Example: CSS (content scrambling system) for DVD encryption [1996]

- Modern stream ciphers (eSTREAM project: 2004-2009)
  - Core design maps 256-bit key, 64-bit nonce, 64-bit counter onto a 512-bit output
    - \(\Rightarrow\) can be very problematic on weak devices (also may not have good sources of entropy)

-RC4 stream cipher (widely used - SSL/TLS protocol, 802.11b)
  - Initial PRG seed (28 bits)
  - 2048-bit internal state
  - 1-byte per round

-Brute force attack: guess the seed \((\sim 2^{40} \text{ time})\)
  - Can do much better with more clever strategy
    - General idea: if we know a few bytes of output of the stream cipher and the output of the 17-bit LFSR, can subtract to obtain output of 25-bit LFSR
    - Brute force the seed of the 17-bit LFSR: each guess induces a state for the 25-bit LFSR
    - Check if output matches or not
  - Attack now runs in \(\sim 2^{46}\) time

By 1999, full key recovery attack can recover key from DVD in just \(\sim 18\) seconds on 450 MHz processor (totally broken!)

Other examples:
- GSM encryption (RS/1.2 stream ciphers for encrypting GSM cell phone traffic)
  - XOR outputs of 3 LFSRs
  - Tried to keep cipher design private, but eventually reverse engineered and attacks found AG/1

  Swedish documents: NSA can process encrypted Bluetooth E0 stream cipher used a design based on 4 LFSRs in conjunction with a 2-bit finite state machine - also not secure!

<table>
<thead>
<tr>
<th>17-bit LFSR</th>
<th>8 bits</th>
</tr>
</thead>
</table>

\[ x+y+c \mod 256 \Rightarrow 8 \text{ bits} \]

\(c:\) carry bit from previous operation (initially 0)

\(\Rightarrow\) actual CSS encryption has a few differences, but the core attack is unaffected
Recall: the one-time pad is not reusable (i.e., the two-time pad is totally broken)

NEVER REUSE THE KEY TO A STREAM CIPHER!

But wait... we “proved” that a stream cipher was secure, and yet, there is an attack?

Recall security game:

\[
\begin{align*}
\text{adversary} & \quad m_0, m_1 \\
\text{challenger} & \quad k \in \mathbb{K} \\
& \quad C_b \leftarrow \text{Encrypt}(k, m_b)
\end{align*}
\]

Observe: adversary only sees one ciphertext key is only used once

$\Rightarrow$ Security in this model says nothing about multiple messages/ciphertexts

Problem: If we want security with multiple ciphertexts, we need a different or stronger definition (CPA security)
Definition: An encryption scheme \( \Pi_E = (\text{Encrypt}, \text{Decrypt}) \) is secure against chosen-plaintext attacks (CPA-secure) if for all efficient adversaries \( A \):

\[
\text{CPAAdv}[A, \Pi_E] = \left| \Pr[W_b = 1] - \Pr[W_b = 1] \right| = \text{negl},
\]

where \( W_b \) (\( b \in \{0,1\} \)) is the output of the following experiment:

\[
\begin{align*}
\text{adversary:} & \quad \text{choose } m_0, m_1, \text{ with } m_0 \neq m_1, \quad k \in \mathbb{K} \\
\text{challenger:} & \quad \text{sample } b \in \{0,1\} \quad \text{from adversary} \\
& \quad \text{Encrypt}(k, m_b) \quad \text{for } b \in \{0,1\} \\
& \quad \text{output } b \text{ if } E_b = c' \\
& \quad \text{output } b \text{ if } E_b = c
\end{align*}
\]

Adversary's goal is to guess which of \( m_0 \) or \( m_1 \) was encrypted, given access to an encryption oracle (i.e., adversary gets to see encryptions of messages of its choice).

Claim: A stream cipher is not CPA-secure.

Proof: Consider the following adversary:

\[
\begin{align*}
\text{adversary:} & \quad \text{choose } m_0, m_1, \text{ with } m_0 \neq m_1, \quad k \in \mathbb{K} \\
\text{challenger:} & \quad \text{sample } b \in \{0,1\} \quad \text{from adversary} \\
& \quad \text{Encrypt}(k, m_b) \quad \text{for } b \in \{0,1\} \\
& \quad \text{output } b \text{ if } E_b = c' \\
& \quad \text{output } b \text{ if } E_b = c
\end{align*}
\]

Output 0 if \( c = c' \)
Output 1 if \( c \neq c' \)

Observe: Above attack works for any deterministic encryption scheme.

\( \Rightarrow \) CPA-secure encryption must be randomized!

\( \Rightarrow \) To be reusable, cannot be deterministic. Encrypting the same message twice should not reveal that identical messages were encrypted.

To build a CPA-secure encryption scheme, we will use a “block cipher”:

- Block cipher is an invertible keyed function that takes a block of \( n \) input bits and produces a block of \( n \) output bits
- Examples include 3DES (key size 168 bits, block size 64 bits)
- AES (key size 128 bits, block size 128 bits)

Will define block ciphers abstractly first: pseudorandom functions (PRFs) and pseudorandom permutations (PRPs)

\( \Rightarrow \) General idea: PRFs behave like random functions
PRPs behave like random permutations
Definition: A function \( F : K \times X \rightarrow Y \) with key-space \( K \), domain \( X \), and range \( Y \) is a pseudorandom function (PRF) if for all efficient adversaries \( A \), \( |W_0 - W_1| = \text{negl} \), where \( W_0 \) is the probability the adversary outputs 1 in the following experiment:

\[
\begin{array}{cc}
\text{adversary} & \text{challenger} \\
\downarrow & \downarrow \\
x & b \in \{0,1\}^* \\
\end{array}
\]

\[
\begin{array}{cc}
f(x) & b \\
\end{array}
\]

- the space of all possible functions from \( X \rightarrow Y \)
- function \( f \in \text{Fns}[X,Y] \) can be represented by a truth table of size \( |\mathbb{S}|^{|X|} \)  — this is usually exponentially large!

\[
\text{PRFAdv} = |W_0 - W_1| = \Pr[A \text{ outputs 1} \mid b = 0] - \Pr[A \text{ outputs 1} \mid b = 1]
\]

Intuitively: input-output behavior of a PRF is indistinguishable from that of a random function (to any computationally-bounded adversary).

3DES: \( \{0,1\}^{168} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64} \)  \( |K| = 2^{168} \)

\[
|\text{Fns}[X,Y]| = (2^{64})^{(2^{168})}
\]

A space of random functions is exponentially-larger than key-space.

Definition: A function \( F : K \times X \rightarrow X \) is a pseudorandom permutation (PRP) if:

- for all keys \( k \), \( F(k, \cdot) \) is a permutation and moreover, there exists an efficient algorithm to compute \( F^{-1}(k, \cdot) \):
  \[
  \forall k \in K : \forall x \in X : F^{-1}(k,F(k,x)) = x
  \]

- for \( k \in K \), the input-output behavior of \( F(k, \cdot) \) is computationally indistinguishable from \( f(\cdot) \) where \( f \in \text{Perm}[X] \) and \( \text{Perm}[X] \) is the set of all permutations on \( X \) (analogous to PRF security)

Note: a block cipher is another term for PRP (just like stream ciphers are PRGs)