Maastricht University

Basics of Symmetric Cryptography

Bart Mennink

Finse Quantum World Cybersecurity Winter School 2026

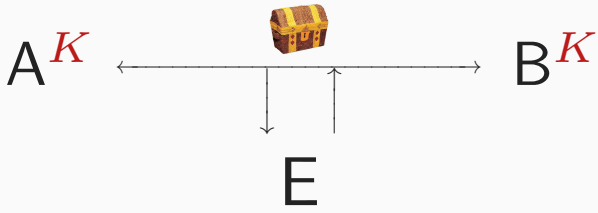
April 22, 2026

Keyed Symmetric Cryptography

General Setting

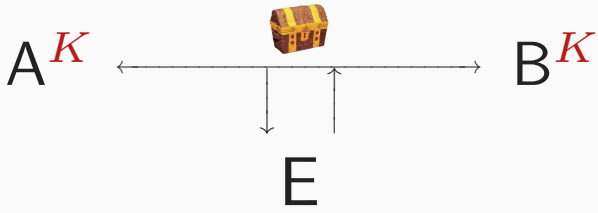
- Two parties, Alice and Bob, communicate over a public channel
  - We assume they have agreed on a joint key K and use it to transmit data

General Setting



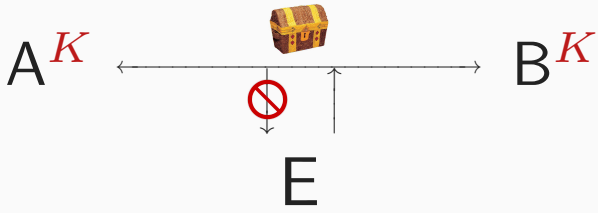
- Two parties, Alice and Bob, communicate over a public channel
  - We assume they have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit disturb/eavesdrop/... communication
- In symmetric cryptography, we are concerned with two main security properties:

General Setting



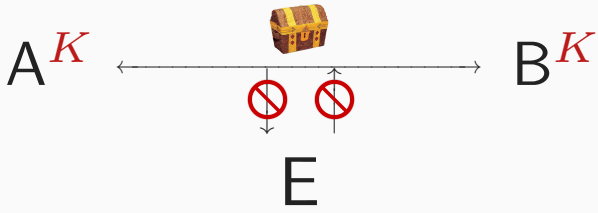
- Two parties, Alice and Bob, communicate over a public channel
  - We assume they have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit disturb/eavesdrop/... communication
- In symmetric cryptography, we are concerned with two main security properties: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

General Setting



- Two parties, Alice and Bob, communicate over a public channel
  - We assume they have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit disturb/eavesdrop/... communication
- In symmetric cryptography, we are concerned with two main security properties:
- 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
  - We assume they have agreed on a joint key K and use it to transmit data
- A malicious party, Eve, may try to exploit disturb/eavesdrop/... communication
- In symmetric cryptography, we are concerned with two main security properties:
- Two main security properties:
  - Confidentiality: Eve cannot learn anything about the content of the data
  - Authenticity: Manipulation/replay/delay of the data gets detected

Encryption

One-Time-Pad Encryption

Encryption:

$$P = \begin{array}{cccccccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

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

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:

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

Decryption:

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

Properties of One-Time Pad

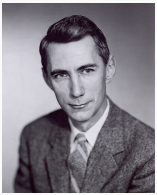
- Perfect secrecy (against an attacker that has no knowledge about the key)
- Key must be as long as the plaintext!

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

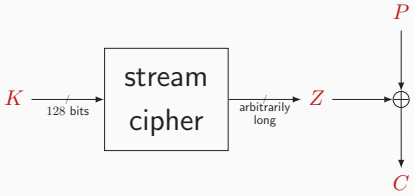


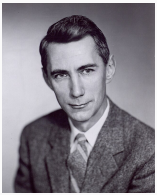
Properties of One-Time Pad

- 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



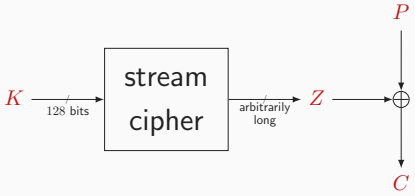


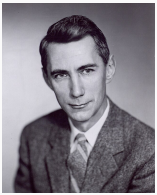
Properties of One-Time Pad

- 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)

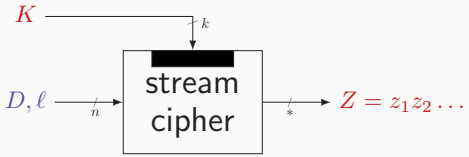
- Key guessing:
  - Exhaustive key search succeeds with probability $\mathbf{Pr}\,(\text{success}) = 1/2^{|K|}$
- Ciphertext Only Attack:
  - Long ciphertexts leak info via letter frequencies
- Known Plaintext Attack:
  - Knowledge of short plaintext sequence reveals full keystream

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

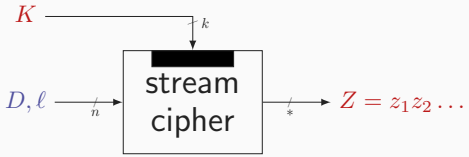
We need something more sophisticated!

Modern Stream Ciphers



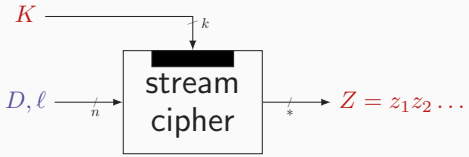
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated

Modern Stream Ciphers



- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- Diversifier is short: typically 96 bits
- Role of diversifier: generate multiple keystreams Z from a single key K

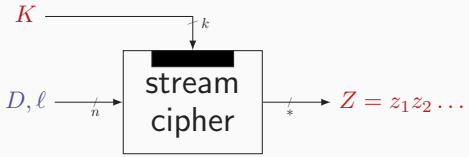
Modern Stream Ciphers



- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- Diversifier is short: typically 96 bits
- Role of diversifier: generate multiple keystreams Z from a single key K

$$C \leftarrow P \oplus \mathsf{SC}_K(D, |P|)$$

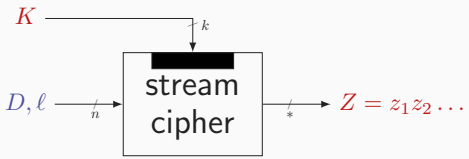
Modern Stream Ciphers



- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- Diversifier is short: typically 96 bits
- Role of diversifier: generate multiple keystreams Z from a single key K

$$C \leftarrow P \oplus \mathsf{SC}_K(D, |P|)$$

When is this stream encryption secure?



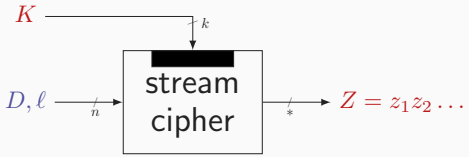
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- Diversifier is short: typically 96 bits
- Role of diversifier: generate multiple keystreams Z from a single key K

$$C \leftarrow P \oplus \mathsf{SC}_K(D, |P|)$$

When is this stream encryption secure?

1. D is unique per message: we call this a nonce: number used once

Modern Stream Ciphers



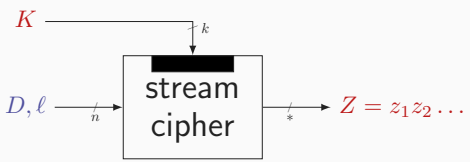
- Using key K, diversifier D, and length ℓ, keystream Z of length ℓ is generated
- Diversifier is short: typically 96 bits
- Role of diversifier: generate multiple keystreams Z from a single key K

$$C \leftarrow P \oplus \mathsf{SC}_K(D, |P|)$$

When is this stream encryption secure?

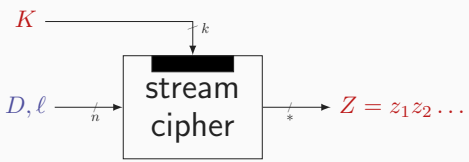
1. D is unique per message: we call this a nonce: number used once
2. Stream cipher itself is secure

Stream Cipher Security, Intuition (1/3)



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

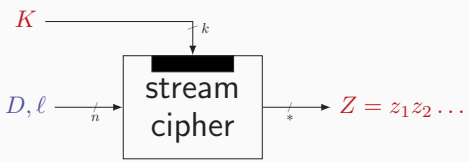
Stream Cipher Security, Intuition (1/3)



- 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
...	...
...	...
...	...
...	...

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
...	...

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
...	...

