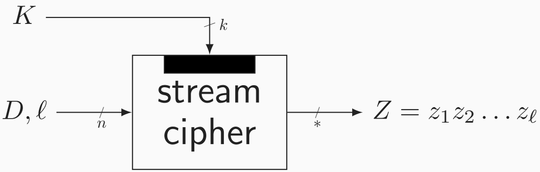
When is a stream cipher strong enough?

Stream Cipher Security, Intuition (1/3)



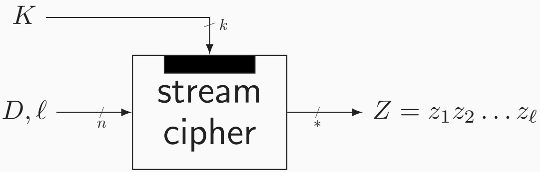
- Kerckhoffs principle: security should be based on secrecy of K
- Thus: attacker knows the algorithm SC

Stream Cipher Security, Intuition (1/3)



- Kerckhoffs principle: security should be based on secrecy of K
- Thus: attacker knows the algorithm SC

- Attacker can also learn some amount of input-output combinations of SC_K
- Intuitively, these data do not expose any irregularities (except for repetition)



- Kerckhoffs principle: security should be based on secrecy of K
- Thus: attacker knows the algorithm SC

- Attacker can also learn some amount of input-output combinations of SC_K
- Intuitively, these data do not expose any irregularities (except for repetition)
- SC_K should behave like a random oracle

Random Oracle

- A database of input-output tuples
- Initially empty

D	Z
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
  - If D is in the database,

D	Z
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$

Intermezzo: Random Oracle

D	Z
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$
$\ldots$	$\ldots$

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database,

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database,

D	Z
1100	101011101010101
...	...
...	...
...	...

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database,

D	Z
1100	101011101010101
1111010101101101	110101
$\ldots$	$\ldots$
$\ldots$	$\ldots$

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database,

D	Z
1100	101011101010101
1111010101101101	110101
001000011100	1010110101110101011
...	...

D	Z
1100	1010111010101 01
1111010101101101	110101
001000011100	10101101011101010 1011
...	...

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$:
    - If $|Z| < \ell$:

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z| < \ell$:

D	Z
1100	101011101010101
1111010101101101	110101
001000011100	10101101011101010101011
...	...

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z| < \ell$:

D	Z
1100	1010111101010101
1111010101101101	110101
001000011100	101011010111010101011
...	...

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z| < \ell$: generate $\ell - |Z|$ random bits Z', append Z' to Z, return $Z \| Z'$

D	Z
1100	101011101010101
1111010101101101	110101
001000011100	10101101011101010101011
...	...

D	Z
1100	101011101010101
1111010101101101	1101011101111101101
001000011100	1010110101110101011
...	...

Random Oracle

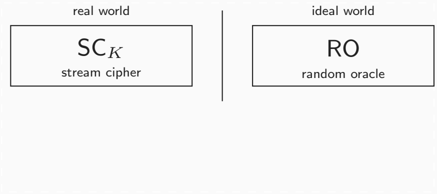
- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z| < \ell$: generate $\ell - |Z|$ random bits Z', append Z' to Z, return $Z \| Z'$

Random Oracle

- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z| < \ell$: generate $\ell - |Z|$ random bits Z', append Z' to Z, return $Z \| Z'$
    - update (D, Z) in the list

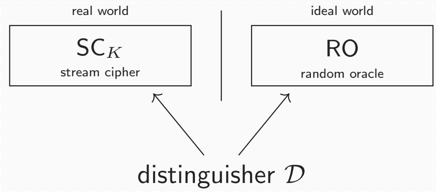
D	Z
1100	101011101010101
1111010101101101	11010111011111101101
001000011100	1010110101110101011
$\ldots$	$\ldots$

Stream Cipher Security, Intuition (2/3)



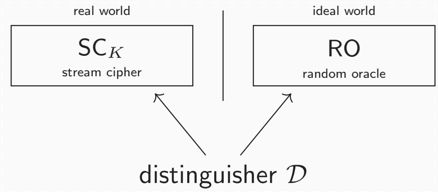
- We thus want to "compare" SC_K with a random oracle RO

Stream Cipher Security, Intuition (2/3)



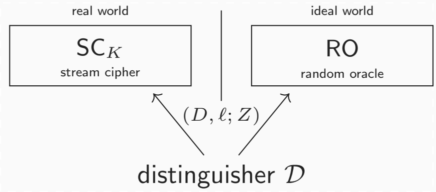
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds

Stream Cipher Security, Intuition (2/3)



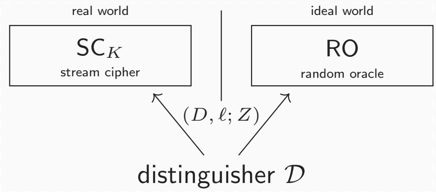
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does a priori not know which oracle it is given access to

Stream Cipher Security, Intuition (2/3)



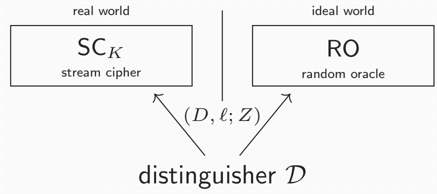
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does a priori not know which oracle it is given access to
  - $\mathcal{D}$ can now make queries (D, ℓ) to receive Z

Stream Cipher Security, Intuition (2/3)



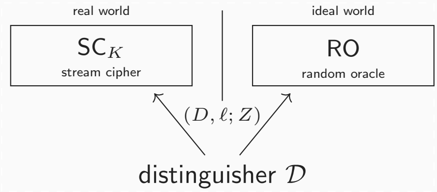
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does a priori not know which oracle it is given access to
  - $\mathcal{D}$ can now make queries (D, ℓ) to receive Z
  - At the end, $\mathcal{D}$ has to guess the outcome of the toss coin (head/tail)

Stream Cipher Security, Intuition (3/3)



- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$

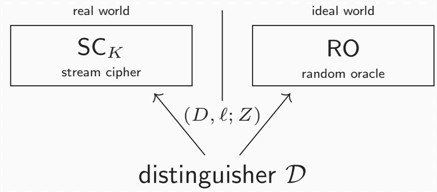
Stream Cipher Security, Intuition (3/3)



- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

$$\mathbf{Adv}(\mathcal{D}) = 2 \cdot \mathbf{Pr}\,(\text{success}) - 1$$

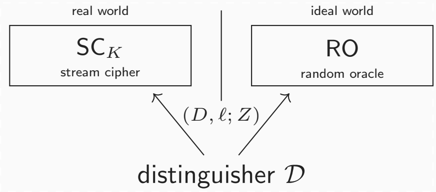
Stream Cipher Security, Intuition (3/3)



- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\left(\text{success}\right)$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

$$\begin{aligned}\mathbf{Adv}\left(\mathcal{D}\right) &= 2 \cdot \mathbf{Pr}\left(\text{success}\right) - 1 \\ &= \mathbf{Pr}\left(\mathcal{D}^{\mathsf{SC}_K} \text{ returns head}\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} \text{ returns head}\right)\end{aligned}$$

Stream Cipher Security, Intuition (3/3)

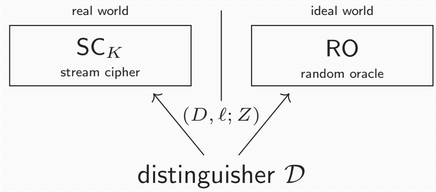


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

$$\begin{aligned}\mathbf{Adv}\,(\mathcal{D}) &= 2 \cdot \mathbf{Pr}\,(\text{success}) - 1 \\ &= \mathbf{Pr}\left(\mathcal{D}^{\mathsf{SC}_K} \text{ returns head}\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} \text{ returns head}\right)\end{aligned}$$

- $\mathcal{D}$ is limited by certain constraints

Stream Cipher Security, Intuition (3/3)

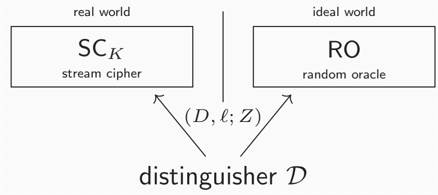


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\left(\text{success}\right)$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

$$\begin{aligned}\mathbf{Adv}(\mathcal{D}) &= 2 \cdot \mathbf{Pr}\left(\text{success}\right) - 1 \\ &= \mathbf{Pr}\left(\mathcal{D}^{\mathsf{SC}_K} \text{ returns head}\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} \text{ returns head}\right)\end{aligned}$$

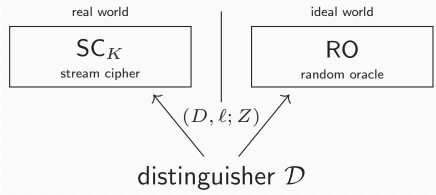
- $\mathcal{D}$ is limited by certain constraints
  - Data (or online) complexity q: total cost of queries $\mathcal{D}$ can make
  - Computation (or time) complexity t: everything that $\mathcal{D}$ can do "on its own"

Stream Cipher Security, Formal



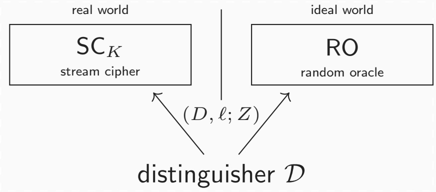
- Two oracles: SC$_K$ (for secret key K) and RO (secret)

Stream Cipher Security, Formal



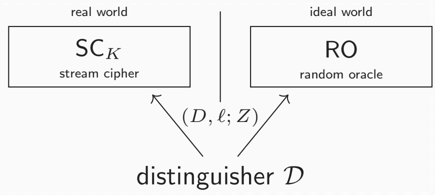
- Two oracles: SC_K (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Stream Cipher Security, Formal



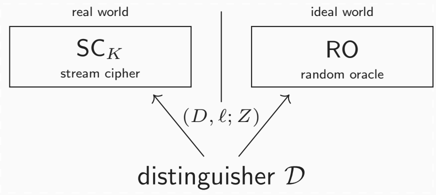
- Two oracles: SC_K (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with

Stream Cipher Security, Formal



- Two oracles: SC_K (for secret key K) and RO (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:

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



- Two oracles: SC_K (for secret key K) and RO (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:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{SC}_K \; ; \; \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{SC}_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

- $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(q, t)$: maximum advantage over any distinguisher with complexity q, t

Generic Stream Cipher Design

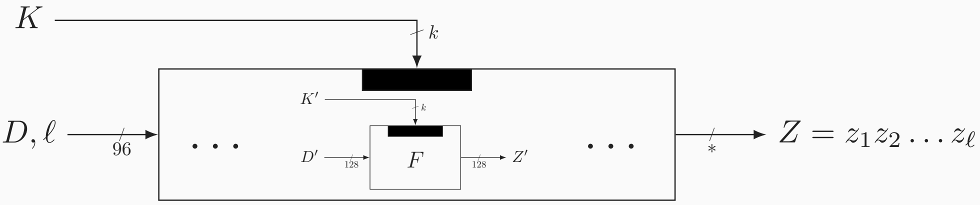
Generic Stream Cipher Design (1/2)

- Classical approach: LFSRs strengthened with non-linear component
- Modern approach: building construction from smaller cryptographic primitive

Generic Stream Cipher Design (1/2)

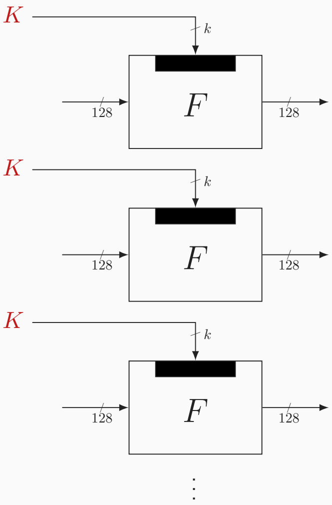
- Classical approach: LFSRs strengthened with non-linear component
- Modern approach: building construction from smaller cryptographic primitive

- Suppose (for the sake of argument):
  - we **know** how to build a strong stream cipher F with fixed-length output
  - we **want** to build a stream cipher with variable-length output



Design

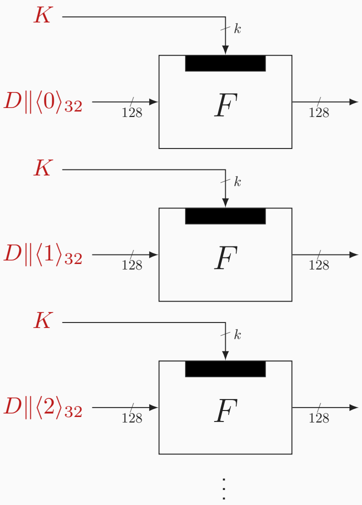
- Feed K to primitive



Generic Stream Cipher Design (2/2)

