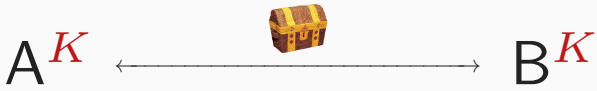
Challenges in Symmetric Cryptography

Bart Mennink
Radboud University (The Netherlands)
Finse Cybersecurity Winter School 2025
April 28, 2025

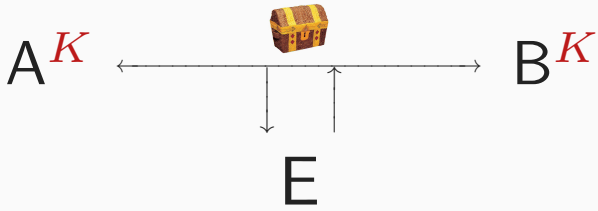
Keyed Symmetric Cryptography

General Setting



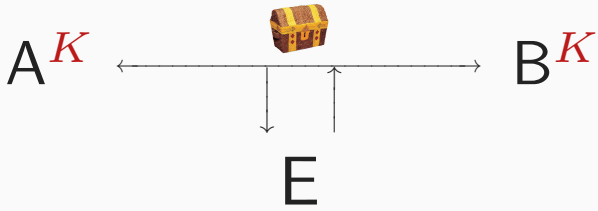
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data

General Setting



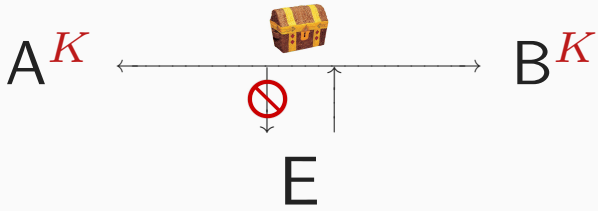
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication

General Setting



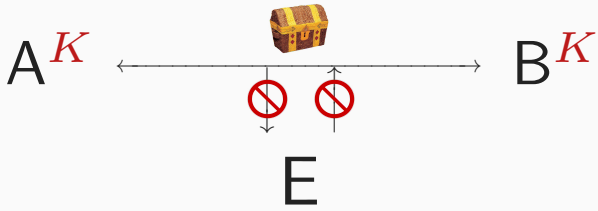
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- Two Main Security Properties:

General Setting



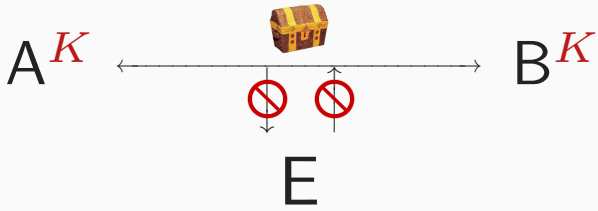
- Two parties, Alice and Bob, communicate over a public channel
  - They have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit/disturb/... the communication
- Two Main Security Properties:
  - Confidentiality: Eve cannot learn anything about the content of 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
- Two Main Security Properties:
  - Confidentiality: Eve cannot learn anything about the content of the data
  - Authenticity: Manipulation/replay/delay of the data gets detected

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
- Two Main Security Properties:
  - Confidentiality: Eve cannot learn anything about the content of the data
  - Authenticity: Manipulation/replay/delay of the data gets detected

Authenticated encryption aims to achieve both

One-Time Pad Encryption

Encryption:

$$
\begin{array}{lcccccccccccccccc}
P = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \bigoplus
\end{array}
$$

One-Time Pad Encryption

Encryption:

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

One-Time Pad Encryption

Encryption:

$$
\begin{array}{lcccccccccccccccc}
P = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \oplus \\
\hline
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 &
\end{array}
$$

Decryption:

$$
\begin{array}{lcccccccccccccccc}
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \oplus
\end{array}
$$

One-Time Pad Encryption

Encryption:

$$
\begin{array}{lcccccccccccccc}
P = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \quad \oplus \\
\hline
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

Decryption:

$$
\begin{array}{lcccccccccccccc}
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \quad \oplus \\
\hline
P = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

Properties of One-Time Pad

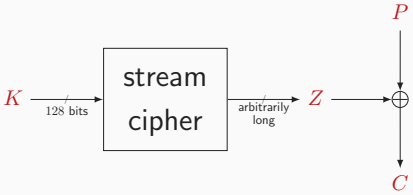
- Perfect secrecy (against an attacker that has no knowledge about the key)
- Key must be as long as the plaintext!

Properties of One-Time Pad

- Perfect secrecy (against an attacker that has no knowledge about the key)
- Key must be as long as the plaintext!

Stream Ciphers

- Generate long keystream Z from short key K

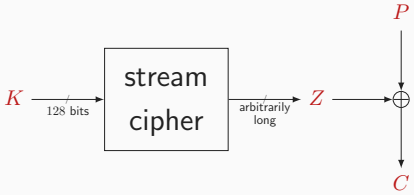


Properties of One-Time Pad

- Perfect secrecy (against an attacker that has no knowledge about the key)
- Key must be as long as the plaintext!

Stream Ciphers

- Generate long keystream Z from short key K



- 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)

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

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

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

- 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

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

- 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

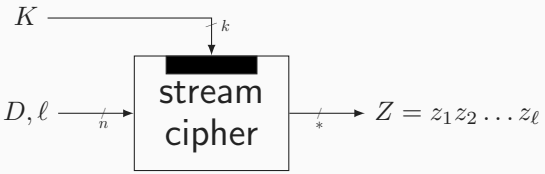
Stream Cipher: Vigenère ($\approx$ 1553, Wikipedia)

- 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!