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 $|Z| < \ell$:

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 $|Z| < \ell$: generate $\ell - |Z|$ random bits Z', append Z' to Z, return $Z \| Z'$

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

Intermezzo: Random Oracle

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

Random Oracle

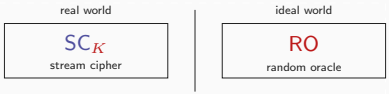
- A database of input-output tuples
- Initially empty
- New query (D, ℓ):
  - If D is not in the database:
    - generate ℓ random bits Z
    - add (D, Z) to the list
    - return Z
  - If D is in the database, look at corresponding Z:
    - If $|Z| \geq \ell$: return first ℓ bits of Z
    - If $|Z|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

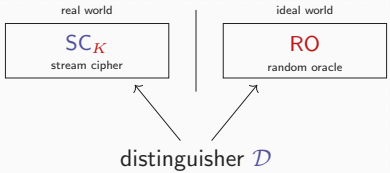
D	Z
1100	101011101010101
1111010101101101	1101011101111101101
001000011100	101011010111010101011
...	...

Stream Cipher Security, Intuition (2/3)



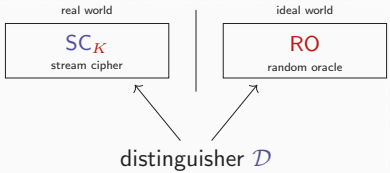
- We thus want to "compare" SC_K with a random oracle RO

Stream Cipher Security, Intuition (2/3)



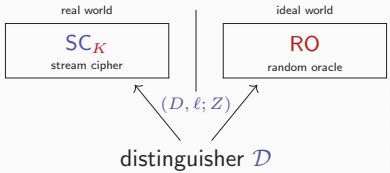
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds

Stream Cipher Security, Intuition (2/3)



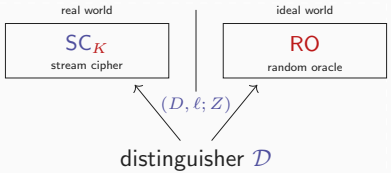
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - Before the experiment, we toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to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

Stream Cipher Security, Intuition (2/3)



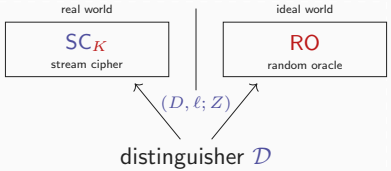
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - Before the experiment, we toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to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



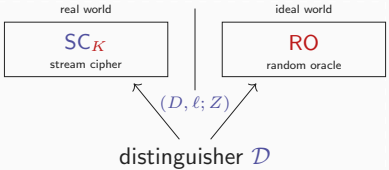
- We thus want to "compare" SC_K with a random oracle RO
- We model a distinguisher $\mathcal{D}$ that is given oracle access to either of the worlds
  - Before the experiment, we toss a coin:
    - head: $\mathcal{D}$ is given oracle access to SC_K
    - tail: $\mathcal{D}$ is given oracle access to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

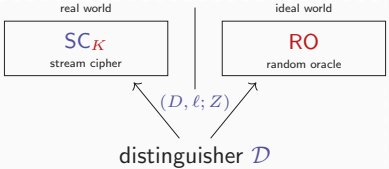
Stream Cipher Security, Intuition (3/3)



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

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

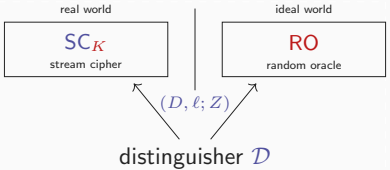
Stream Cipher Security, Intuition (3/3)



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

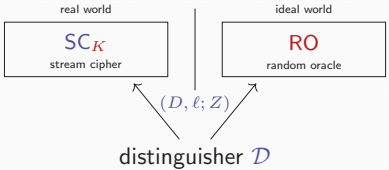


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\,(\text{success})$
- $\mathcal{D}$ can always guess and succeed with prob. $\geq 1/2$, so we scale to $\mathcal{D}$'s advantage:

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

- Resources of $\mathcal{D}$ must be well-defined:
  - Data (or online) complexity q: total cost of queries $\mathcal{D}$ can make, e.g., expressed in number of bits exchanged

Stream Cipher Security, Intuition (3/3)

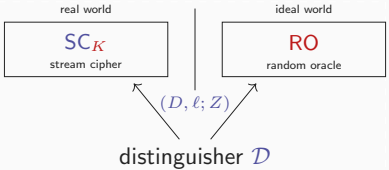


- Denote $\mathcal{D}$'s success probability in correctly guessing head/tail by $\mathbf{Pr}\left(\text{success}\right)$
- $\mathcal{D}$ can always guess and succeed with prob. $\geq 1/2$, so we scale to $\mathcal{D}$'s advantage:

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

- Resources of $\mathcal{D}$ must be well-defined:
  - Data (or online) complexity q: total cost of queries $\mathcal{D}$ can make, e.g., expressed in number of bits exchanged
  - 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 succeed with prob. $\geq 1/2$, so we scale to $\mathcal{D}$'s advantage:

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

- Resources of $\mathcal{D}$ must be well-defined:
  - Data (or online) complexity q: total cost of queries $\mathcal{D}$ can make, e.g., expressed in number of bits exchanged
  - 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: Problem

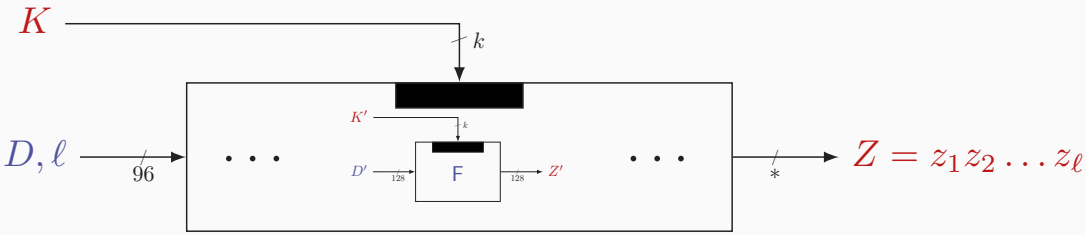
- Stream cipher built from smaller cryptographic primitive

Generic Stream Cipher Design: Problem

- 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: Problem

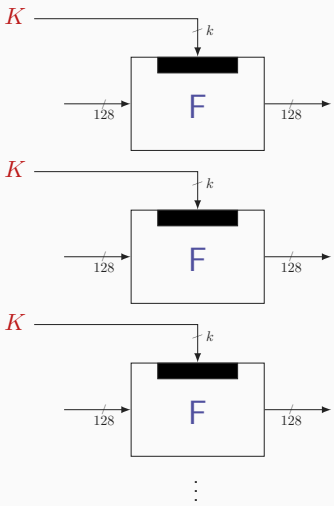
- Stream cipher built from smaller cryptographic primitive
- Suppose (for the sake of argument):
  - we **know** how to build a strong stream cipher F with fixed-length output
  - we **want** to build a stream cipher with variable-length output



Generic Stream Cipher Design: Solution

Design

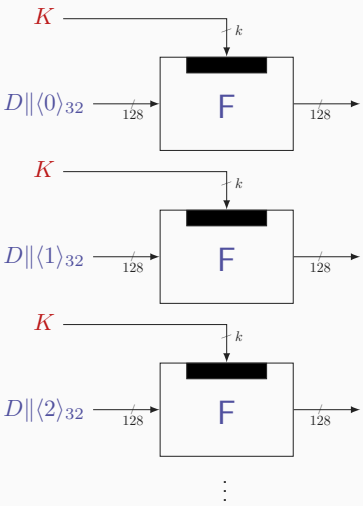
- Feed K to primitive



Generic Stream Cipher Design: Solution

Design

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

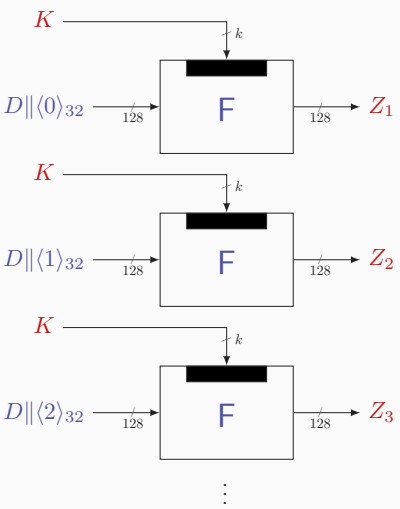


Generic Stream Cipher Design: Solution

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: Solution

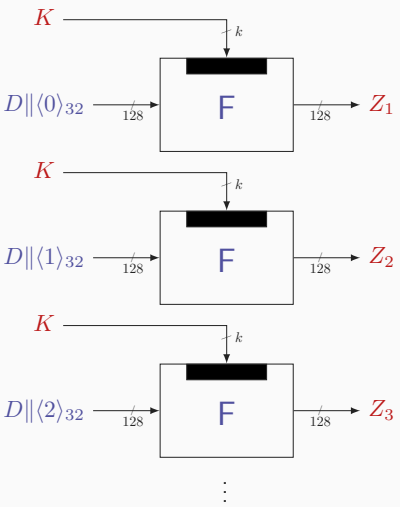
Design

- 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'