Design

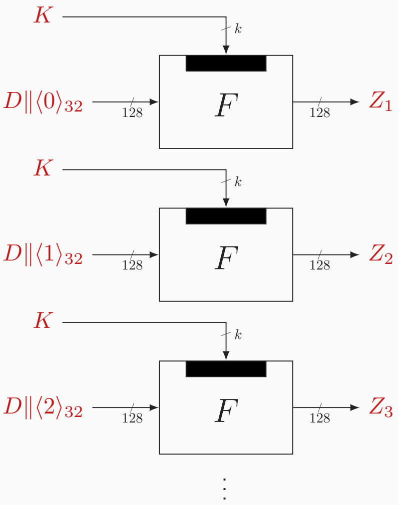
- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter



Design

- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter
- Concatenate outputs:

$$Z = Z_1 \parallel Z_2 \parallel Z_3 \parallel \cdots$$



Generic Stream Cipher Design (2/2)

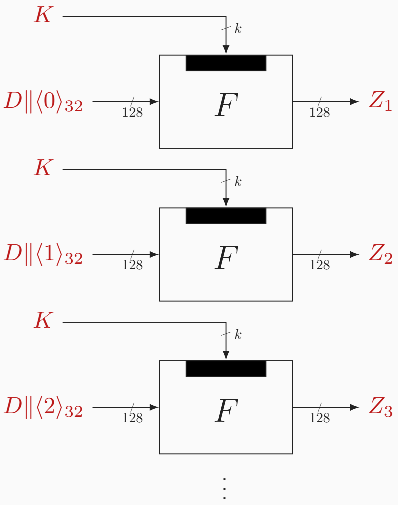
Design

- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter
- Concatenate outputs:

$$Z = Z_1 \parallel Z_2 \parallel Z_3 \parallel \cdots$$

Security

- If F_K is hard to distinguish from a RO′



Generic Stream Cipher Design (2/2)

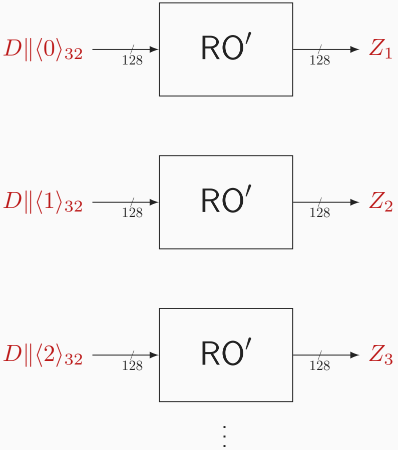
Design

- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter
- Concatenate outputs:

$$Z = Z_1 \parallel Z_2 \parallel Z_3 \parallel \cdots$$

Security

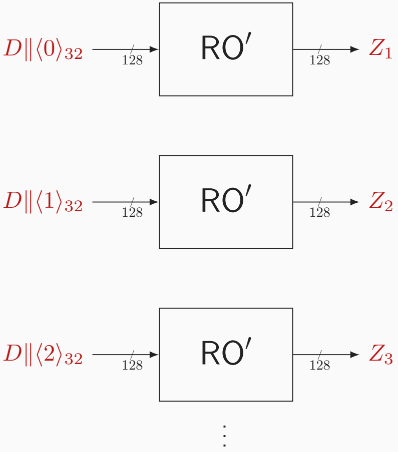
- If F_K is hard to distinguish from a RO′



Design

- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter
- Concatenate outputs:

$$Z = Z_1 \parallel Z_2 \parallel Z_3 \parallel \cdots$$



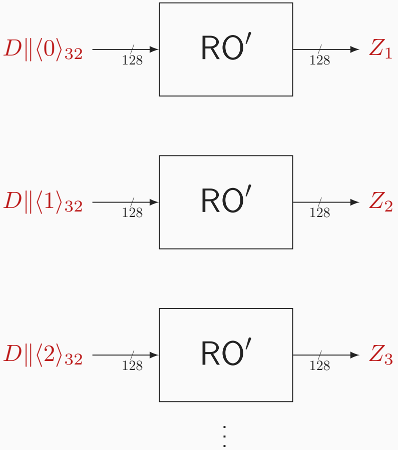
Security

- If F_K is hard to distinguish from a RO′
- Then construction is hard to distinguish from RO

Design

- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with counter
- Concatenate outputs:

$$Z = Z_1 \parallel Z_2 \parallel Z_3 \parallel \cdots$$



Security

- If F_K is hard to distinguish from a RO′
- Then construction is hard to distinguish from RO
- For the purists: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}[F]}(q,t) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q,t')$

Design

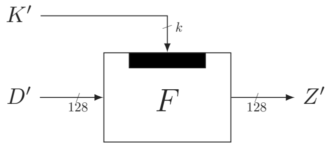
- Feed K to primitive
- Evaluate primitive as
  concatenated with co
- Concatenate outputs:

  $Z = Z_1$

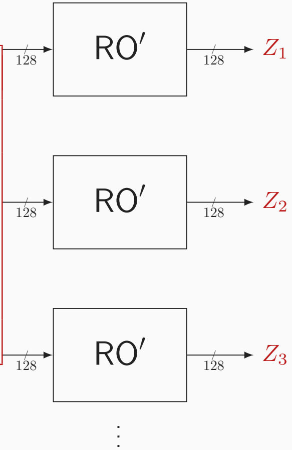
Security

- If F_K is hard to disti
- Then construction is hard to distinguish from RO
- For the purists: $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}[F]}(q,t) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q,t')$

Unfortunately, we do not know how to easily construct a function

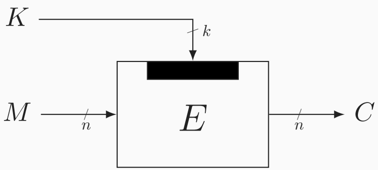


that behaves like a RO′



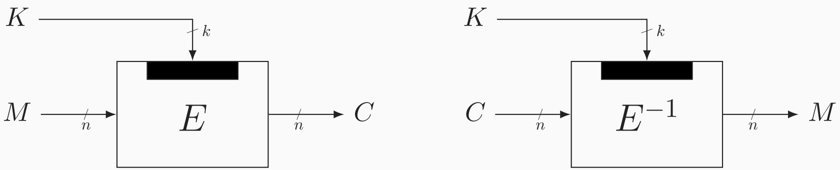
Block Ciphers

Block Ciphers

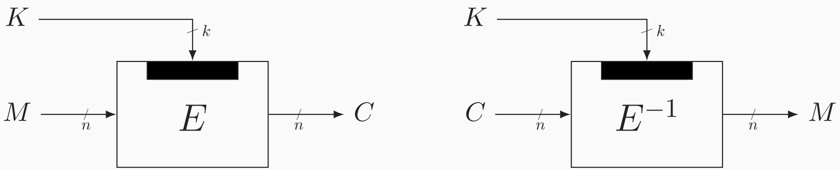


- Using key K, message M is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size

Block Ciphers



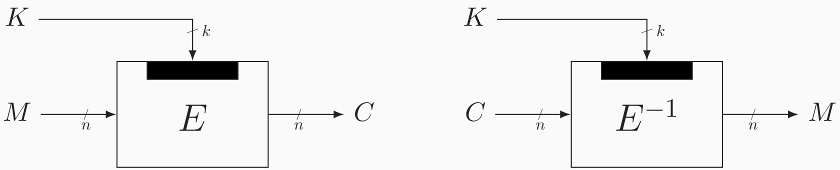
- Using key K, message M is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}



- Using key K, message M is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}
- Example [DR02]:

$$\text{AES-128}\colon \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$$
$$(K, M) \mapsto C$$

Block Ciphers

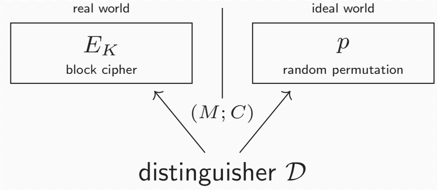


- Using key K, message M is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}
- Example [DR02]:

$$\begin{aligned} \text{AES-128}\colon \{0,1\}^{128} \times \{0,1\}^{128} &\to \{0,1\}^{128} \\ (K, M) &\mapsto C \end{aligned}$$

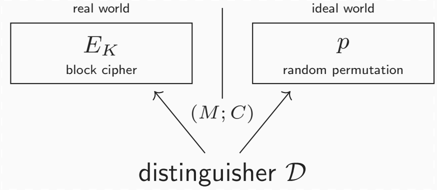
- A good block cipher should behave like a random permutation

Block Cipher Security



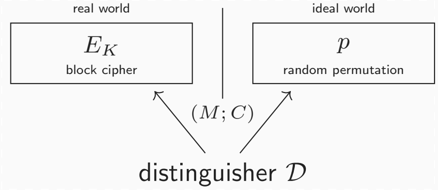
- Two oracles: E_K (for secret key K) and p (secret)

Block Cipher Security

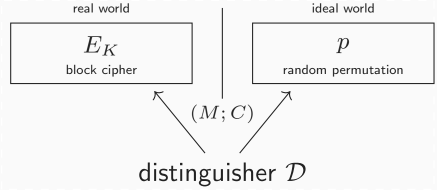


- Two oracles: E_K (for secret key K) and p (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Block Cipher Security



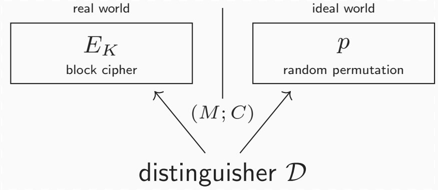
- Two oracles: E_K (for secret key K) and p (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with



- Two oracles: E_K (for secret key K) and 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:

$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(E_K\ ;\ p\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{E_K}=1\right) - \mathbf{Pr}\left(\mathcal{D}^{p}=1\right)\right|$$

Block Cipher Security



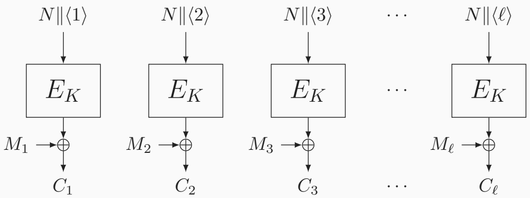
- Two oracles: E_K (for secret key K) and p (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(E_K\ ;\ p\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{E_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{p} = 1\right)\right|$$

- $\mathbf{Adv}_E^{\mathrm{prp}}(q,t)$: maximum advantage over any $\mathcal{D}$ with query/time complexity q/t

Counter Mode Encryption

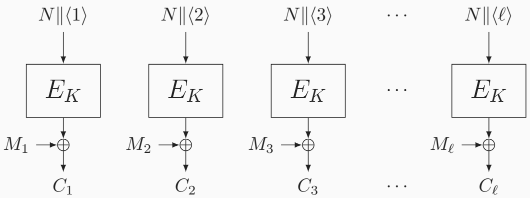
Counter (CTR) Mode



Features

- Stream-based encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple
- Decryption needs no E_K^{-1}

Counter (CTR) Mode



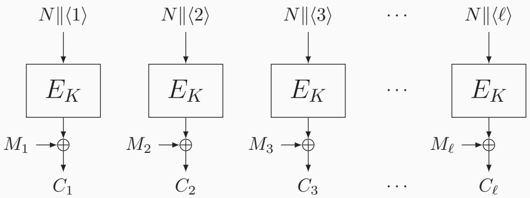
Features

- Stream-based encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple
- Decryption needs no E_K^{-1}

Security

- "Hopefully" secure as long as N is never repeated and E_K is a secure PRP

Counter (CTR) Mode



Features

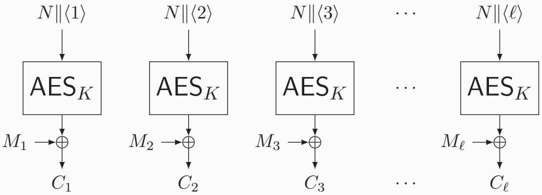
- Stream-based encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple
- Decryption needs no E_K^{-1}

Security

- "Hopefully" secure as long as N is never repeated and E_K is a secure PRP
- Let us investigate that!