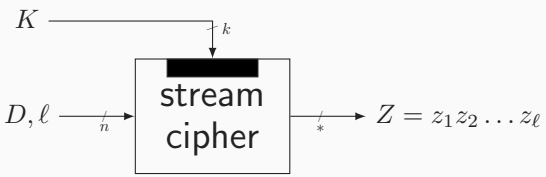
Modern Stream Cipher Design

Modern Stream Ciphers



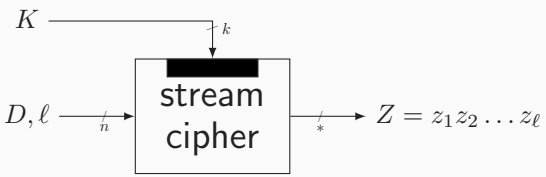
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated

Modern Stream Ciphers



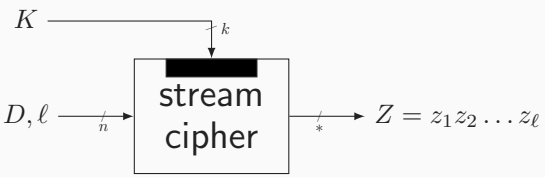
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- The diversifier must be different for each plaintext that is transmitted

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 plaintext 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:

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

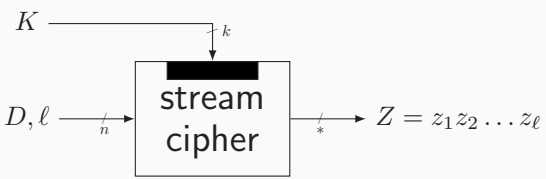


- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- The diversifier must be different for each plaintext that is transmitted
- Example: data streams, e.g., pay TV and telephone, often split data in relatively short, numbered, frames. The frame number may serve as diversifier:

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

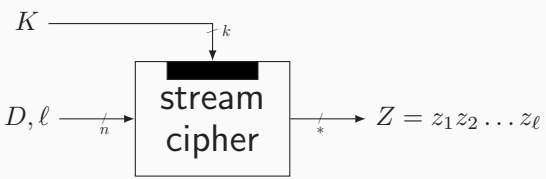
When is a stream cipher strong enough?

Stream Cipher Security, Intuition (1/3)



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

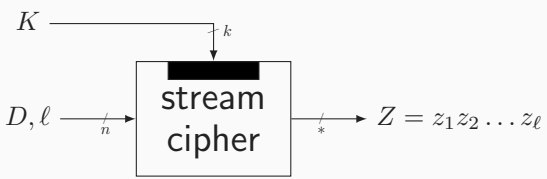
Stream Cipher Security, Intuition (1/3)



- Kerckhoffs principle: security should only 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)

Stream Cipher Security, Intuition (1/3)



- Kerckhoffs principle: security should only 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

Intermezzo: Random Oracle

Random Oracle

- A database of input-output tuples
- Initially empty

D	Z
...	...
...	...
...	...
...	...

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
...	...
...	...
...	...
...	...

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
...	...
...	...
...	...
...	...

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
...	...
...	...

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	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 $|Z| \geq \ell$:
    - If $|Z| < \ell$:

D	Z
1100	101011101010101
1111010101101101	110101
001000011100	101011010111010101011
...	...

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	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 $|Z| < \ell$:

D	Z
1100	101011101010101
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	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	1101011101111101101
001000011100	101011010111010101011
...	...

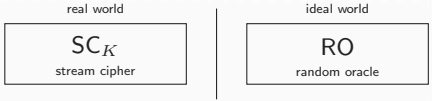
Intermezzo: Random Oracle

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 $Z \| Z'$
    - update (D, Z) in the list

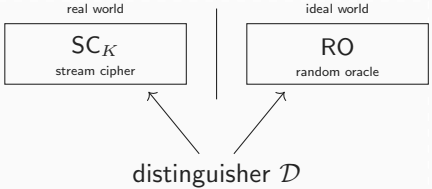
D	Z
1100	101011101010101
1111010101101101	1101011101111101101
001000011100	101011010111010101011
...	...

Stream Cipher Security, Intuition (2/3)



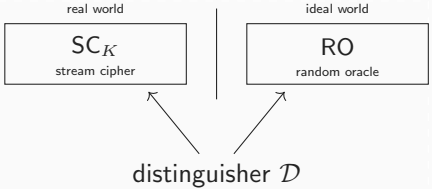
- We thus want to "compare" SC_K with a random oracle RO

Stream Cipher Security, Intuition (2/3)



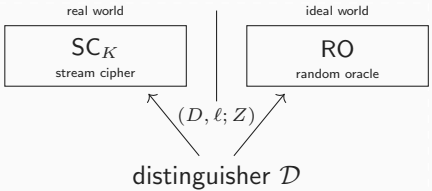
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds

Stream Cipher Security, Intuition (2/3)



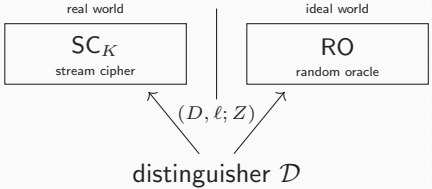
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does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

Stream Cipher Security, Intuition (2/3)



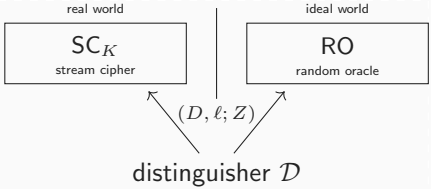
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does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