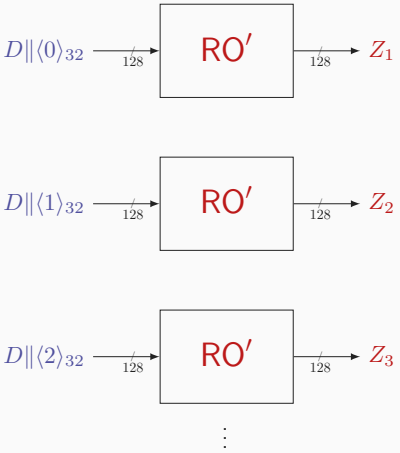


Generic Stream Cipher Design: Solution

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

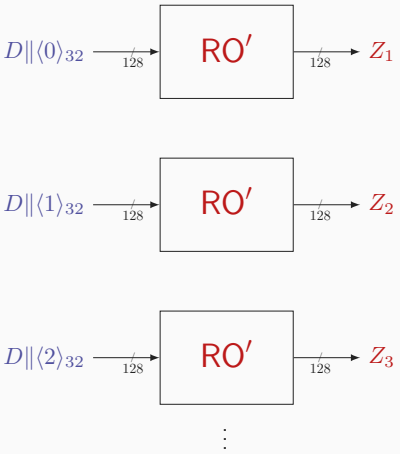
- If F_K is hard to distinguish from a RO'

Generic Stream Cipher Design: Solution

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

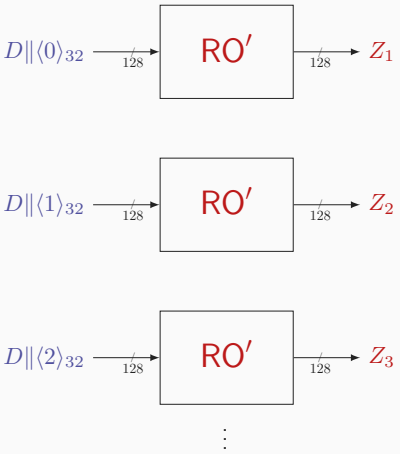
- If F_K is hard to distinguish from a RO′
- Then construction is hard to distinguish from RO

Generic Stream Cipher Design: Solution

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

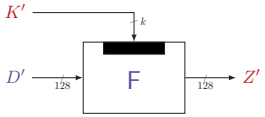
Generic Stream Cipher Design: Solution

Design

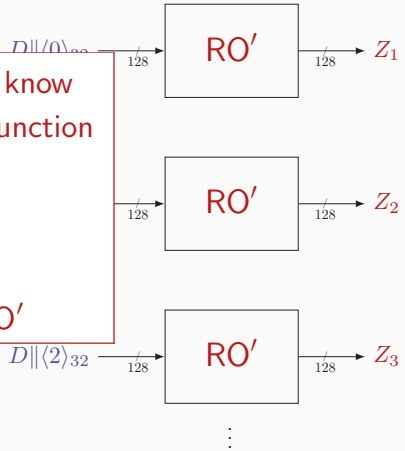
- Feed K to primitive
- Evaluate primitive as often as needed, with D concatenated with co
- Concatenate outputs:

$$Z = Z_1$$

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



that behaves like a RO′

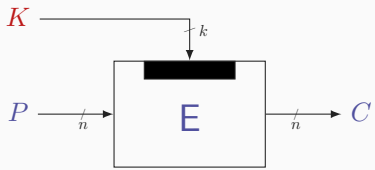


Security

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

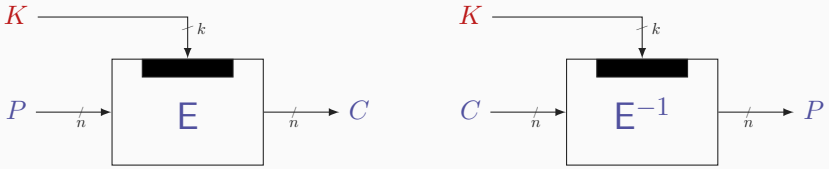
Block Ciphers and AES

Block Ciphers



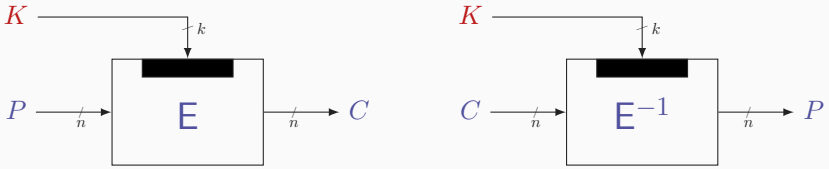
- Using key K, plaintext P is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size

Block Ciphers



- Using key K, plaintext P 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}

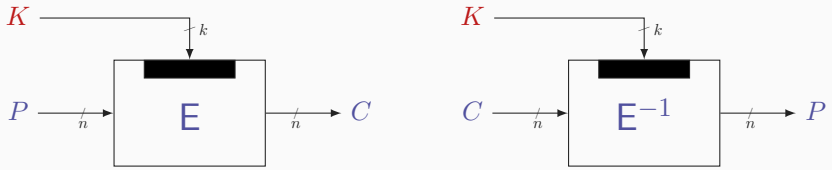
Block Ciphers



- Using key K, plaintext P 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]:

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

Block Ciphers

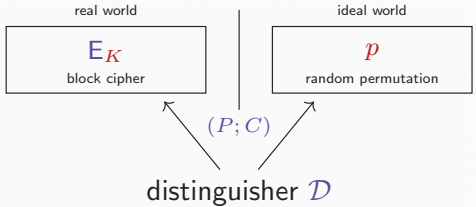


- Using key K, plaintext P is bijectively transformed to ciphertext C
- Key, plaintext, and ciphertext are typically of fixed size
- For fixed key, E_K is invertible and the inverse is denoted as E_K^{-1}
- Example [DR02]:

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

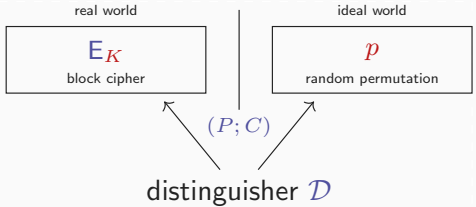
- A good E should behave like a random permutation if one does not know K

Block cipher (PRP) security



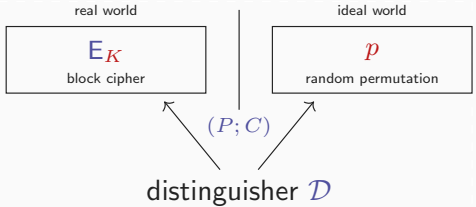
- Two oracles: E_K (for secret key K) and p (secret)

Block cipher (PRP) security



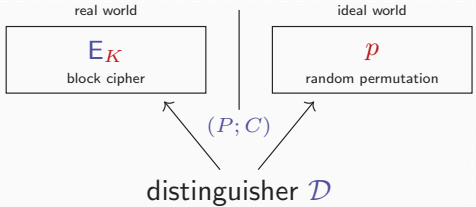
- Two oracles: E_K (for secret key K) and p (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these

Block cipher (PRP) 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

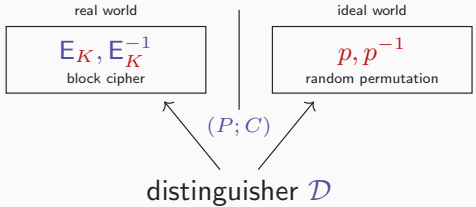
Block cipher (PRP) security



- 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:

$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{E}_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{p} = 1\right)\right|$$

Block cipher (PRP) security

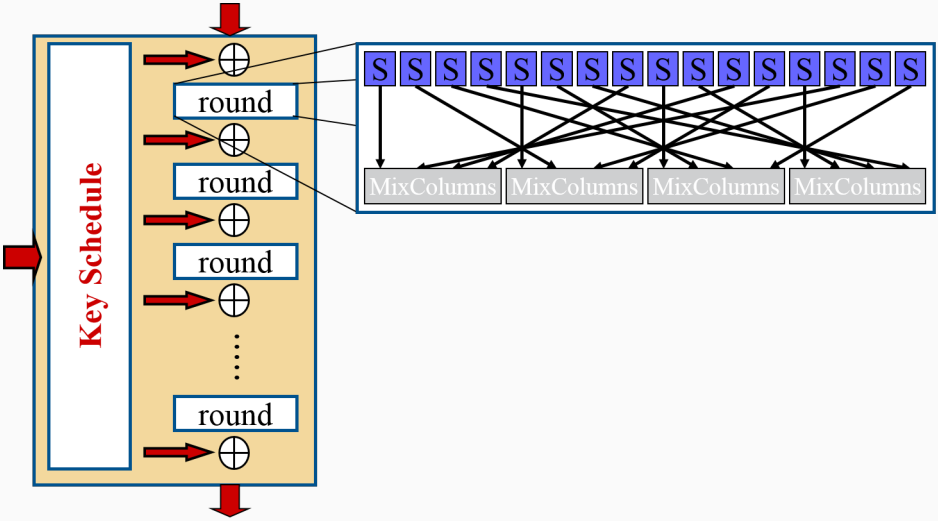


- Two oracles: E_K (for secret key K) and p (secret)
- Distinguisher $\mathcal{D}$ has query access to one of these
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

