PRPs and PRFs
Recap

Simple stream ciphers: can only use key to encrypt one message

Next goal: ciphers where a single key can be used to encrypt many messages

First, let’s introduce block ciphers.
Quick Recap

A block cipher is a pair of efficient algs. \((E, D)\):

- **PT Block** \(n\) bits
- **Key** \(k\) bits
- **CT Block** \(n\) bits

Canonical examples:

1. **AES:** \(n=128\) bits, \(k = 128, 192, 256\) bits
2. **3DES:** \(n= 64\) bits, \(k = 168\) bits (historical)
Block Ciphers Built by Iteration

R(k,m) is called a round function

3DES: n=48, \hspace{1cm} AES128: n=10, \hspace{1cm} AES256: n=14
AES: an iterated Even-Mansour cipher

\[ \pi: \{0,1\}^n \rightarrow \{0,1\}^n \quad \text{invertible function} \]
AES256: 14 rounds of EM

Key expansion: 16 bytes → 240 bytes

Key: 32 bytes

Input: 4

Output: 4
The permutation $\pi$

(1) **ByteSub**: a 1 byte S-box. 256 byte table. (invertible)

(2) **ShiftRows**:

(3) **MixColumns**:
Recall the AES pledge

I promise that I will not implement AES myself in production code, even though it might be fun. This agreement will remain in effect until I learn all about side-channel attacks and countermeasures to the point where I lose all interest in implementing AES myself.
## Performance (no HW acceleration)

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Block/key size</th>
<th>Speed (MB/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChaCha20</td>
<td>- / 256</td>
<td>643</td>
</tr>
<tr>
<td>3DES</td>
<td>64 / 168</td>
<td>30</td>
</tr>
<tr>
<td>AES128</td>
<td>128 / 128</td>
<td>163</td>
</tr>
<tr>
<td>AES256</td>
<td>128 / 256</td>
<td>115</td>
</tr>
</tbody>
</table>
AES-NI: AES hardware instructions

AES instructions (Intel, AMD, ARM, …)

- **aesenc, aesenclast**: do one round of AES
  
  128-bit registers: xmm1=state, xmm2=round key
  
  \[\text{aesenc } \text{xmm1, xmm2} \quad ; \quad \text{puts result in xmm1}\]

- **aesdec, aesdeclast**: one round of AES\(^{-1}\)

- **aeskeygenassist**: performs AES key expansion

**Claim 1**: 20 x speed-up over OpenSSL on same hardware

**Claim 2**: constant time execution
AES-NI: Encrypting one block (AES256)

step 0: \texttt{aeskeygenassist}(256-bit key) \rightarrow \\
\text{round keys in xmm2, xmm3, \ldots, xmm16}

step 1: load plaintext block into \texttt{xmm1} (128-bit block)

step 2: \begin{align*}
\text{xor} & \texttt{xmm1, xmm2} \\
\text{aesenc} & \texttt{xmm1, xmm3} \\
\text{aesenc} & \texttt{xmm1, xmm4} \\
\text{aesenc} & \texttt{xmm1, xmm5} \\
\ldots & \\
\text{aesenclast} & \texttt{xmm1, xmm16}
\end{align*}

15 instructions
AES-NI: parallelism and pipelining

- Intel Skylake (old): 4 cycles for one aesenc
  - fully pipelined: can issue one instruction every cycle

- Intel Icelake (2019): vectorized aesenc (vaesenc)
  - vaesenc: compute aesenc on four blocks in parallel
  - fully pipelined

Implications:

- AES256 encrypt a single block takes **56 cycles** (14 rounds)
- AES256 encrypt 16 blocks on Icelake takes **59 cycles**
AES256 encrypt on Icelake

To encrypt 16 blocks do: \( m_0, \ldots, m_{15} \in \{0,1\}^{128} \)

0: \[
\begin{array}{cccc}
  m_0 & m_1 & m_2 & m_3 \\
\end{array}
\]

(vaesenc -- 512 bit register zmm1)

1: \[
\begin{array}{cccc}
  m_4 & m_5 & m_6 & m_7 \\
\end{array}
\]

(vaesenc)

2: \[
\begin{array}{cccc}
  m_8 & m_9 & m_{10} & m_{11} \\
\end{array}
\]