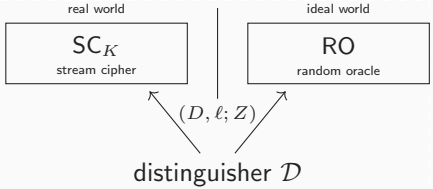
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - We toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to RO
  - $\mathcal{D}$ does 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 $\mathbf{Pr}\,(\text{success})$

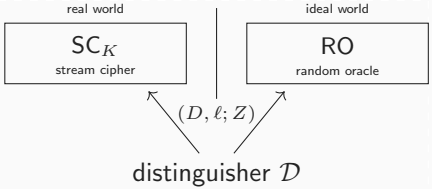
Stream Cipher Security, Intuition (3/3)



- 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:

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

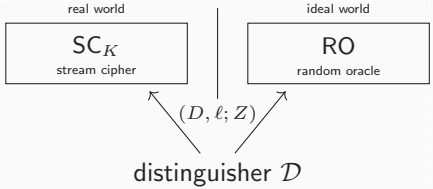
Stream Cipher Security, Intuition (3/3)



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

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

Stream Cipher Security, Intuition (3/3)

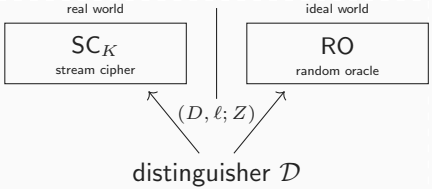


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

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

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

Stream Cipher Security, Intuition (3/3)

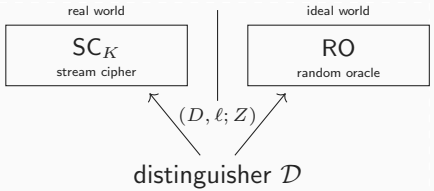


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$
- $\mathcal{D}$ can always guess and succeeds with probability $\geq 1/2$, so we scale the probability to $\mathcal{D}$'s advantage:

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

- $\mathcal{D}$ 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, 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:

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

- $\mathcal{D}$ 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"
- SC is secure as long as $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{SC}}(\mathcal{D}) \ll 1$ for any such $\mathcal{D}$

Generic Stream Cipher Design

Generic Stream Cipher Design (1/2)

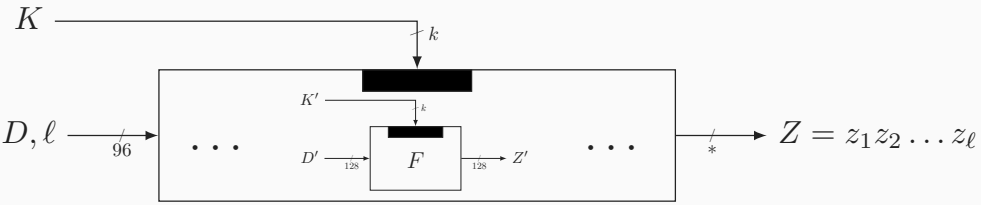
Modern Approach

- Stream cipher built from smaller cryptographic primitive

Generic Stream Cipher Design (1/2)

Modern Approach

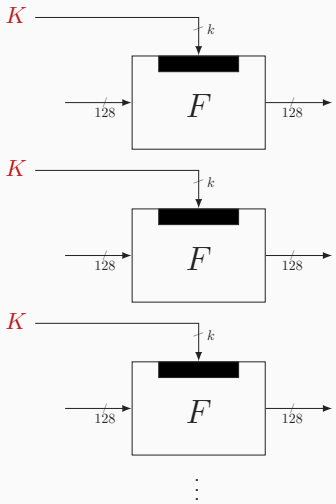
- Stream cipher built from smaller cryptographic primitive
- Suppose (for the sake of argument):
  - we **know** how to build a strong stream cipher F with fixed-length output
  - we **want** to build a stream cipher with variable-length output



Generic Stream Cipher Design (2/2)

Design

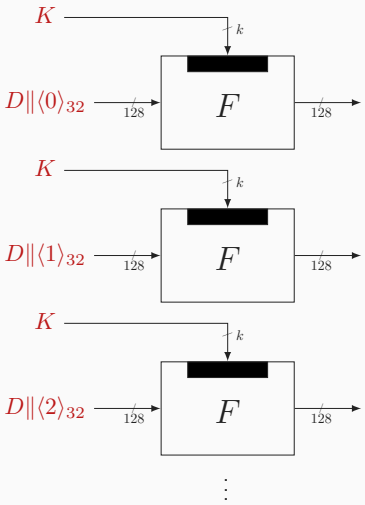
- Feed K to primitive



Generic Stream Cipher Design (2/2)

Design

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

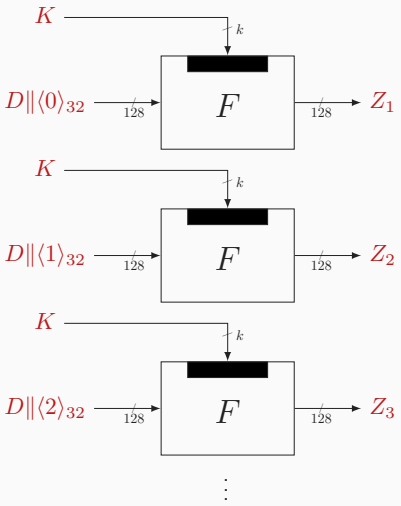


Generic Stream Cipher Design (2/2)

Design

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

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



Generic Stream Cipher Design (2/2)