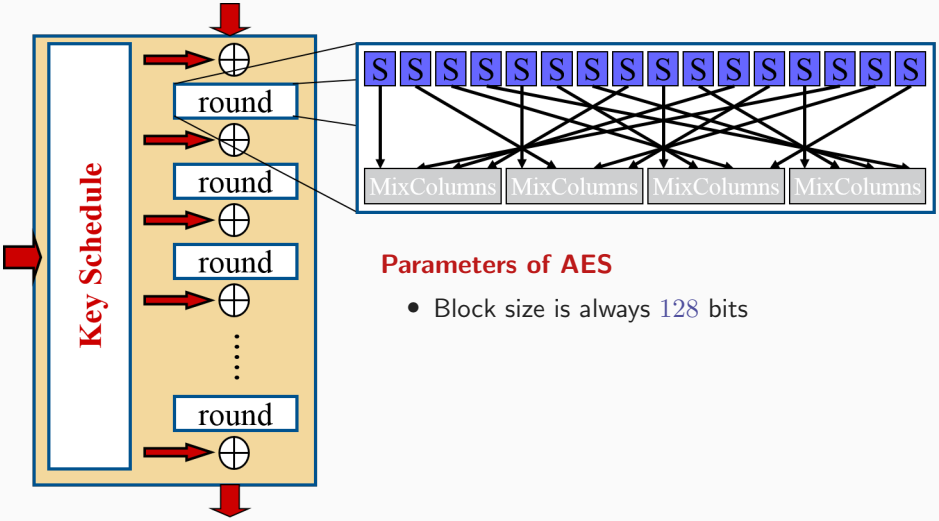
$$\mathbf{Adv}_E^{\mathrm{prp}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathsf{E}_K} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{p} = 1\right)\right|$$

- Extension to SPRP security: $\mathcal{D}$ gets both forward and inverse query access

AES in a Nutshell



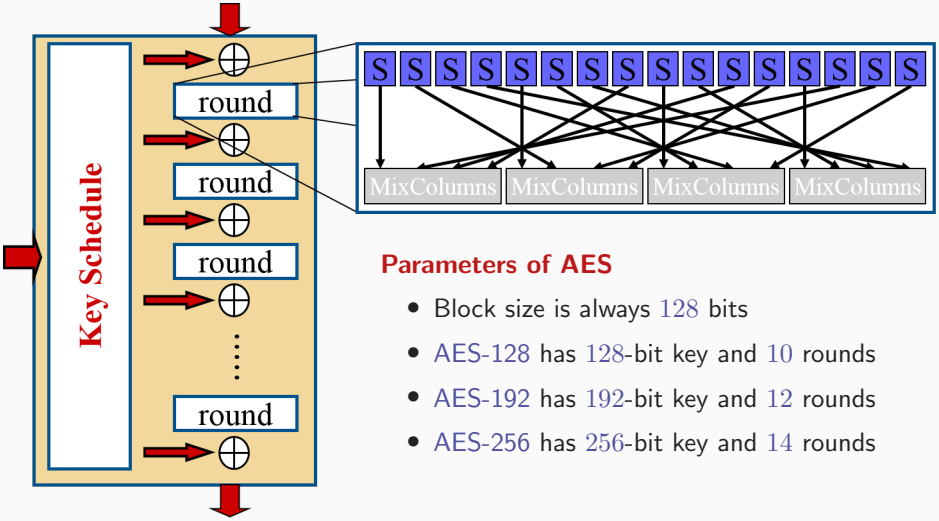
AES in a Nutshell



Parameters of AES

- Block size is always 128 bits

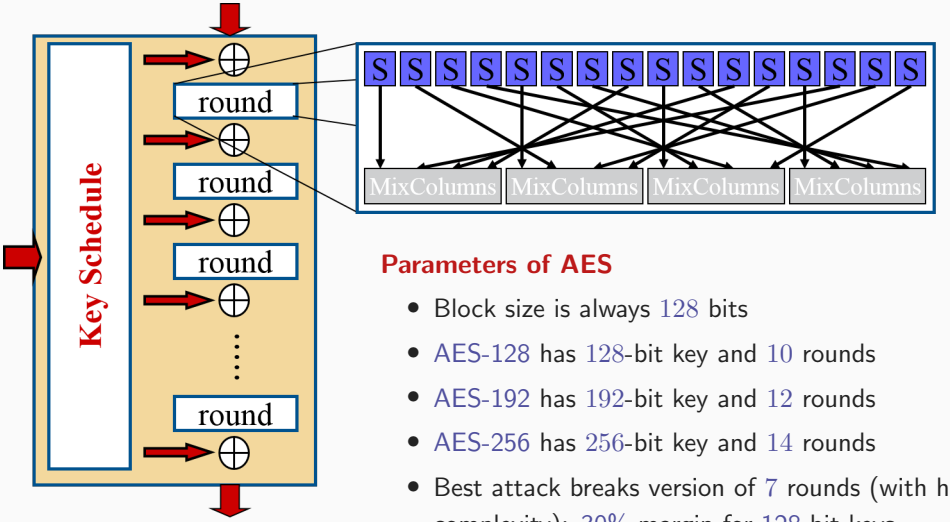
AES in a Nutshell



Parameters of AES

- Block size is always 128 bits
- AES-128 has 128-bit key and 10 rounds
- AES-192 has 192-bit key and 12 rounds
- AES-256 has 256-bit key and 14 rounds

AES in a Nutshell



Parameters of AES

- Block size is always 128 bits
- AES-128 has 128-bit key and 10 rounds
- AES-192 has 192-bit key and 12 rounds
- AES-256 has 256-bit key and 14 rounds
- Best attack breaks version of 7 rounds (with high complexity): 30% margin for 128-bit keys

Security Claim

- For any distinguisher $\mathcal{D}$ with q queries and t time,

$$\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{AES}}(\mathcal{D}) \leq \frac{t}{2^k}$$

- Very similar to claim in the original specification of AES [DR99]

Security Claim

- For any distinguisher $\mathcal{D}$ with q queries and t time,

$$\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{AES}}(\mathcal{D}) \leq \frac{t}{2^k}$$

- Very similar to claim in the original specification of AES [DR99]
- In normal English: exhaustive key search is the best possible attack

Classical Security of AES

Security Claim

- For any distinguisher $\mathcal{D}$ with q queries and t time,

$$\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{AES}}(\mathcal{D}) \leq \frac{t}{2^k}$$

- Very similar to claim in the original specification of AES [DR99]
- In normal English: exhaustive key search is the best possible attack

State of the Art After 25 Years

- No attacks on full-round AES
- Attacks on round-reduced AES have huge data complexity

Classical Security of AES

Security Claim

- For any distinguisher $\mathcal{D}$ with q queries and t time,

$$\mathbf{Adv}^{\mathrm{sprp}}_{\mathsf{AES}}(\mathcal{D}) \leq \frac{t}{2^k}$$

- Very similar to claim in the original specification of AES [DR99]
- In normal English: exhaustive key search is the best possible attack

State of the Art After 25 Years

- No attacks on full-round AES
- Attacks on round-reduced AES have huge data complexity
- Best attacks are implementation/side-channel attacks

How Post-Quantum Secure is AES?

How Post-Quantum Secure is AES?

Grover's Algorithm [Gro96]

- Quadratic speed-up of exhaustive key search
- In other words: key search requires $t = 2^{k/2}$ time

How Post-Quantum Secure is AES?

Grover's Algorithm [Gro96]

- Quadratic speed-up of exhaustive key search
- In other words: key search requires $t = 2^{k/2}$ time
- Tiny problem...

How Post-Quantum Secure is AES?

Grover's Algorithm [Gro96]

- Quadratic speed-up of exhaustive key search
- In other words: key search requires $t = 2^{k/2}$ time
- Tiny problem... Grover's algorithm cannot be parallelized [Zal99]

How Post-Quantum Secure is AES?

Grover's Algorithm [Gro96]

- Quadratic speed-up of exhaustive key search
- In other words: key search requires $t = 2^{k/2}$ time
- Tiny problem... Grover's algorithm cannot be parallelized [Zal99]

National Institute of Standards and Technology [NISa, NISb]

It is worth noting that the security categories based on these reference primitives provide substantially more quantum security than a naïve analysis might suggest. For example, categories 1, 3 and 5 are defined in terms of block ciphers, which can be broken using Grover's algorithm, with a quadratic quantum speedup. But Grover's algorithm requires a long-running serial computation, which is difficult to implement in practice. In a realistic attack, one has to run many smaller instances of the algorithm in parallel, which makes the quantum speedup less dramatic[6]

How Post-Quantum Secure is AES?