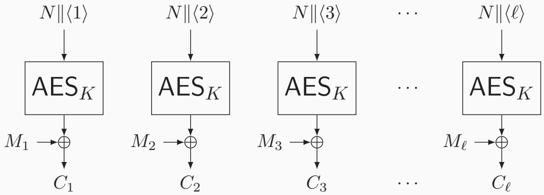
Security of Counter Mode Based on AES

- Let us consider counter mode based on AES: CTR[AES_K]



Security of Counter Mode Based on AES

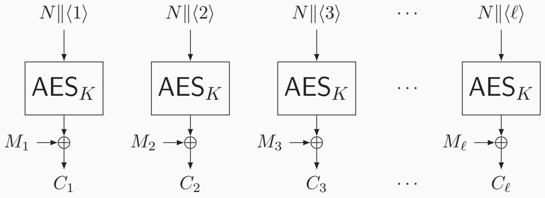
- Let us consider counter mode based on AES: CTR[AES$_K$]



- We focus on the keystream generation portion

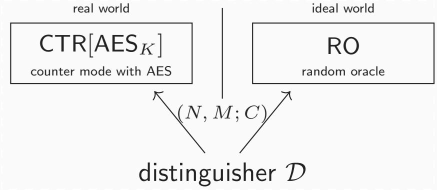
Security of Counter Mode Based on AES

- Let us consider counter mode based on AES: CTR[AES$_K$]



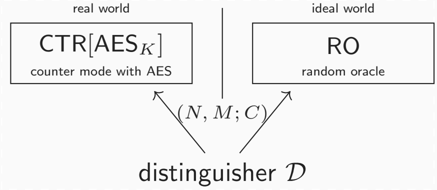
- We focus on the keystream generation portion
- Assumptions
  - Distinguisher never repeats nonce N
  - AES itself is sufficiently secure: $\mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(q, t)$ is small

Security of Counter Mode Based on AES: Model



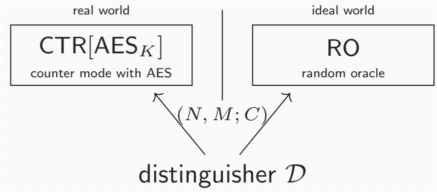
- Two oracles: CTR[AES$_K$] (for secret key K) and RO (secret)

Security of Counter Mode Based on AES: Model



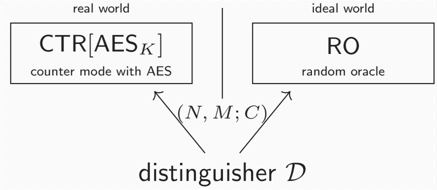
- Two oracles: CTR[AES$_K$] (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Security of Counter Mode Based on AES: Model



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

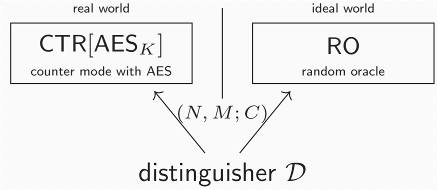
Security of Counter Mode Based on AES: Model



- Two oracles: CTR[AES$_K$] (for secret key K) and RO (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:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR}[\mathsf{AES}_K]\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR}[\mathsf{AES}_K]} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

Security of Counter Mode Based on AES: Model

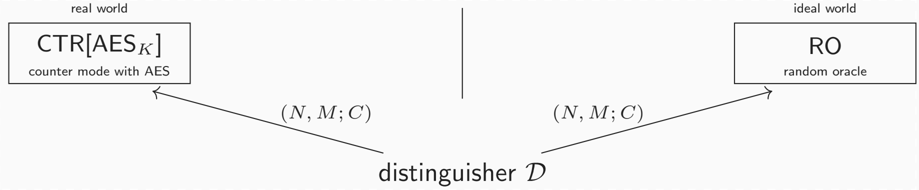


- Two oracles: CTR[AES$_K$] (for secret key K) and RO (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR}[\mathsf{AES}_K]\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR}[\mathsf{AES}_K]} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

- $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(q, t)$: maximum advantage over any $\mathcal{D}$ with q/t blocks/time

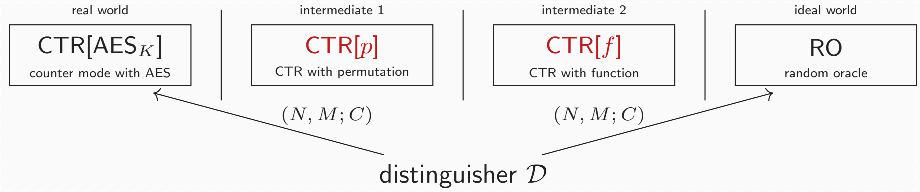
Proof: Overview



- For any (fixed) distinguisher $\mathcal{D}$ (later, we maximize over all), we have to bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR[AES}_K\mathsf{]}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

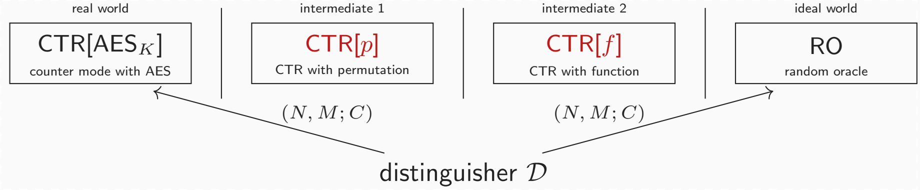
Proof: Overview



- For any (fixed) distinguisher $\mathcal{D}$ (later, we maximize over all), we have to bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR[AES}_K\mathsf{]}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

- We add intermediate worlds CTR[p] and CTR[f] for random p and f



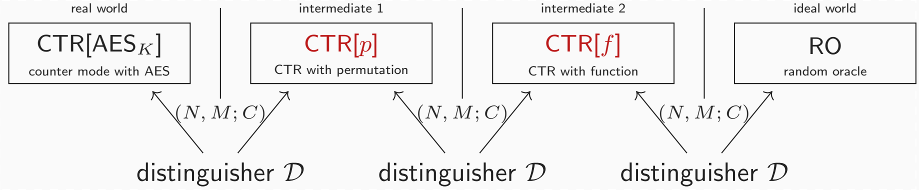
- For any (fixed) distinguisher $\mathcal{D}$ (later, we maximize over all), we have to bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR[AES}_K\mathsf{]}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

- We add intermediate worlds CTR[p] and CTR[f] for random p and f
- By the triangle inequality:

$$\Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right) \le \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{CTR}[p]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[p]\ ;\ \mathsf{CTR}[f]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[f]\ ;\ \mathsf{RO}\right)$$

Proof: Overview



- For any (fixed) distinguisher $\mathcal{D}$ (later, we maximize over all), we have to bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(\mathsf{CTR}[\mathsf{AES}_K]\ ;\ \mathsf{RO}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{CTR}[\mathsf{AES}_K]} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathsf{RO}} = 1\right)\right|$$

- We add intermediate worlds $\mathsf{CTR}[p]$ and $\mathsf{CTR}[f]$ for random p and f
- By the triangle inequality:

$$\Delta_{\mathcal{D}}\left(\mathsf{CTR}[\mathsf{AES}_K]\ ;\ \mathsf{RO}\right) \le \Delta_{\mathcal{D}}\left(\mathsf{CTR}[\mathsf{AES}_K]\ ;\ \mathsf{CTR}[p]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[p]\ ;\ \mathsf{CTR}[f]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[f]\ ;\ \mathsf{RO}\right)$$

Proof: From $\mathsf{CTR}[\mathsf{AES}_K]$ to $\mathsf{CTR}[p]$

- $\mathcal{D}$'s goal: distinguish $\mathsf{CTR}[\mathsf{AES}_K]$ from $\mathsf{CTR}[p]$

Proof: From CTR[AES_K] to CTR[p]

- $\mathcal{D}$'s goal: distinguish CTR[AES_K] from CTR[p]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power

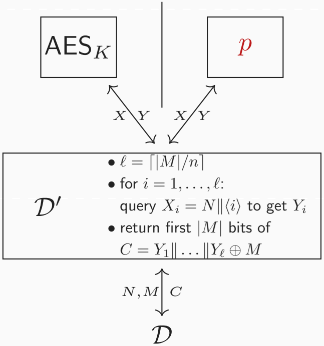
Proof: From CTR[AES_K] to CTR[p]

- $\mathcal{D}$'s goal: distinguish CTR[AES_K] from CTR[p]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish AES_K from p

Proof: From CTR[AES_K] to CTR[p]

- $\mathcal{D}$'s goal: distinguish CTR[AES_K] from CTR[p]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish AES_K from p

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess



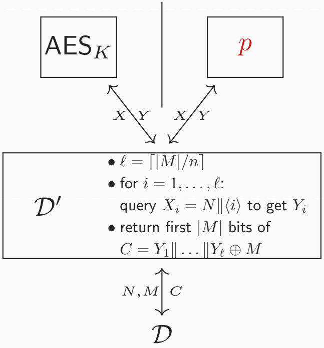
Proof: From CTR[AES_K] to CTR[p]

- $\mathcal{D}$'s goal: distinguish CTR[AES_K] from CTR[p]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish AES_K from p

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess

- $\mathcal{D}'$ success probability turns out to be at least that of $\mathcal{D}$:

$$\Delta_{\mathcal{D}}\,(\text{CTR}[\text{AES}_K]\ ;\ \text{CTR}[p]) \leq \Delta_{\mathcal{D}'}\,(\text{AES}_K\ ;\ p)$$

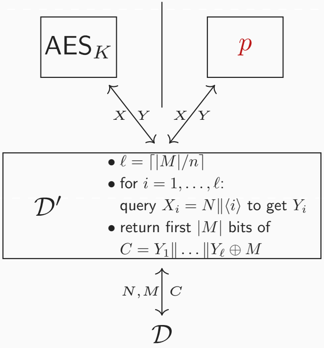


Proof: From CTR[AES_K] to CTR[p]

- $\mathcal{D}$'s goal: distinguish CTR[AES_K] from CTR[p]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish AES_K from p

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess

- $\mathcal{D}'$ success probability turns out to be at least that of $\mathcal{D}$:
  $\Delta_{\mathcal{D}}\left(\mathrm{CTR}[\mathsf{AES}_K]\ ;\ \mathrm{CTR}[p]\right) \leq \Delta_{\mathcal{D}'}\left(\mathsf{AES}_K\ ;\ p\right)$
- But we have seen this distance before:
  $\Delta_{\mathcal{D}'}\left(\mathsf{AES}_K\ ;\ p\right) = \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(\mathcal{D}') \leq \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(q, t')$
  (t' slightly larger than t)



Proof: From $\mathsf{CTR}[p]$ to $\mathsf{CTR}[f]$ (1/2)

- $\mathcal{D}$'s goal: distinguish $\mathsf{CTR}[p]$ from $\mathsf{CTR}[f]$

Proof: From $\mathsf{CTR}[p]$ to $\mathsf{CTR}[f]$ (1/2)

- $\mathcal{D}$'s goal: distinguish $\mathsf{CTR}[p]$ from $\mathsf{CTR}[f]$
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power

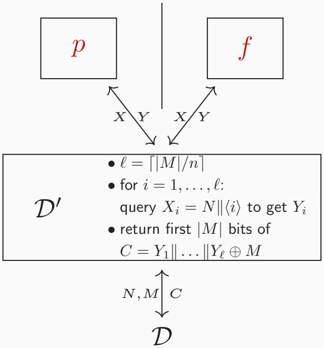
Proof: From CTR$[p]$ to CTR$[f]$ (1/2)

- $\mathcal{D}$'s goal: distinguish CTR$[p]$ from CTR$[f]$
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish p from f

Proof: From CTR[p] to CTR[f] (1/2)

- $\mathcal{D}$'s goal: distinguish CTR[p] from CTR[f]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish p from f

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess



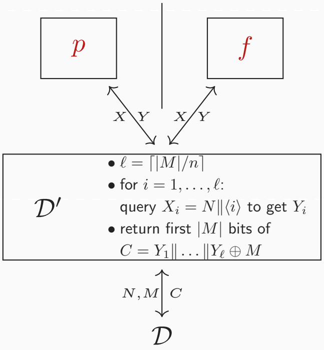
Proof: From CTR$[p]$ to CTR$[f]$ (1/2)

- $\mathcal{D}$'s goal: distinguish CTR$[p]$ from CTR$[f]$
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish p from f

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess

- $\mathcal{D}'$ success probability turns out to be at least that of $\mathcal{D}$:

  $\Delta_{\mathcal{D}}\left(\text{CTR}[p]\ ;\ \text{CTR}[f]\right) \leq \Delta_{\mathcal{D}'}\left(p\ ;\ f\right)$



Proof: From CTR[p] to CTR[f] (1/2)

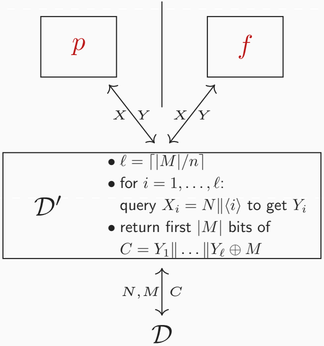
- $\mathcal{D}$'s goal: distinguish CTR[p] from CTR[f]
- We replace $\mathcal{D}$ by a distinguisher $\mathcal{D}'$ that has more power
- $\mathcal{D}'$'s goal: distinguish p from f

- $\mathcal{D}'$ simulates the oracles of $\mathcal{D}$:
- Once $\mathcal{D}$ makes its final guess, $\mathcal{D}'$ makes the same guess

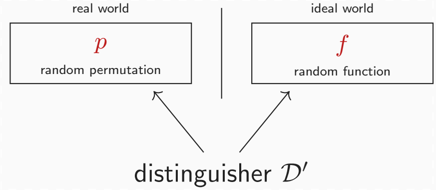
- $\mathcal{D}'$ success probability turns out to be at least that of $\mathcal{D}$:

$$\Delta_{\mathcal{D}}(\text{CTR}[p]\ ;\ \text{CTR}[f]) \leq \Delta_{\mathcal{D}'}(p\ ;\ f)$$

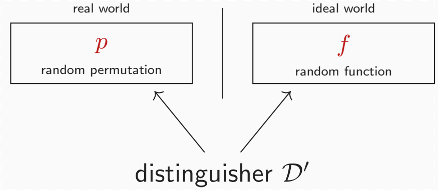
- This is a well-known distance, called the RP-RF switch