(vaesenc)

3: \[
\begin{array}{cccc}
  m_{12} & m_{13} & m_{14} & m_{15} \\
\end{array}
\]

(vaesenc)

4: \[
\begin{array}{cccc}
  m_0' & m_1' & m_2' & m_3' \\
\end{array}
\]

(vaesenc)

5: \[
\begin{array}{cccc}
  m_4' & m_5' & m_6' & m_7' \\
\end{array}
\]

(vaesenc)

... finish all 14 rounds after 59 cycles
Abstract view of a block cipher: PRPs and PRFs

Topics:
1. Abstract block ciphers: PRPs and PRFs
2. Security models for encryption
3. Analysis of CBC and counter mode
PRPs and PRFs

• Pseudo Random Function (PRF) defined over (K,X,Y):
  \[ F: K \times X \rightarrow Y \]
  such that exists “efficient” algorithm to evaluate \( F(k,x) \)

• Pseudo Random Permutation (PRP) defined over (K,X):
  \[ E: K \times X \rightarrow X \]
  such that:
  1. Exists “efficient” algorithm to evaluate \( E(k,x) \)
  2. The function \( E( k, \cdot ) \) is one-to-one
  3. Exists “efficient” inversion algorithm \( D(k,x) \)
Running example

• Example PRPs: 3DES, AES, …

  AES128: $K \times X \rightarrow X$  where  $K = X = \{0,1\}^{128}$

  DES: $K \times X \rightarrow X$  where  $X = \{0,1\}^{64}$,  $K = \{0,1\}^{56}$

  3DES: $K \times X \rightarrow X$  where  $X = \{0,1\}^{64}$,  $K = \{0,1\}^{168}$

• Functionally, any PRP is also a PRF.
  – A PRP is a PRF where $X=Y$ and is efficiently invertible
  – A PRP is sometimes called a block cipher
Secure PRFs

- Let \( F: K \times X \to Y \) be a PRF.
  
  \[
  \begin{align*}
    &\text{Funs}[X,Y]: \text{the set of all functions from } X \text{ to } Y \\
    &S_F = \{ F(k, \cdot) \text{ s.t. } k \in K \} \subseteq \text{Funs}[X,Y]
  \end{align*}
  \]

- **Intuition:** a PRF is secure if a random function in Funs[X,Y] is indistinguishable from a random function in \( S_F \).
Secure PRFs

- Let $F: K \times X \rightarrow Y$ be a PRF

\[
\begin{align*}
\text{Funs}[X,Y]: & \quad \text{the set of all functions from } X \text{ to } Y \\
S_F = \{ F(k,\cdot) \text{ s.t. } k \in K \} & \subseteq \text{Funs}[X,Y]
\end{align*}
\]

- **Intuition**: a PRF is **secure** if a random function in $\text{Funs}[X,Y]$ is indistinguishable from a random function in $S_F$

???

???
Secure PRF: definition

• For $b=0,1$ define experiment $\text{EXP}(b)$ as:

  - $b=0$: $k \leftarrow K$, $f \leftarrow F(k, \cdot)$
  - $b=1$: $f \leftarrow \text{Funs}[X, Y]$

  $x_i \in X$
  $f(x_i)$

  $\text{Adv. } \mathcal{A}$

  $b' \in \{0,1\}$

• **Def:** $F$ is a **secure PRF** if for all “efficient” $\mathcal{A}$:

  $$\text{Adv}_{\text{PRF}}[\mathcal{A}, F] = \left| \Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1] \right|$$

  is “negligible.”
An example

Let $K = X = \{0,1\}^n$.

Consider the PRF: $F(k, x) = k \oplus x$ defined over $(K, X, X)$

Let’s show that $F$ is insecure:

Adversary $\mathcal{A}$:

(1) choose arbitrary $x_0 \neq x_1 \in X$

(2) query for $y_0 = f(x_0)$ and $y_1 = f(x_1)$

(3) output `0' if $y_0 \oplus y_1 = x_0 \oplus x_1$, else `1'

$\Pr[\text{EXP}(0) = 0] = 1$

$\Pr[\text{EXP}(1) = 0] = 1/2^n$

$\Rightarrow \text{Adv}_{\text{PRF}}[\mathcal{A}, F] = 1 - (1/2^n)$ (not negligible)
Secure PRP