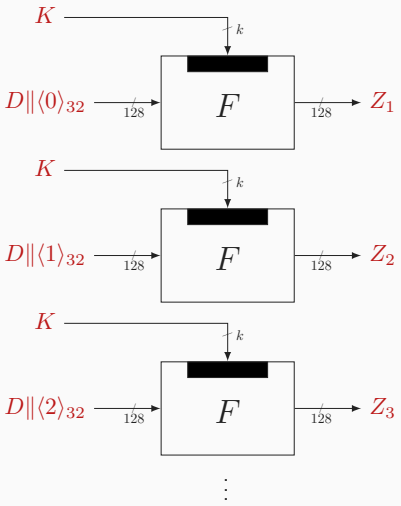
Design

- 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)

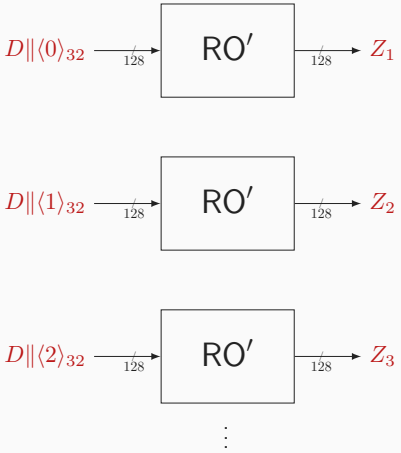
Design

- 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$'$

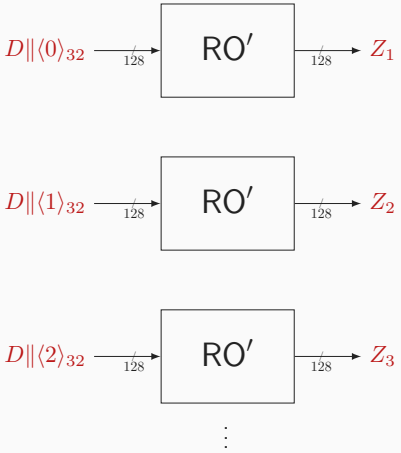


Generic Stream Cipher Design (2/2)

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

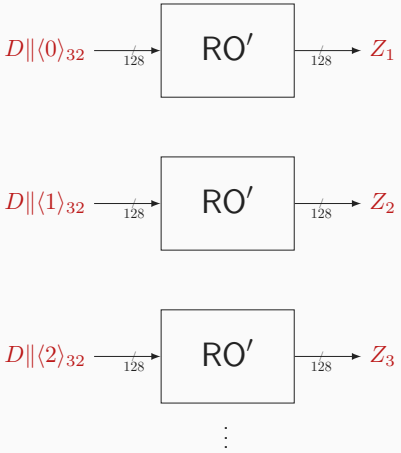
- If F_K is hard to distinguish from a RO′
- Then construction is hard to distinguish from RO

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

Generic Stream Cipher Design (2/2)

Design

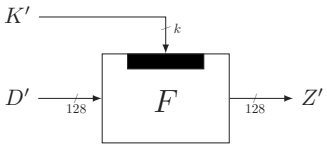
- Feed K to primitive
- Evaluate primitive as
  concatenated with co
- Concatenate outputs:

$$Z = Z_1$$

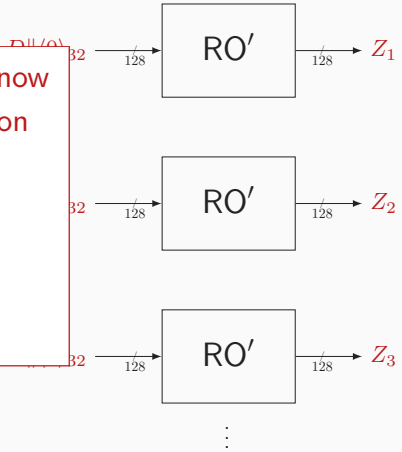
Security

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

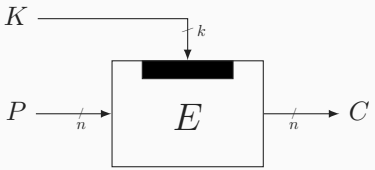
Unfortunately, we do not know how to construct a function



that behaves like a RO′

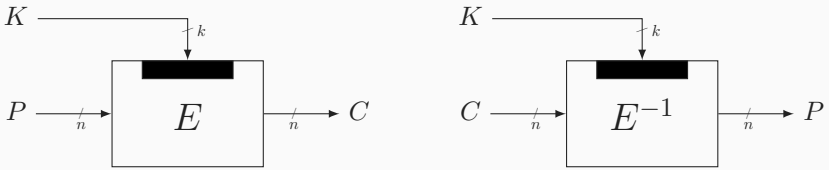


Block Ciphers



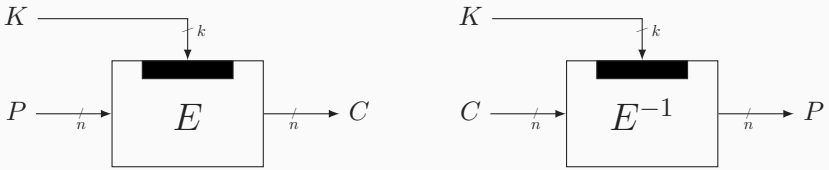
- Using key K, plaintext P is bijectively transformed to ciphertext C

Block Ciphers



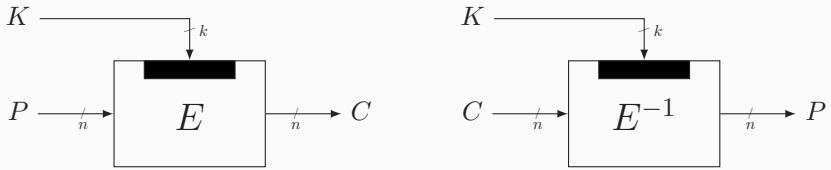
- Using key K, plaintext P is bijectively transformed to ciphertext C
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}