Proof: From CTR$[p]$ to CTR$[f]$ (2/2)

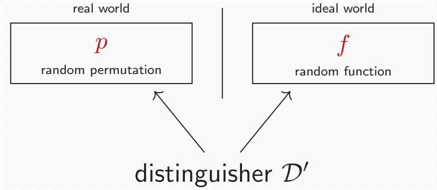


- Distinguisher $\mathcal{D}'$ gets q random n-bit samples:
  - real world: without replacement
  - ideal world: with replacement

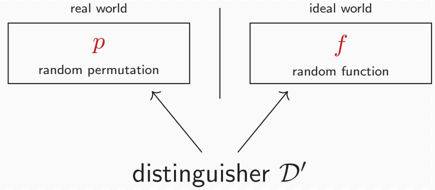


- Distinguisher $\mathcal{D}'$ gets q random n-bit samples:
  - real world: without replacement
  - ideal world: with replacement
- The two worlds can only be distinguished if f ever outputs colliding samples

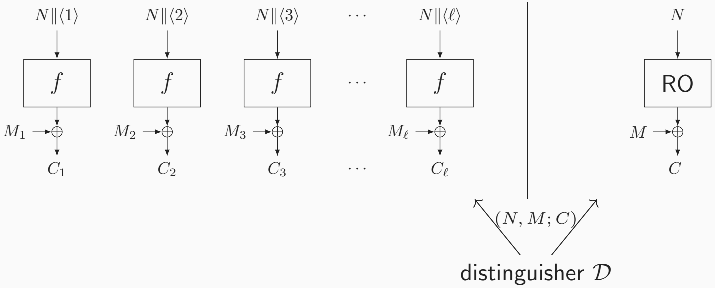
Proof: From CTR[p] to CTR[f] (2/2)



- Distinguisher $\mathcal{D}'$ gets q random n-bit samples:
  - real world: without replacement
  - ideal world: with replacement
- The two worlds can only be distinguished if f ever outputs colliding samples
- This happens with probability at most $\binom{q}{2}/2^n$

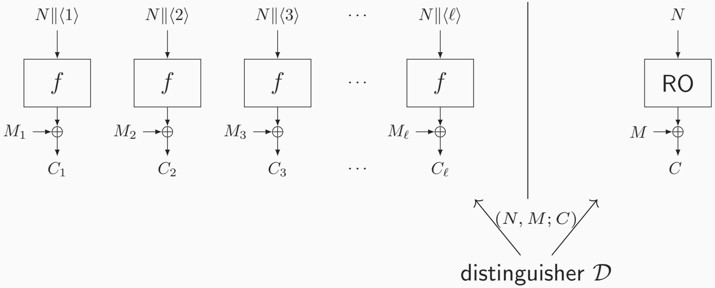


- Distinguisher $\mathcal{D}'$ gets q random n-bit samples:
  - real world: without replacement
  - ideal world: with replacement
- The two worlds can only be distinguished if f ever outputs colliding samples
- This happens with probability at most $\binom{q}{2}/2^n$
- Hence: $\Delta_{\mathcal{D}'}(p\ ;\ f) \leq \binom{q}{2}/2^n$



- In real world: f is a random function that is never evaluated for repeated $N\|\langle i\rangle$
- In ideal world: RO is a random oracle that is never evaluated for repeated N

Proof: From CTR$[f]$ to RO



- In real world: f is a random function that is never evaluated for repeated $N\|\langle i\rangle$
- In ideal world: RO is a random oracle that is never evaluated for repeated N
- Hence: $\Delta_{\mathcal{D}}\left(\text{CTR}[f]\ ;\ \text{RO}\right) = 0$

Proof: Conclusion

- Recall goal: bounding $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D})$ for any $\mathcal{D}$ querying q blocks in t time

Proof: Conclusion

- Recall goal: bounding $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D})$ for any $\mathcal{D}$ querying q blocks in t time
- From the triangle inequality and bounds on the three individual terms:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) &= \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right) \\
&\leq \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{CTR}[p]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[p]\ ;\ \mathsf{CTR}[f]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[f]\ ;\ \mathsf{RO}\right) \\
&\leq \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(q, t') + \binom{q}{2}/2^n + 0
\end{aligned}$$

Proof: Conclusion

- Recall goal: bounding $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D})$ for any $\mathcal{D}$ querying q blocks in t time
- From the triangle inequality and bounds on the three individual terms:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(\mathcal{D}) &= \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{RO}\right)\\ &\leq \Delta_{\mathcal{D}}\left(\mathsf{CTR[AES}_K\mathsf{]}\ ;\ \mathsf{CTR}[p]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[p]\ ;\ \mathsf{CTR}[f]\right) + \Delta_{\mathcal{D}}\left(\mathsf{CTR}[f]\ ;\ \mathsf{RO}\right)\\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(q,t') + \binom{q}{2}/2^n + 0\end{aligned}$$

- As this reasoning holds for all distinguishers $\mathcal{D}$ querying q blocks in t time, we obtain:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[AES]}}(q,t) \leq \mathbf{Adv}^{\mathrm{prp}}_{\mathsf{AES}}(q,t') + \binom{q}{2}/2^n$$

Beyond Birthday Bound Security

Birthday Paradox

For a random selection of 23 people, with a probability at least 50% two of them share the same birthday

For a random selection of 23 people, with a probability at least 50% two of them share the same birthday

General Birthday Paradox

- Consider space $\mathcal{S} = \{0,1\}^n$
- Randomly draw q elements from $\mathcal{S}$
- Expected number of collisions:

$$\mathbf{Ex}\left[\text{collisions}\right] = \binom{q}{2}/2^n$$

For a random selection of 23 people, with a probability at least 50% two of them share the same birthday

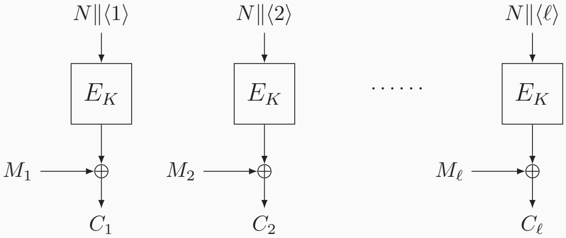
General Birthday Paradox

- Consider space $\mathcal{S} = \{0,1\}^n$
- Randomly draw q elements from $\mathcal{S}$
- Expected number of collisions:

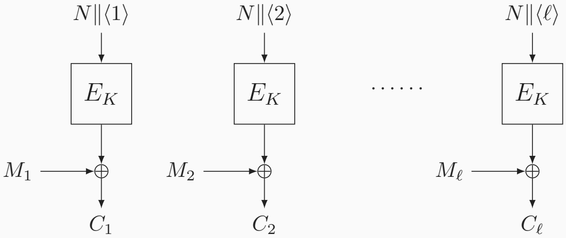
$$\mathbf{Ex}\left[\text{collisions}\right] = \binom{q}{2} / 2^n$$

- Important phenomenon in cryptography

Counter Mode Based on Pseudorandom Permutation



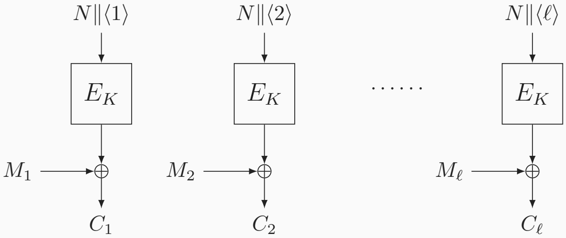
Counter Mode Based on Pseudorandom Permutation



- Security bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[E]}(q,t) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(q,t') + \binom{q}{2}/2^n$$

Counter Mode Based on Pseudorandom Permutation

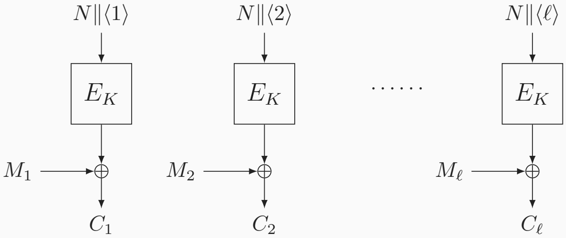


- Security bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[E]}(q,t) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}(q,t') + \binom{q}{2}/2^n$$

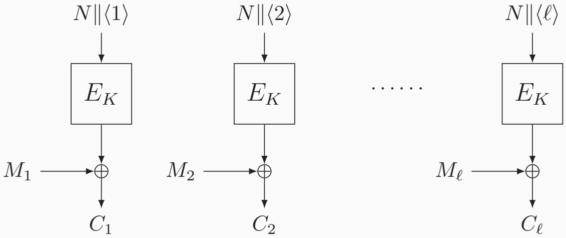
- $\mathsf{CTR}[E]$ is secure as long as:
  - E_K is a secure PRP
  - Number of encrypted blocks $q \ll 2^{n/2}$

Counter Mode Based on Pseudorandom Permutation



- $M_i \oplus C_i$ is distinct for all q blocks
- Unlikely to happen for random string

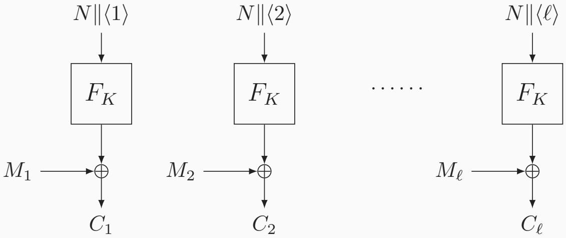
Counter Mode Based on Pseudorandom Permutation



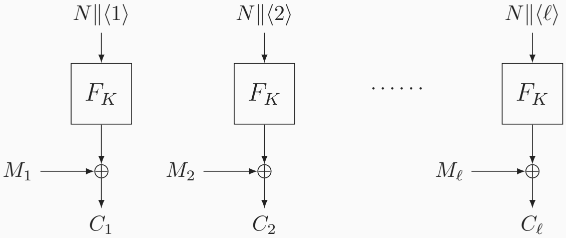
- $M_i \oplus C_i$ is distinct for all q blocks
- Unlikely to happen for random string
- Distinguishing attack in $q \approx 2^{n/2}$ blocks:

$$\binom{q}{2}/2^n \lesssim \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[E]}(q,t)$$

Counter Mode Based on Pseudorandom Function



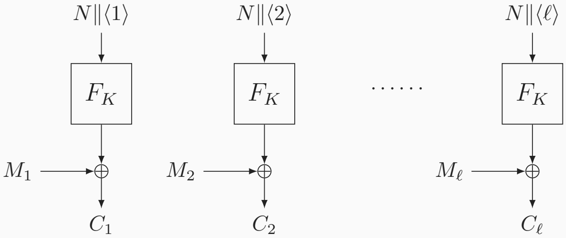
Counter Mode Based on Pseudorandom Function



- Security bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[F]}(q,t) \leq \mathbf{Adv}^{\mathrm{prf}}_{F}(q,t')$$

Counter Mode Based on Pseudorandom Function

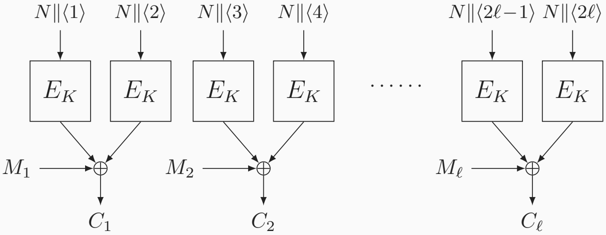


- Security bound:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[F]}(q,t) \le \mathbf{Adv}^{\mathrm{prf}}_{F}(q,t')$$

- $\mathsf{CTR}[F]$ is secure as long as F_K is a secure PRF
- Birthday bound security loss disappeared

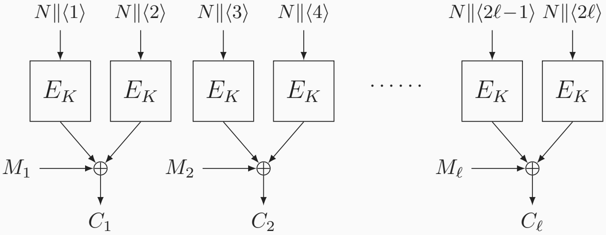
Counter Mode Based on XoP



- Security bound [Pat08a, DHT17]:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[XoP]}}(q,t) \leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{XoP}}(q,t')$$

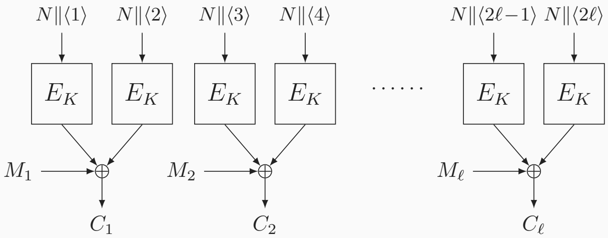
Counter Mode Based on XoP



- Security bound [Pat08a, DHT17]:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[XoP]}}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{XoP}}(q,t') \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{E}(2q,t'') + q/2^n\end{aligned}$$