- For $b = 0, 1$ define experiment $\text{EXP}(b)$ as:

  - **Chal.** $b = 0$: $k \leftarrow K$, $f \leftarrow E(k, \cdot)$
  - $b = 1$: $f \leftarrow \text{Perms}[X]$  

- **Def:** $E$ is a secure PRP if for all "efficient" $\mathcal{A}$:

  $$\text{Adv}_{\text{PRP}}[\mathcal{A}, E] = \left| \Pr[\text{EXP}(0) = 1] - \Pr[\text{EXP}(1) = 1] \right|$$

  is "negligible."
Example secure PRPs

- Example secure PRPs: 3DES, AES, ...

  AES256: \( K \times X \to X \) where \( X = \{0,1\}^{128} \)
  \[
  K = \{0,1\}^{256}
  \]

- AES256 PRP Assumption (example):

  For all \( \mathcal{A} \) s.t. \( \text{time}(\mathcal{A}) < 2^{80} \):
  \[
  \text{Adv}_{\text{PRP}}[\mathcal{A}, \text{AES256}] < 2^{-40}
  \]
The PRP-PRF Switching Lemma

Any secure PRP is also a secure PRF.

Lemma: Let \( E \) be a PRP over \((K, X)\).
Then for any \( q \)-query adversary \( \mathcal{A} \):

\[
\left| \text{Adv}_{\text{PRF}}[\mathcal{A}, E] - \text{Adv}_{\text{PRP}}[\mathcal{A}, E] \right| < \frac{q^2}{2|X|}
\]

\( \Rightarrow \) Suppose \( |X| \) is large so that \( \frac{q^2}{2|X|} \) is “negligible”

Then \( \text{Adv}_{\text{PRP}}[\mathcal{A}, E] \) “negligible” \( \Rightarrow \) \( \text{Adv}_{\text{PRF}}[\mathcal{A}, E] \) “negligible”
Using PRPs and PRFs

• **Goal:** build “secure” encryption from a PRP.

• Security is always defined using two parameters:

1. What “**power**” does adversary have?
   examples:
   - Adv sees only one ciphertext (one-time key)
   - Adv sees many PT/CT pairs (many-time key, CPA)

2. What “**goal**” is adversary trying to achieve?
   examples:
   - Fully decrypt a challenge ciphertext.
   - Learn info about PT from CT (semantic security)
Incorrect use of a PRP

Electronic Code Book (ECB):

PT: \[ \begin{array}{ccc}
      & m_1 & m_2 & \cdots \\
\end{array} \]

CT: \[ \begin{array}{ccc}
      & c_1 & c_2 & \cdots \\
\end{array} \]

Problem:
- if \[ m_1 = m_2 \] then \[ c_1 = c_2 \]
In pictures

An example plaintext

Encrypted with AES in ECB mode

(courtesy B. Preneel)
How to use a block cipher?

Modes of Operation for One-time Use Key

Example application:

- Encrypted email. New key for every message.
Semantic Security for one-time key

- \( \mathbb{E} = (E,D) \) a cipher defined over \((K,M,C)\)
- For \( b=0,1 \) define \( \text{EXP}(b) \) as:

\[
\text{Adv}_{SS}[\mathcal{A},\mathbb{E}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right|
\]

is “negligible.”
Semantic security (cont.)

Sem. Sec. $\Rightarrow$ no “efficient” adversary learns “info” about PT from a **single** CT.

Example: suppose efficient $\mathcal{A}$ can deduce LSB of PT from CT. Then $E = (E,D)$ is not semantically secure.

Then $\text{Adv}_{\text{SS}}[B, E] = 1 \Rightarrow E$ is not sem. sec.
Note: ECB is not Sem. Sec.

