Project Idea:
- Security of automotive info systems

Misc:
- TLS bugs (Apple, not Linux)

Today: Block Ciphers
- Modes of operation: ECB, CTR, CBC, CFB
- IND-CCA security definition
- UFE mode
- Start MACs (if time)
For practical purposes, can treat AES as ideal block cipher:
- For each key, mapping \( \text{Enc}(k, o) \) is a random independent permutation of \( \{0,1\}^{28} \) to itself.

### Modes of Operation:

How to encrypt variable-length messages? (using AES)

- **ECB** = "Electronic code book"
- **CTR** = "Counter mode"
- **CBE" = "Cipher-block chaining" (CBC-MAC)
- **CFB" = "Cipher Feedback"
  - (others...)

**ECB:**

\[
\begin{align*}
M_1 & \quad \cdots \quad M_n \\
E_1 & \quad \cdots \quad E_n
\end{align*}
\]

- Divide data into \( b \)-bit blocks, where \( b \) = input block size.

To handle data that is not a multiple of \( b \) bits in length:
- Append a "1" bit (always)
- Append enough "0" bits to make length a multiple of \( b \) bits.

This gives invertible \((1 + 1)\) "padding" operation.

Pad before encryption; unpad after decryption.

**ECB** preserves many patterns:

- repeated message blocks
- repeated ciphertext blocks

**ECB** really only good for encrypting random data (e.g., keys).
CTR (Counter mode):

Generate a PR (pseudorandom) sequence by encrypting $i, i+1, \ldots$

XOR with message to obtain ciphertext.

\[ \begin{array}{c}
K \\ E \\ X_i \\ M_i \\
\downarrow \\ \downarrow \\ \downarrow \\
K \\ E \\ X_{i+1} \\ M_{i+1} \\
\downarrow \\ \downarrow \\ \downarrow \\
\ldots \\
K \\ E \\ X_{i+2} \\ M_{i+2} \\
\downarrow \\ \downarrow \\ \downarrow \\
C_1 \\ C_2 \\ C_3
\end{array} \]

Initial counter value can be transmitted first:

\[ C, C_1, C_2, \ldots \]

Of course, no counter value should be re-used!
CBC (Cipher-block chaining):

Choose IV ("initialization vector") randomly, then use each $C_i$ as "IV" for $M_i$. Transmit IV with ciphertext:

$$IV, C_1, C_2, \ldots, C_n$$

Description easy, and parallelizable (little error propagation)

Lookup "ciphertext stealing" for cute way of handling messages that are not a multiple of 64 bits in length. This method gives ciphertext length = message length.

Last block $C_n$ is the "CBC-MAC" (CBC Message Authentication code) for message $M$. [A fixed IV is used here.] The MAC is a "cryptographic checksum" (more later...). (If messages have variable length, then key for last block should be different.)

fixed IV is usually all 0's
**CFB (Cipher Feedback mode)**

Similar to CBC mode. Uses random IV transmitted with ciphertext.

If M is not a multiple of b bits in length, cut just transmit shortened ciphertext. (No need for ciphertext stealing.)
Are these modes good ones? What do we want?

If block cipher is indistinguishable from ideal block cipher then mode provides indistinguishability based on chosen ciphertext attack (IND-CCA):

- Define as game with adversary.
- Mode is IND-CCA secure if adversary can win with probability at most $\frac{1}{2} + \epsilon$ for "negligible" $\epsilon$.
- Let $K$ be randomly chosen key.
- Let $E_K$ denote encryption (using mode) with key $K$.
- Let $D_K$ denote decryption.

Phase I ("Find"):

- Adversary given black-box access to $E_K$, $D_K$ (can encrypt/decrypt whatever it likes).
- Adversary outputs two messages $m_0$, $m_1$ of same length, plus state information $s$.

Phase II ("Guess"):

- Examiner secretly picks $d \in \{0,1\}$.
- Examiner computes $y = E_K(m_d)$.
- Adversary given $y$, $s$, access to $E_K$, and access to $D_K$ (except on $y$).
- Adversary computes for a while, then must produce bit $d'$ as its guess for $d$.
- Adversary's advantage is $|P(d'=d) - \frac{1}{2}|$.

Encryption secure against CCA attack if advantage is negligible.
Theorem: Modes ECB, CTR, CBC, CFB are not IND-CCA secure.

Proof: Adversary picks $m_0 = 0^x$, $m_1 = 1^x$ for large $x$.

Then $y = E_k(m_d)$

Let $z = \text{1st half of } y$.

Since $z \neq y$, Adversary allowed in phase II to ask for $D_k(z)$.

This gives first half of $m_d$, revealing $d$.

Adversary always wins. \[ \Box \]

Can one design a IND-CCA scheme?

Given a ciphertext $y$ for a message $m$,

Adversary should not be able to construct a ciphertext $z$ for a related (e.g. truncated) message.

(nonmalleability)
Fact:

To be IND-CCA secure, encryption method must be randomized or stateful!

(else Adv can encrypt m₀ & m₁ during Phase I, and compare to y in Phase II.)

(Randomization used should be unknown to Adv, too.)

(If state is used, encryption method can use pseudo-random values rather than random values, e.g. have another key k' and a counter i, and generate pseudo-random values from k' and i to use instead of the truly random values.)
Here is a sketch of one IND-CCA secure method, (due to Desai, UBE = "Unbalanced Feistel encryption")

\[ M = \text{long message, sequence } M_1, M_2, \ldots, M_n \text{ of } b\text{-bit blocks.} \]

\[ K = (K_1, K_2, K_3) \] Three indep. keys for block ciphers

\[ r \leftarrow \mathcal{R} \{0, 1\}^b \] starting counter value

\[ \text{pad } P = P_1, P_2, \ldots, P_n \] where \( P_i = E_{K_1}(r + i) \leftarrow (\text{CTR mode}) \)

\[ \text{ciphertext } C = C_1, C_2, \ldots, C_n \] where \( C_i = M_i \oplus P_i \)

\[ \text{CBC-MAC: } X_0 = 0^b \]

\[ X_i = E_{K_2}(X_{i-1} \oplus C_i) \quad 1 \leq i \leq n \]

\[ X_n = E_{K_3}(X_{n-1} \oplus C_n) \quad \text{(MAC)} \]

\[ \sigma = r \oplus X_n \] use MAC to mask \( r \) (no message authentication)

Output: \( C_1, C_2, \ldots, C_n, \sigma \)

---

CBC-MAC uses \( K_2 \) mostly, bit \( K_3 \) on last block.
• Encryption with UFE can be done in single pass over data, but decryption requires two passes:
  - first to compute \( \text{mac } X_n \), then to get \( r \)
  - second to decrypt \( C \) to get \( M \)

• Only designed for confidentiality (there is no way provided for receiver to tell if cipher text has been tampered with.) (Need to use MAC on top of all of this, or some "combined mode" providing both confidentiality & integrity.)

• Note "unbalanced Feistel structure".

• Length of ciphertext \((C, o) = |M| + |r|\); expansion only as needed for randomization.
  No need for "cipher text stealing" since we use CTR mode.