Counter Mode Based on XoP

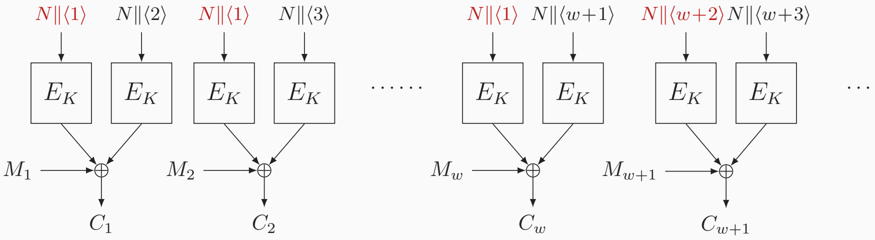


- Security bound [Pat08a, DHT17]:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR[XoP]}}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{XoP}}(q,t') \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{E}(2q,t'') + q/2^n\end{aligned}$$

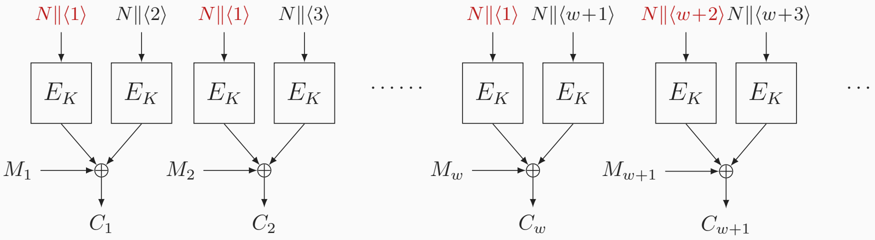
- Beyond birthday bound but 2x as expensive as CTR[E]

CENC by Iwata [Iwa06]



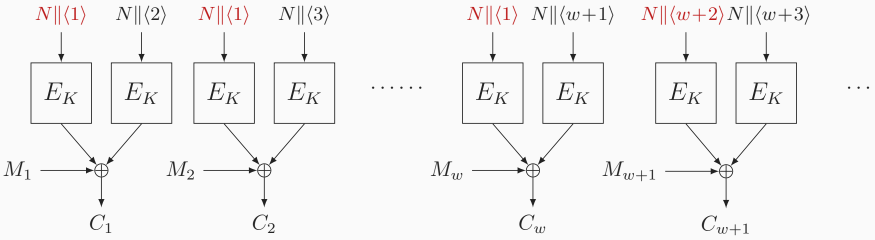
- One subkey used for $w \geq 1$ encryptions

CENC by Iwata [Iwa06]



- One subkey used for $w \geq 1$ encryptions
- Almost as expensive as CTR$[E]$

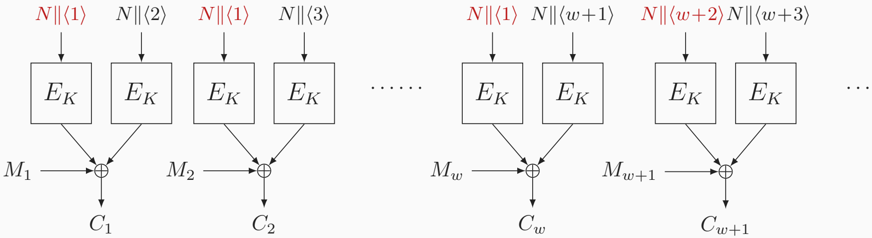
CENC by Iwata [Iwa06]



- One subkey used for $w \geq 1$ encryptions
- Almost as expensive as CTR$[E]$
- Security bound [IMV16]:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[\mathsf{XoP}[w]]}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{XoP}[w]}(q,t') \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{E}((w+1)q,t'') + wq/2^n\end{aligned}$$

CENC by Iwata [Iwa06]



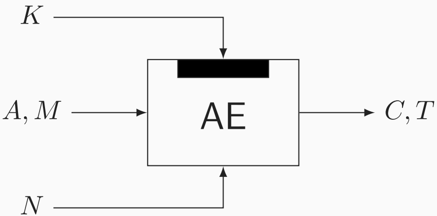
- One subkey used for $w \geq 1$ encryptions
- Almost as expensive as CTR$[E]$
- Security bound [IMV16]:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CTR}[\mathsf{XoP}[w]]}(q,t) &\leq \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{XoP}[w]}(q,t') \\ &\leq \mathbf{Adv}^{\mathrm{prp}}_{E}((w+1)q,t'') + wq/2^n\end{aligned}$$

- Security of XoP and XoP$[w]$ can be proven using mirror theory [Pat03]

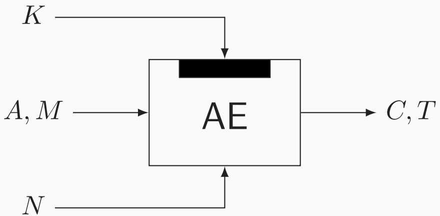
Authenticated Encryption and GCM

Authenticated Encryption



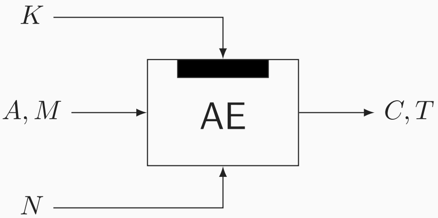
- Using key K:
  - Message M is encrypted in ciphertext C
  - Associated data A and message M are authenticated using T

Authenticated Encryption



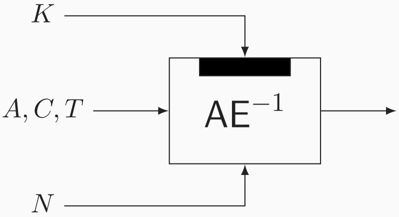
- Using key K:
  - Message M is encrypted in ciphertext C
  - Associated data A and message M are authenticated using T
- Nonce N randomizes the scheme

Authenticated Encryption



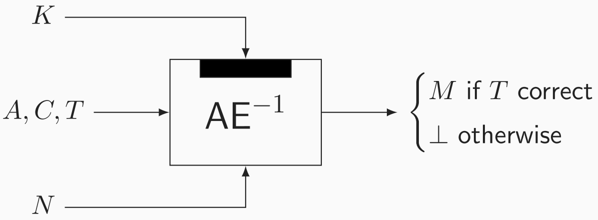
- Using key K:
  - Message M is encrypted in ciphertext C
  - Associated data A and message M are authenticated using T
- Nonce N randomizes the scheme
- Key, nonce, and tag are typically of fixed size
- Associated data, message, and ciphertext could be arbitrary length

Authenticated Decryption



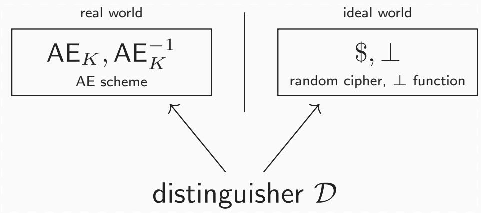
- Authenticated decryption needs to satisfy that
  - Message disclosed if tag is correct
  - Message is not leaked if tag is incorrect

Authenticated Decryption



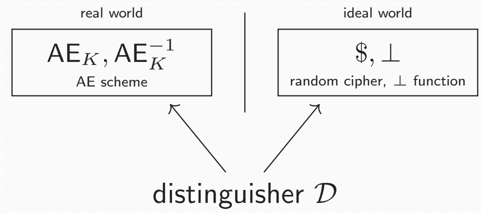
- Authenticated decryption needs to satisfy that
  - Message disclosed if tag is correct
  - Message is not leaked if tag is incorrect

Authenticated Encryption Security



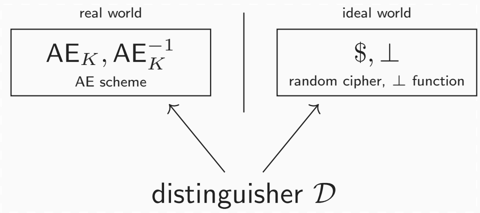
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\$, \perp)$ (secret)

Authenticated Encryption Security



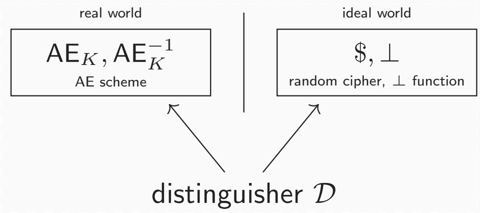
- Two oracles: (AE_K, AE_K^{-1}) (for secret key K) and $(\$, \perp)$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ unique nonce for each encryption query, and no trivial queries

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
  $\to$ unique nonce for each encryption query, and no trivial queries
- $\mathcal{D}$ tries to determine which oracle it communicates with

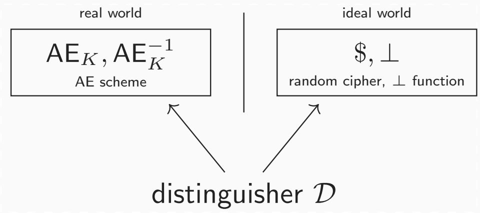
Authenticated Encryption Security



- 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:

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

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:

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

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

Universal Hash Functions

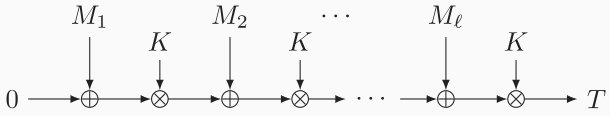
- Consider hash function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$
- H is ε-XOR-universal if $\mathbf{Pr}_K\left(H_K(M) \oplus H_K(M') = T\right) \leq \varepsilon \quad (\forall\ M \neq M', T)$

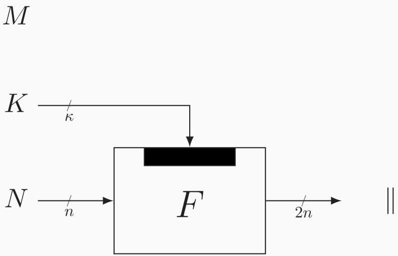
Universal Hash Functions

- Consider hash function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$
- H is ε-XOR-universal if $\mathbf{Pr}_K\left(H_K(M) \oplus H_K(M') = T\right) \le \varepsilon \quad (\forall\ M \neq M', T)$

GHASH

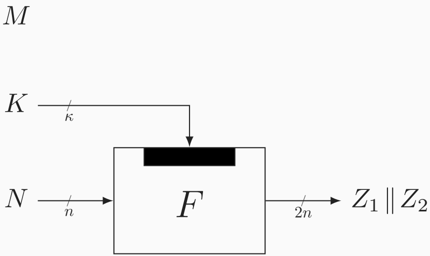
- Addition and multiplication over finite field
- $\ell 2^{-t}$-XOR-universal [MV04]





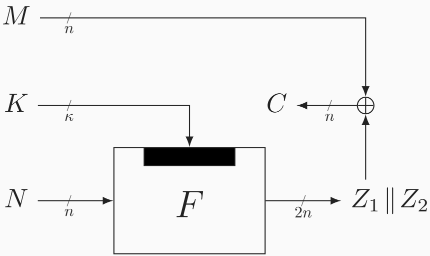
Encryption

- Input: (N, M)



Encryption

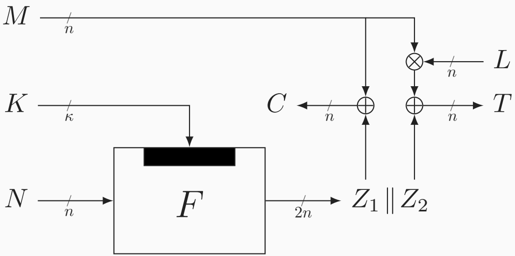
- Input: (N, M)
- Compute keystream $Z_1 \parallel Z_2$



Encryption

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

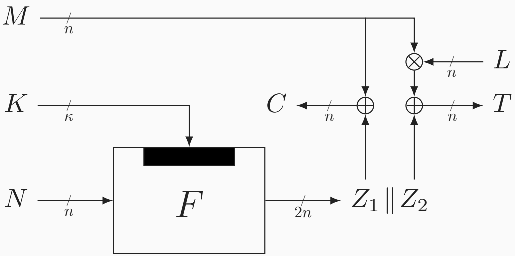
Simple Example



Encryption

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

Simple Example



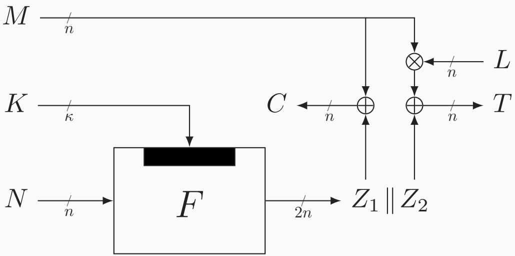
Encryption

- 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)