Grover's Algorithm [Gro96]

- Quadratic speed-up of exhaustive key search
- In other words: key search requires $t = 2^{k/2}$ time
- Tiny problem... Grover's algorithm cannot be parallelized [Zal99]

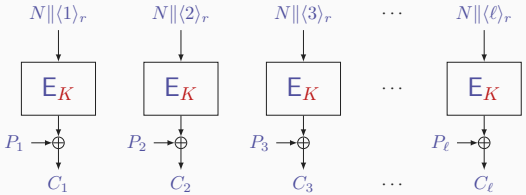
National Institute of Standards and Technology [NISa, NISb]

It is worth noting that the security categories based on these reference primitives provide substantially more quantum security than a naïve analysis might suggest. For example, categories 1, 3 and 5 are defined in terms of block ciphers, which can be broken using Grover's algorithm, with a quadratic quantum speedup. But Grover's algorithm requires a long-running serial computation, which is difficult to implement in practice. In a realistic attack, one has to run many smaller instances of the algorithm in parallel, which makes the quantum speedup less dramatic.[6]

Bottom Line: Don't Worry, Be Happy!

Encryption (continued)

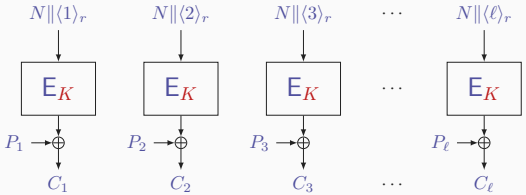
Counter (CTR) Mode



Features

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

Counter (CTR) Mode



Features

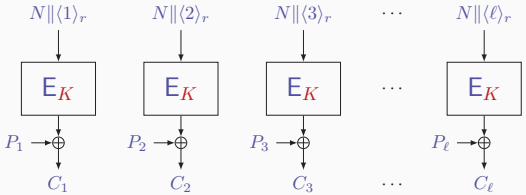
- Most widely used encryption mode
- Fully parallelizable (encryption and decryption) and extremely simple

Security

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

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

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 $q \ll 2^{n/2}$:

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

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:

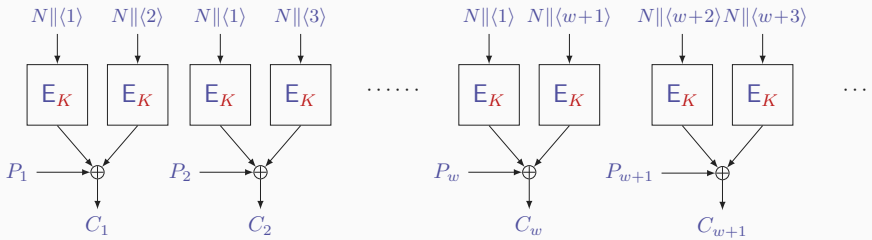
$$\mathbf{Ex}\left[\text{collisions}\right] = \binom{q}{2}/2^n$$

- Important phenomenon in cryptography

Lightweight Cryptography

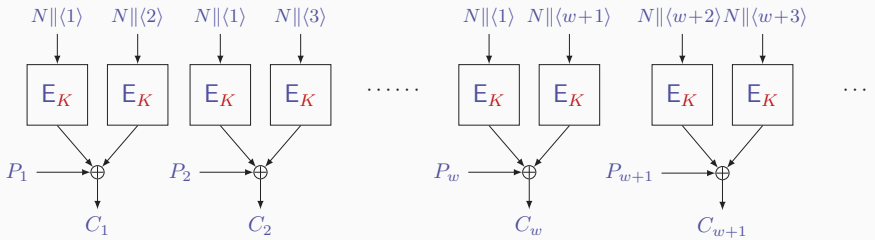
- AES has block size of 128 bits: birthday paradox problem only in certain settings
- Lightweight cryptography operates in constrained environments
  - IoT, healthcare devices, sensors, RFID tags, ...
- Lightweight block ciphers have a block size of 64 bits or even less [BKL+07]

CENC [Iwa06]



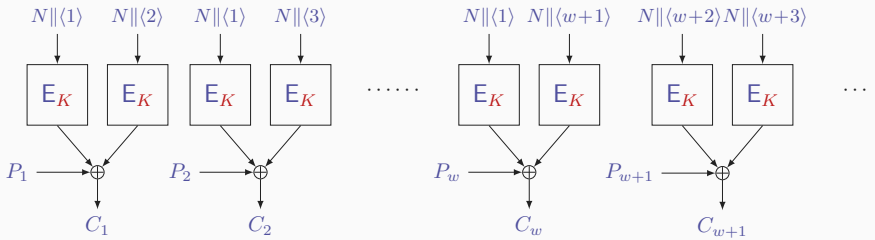
- One subkey used for $w \geq 1$ encryptions

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]:

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

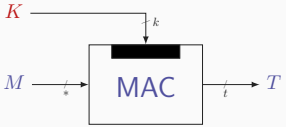
Authentication

Authenticity and MAC Functions

- Encryption does not guarantee that Alice can detect message tampering by Eve

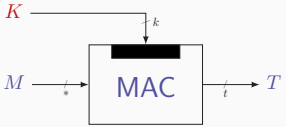
Authenticity and MAC Functions

- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



Authenticity and MAC Functions

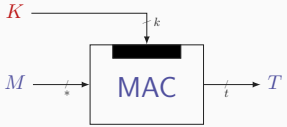
- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



- Alice computes over the (possibly long) message M a short authentication tag T

Authenticity and MAC Functions

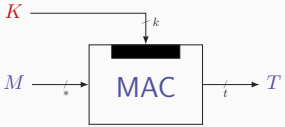
- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



- Alice computes over the (possibly long) message M a short authentication tag T
  - Computation of T requires a secret key K
  - With key it is easy to compute T, without key not

Authenticity and MAC Functions

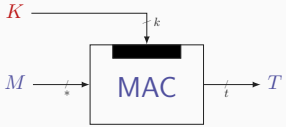
- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



- Alice computes over the (possibly long) message M a short authentication tag T
  - Computation of T requires a secret key K
  - With key it is easy to compute T, without key not
- Alice sends (M, T) to Bob
  - Bob uses his key K to compute T' from M
  - If $T = T'$, he accepts the message as coming from Alice

Authenticity and MAC Functions

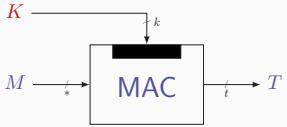
- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



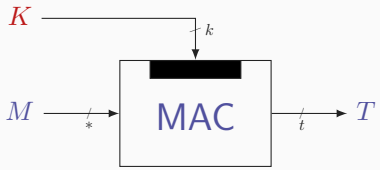
- Alice computes over the (possibly long) message M a short authentication tag T
  - Computation of T requires a secret key K
  - With key it is easy to compute T, without key not
- Alice sends (M, T) to Bob
  - Bob uses his key K to compute T' from M
  - If $T = T'$, he accepts the message as coming from Alice
- A *forgery* is a pair (M, T) not coming from Alice, accepted by Bob

Authenticity and MAC Functions

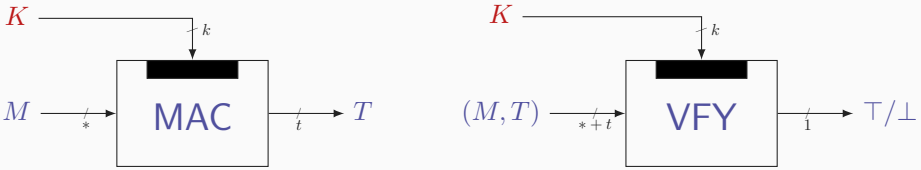
- Encryption does not guarantee that Alice can detect message tampering by Eve
- Use a Message Authentication Code (MAC) function:



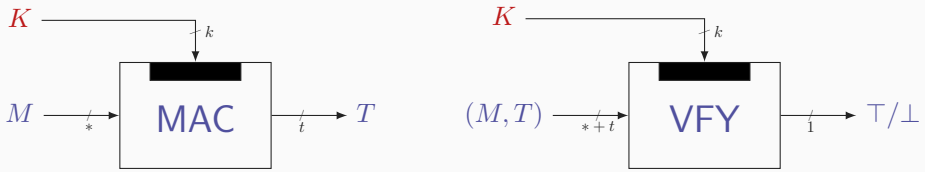
- Alice computes over the (possibly long) message M a short authentication tag T
  - Computation of T requires a secret key K
  - With key it is easy to compute T, without key not
- Alice sends (M, T) to Bob
  - Bob uses his key K to compute T' from M
  - If $T = T'$, he accepts the message as coming from Alice
- A *forgery* is a pair (M, T) not coming from Alice, accepted by Bob
- Often, Bob wants to authenticate more, requiring *active checks* from her part