ECB is not semantically secure for messages that contain two or more blocks.

\[
\begin{align*}
\text{Chal.} & \quad k \leftarrow K \\
\text{b} & \in \{0, 1\} \\
\text{Two blocks} & \\
\text{m}_0 &= \text{“Hello World”} \\
\text{m}_1 &= \text{“Hello Hello”} \\
(c_1, c_2) &\leftarrow E(k, m_b) \\
\text{Adv. } \mathcal{A} & \\
\text{If } c_1 = c_2 \text{ output 1, else output 0}
\end{align*}
\]

Then \( \text{Adv}_{\text{SS}}[\mathcal{A}, \text{ECB}] = 1 \)
Secure Constructions

Examples of sem. sec. systems:

1. $\text{Adv}_{\text{SS}}[\mathcal{A}, \text{OTP}] = 0$ for all $\mathcal{A}$

2. Deterministic counter mode from a PRF $F$:
   - $E_{\text{DETCTR}}(k,m) =$
     
     $\begin{array}{cccc}
     m[0] & m[1] & \cdots & m[L] \\
     \oplus & & & \\
     F(k,0) & F(k,1) & \cdots & F(k,L) \\
     \hline \\
     c[0] & c[1] & \cdots & c[L]
     \end{array}$
   - Stream cipher built from PRF (e.g. AES)
**Theorem:** For any $L > 0$.

If $F$ is a secure PRF over $(K,X,X)$ then $E_{DETCTR}$ is sem. sec. cipher over $(K,X^L,X^L)$.

In particular, for any adversary $\mathcal{A}$ attacking $E_{DETCTR}$ there exists a PRF adversary $B$ s.t.:

$$\text{Adv}_{SS}[\mathcal{A}, E_{DETCTR}] = 2 \cdot \text{Adv}_{PRF}[B, F]$$

---

$\text{Adv}_{PRF}[B, F]$ is negligible (since $F$ is a secure PRF)

$$\Rightarrow$$ $\text{Adv}_{SS}[\mathcal{A}, E_{DETCTR}]$ must be negligible.
Modes of Operation for Many-time Key

Example applications:
1. File systems: Same AES key used to encrypt many files.
2. IPsec: Same AES key used to encrypt many packets.
Semantic Security for many-time key (CPA security)

Cipher $E = (E,D)$ defined over $(K,M,C)$.
For $b=0,1$ define $\text{EXP}(b)$ as:

- Chal. \quad k \leftarrow K
- Adv. $\mathcal{A}$

for $i=1,\ldots,q$:

$$m_{i,0}, m_{i,1} \in M: \quad |m_{i,0}| = |m_{i,1}|$$

$$c_i \leftarrow E(k, m_{i,b})$$

if adv. wants $c = E(k, m)$ it queries with $m_{j,0} = m_{j,1} = m$

Def: $E$ is sem. sec. under CPA if for all “efficient” $\mathcal{A}$:

$$\operatorname{Adv}_{\text{CPA}}[\mathcal{A}, E] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right|$$

is “negligible.”
Security for many-time key

Fact: stream ciphers are insecure under CPA.

- More generally: if \( E(k, m) \) always produces same ciphertext, then cipher is insecure under CPA.

If secret key is to be used multiple times \( \Rightarrow \) given the same plaintext message twice, the encryption alg. must produce different outputs.
**Nonce-based Encryption**

nonce $n$: a value that changes from msg to msg 
(k,n) pair never used more than once

- **method 1**: encryptor chooses a random nonce, $n \leftarrow \mathcal{N}$
- **method 2**: nonce is a counter (e.g. packet counter) 
  – used when encryptor keeps state from msg to msg 
  – if decryptor has same state, need not send nonce with CT
Construction 1: CBC with random nonce

Cipher block chaining with a random IV \((IV = nonce)\)

\[
\begin{align*}
IV & \quad m[0] \quad m[1] \quad m[2] \quad m[3] \\
\oplus & \quad E(k, \cdot) \quad \oplus \quad E(k, \cdot) \quad \oplus \quad E(k, \cdot) \quad \oplus \quad E(k, \cdot) \\
\downarrow & \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
IV & \quad c[0] \quad c[1] \quad c[2] \quad c[3]
\end{align*}
\]

ciphertext

note: CBC where attacker can predict the IV is not CPA-secure. HW.
CBC: CPA Analysis

CBC Theorem: For any $L > 0$,
If $E$ is a secure PRP over $(K, X)$ then
$E_{CBC}$ is a semi. sec. under CPA over $(K, X^L, X^{L+1})$.