Simple Example



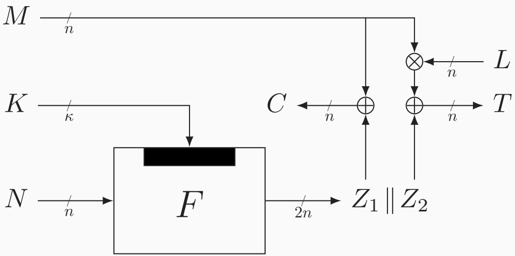
Encryption

- 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 $M = Z_1 \oplus C$

Simple Example



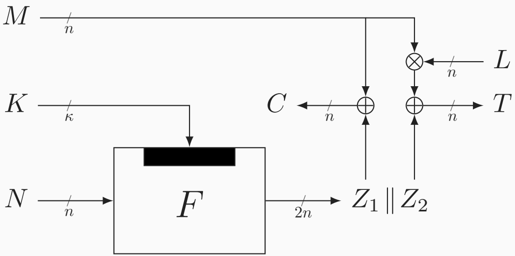
Encryption

- 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 $T^\star = Z_2 \oplus (M \otimes L)$

Simple Example

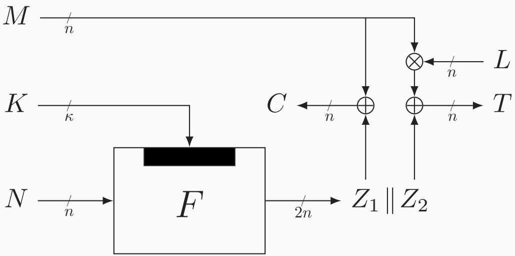


Encryption

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

Decryption

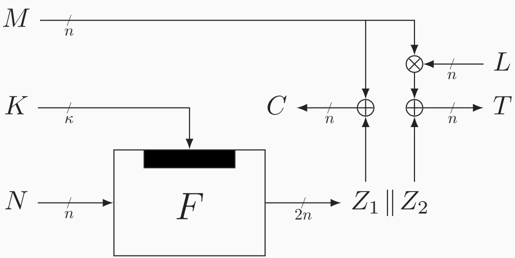
- Input: (N, C, T)
- Compute keystream $Z_1 \parallel Z_2$
- Compute $M = Z_1 \oplus C$
- Compute $T^\star = Z_2 \oplus (M \otimes L)$
- Output: $\begin{cases} M \text{ if } T = T^\star \\ \perp \text{ otherwise} \end{cases}$



Confidentiality

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

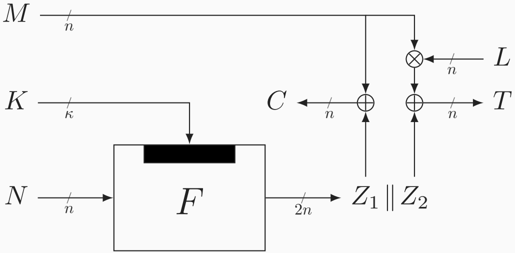
Simple Example: Confidentiality



Confidentiality

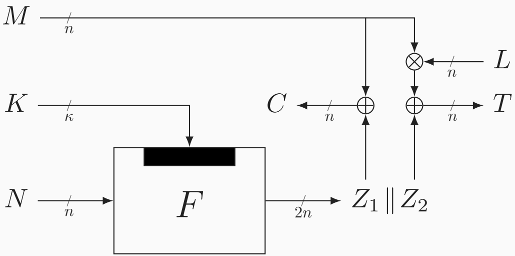
- Consider new query (N, M)
- N should be fresh
- Random $Z_1 \parallel Z_2$
  (if F is a good stream cipher)

Simple Example: Confidentiality



Confidentiality

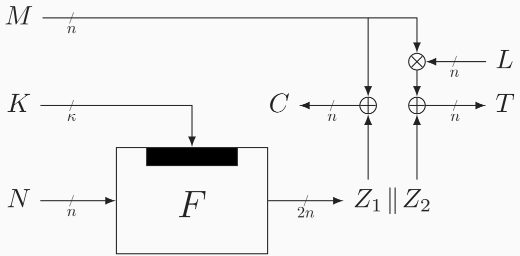
- Consider new query (N, M)
- N should be fresh
- Random $Z_1 \parallel Z_2$
  (if F is a good stream cipher)
- Random (C, T)



Authenticity

- Consider forgery attempt (N, C, T)

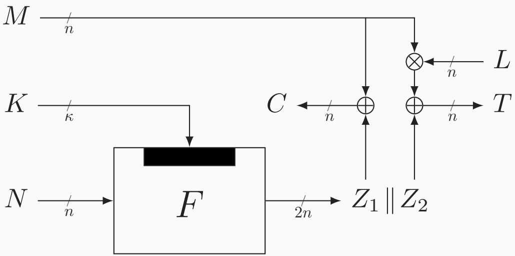
Simple Example: Authenticity



Authenticity

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

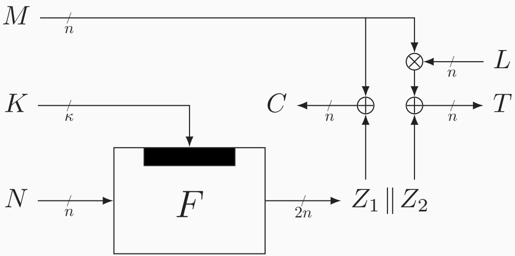
Simple Example: Authenticity



Authenticity

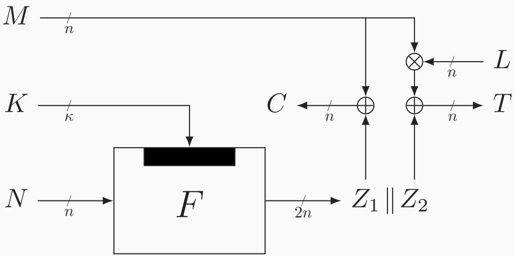
- Consider forgery attempt (N, C, T)
- N could be repeated nonce
- N fresh:
  - $T^\star$ is random, unpredictable

Simple Example: Authenticity



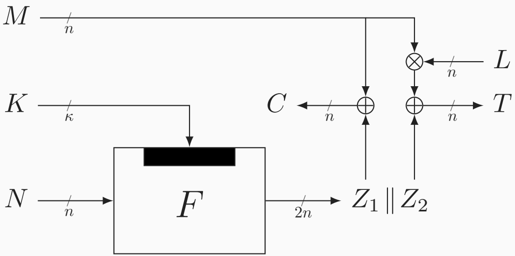
Authenticity

- 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



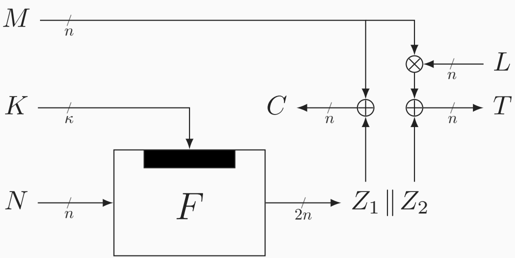
Authenticity

- 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
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$



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
  - $M = Z_1 \oplus C = M' \oplus C' \oplus C$
  - $T^\star = Z_2 \oplus (M \otimes L)$
    $= T' \oplus ((M \oplus M') \otimes L)$
    $= T' \oplus ((C \oplus C') \otimes L)$

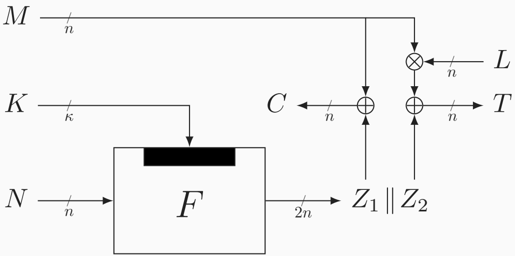


Authenticity

- 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
    $T \oplus T' = (C \oplus C') \otimes L$

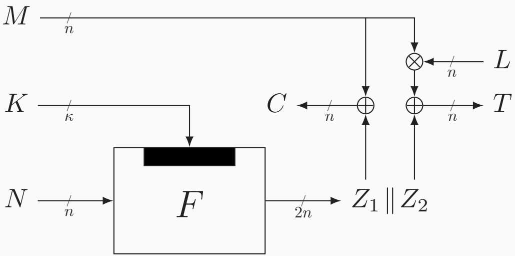
Simple Example: Authenticity



Authenticity

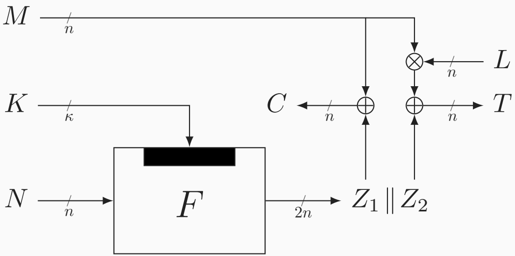
- Consider forgery attempt (N, C, T)
- N could be repeated nonce

- 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



Suppose M is Variable-Length?

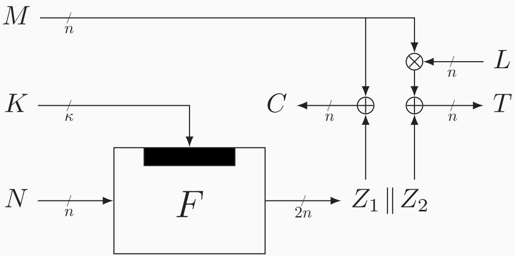
Simple Example: How to Support Variable Length?



Suppose M is Variable-Length?

- $|M| + n$ bits of keystream suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

Simple Example: How to Support Variable Length?



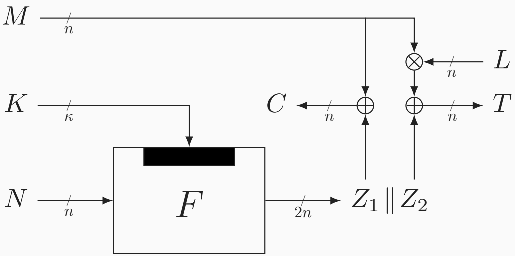
Suppose M is Variable-Length?

- $|M| + n$ bits of keystream 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?

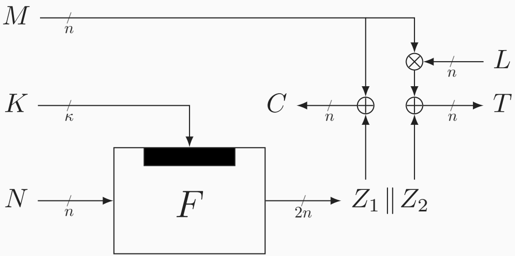
- $|M| + n$ bits of keystream suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

What about AD A?

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

This is almost exactly GCM!

Simple Example: How to Support Variable Length?



Suppose M is Variable-Length?

- $|M| + n$ bits of keystream suffice:
  - Use streaming mode for F
  - Replace $M \otimes L$ by $H_L(M)$

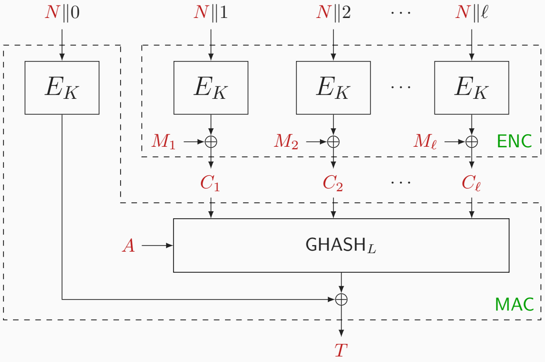
What about AD A?

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

This is almost exactly GCM!

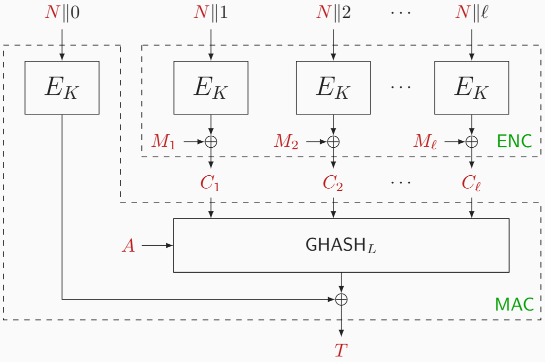
- Encrypt-*then*-MAC: $H_L(A, C)$
- Take CTR mode for F

GCM for 96-bit Nonce N



- McGrew and Viega (2004)
- Widely used (TLS!)
- $L = E_K(0^n)$

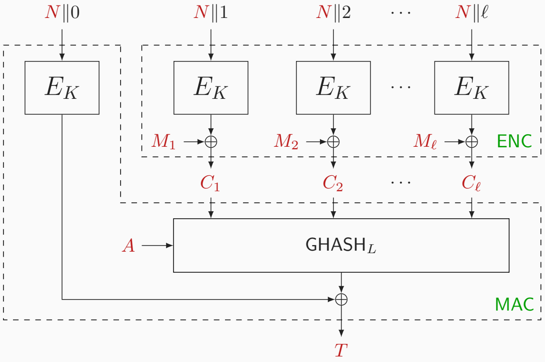
GCM for 96-bit Nonce N



- McGrew and Viega (2004)
- Widely used (TLS!)
- $L = E_K(0^n)$

- Parallelizable
- Evaluates E only (no E^{-1})
- Provably secure (if E is PRP)

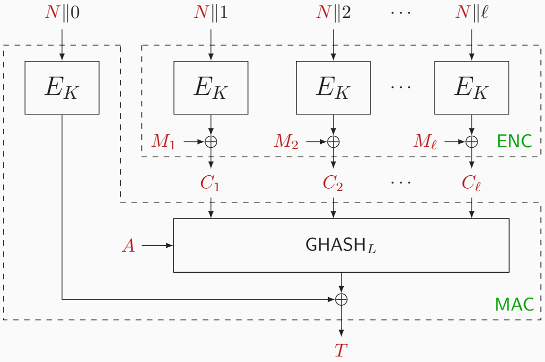
GCM for 96-bit Nonce N



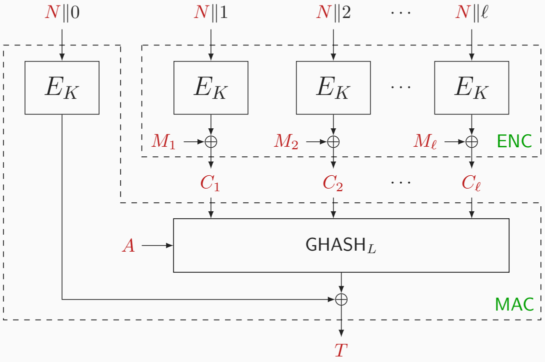
- McGrew and Viega (2004)
- Widely used (TLS!)
- $L = E_K(0^n)$

- Parallelizable
- Evaluates E only (no E^{-1})
- Provably secure (if E is PRP)
- Note: equally popular is ChaCha20-Poly1305!