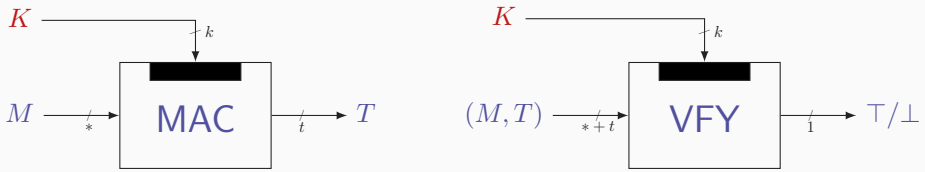
- Using key K, message M is signed with tag T



- Using key K, message M is signed with tag T
- Associated verification function takes K and (M, T) and outputs
  - $\top$ if tag is correct
  - $\bot$ if tag is incorrect

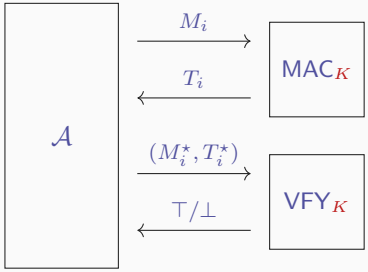


- Using key K, message M is signed with tag T
- Associated verification function takes K and (M, T) and outputs
  - $\top$ if tag is correct
  - $\bot$ if tag is incorrect
- Security goal: MAC_K should behave like a random function
  - I.e., $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{MAC}}(\mathcal{D})$ should be small

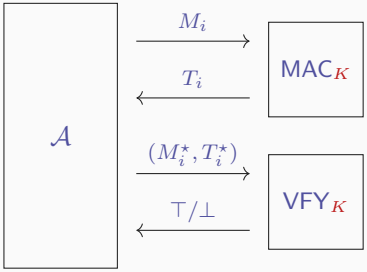


- Using key K, message M is signed with tag T
- Associated verification function takes K and (M, T) and outputs
  - $\top$ if tag is correct
  - $\bot$ if tag is incorrect
- Security goal: MAC_K should behave like a random function
  - I.e., $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{MAC}}(\mathcal{D})$ should be small
- PRF security implies a weaker notion, called unforgeability

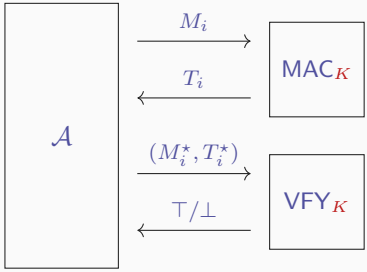
- MAC security defined using unforgeability



- MAC security defined using unforgeability
- Adversary $\mathcal{A}$ has access to MAC_K and VFY_K

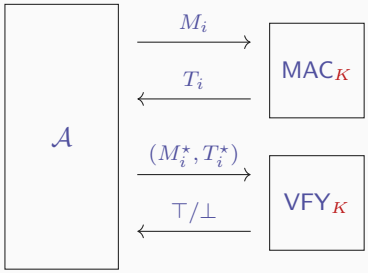


- MAC security defined using unforgeability
- Adversary $\mathcal{A}$ has access to MAC_K and VFY_K
- It can query both oracles interchangeably
  - No query VFY_K for an output of MAC_K

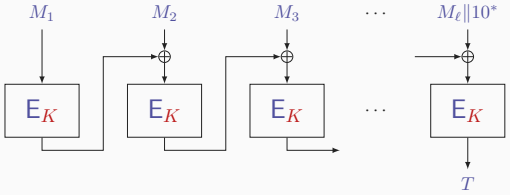


- MAC security defined using unforgeability
- Adversary $\mathcal{A}$ has access to MAC_K and VFY_K
- It can query both oracles interchangeably
  - No query VFY_K for an output of MAC_K
- $\mathcal{A}$ mounts a forgery if VFY_K ever outputs $\top$
- Its advantage is defined as:

$$\mathbf{Adv}^{\mathrm{unf}}_{\mathsf{MAC}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}^{\mathsf{MAC}_K,\mathsf{VFY}_K} \text{ forges}\right)$$

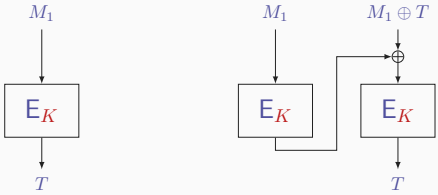


Cipher Block Chaining MAC (CBC-MAC): MAC Function From Block Cipher



- Apply E_K to first block of padded message M_1 to give initial CV
- Absorb message block M_i in CV by computing $CV \leftarrow \mathsf{E}_K(CV \oplus M_i)$
- Tag T is the final value of CV

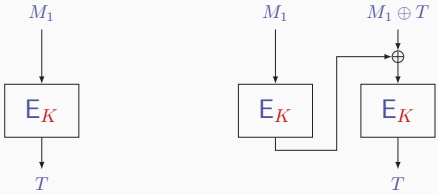
Cipher Block Chaining MAC (CBC-MAC): Weakness



- In general, CBC-MAC can be distinguished from random in two queries:
  - Query M_1, tag equals $T = \mathsf{E}_K(M_1)$
  - Query $M_1 \| (M_1 \oplus T)$, tag equals

$$\mathsf{E}_K(\mathsf{E}_K(M_1) \oplus M_1 \oplus T) = \mathsf{E}_K(T \oplus M_1 \oplus T) = \mathsf{E}_K(M_1) = T$$

Cipher Block Chaining MAC (CBC-MAC): Weakness

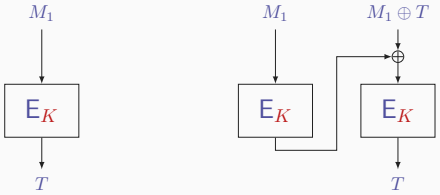


- In general, CBC-MAC can be distinguished from random in two queries:
  - Query M_1, tag equals $T = \mathsf{E}_K(M_1)$
  - Query $M_1 \| (M_1 \oplus T)$, tag equals

$$\mathsf{E}_K(\mathsf{E}_K(M_1) \oplus M_1 \oplus T) = \mathsf{E}_K(T \oplus M_1 \oplus T) = \mathsf{E}_K(M_1) = T$$

- CBC-MAC is not unforgeable and definitely not PRF-secure

Cipher Block Chaining MAC (CBC-MAC): Weakness

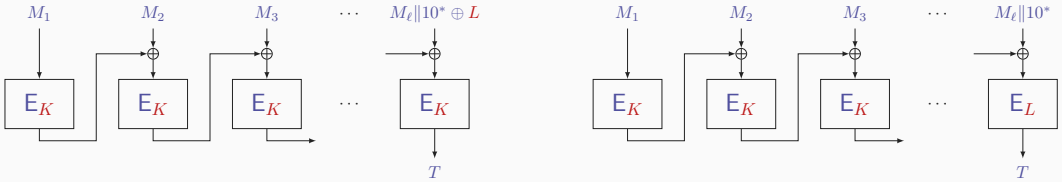


- In general, CBC-MAC can be distinguished from random in two queries:
  - Query M_1, tag equals $T = \mathsf{E}_K(M_1)$
  - Query $M_1 \| (M_1 \oplus T)$, tag equals

$$\mathsf{E}_K(\mathsf{E}_K(M_1) \oplus M_1 \oplus T) = \mathsf{E}_K(T \oplus M_1 \oplus T) = \mathsf{E}_K(M_1) = T$$

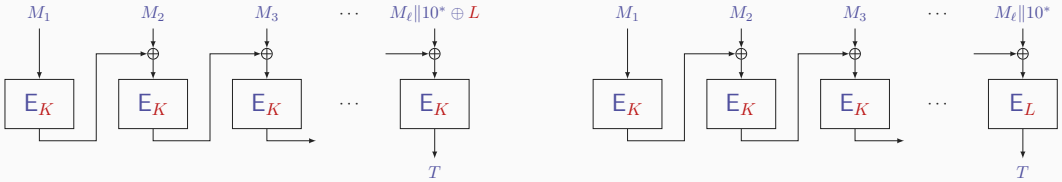
- CBC-MAC is not unforgeable and definitely not PRF-secure
- Note: attack ignores padding, but this can be dealt with

CBC-MAC(-like): MAC Function From Block Cipher



- Solution 1: mask last block with dedicated key L (known as C-MAC [Mor05])
- Solution 2: apply independent last primitive call

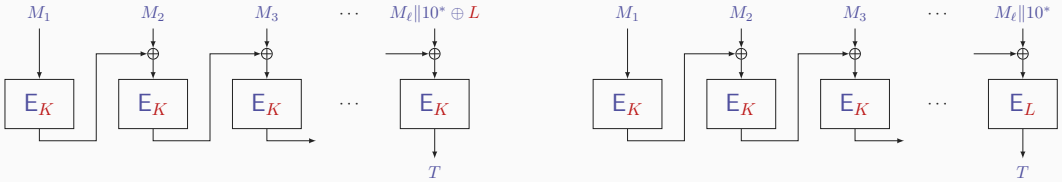
CBC-MAC(-like): MAC Function From Block Cipher



- Solution 1: mask last block with dedicated key L (known as C-MAC [Mor05])
- Solution 2: apply independent last primitive call

- Secure if E_K behaves like a random permutation

CBC-MAC(-like): MAC Function From Block Cipher



- Solution 1: mask last block with dedicated key L (known as C-MAC [Mor05])
- Solution 2: apply independent last primitive call

- Secure if E_K behaves like a random permutation
- Security again only up to the birthday bound: $2^{n/2}$ blocks

But Wait, Quantum Period Finding!