Block Ciphers



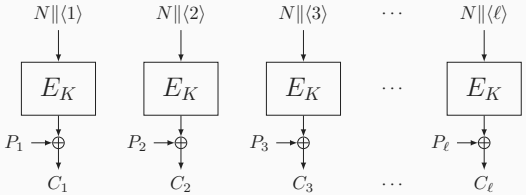
- Using key K, plaintext P is bijectively transformed to ciphertext C
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}
- Example [DR02]: AES-128: $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$

Block Ciphers



- Using key K, plaintext P is bijectively transformed to ciphertext C
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}
- Example [DR02]: AES-128: $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$
- A good block cipher should behave like a random permutation
  - We denote the advantage of distinguisher $\mathcal{D}$ by $\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D})$
  - $\mathcal{D}$ is limited by query/time complexity q/t

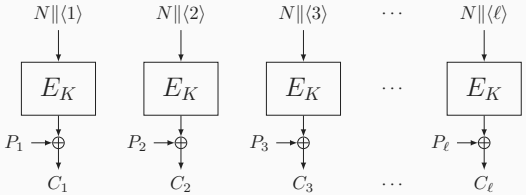
Counter (CTR) Mode



Features

- Most widely used encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple

Counter (CTR) Mode



Features

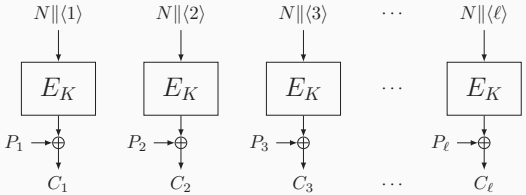
- Most widely used encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple

Security

- Secure as long as N is unique, E_K is a secure PRP, and $q \ll 2^{n/2}$:

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

Counter (CTR) Mode



Features

- Most widely used encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple

Security

- Secure as long as N is unique, E_K is a secure PRP, and $q \ll 2^{n/2}$:

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

birthday bound

General Birthday Paradox

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

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

The Birthday Bound

General Birthday Paradox

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

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

- Important phenomenon in cryptography

The Birthday Bound

General Birthday Paradox

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

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

- Important phenomenon in cryptography

Lightweight Cryptography

- AES has block size of 128 bits: birthday paradox problem only in certain settings

The Birthday Bound

General Birthday Paradox

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

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

- Important phenomenon in cryptography

Lightweight Cryptography

- AES has block size of 128 bits: birthday paradox problem only in certain settings
- Lightweight cryptography operates in constrained environments
  - IoT, healthcare devices, sensors, RFID tags, ...

The Birthday Bound

General Birthday Paradox

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

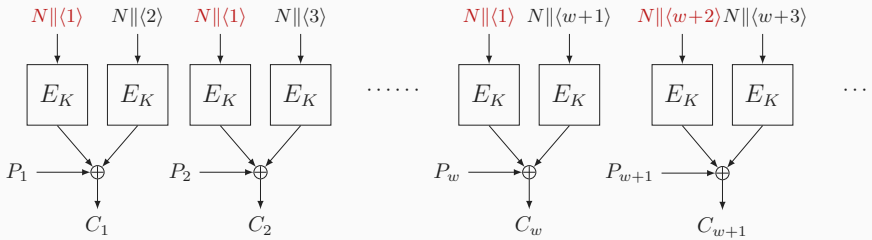
$$\mathbf{Ex}\left[\text{collisions}\right] = \binom{q}{2}/2^n$$

- Important phenomenon in cryptography

Lightweight Cryptography

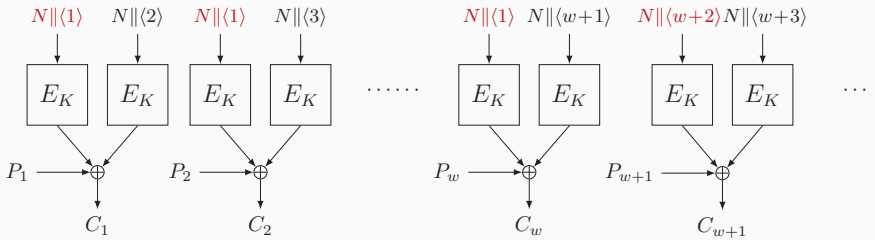
- AES has block size of 128 bits: birthday paradox problem only in certain settings
- Lightweight cryptography operates in constrained environments
  - IoT, healthcare devices, sensors, RFID tags, ...
- Lightweight block ciphers have a block size of 64 bits or even less [BKL+07]

CENC [Iwa06]



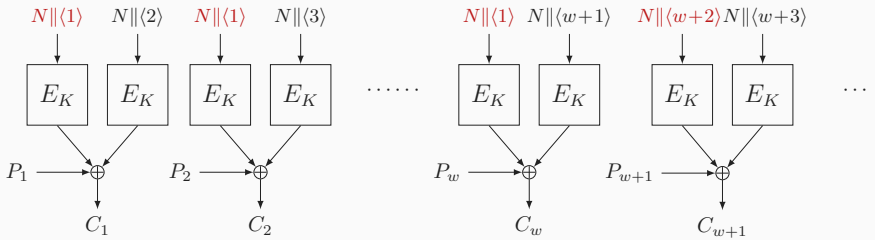
- One subkey used for $w \geq 1$ encryptions

CENC [Iwa06]



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

CENC [Iwa06]

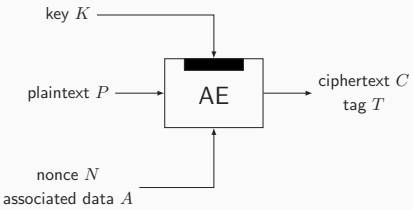


- One subkey used for $w \geq 1$ encryptions
- Almost as expensive as CTR
- Secure as long as N is unique, E_K is a secure PRP, and $q \ll 2^n/w$ [IMV16]:

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{CENC}}(q,t) \leq \mathbf{Adv}^{\mathrm{prp}}_{E}((w+1)q,t') + wq/2^n$$

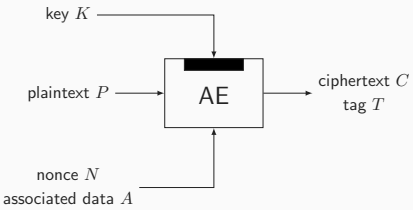
Authenticated Encryption and GCM

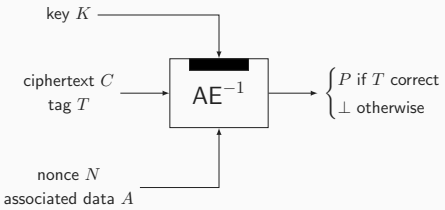
Authenticated Encryption



- Using key K:
  - Ciphertext C encrypts plaintext P
  - Tag T authenticates (N, A, P)

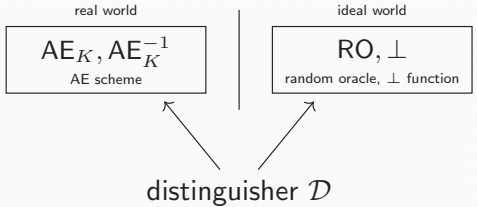
Authenticated Encryption





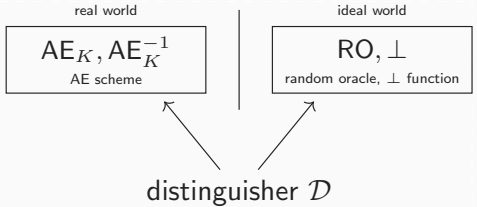
- Using key K:
  - Ciphertext C encrypts plaintext P
  - Tag T authenticates (N, A, P)
- Unwrapping needs to satisfy that
  - Plaintext disclosed if tag is correct
  - Plaintext is not leaked if tag is incorrect

Authenticated Encryption Security



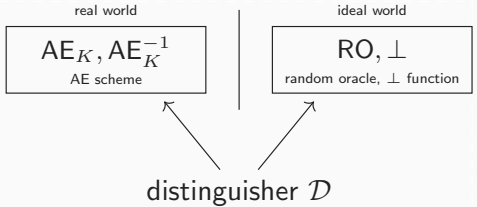
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (secret)

Authenticated Encryption Security



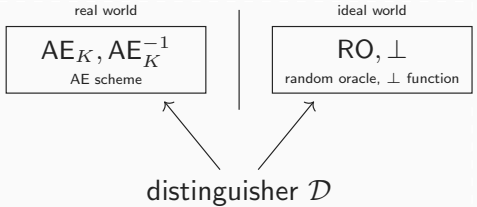
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ unique nonce for each encryption query, and no trivial queries

Authenticated Encryption Security



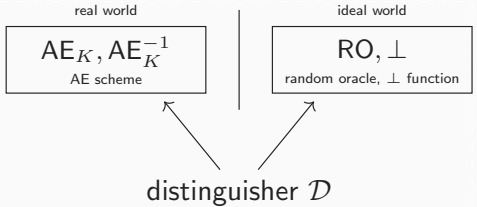
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ unique nonce for each encryption query, and no trivial queries
- $\mathcal{D}$ tries to determine which oracle it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

Authenticated Encryption Security



- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (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 $\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(\mathcal{D})$

Authenticated Encryption Security



- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (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 $\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(\mathcal{D})$
- $\mathcal{D}$ is limited by query/time complexity $(q_e, q_d)/t$

Universal Hash Functions

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

Universal Hash Functions

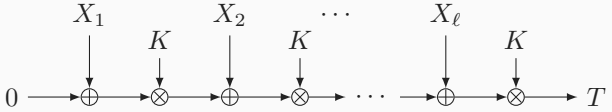
- Consider a function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$
- H is ε-XOR-universal if $\mathbf{Pr}_K\left(H_K(X) \oplus H_K(X') = T\right) \leq \varepsilon$ for all $X \neq X', T$
- Example: finite field multiplication $H_K(X) = K \otimes X$ is 2^{-k}-XOR-universal

Universal Hash Functions

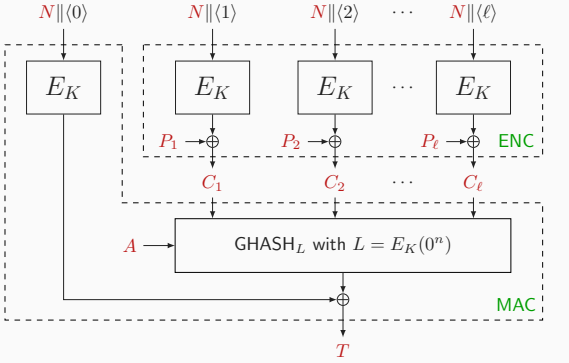
- Consider a function $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^t$
- H is ε-XOR-universal if $\mathbf{Pr}_K\left(H_K(X) \oplus H_K(X') = T\right) \leq \varepsilon$ for all $X \neq X', T$
- Example: finite field multiplication $H_K(X) = K \otimes X$ is 2^{-k}-XOR-universal

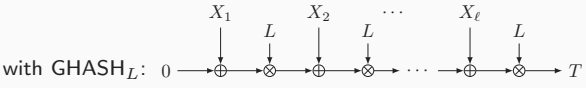
GHASH

- Addition and multiplication over finite field (with proper input encoding)
- $\ell 2^{-k}$-XOR-universal [MV04]