Problems With GCM for 96-bit Nonce N



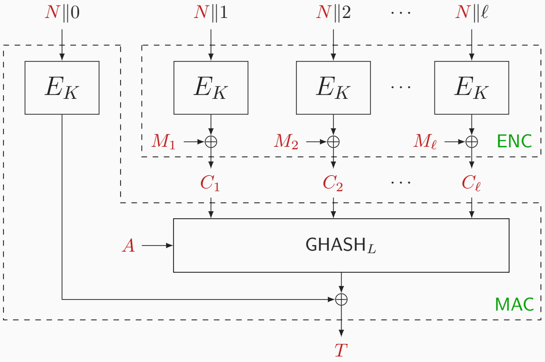
Problems With GCM for 96-bit Nonce N



Nonce Reuse

- Leaks $M \oplus M' = C \oplus C'$ and L

Problems With GCM for 96-bit Nonce N



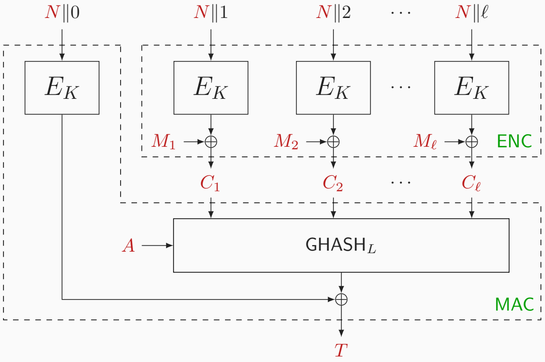
Nonce Reuse

- Leaks $M \oplus M' = C \oplus C'$ and L

Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- AES-192/AES-256?

Problems With GCM for 96-bit Nonce N



Nonce Reuse

- Leaks $M \oplus M' = C \oplus C'$ and L

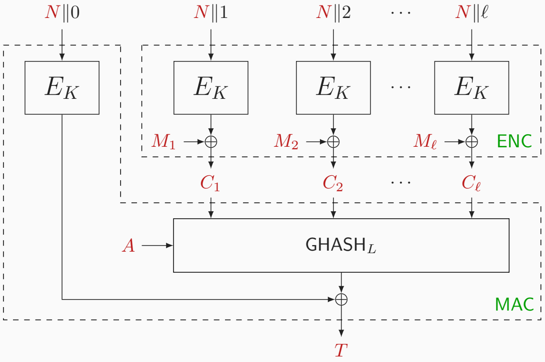
Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- AES-192/AES-256?

Short Nonce

- Random nonces are dangerous
- Nonce-dependent key? [Gue24]

Problems With GCM for 96-bit Nonce N



Nonce Reuse

- Leaks $M \oplus M' = C \oplus C'$ and L

Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- AES-192/AES-256?

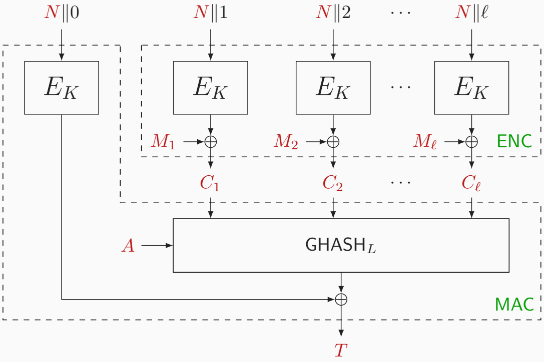
Short Nonce

- Random nonces are dangerous
- Nonce-dependent key? [Gue24]

Short Block Size

- Could be problematic in general
- Rijndael-256? [KCCP23, PST23]

Problems With GCM for 96-bit Nonce N



Nonce Reuse

- Leaks $M \oplus M' = C \oplus C'$ and L

Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- AES-192/AES-256?

Short Nonce

- Random nonces are dangerous
- Nonce-dependent key? [Gue24]

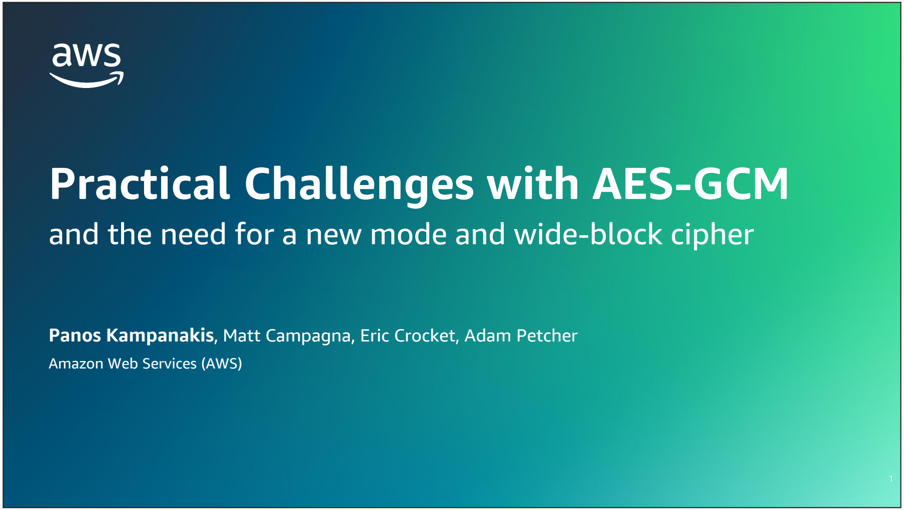
Short Block Size

- Could be problematic in general
- Rijndael-256? [KCCP23, PST23]

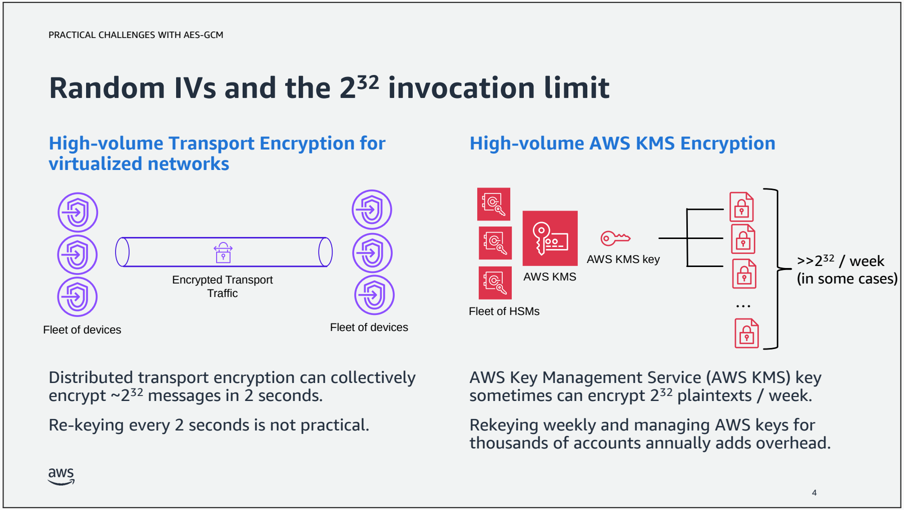
No Tag Truncation

- Easier subkey recovery [Fer05]
- Alternative hashing? [CMP23]

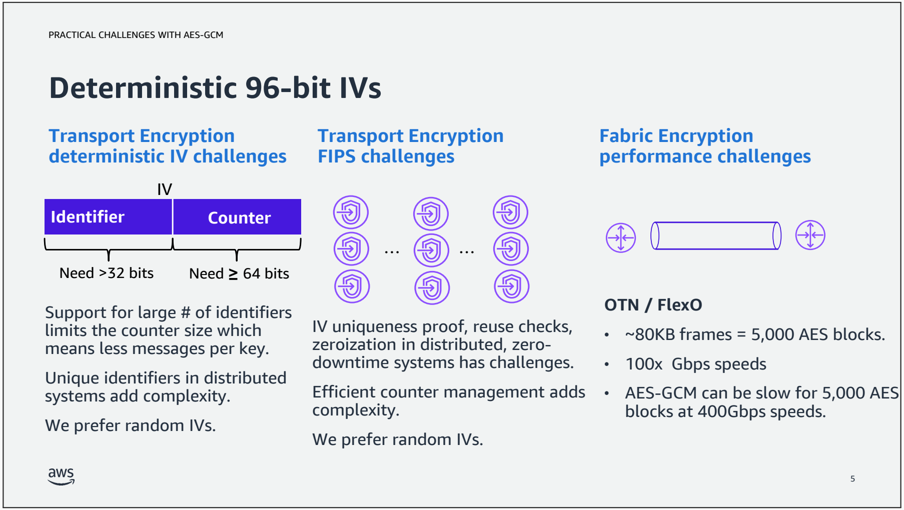
Use Case: Amazon Web Services [KCCP23]



Use Case: Amazon Web Services [KCCP23]



Use Case: Amazon Web Services [KCCP23]



Conclusion

Conclusion

Provable Security in Symmetric Cryptography

- Basic modes proved secure using quite simple ideas
- More sophisticated modes require nice tricks in graph theory
- Often this boils down to trying to upper or lower bound solutions

Conclusion

Provable Security in Symmetric Cryptography

- Basic modes proved secure using quite simple ideas
- More sophisticated modes require nice tricks in graph theory
- Often this boils down to trying to upper or lower bound solutions

Current Directions in Provable Security

- Difficulties in beyond birthday bound security
- Accordion modes
- Arithmetization-oriented modes

Provable Security in Symmetric Cryptography

- Basic modes proved secure using quite simple ideas
- More sophisticated modes require nice tricks in graph theory
- Often this boils down to trying to upper or lower bound solutions

Current Directions in Provable Security

- Difficulties in beyond birthday bound security
- Accordion modes
- Arithmetization-oriented modes

Thank you for your attention!

Mihir Bellare and Björn Tackmann.
The Multi-user Security of Authenticated Encryption: AES-GCM in TLS 1.3.
In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 247–276. Springer, 2016.

Matthew Campagna, Alexander Maximov, and John Preuß Mattsson.
Galois counter mode with secure short tags (GCM-SST).
Third NIST Workshop on Block Cipher Modes of Operation 2023, October 2023.
https://www.amazon.science/publications/galois-counter-mode-with-secure-short-tags-gcm-sst.

Shan Chen and John P. Steinberger.
Tight Security Bounds for Key-Alternating Ciphers.
In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 327–350. Springer, 2014.

Wei Dai, Viet Tung Hoang, and Stefano Tessaro.
Information-Theoretic Indistinguishability via the Chi-Squared Method.
In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 497–523. Springer, 2017.

Joan Daemen and Vincent Rijmen.
The Design of Rijndael: AES - The Advanced Encryption Standard.
Information Security and Cryptography. Springer, 2002.

Shimon Even and Yishay Mansour.
A Construction of a Cipher From a Single Pseudorandom Permutation.
In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1991.

Niels Ferguson.
Authentication Weaknesses in GCM.
Public Comment to NIST, 2005.
`http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html`.

Shay Gueron.
Double-Nonce-Derive-Key-GCM (DNDK-GCM): General design paradigms and application.
NIST Workshop on the Requirements for an Accordion Cipher Mode 2024, June 2024.
https://csrc.nist.gov/csrc/media/Presentations/2024/double-nonce-derive-key-gcm-dndk-gcm/images-media/sess-6-gueron-acm-workshop-2024.pdf.

Tetsu Iwata, Bart Mennink, and Damian Vizár.
CENC is Optimally Secure.
Cryptology ePrint Archive, Report 2016/1087, 2016.
http://eprint.iacr.org/2016/1087.

Tetsu Iwata.
New Blockcipher Modes of Operation with Beyond the Birthday Bound Security.
In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2006.

Panos Kampanakis, Matt Campagna, Eric Crocket, and Adam Petcher.
Practical Challenges with AES-GCM and the need for a new cipher.
Third NIST Workshop on Block Cipher Modes of Operation 2023, October 2023.
https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/Practical%20Challenges%20with%20AES-GCM.pdf.