Simon's Algorithm [Sim97]

- Period finding can be done in polynomial time
- C-MAC permits a period and can be broken in $O(n)$ time [KLLN16]

But Wait, Quantum Period Finding!

Simon's Algorithm [Sim97]

- Period finding can be done in polynomial time
- C-MAC permits a period and can be broken in $O(n)$ time [KLLN16]
- However... attack only works if distinguisher has superposition query capabilities

But Wait, Quantum Period Finding!

Simon's Algorithm [Sim97]

- Period finding can be done in polynomial time
- C-MAC permits a period and can be broken in $O(n)$ time [KLLN16]
- However... attack only works if distinguisher has superposition query capabilities
  - Solution: do not implement your keyed algorithm on a quantum computer

But Wait, Quantum Period Finding!

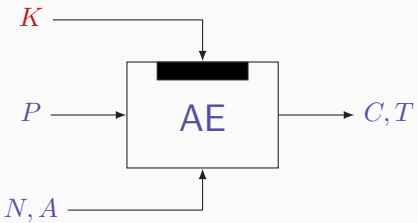
Simon's Algorithm [Sim97]

- Period finding can be done in polynomial time
- C-MAC permits a period and can be broken in $O(n)$ time [KLLN16]
- However... attack only works if distinguisher has superposition query capabilities
  - Solution: do not implement your keyed algorithm on a quantum computer

Bottom Line: Don't Worry, Be Happy!

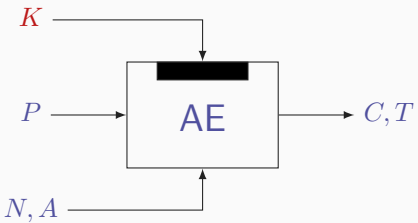
Authenticated Encryption and GCM

Authenticated Encryption (AE): Wrap



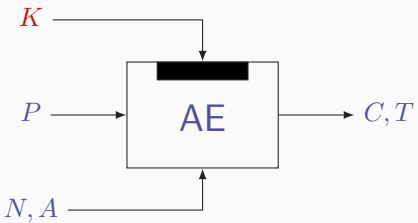
- Input: nonce N, associated data A, and plaintext P
- Ciphertext C encrypts P, and tag T authenticates (N, A, P)

Authenticated Encryption (AE): Wrap



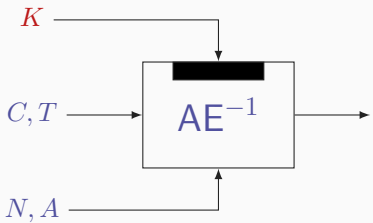
- Input: nonce N, associated data A, and plaintext P
- Ciphertext C encrypts P, and tag T authenticates (N, A, P)
- Schemes typically require uniqueness of N
  - if not unique, typically ciphertexts leak plaintext difference...
  - ... or even key material!

Authenticated Encryption (AE): Wrap



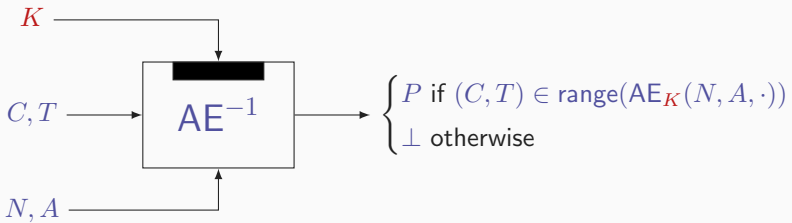
- Input: nonce N, associated data A, and plaintext P
- Ciphertext C encrypts P, and tag T authenticates (N, A, P)
- Schemes typically require uniqueness of N
  - if not unique, typically ciphertexts leak plaintext difference...
  - ... or even key material!
- There exist AE schemes that are secure without a nonce: more expensive

Authenticated Encryption (AE): Unwrap

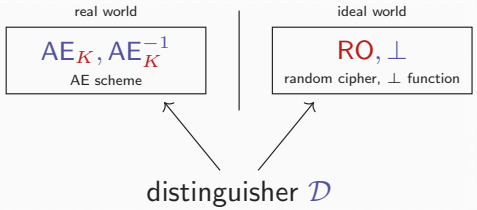


- Unwrapping needs to satisfy that
  - plaintext P disclosed if (C, T) comes from an evaluation of wrap
  - error symbol otherwise

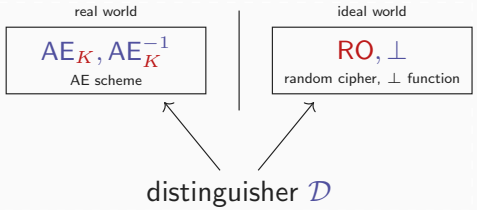
Authenticated Encryption (AE): Unwrap



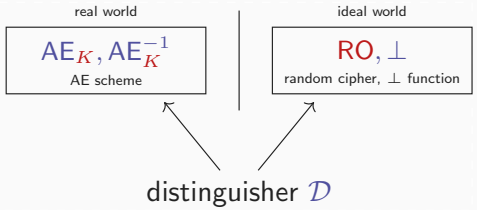
- Unwrapping needs to satisfy that
  - plaintext P disclosed if (C, T) comes from an evaluation of wrap
  - error symbol otherwise



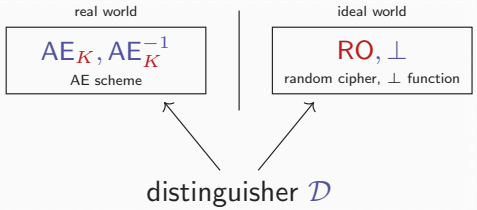
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (ideal)



- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (ideal)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ no trivial queries: query AE_K^{-1} on result of AE_K
  $\rightarrow$ should respect uniqueness of N



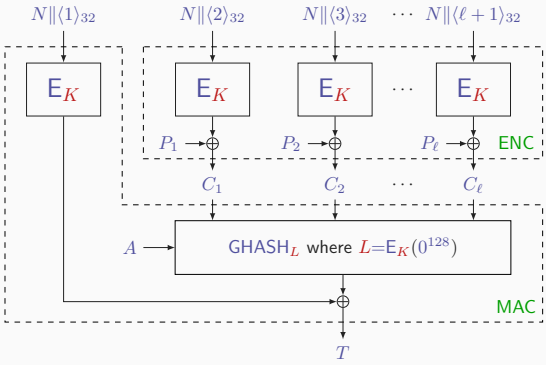
- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (ideal)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ no trivial queries: query AE_K^{-1} on result of AE_K
  $\rightarrow$ should respect uniqueness of N
- $\mathcal{D}$ tries to determine which oracle it communicates with

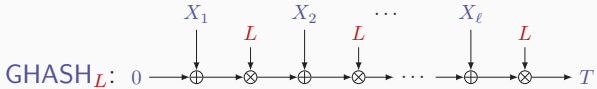


- Two oracles: $(\mathsf{AE}_K, \mathsf{AE}_K^{-1})$ (for secret key K) and $(\mathsf{RO}, \perp)$ (ideal)
- Distinguisher $\mathcal{D}$ has query access to one of these
  $\rightarrow$ no trivial queries: query AE_K^{-1} on result of AE_K
  $\rightarrow$ should respect uniqueness of N
- $\mathcal{D}$ tries to determine which oracle it communicates with
- Its advantage is defined as:

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

GCM for 96-bit Nonce N [MV04]

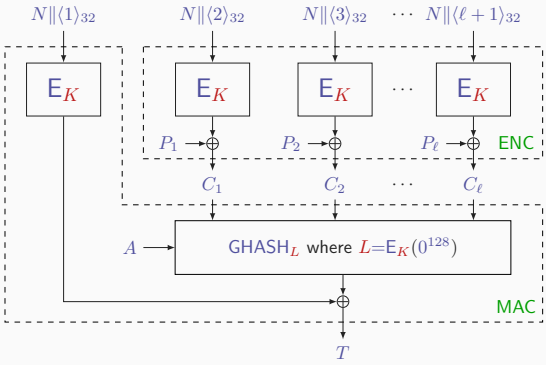


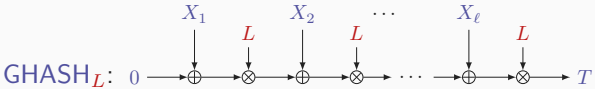


Features

- Efficient, parallelizable, inverse-free
- Widely used:
  - TLS, WPA3, IPsec, ...
- Cooler variant: ChaCha20-Poly1305!