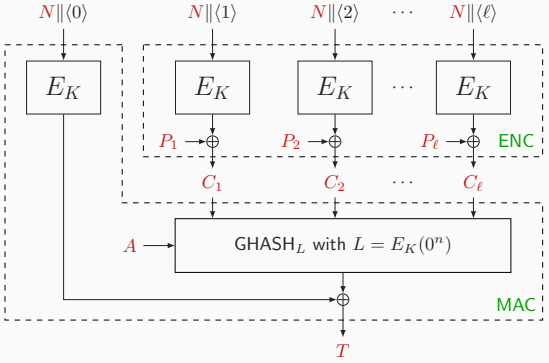
GCM for 96-bit Nonce N [MV04]

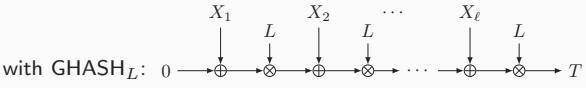




- Widely used:
  - TLS, WPA3, IPsec, ...
- Equally popular: ChaCha20-Poly1305!

GCM for 96-bit Nonce N [MV04]

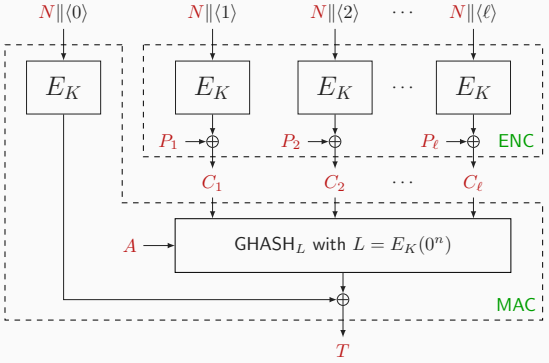


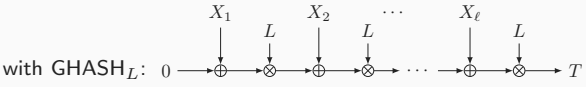


- Widely used:
  - TLS, WPA3, IPsec, ...
- Equally popular: ChaCha20-Poly1305!

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

GCM for 96-bit Nonce N [MV04]



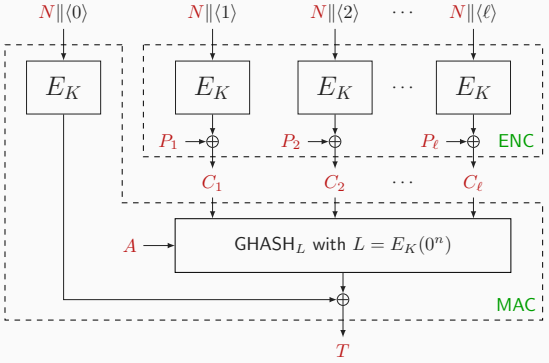


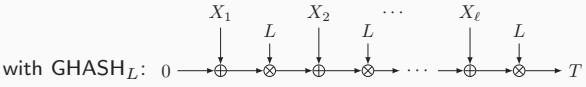
- Widely used:
  - TLS, WPA3, IPsec, ...
- Equally popular: ChaCha20-Poly1305!

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

- Nonce reuse is devastating
  - Leaks $P \oplus P' = C \oplus C'$ and L

Problems With GCM for 96-bit Nonce N (1/2)

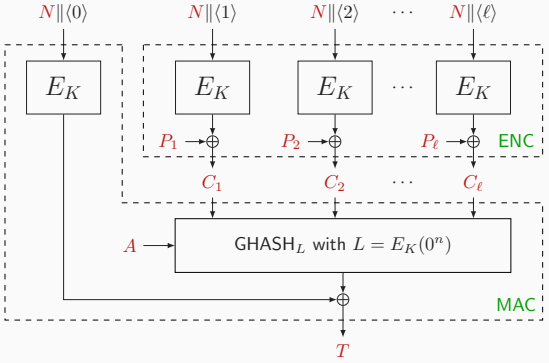


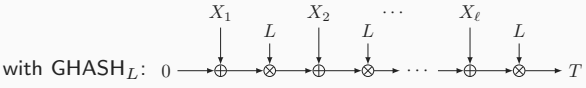


Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]

Problems With GCM for 96-bit Nonce N (1/2)

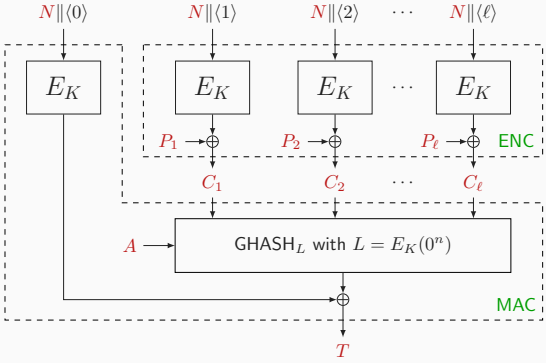


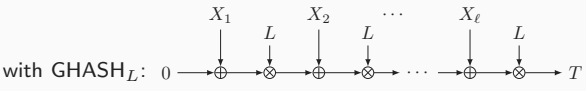


Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- Better solutions for NIST lightweight cryptography standard [DM24]

Problems With GCM for 96-bit Nonce N (1/2)