In particular, for a $q$-query adversary $A$ attacking $E_{CBC}$
there exists a PRP adversary $B$ s.t.:

$$\text{Adv}_{\text{CPA}}[A, E_{CBC}] \leq 2 \cdot \text{Adv}_{\text{PRP}}[B, E] + 2 \frac{q^2 L^2}{|X|}$$

Note: CBC is only secure as long as $q^2 \cdot L^2 \ll |X|$
Construction 1’: CBC with unique nonce

Cipher block chaining with unique IV (IV = nonce)

unique IV means: (key, IV) pair is used for only one message

Diagram:
- IV
- m[0] → E(k₁, ·) → c[0]
- m[1] → E(k₁, ·) → c[1]
- m[3] → E(k₁, ·) → c[3]

IV’ = unique IV

included only if unknown to decryptor
A CBC technicality: padding

TLS 1.0: if need n-byte pad, n>0, use: \[ \text{n-1} \text{n-1} \cdots \text{n-1} \]
if no pad needed, add a dummy block

pad is removed during decryption
Construction 2: rand ctr-mode

F: PRF defined over (K,X,Y) where \( X = \{0,1, \ldots, 2^n-1\} \) and \( Y = \{0,1\}^n \)

(e.g., \( n=128 \))

IV - chosen at random for every message

note: parallelizable (unlike CBC)
Why is this CPA secure?

CPA security holds as long as intervals do not intersect

- $q$ msgs, $L$ blocks each $\Rightarrow$ $\Pr[\text{intersection}] \leq 2 \frac{q^2 L}{|X|}$ needs to be negligible
random ctr-mode: CPA analysis

Randomized counter mode: random IV.

Counter-mode Theorem: For any $L>0$,

If $F$ is a secure PRF over $(K,X,X)$ then $E_{\text{CTR}}$ is a sem. sec. under CPA over $(K,X^L,X^{L+1})$.

In particular, for a $q$-query adversary $A$ attacking $E_{\text{CTR}}$ there exists a PRF adversary $B$ s.t.:

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 2 \cdot \text{Adv}_{\text{PRF}}[B, F] + 2 q^2 L / |X|$$

Note: ctr-mode only secure as long as $q^2 \cdot L \ll |X|$.

Better than CBC!
An example

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 2 \cdot \text{Adv}_{\text{PRF}}[B, E] + 2 \frac{q^2 L}{|X|}$$

$q = \# \text{ messages encrypted with } k$, $L = \text{ length of max msg}$

Suppose we want $\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 1/2^{31}$

- Then need: $q^2 L / |X| \leq 1/2^{32}$

- AES: $|X| = 2^{128} \Rightarrow q L^{1/2} < 2^{48}$

So, after $2^{32}$ CTs each of length $2^{32}$, must change key (total of $2^{64}$ AES blocks)
Construction 2’: nonce ctr-mode

To ensure $F(k, x)$ is never used more than once, choose IV as:

- **IV:**
  - nonce: 00000000
  - 96 bits: 00000000
  - 32 bits: 00000000

- **IV+1:**
  - nonce: 00000001

- **IV+2:**
  - nonce: 00000002

**IV** starts at 0 for every msg.
## Comparison: ctr vs. CBC

<table>
<thead>
<tr>
<th></th>
<th>CBC</th>
<th>ctr mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>required primitive</td>
<td>PRP</td>
<td>PRF</td>
</tr>
<tr>
<td>parallel processing</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>security</td>
<td>$q^2 L^2 &lt;&lt;</td>
<td>X</td>
</tr>
<tr>
<td>dummy padding block</td>
<td>Yes*</td>
<td>No</td>
</tr>
<tr>
<td>1 byte msgs (nonce-based)</td>
<td>16x expansion</td>
<td>no expansion</td>
</tr>
</tbody>
</table>

* for CBC, dummy padding block can be avoided using *ciphertext stealing*
Summary

PRPs and PRFs: a useful abstraction of block ciphers.

We examined two security notions:

1. Semantic security against one-time.
2. Semantic security against many-time CPA.

Note: neither mode ensures data integrity.