David A. McGrew and John Viega.
The Security and Performance of the Galois/Counter Mode (GCM) of Operation.
In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

Jacques Patarin.
Étude des Générateurs de Permutations Basés sur le Schéma du D.E.S.
PhD thesis, Université Paris 6, Paris, France, November 1991.

Jacques Patarin.
Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security.
In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 513–529. Springer, 2003.

Jacques Patarin.
A Proof of Security in $O(2^n)$ for the Xor of Two Random Permutations.
In Reihaneh Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 232–248. Springer, 2008.

Jacques Patarin.
The "Coefficients H" Technique.
In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008.

John Preuß Mattsson, Ben Smeets, and Erik Thormarker.
Proposals for Standardization of Encryption Schemes.
Third NIST Workshop on Block Cipher Modes of Operation 2023, October 2023.
https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/Proposals%20for%20Standardization%20of%20Encryption%20Schemes%20Final.pdf.

Supporting Slides

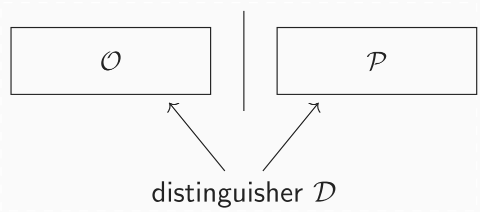
H-Coefficient Technique and Security of Even-Mansour

H-Coefficient Technique

- Patarin [Pat91, Pat08b]
- Popularized by Chen and Steinberger [CS14]

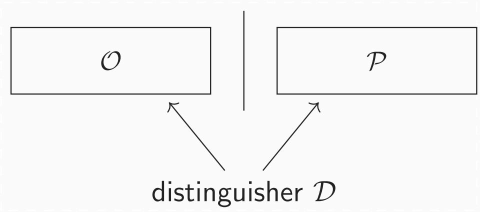
H-Coefficient Technique

- Patarin [Pat91, Pat08b]
- Popularized by Chen and Steinberger [CS14]



H-Coefficient Technique

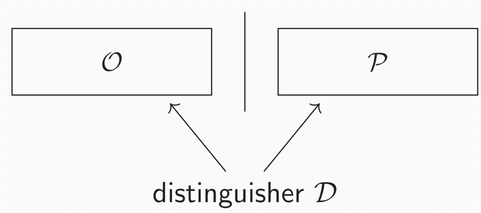
- Patarin [Pat91, Pat08b]
- Popularized by Chen and Steinberger [CS14]



- Basic idea:
  - Each conversation defines a transcript τ

H-Coefficient Technique

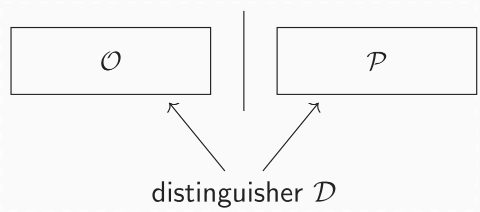
- Patarin [Pat91, Pat08b]
- Popularized by Chen and Steinberger [CS14]



- Basic idea:
  - Each conversation defines a transcript τ
  - $\mathcal{O} \approx \mathcal{P}$ for most of the transcripts

H-Coefficient Technique

- Patarin [Pat91, Pat08b]
- Popularized by Chen and Steinberger [CS14]



- Basic idea:
  - Each conversation defines a transcript τ
  - $\mathcal{O} \approx \mathcal{P}$ for most of the transcripts
  - Remaining transcripts occur with small probability

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Complexity only measured by the number of queries
- Each conversation defines a transcript τ

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Complexity only measured by the number of queries
- Each conversation defines a transcript τ
- Consider good and bad transcripts

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Complexity only measured by the number of queries
- Each conversation defines a transcript τ
- Consider good and bad transcripts

Lemma

Let $\varepsilon \geq 0$ be such that for all good transcripts τ:

$$\frac{\mathbf{Pr}\left(\mathcal{O} \text{ gives } \tau\right)}{\mathbf{Pr}\left(\mathcal{P} \text{ gives } \tau\right)} \geq 1 - \varepsilon$$

Then, $\Delta_{\mathcal{D}}(\mathcal{O}; P) \leq \varepsilon + \mathbf{Pr}\left(\text{bad transcript for } \mathcal{P}\right)$

H-Coefficient Technique

- $\mathcal{D}$ is computationally unbounded and deterministic
- Complexity only measured by the number of queries
- Each conversation defines a transcript τ
- Consider good and bad transcripts

Lemma

Let $\varepsilon \geq 0$ be such that for all good transcripts τ:

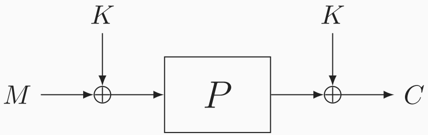
$$\frac{\mathbf{Pr}\left(\mathcal{O} \text{ gives } \tau\right)}{\mathbf{Pr}\left(\mathcal{P} \text{ gives } \tau\right)} \geq 1 - \varepsilon$$

Then, $\Delta_{\mathcal{D}}(\mathcal{O}; P) \leq \varepsilon + \mathbf{Pr}\left(\text{bad transcript for } \mathcal{P}\right)$

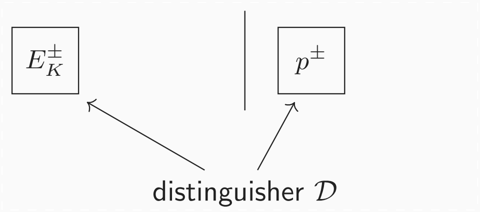
Trade-off: define bad transcripts smartly!

Security of Even-Mansour (1/6)

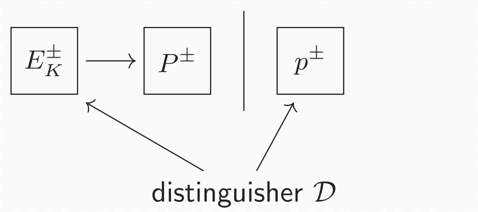
- Even-Mansour construction [EM91]:



$$E_K(M) = P(M \oplus K) \oplus K$$



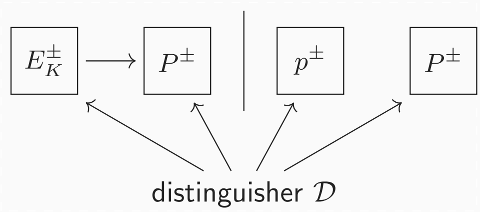
Slightly Different Security Model



Slightly Different Security Model

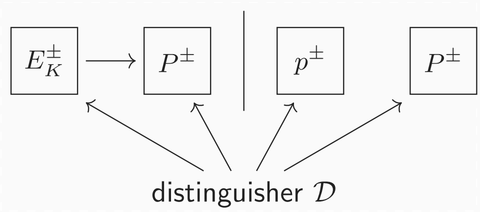
- Underlying permutation

Security of Even-Mansour (2/6)



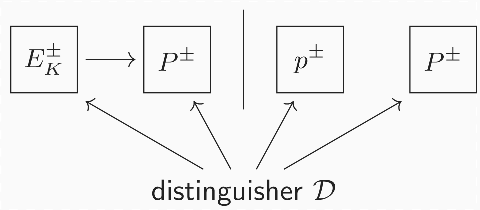
Slightly Different Security Model

- Underlying permutation randomized
- Information-theoretic distinguisher $\mathcal{D}$
  - q construction queries
  - t offline evaluations $\approx t$ primitive queries



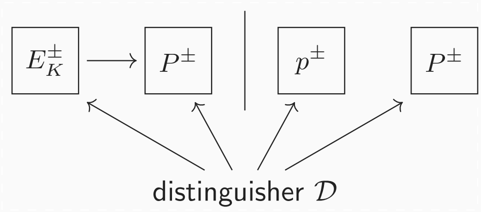
Slightly Different Security Model

- Underlying permutation randomized
- Information-theoretic distinguisher $\mathcal{D}$
  - q construction queries
  - t offline evaluations $\approx t$ primitive queries
  - Unbounded computational power

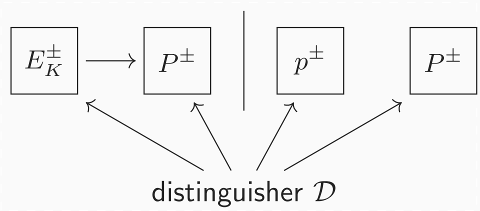


- Two construction oracles: (E_K, E_K^{-1}) (for secret key K) and (p, p^{-1}) (secret)
- Two primitive oracles: (P, P^{-1}) (secret)

Security of Even-Mansour (3/6)

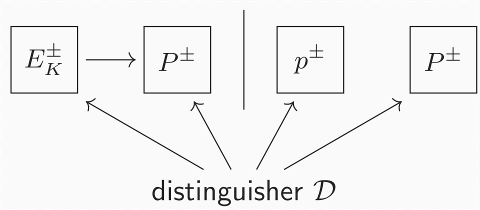


- Two construction oracles: (E_K, E_K^{-1}) (for secret key K) and (p, p^{-1}) (secret)
- Two primitive oracles: (P, P^{-1}) (secret)
- $\mathcal{D}$ tries to determine which oracle it communicates with



- Two construction oracles: (E_K, E_K^{-1}) (for secret key K) and (p, p^{-1}) (secret)
- Two primitive oracles: (P, P^{-1}) (secret)
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

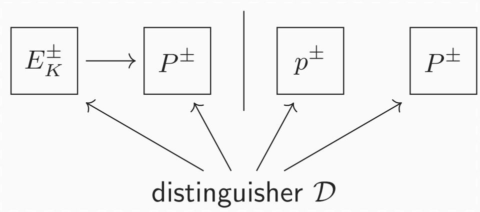
$$\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(E_K, E_K^{-1}\ ;\ p, p^{-1}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{E_K, E_K^{-1}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{p, p^{-1}} = 1\right)\right|$$



- Two construction oracles: (E_K, E_K^{-1}) (for secret key K) and (p, p^{-1}) (secret)
- Two primitive oracles: (P, P^{-1}) (secret)
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

$$\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}\left(E_K, E_K^{-1}\ ;\ p, p^{-1}\right) = \left|\mathbf{Pr}\left(\mathcal{D}^{E_K, E_K^{-1}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{p, p^{-1}} = 1\right)\right|$$

- $\mathbf{Adv}_E^{\mathrm{sprp}}(q, t)$: maximum advantage over any $\mathcal{A}$ with query/time complexity q/t



Theorem

For any distinguisher $\mathcal{D}$ making q queries to $E_K^\pm/p^\pm$ and t primitive queries

$$\mathbf{Adv}_E^{\mathrm{sprp}}(\mathcal{D}) = \Delta_{\mathcal{D}}(E_K^\pm, P^\pm; p^\pm, P^\pm) \leq ???$$

Security of Even-Mansour (5/6)

Step 1. Define how transcripts look like

Step 2. Define good and bad transcripts

Step 3. Upper bound $\mathbf{Pr}\,(\text{bad transcript for } (p^{\pm}, P^{\pm}))$

Step 4. Lower bound $\frac{\mathbf{Pr}\big((E_K^{\pm}, P^{\pm}) \text{ gives } \tau\big)}{\mathbf{Pr}\big((p^{\pm}, P^{\pm}) \text{ gives } \tau\big)} \geq 1 - \varepsilon$ ($\forall$ good τ)

1. **Define how transcripts look like**

- Construction queries:

$$\tau_E = \{(M_1, C_1), \ldots, (M_q, C_q)\}$$

- Primitive queries:

$$\tau_P = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$$

Security of Even-Mansour (6/6)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(M_1, C_1), \ldots, (M_q, C_q)\}$$

- Primitive queries:

$$\tau_P = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(M_1, C_1), \ldots, (M_q, C_q)\}$$

- Primitive queries:

$$\tau_P = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

- Bonus information!
  - After interaction of $\mathcal{D}$ with oracles: reveal the key

Security of Even-Mansour (6/6)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(M_1, C_1), \ldots, (M_q, C_q)\}$$

- Primitive queries:

$$\tau_P = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

- Bonus information!
  - After interaction of $\mathcal{D}$ with oracles: reveal the key
    - Real world $(E_K^{\pm}, P^{\pm})$: key used for encryption

Security of Even-Mansour (6/6)

1. Define how transcripts look like

- Construction queries:

$$\tau_E = \{(M_1, C_1), \ldots, (M_q, C_q)\}$$

- Primitive queries:

$$\tau_P = \{(X_1, Y_1), \ldots, (X_t, Y_t)\}$$

- Unordered lists (ordering not needed in current proof)
- 1-to-1 correspondence between any $\mathcal{D}$ and any (τ_E, τ_P)

- Bonus information!
  - After interaction of $\mathcal{D}$ with oracles: reveal the key
    - Real world $(E_K^{\pm}, P^{\pm})$: key used for encryption
    - Ideal world $(p^{\pm}, P^{\pm})$: dummy key $K \xleftarrow{\$} \{0,1\}^n$