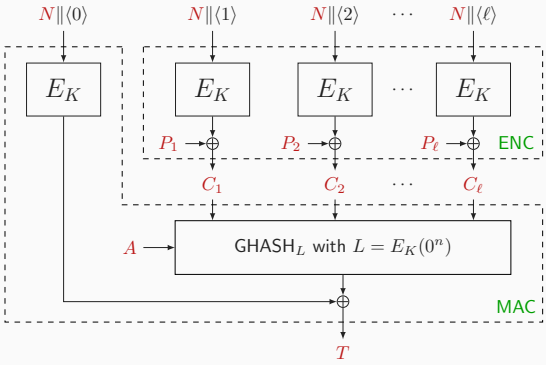
Short Key

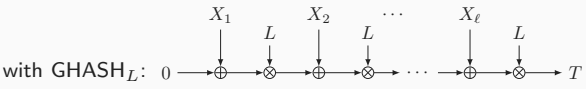
- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- Better solutions for NIST lightweight cryptography standard [DM24]

No Tag Truncation

- Easier subkey recovery [Fer05]
- Alternative: GCM-SST [CMP23]
  - Recently proven secure [IJMM25]

Problems With GCM for 96-bit Nonce N (1/2)





Short Key

- Problematic in multi-user setting
- TLS 1.3 masks N with K' [BT16]
- Better solutions for NIST lightweight cryptography standard [DM24]

No Tag Truncation

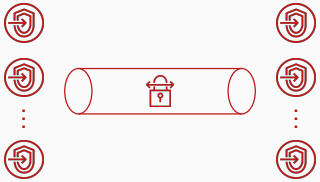
- Easier subkey recovery [Fer05]
- Alternative: GCM-SST [CMP23]
  - Recently proven secure [IJMM25]

Short Nonce

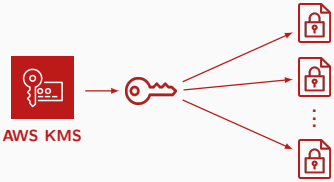
- Biggest problem of all

Problems With GCM for 96-bit Nonce N (2/2)

Use Case: Amazon Web Services [KCCP23]



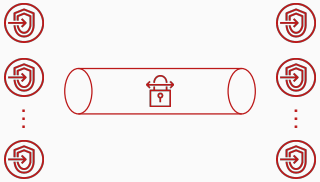
AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds



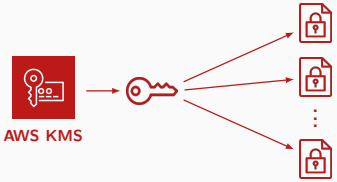
AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

Problems With GCM for 96-bit Nonce N (2/2)

Use Case: Amazon Web Services [KCCP23]



AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds

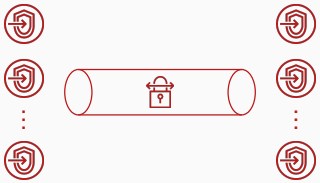


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

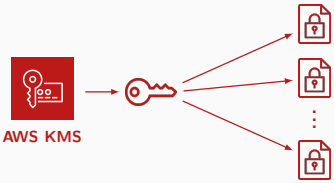
- Counter management is problematic

Problems With GCM for 96-bit Nonce N (2/2)

Use Case: Amazon Web Services [KCCP23]



AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds

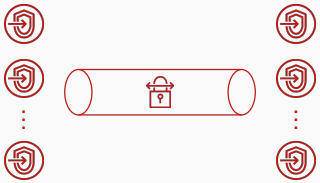


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

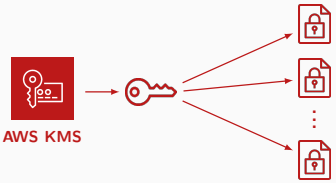
- Counter management is problematic
- Random 96-bit nonces tend to collide

Problems With GCM for 96-bit Nonce N (2/2)

Use Case: Amazon Web Services [KCCP23]



AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds

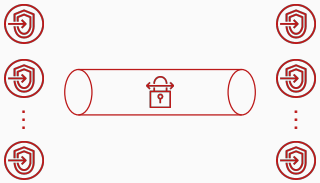


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

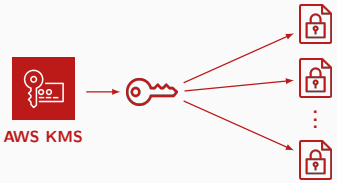
- Counter management is problematic
- Random 96-bit nonces tend to collide
- AWS also needs 32-bit identifiers, leaving only 64 bits of nonce

Problems With GCM for 96-bit Nonce N (2/2)

Use Case: Amazon Web Services [KCCP23]



AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds



AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

- Counter management is problematic
- Random 96-bit nonces tend to collide
- AWS also needs 32-bit identifiers, leaving only 64 bits of nonce
- Limitations of GCM are a real world problem!

Conclusion

Conclusion

Symmetric Cryptography

- Basic modes proved secure using quite simple ideas
- Certain use cases expose limitations of current standards

Conclusion

Symmetric Cryptography

- Basic modes proved secure using quite simple ideas
- Certain use cases expose limitations of current standards

Outline

- Accordion modes $\longrightarrow$ ddd-AES
- Lightweight cryptography $\longrightarrow$ Ascon

References i

Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe.
PRESENT: An Ultra-Lightweight Block Cipher.
In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

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.

References ii


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.


Christoph Dobraunig and Bart Mennink.
Generalized Initialization of the Duplex Construction.
In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 460–484. Springer, 2024.


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

References iii

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

Akiko Inoue, Ashwin Jha, Bart Mennink, and Kazuhiko Minematsu.
Generic Security of GCM-SST.
In Marc Fischlin and Veelasha Moonsamy, editors, *Applied Cryptography and Network Security - 23rd International Conference, ACNS 2025, Munich, Germany, June 23-26, 2025, Proceedings*, Lecture Notes in Computer Science. Springer, 2025.
To appear.

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.