GCM for 96-bit Nonce N [MV04]





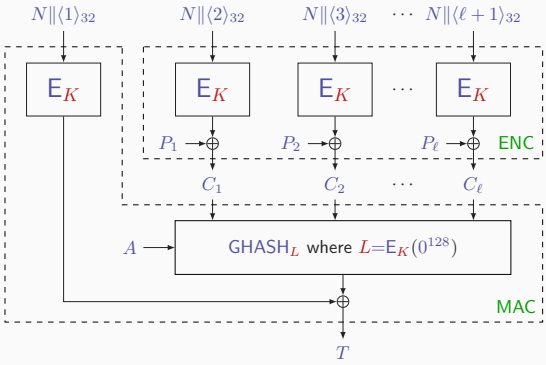
Features

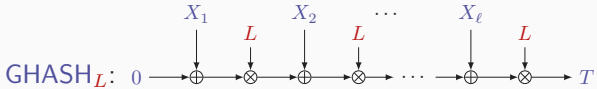
- Efficient, parallelizable, inverse-free
- Widely used:
  - TLS, WPA3, IPsec, ...
- Cooler variant: ChaCha20-Poly1305!

Secure as Long as...

- E_K is a secure block cipher
- Number of blocks doesn't exceed $2^{n/2}$
- N is never repeated...

GCM for 96-bit Nonce N [MV04]





Features

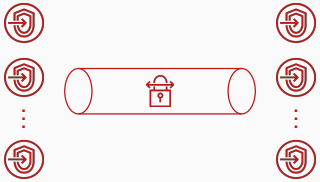
- Efficient, parallelizable, inverse-free
- Widely used:
  - TLS, WPA3, IPsec, ...
- Cooler variant: ChaCha20-Poly1305!

Secure as Long as...

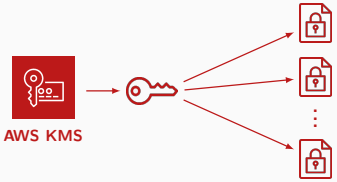
- E_K is a secure block cipher
- Number of blocks doesn't exceed $2^{n/2}$
- N is never repeated...
- ... but nonce reuse is devastating
  - Leaks $P \oplus P' = C \oplus C'$ and L

Problems With GCM for 96-bit Nonce N

Use Case: Amazon Web Services [KCCP23]



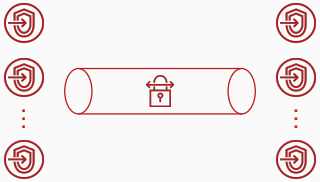
AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds



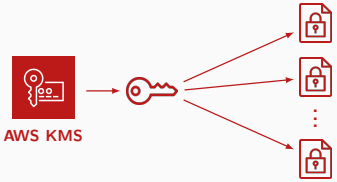
AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

Problems With GCM for 96-bit Nonce N

Use Case: Amazon Web Services [KCCP23]



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

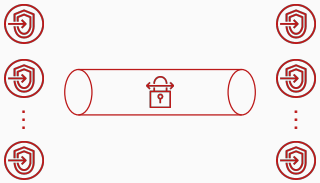


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

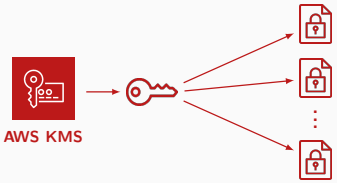
- Counter management is problematic

Problems With GCM for 96-bit Nonce N

Use Case: Amazon Web Services [KCCP23]



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

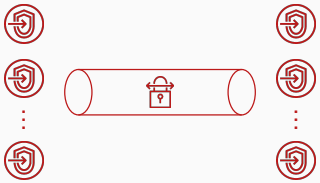


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

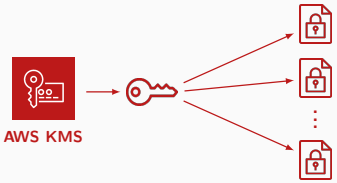
- Counter management is problematic
- Random 96-bit nonces tend to collide

Problems With GCM for 96-bit Nonce N

Use Case: Amazon Web Services [KCCP23]



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

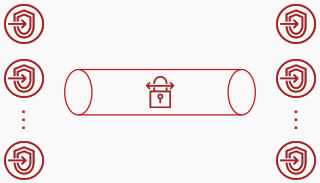


AWS KMS key encrypts
up to $\approx 2^{32}$ plaintexts per week

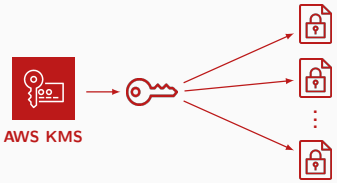
- Counter management is problematic
- Random 96-bit nonces tend to collide
- AWS also needs 32-bit identifiers, leaving only 64 bits of nonce

Problems With GCM for 96-bit Nonce N

Use Case: Amazon Web Services [KCCP23]



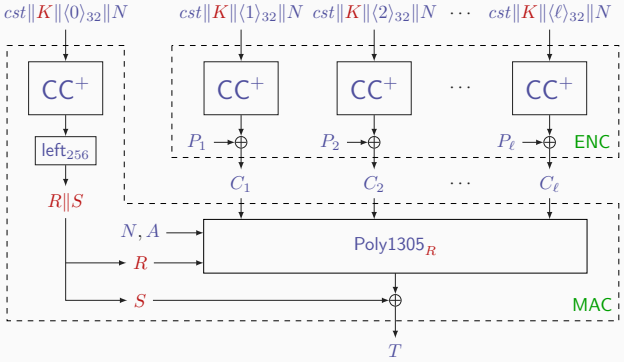
AWS distributed systems encrypt
up to $\approx 2^{32}$ plaintexts per 2 seconds



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!

ChaCha20-Poly1305

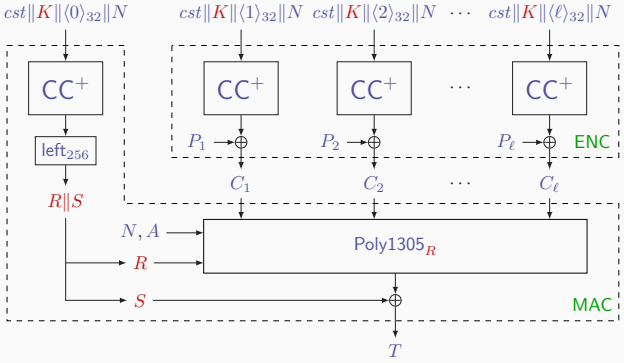


- Bernstein [Ber05, Ber08]
- RFC 8439 [NL18]

- CC: 512-bit permutation
- CC$^+$: CC plus feed-forward

$$\text{Poly1305}_R(X) = (1\|X_1)\cdot R^\ell + (1\|X_2)\cdot R^{\ell-1} + \cdots + (1\|X_{\ell-1})\cdot R^2 + X_\ell \cdot R \bmod 2^{130} - 5$$

ChaCha20-Poly1305



- Bernstein [Ber05, Ber08]
- RFC 8439 [NL18]

- CC: 512-bit permutation
- CC$^+$: CC plus feed-forward

- Both key and mask of Poly1305 are nonce-dependent

$$\text{Poly1305}_R(X) = (1\|X_1)\cdot R^\ell + (1\|X_2)\cdot R^{\ell-1} + \cdots + (1\|X_{\ell-1})\cdot R^2 + X_\ell \cdot R \bmod 2^{130} - 5$$

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

Advances in the Field

- Accordion modes
- Permutation-based cryptography
- Lightweight cryptography

Conclusion

Symmetric Cryptography

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

Advances in the Field

- Accordion modes
- Permutation-based cryptography
- Lightweight cryptography

Thank you for your attention!

References i


Daniel J. Bernstein.
The Poly1305-AES Message-Authentication Code.
In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.


Daniel J. Bernstein.
The ChaCha family of stream ciphers.
`https://cr.yp.to/chacha.html`, January 2008.

References ii

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.

Joan Daemen and Vincent Rijmen.
AES Proposal: Rijndael.
Submission to the NIST AES Competition, September 1999.

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

References iii


Lov K. Grover.
A Fast Quantum Mechanical Algorithm for Database Search.
In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.


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`.


Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia.
Breaking Symmetric Cryptosystems Using Quantum Period Finding.
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 II*, Lecture Notes in Computer Science, pages 207–237. Springer, 2016.


Morris Dworkin.
Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, May 2005.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.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.

NIST.
Post-Quantum Cryptography – FAQs.
https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs.
Accessed: 2026-04-17.

NIST.
Post-Quantum Cryptography – Security (Evaluation Criteria).
https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria).
Accessed: 2026-04-17.

Yoav Nir and Adam Langley.
ChaCha20 and Poly1305 for IETF Protocols.
Request for Comments (RFC) 8439, June 2018.
https://tools.ietf.org/html/rfc8439.


Daniel R. Simon.
On the Power of Quantum Computation.
SIAM J. Comput., 26(5):1474–1483, 1997.


Christof Zalka.
Grover's quantum searching algorithm is optimal.
Physical Review A, 60(4):2746–2751, October 1999.