Stated security results summarized in the following table:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Power</th>
<th>one-time key</th>
<th>Many-time key (CPA)</th>
<th>CPA and CT integrity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sem. Sec.</td>
<td>steam-ciphers</td>
<td>rand CBC</td>
<td>later</td>
<td></td>
</tr>
<tr>
<td></td>
<td>det. ctr-mode</td>
<td>rand ctr-mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Attacks on block ciphers

Goal: distinguish block cipher from a random permutation
- if this can be done efficiently then block cipher is broken

Harder goal:
\[ \text{find key } k \text{ given many } c_i = E(k, m_i) \text{ for random } m_i \]
(1) Linear and differential attacks

[BS’89,M’93]

Given many \((m_i, c_i)\) pairs, can recover key much faster than exhaustive search

**Linear cryptanalysis** (overview): let \(c = \text{DES}(k, m)\)

Suppose for random \(k, m:\)

\[
\Pr \left[ m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j_j] \oplus \cdots \oplus c[j_v] = k[l_1] \oplus \cdots \oplus k[l_u] \right] = \frac{1}{2} + \varepsilon
\]

For some \(\varepsilon\).

For DES, this exists with \(\varepsilon = 1/2^{21} \approx 0.0000000477\) !!
Linear attacks

Pr\left[ m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j] \oplus \cdots \oplus c[j_v] = k[l_1] \oplus \cdots \oplus k[l_u] \right] = \frac{1}{2} + \varepsilon

**Thm:** given $1/\varepsilon^2$ random pairs $(m, c=\text{DES}(k, m))$ then

\[ k[l_1] \oplus \cdots \oplus k[l_u] = \text{MAJ}\left[ m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j] \oplus \cdots \oplus c[j_v] \right] \]

with prob. $\geq 97.7\%$

$\Rightarrow$ with $1/\varepsilon^2$ inp/out pairs can find $k[l_1] \oplus \cdots \oplus k[l_u]$ in time $\approx 1/\varepsilon^2$
Linear attacks

For DES, $\epsilon = 1/2^{21} \Rightarrow$

with $2^{42}$ inp/out pairs can find $k[l_1] \oplus \ldots \oplus k[l_u]$ in time $2^{42}$

Roughly speaking: can find 14 key “bits” this way in time $2^{42}$

Brute force remaining $56-14=42$ bits in time $2^{42}$

Attack time: $\approx 2^{43} (<< 2^{56})$ with $2^{42}$ random inp/out pairs
Lesson

A tiny bit of linearly leads to a $2^{42}$ time attack.

⇒ don’t design ciphers yourself !!
(2) Side channel attacks on software AES

Attacker measures the time to compute AES128(k,m) for many random blocks m.

- Suppose that the 256-byte S table is not in L1 cache at the start of each invocation

  \[ \Rightarrow \text{time to encrypt reveals the order in which S entries are accessed} \]

  \[ \Rightarrow \text{leaks info. that can compromise entire key} \]

Lesson: don’t implement AES yourself!

Mitigation: AES-NI or use vetted software (e.g., BoringSSL)
(3) Quantum attacks

Generic search problem:

Let \( f: X \rightarrow \{0,1\} \) be a function.

Goal: find \( x \in X \) s.t. \( f(x) = 1 \).

Classical computer: best generic algorithm time = \( O(|X|) \)

Quantum computer [Grover ’96]: time = \( O(|X|^{1/2}) \)

(requires a long running quantum computation)
Quantum exhaustive search

Given $m$, $c = E(k,m)$ define

$$f(k) = \begin{cases} 1 & \text{if } E(k,m) = c \\ 0 & \text{otherwise} \end{cases}$$

Grover $\Rightarrow$ quantum computer can find $k$ in time $O(|K|^{1/2})$

AES128: quantum key recovery time $\approx 2^{64}$

Adversary has access to a quantum computer $\Rightarrow$

encrypt data using a cipher with 256-bit keys (AES256)
THE END
Recap

**Secure PRF:** \( F: K \times X \rightarrow Y \) and
\[
\{ f(x) = F(k,x) \text{ for } k \leftarrow K \} \text{ is indist. from random } f \text{ in } \text{Funs}[X,Y]
\]

**Secure PRP:** \( E: K \times X \rightarrow X \), efficiently invertible, and
\[
\{ \pi(x) = E(k,x) \text{ for } k \leftarrow K \} \text{ is indist. from random } \pi \text{ in } \text{Perms}[X]
\]

How to use a secure PRF and a secure PRP for encryption?

- One-time key (semantic security): det. CTR-mode
- Many-time key (CPA security):
  - nonce-based CBC,
  - nonce-based CTR mode