Review of Symmetric-Key Encryption

Breno de Medeiros

Department of Computer Science
Florida State University
A **block cipher** is a deterministic, stateless function from a *key space* $\mathcal{K}$ to a *permutation* of bit-strings (blocks) of a given, fixed length (the cipher’s *block size*). Being a permutation, it has an inverse.

For each key, the block cipher induces a permutation on the blockspace.
Encrypting multiple blocks

- It is possible to encrypt multiple blocks of data by simply repeatedly invoking the block cipher to encrypt each block *independently*.
- This is called the *Electronic Codebook (ECB)* mode of encryption.
- ECB reveals patterns about the data. The following images (example from Wikipedia article on modes of encryption), illustrate the effect of encrypting multiple data chunks with ECB:
Stateful Encryption

To safely use a block cipher with the same key to encrypt multiple blocks, the *stateless* block cipher is used as a component of a *stateful* encryption engine. (A finite state automata.)

There is an input buffer, an internal buffer $S$ which initially stores the *initialization vector* (IV), and an output buffer.

At each step, a block of data is read from the input buffer, the value of the internal buffer is updated, and some value is written to the output buffer.

The block cipher implements (part of) the state transition function *and* or the function that computes the output value from the internal buffer state and input.
Cipher Block Chaining (CBC) mode encryption

Counter mode (CTR)

Counter (CTR) mode encryption

Counter (CTR) mode decryption

Security Notions for Symmetric Cryptography

Breno de Medeiros

Department of Computer Science
Florida State University
Symmetric Encryption

Symmetric encryption is one of the core primitives of cryptography. A symmetric encryption scheme $SE$ consists of three algorithms:

- **Key generation** (probabilistic): using a *source of randomness*, outputs a *key* $K$—for simplicity, assume it to be an arbitrary bit-string of fixed length $\tau$, called the *security parameter*.

- **Encryption** (probabilistic or deterministic, stateful or stateless): takes as input a key $K$, a state $s$ (if stateful), a *plaintext* (arbitrary-length bit-string) $M$, and a source of randomness (if probabilistic), and outputs the *ciphertext* $C$ (a bit-string) and if stateful, a new state $s'$.

- **Decryption** (deterministic and stateless): On input a key $K$ and a ciphertext $C$ (presumably encrypted under $K$), outputs a plaintext $M$ (a bit-string) or FAIL.
At a minimum, a symmetric encryption scheme must enable communication between two parties, i.e., the decryption algorithm must be the reverse of the encryption. Therefore, in all cases, if \( C \) is the encryption of message \( M \) under key \( K \), then decrypting \( C \) under the same key recovers \( M \):

**Correctness:** \( \text{Dec}(K, \text{Enc}(K, M)) = M \).

Intuitively, *security* could be defined as “without prior knowledge of the key, an attacker who obtains the ciphertext \( C \) does not learn anything new about the message \( M \) (except its approximate length).” In what follows, we will define more rigorously this concept as a concrete notion in computational complexity.
In order to formalize the security of a block cipher, consider the following scenario. Two parties, $A$ and $B$, wish to use a block cipher to communicate securely. After observing a number of ciphertext blocks being sent from $A$ to $B$, an eavesdropper should not infer anything about the corresponding plaintext values, or about the key used.

Let $\mathcal{F}_b$ the the set of all functions with domain and image equal to the space of blocks of length $b$. The block cipher is an element of $\mathcal{F}_b$, but typically chosen from a much smaller subset $f_b$.

Example: 128-bit AES has 128-bit block size and 128-bit keys. So, in this case, $|f_b| = 128 \cdot 2^{128}$ possible variations of AES), while $|\mathcal{F}_b| = 128 \cdot 2^{128}$, as there are $2^{128} \cdot 2^{128}$ functions with domain and image equal to the space of blocks of length 128, and a similar number of permutations of such blocks.

Security of block ciphers is (formally) the inability of distinguishing between outputs of the block cipher (for a particular key) and outputs of a randomly chosen function (permutation) of blocks.
To define concretely the security of block ciphers and symmetric encryption, we need to capture the notion of an adversary.

An adversary has more resources than simply computing an algorithm. For instance, it may be able to obtain encryptions and/or decryptions of data blocks.

This is modeled as an oracle.

- An encryption oracle $O^K(·)$ accepts as input a data block, and returns its encryption under key $K$.
- A decryption oracle $O^{K^{-1}}(·)$ accepts as input a data block, and returns its decryption under key $K$. 
An adaptive-chosen plaintext attack against indistinguishability is a probabilistic algorithm (Turing machine) with access to an encryption oracle $\mathcal{O}^K$, that works in stages:

1\textsuperscript{st} computational stage: The attacker can perform arbitrary computations and query $\mathcal{O}^K$ for encryptions of blocks of his choice.

Proposal and challenge: The adversary submits two messages (of the same length in blocks) $m_0$ and $m_1$ to a challenger, which selects one of two messages $m_b$, $b \in \{0, 1\}$ at random and returns $Enc_K(m_b)$ (but not the value $b$) to the adversary.

2\textsuperscript{nd} computational stage: Like the 1\textsuperscript{st} stage.

Guess: The adversary outputs a guess bit $b'$ and wins the indistinguishability game if $b' = b$. 

---

An instance of the IND-CPA game

**Adversary**
- p₁, p₂, p₃, ..., pₖ
- c₁, c₂, c₃, ..., cₖ
- Proposal: m₀, m₁
- Challenge: C = Enc(K, mb)
- pₖ₊₁, pₖ₊₂, ..., pₙ
- cₖ₊₁, cₖ₊₂, ..., cₙ

**Challenger**

b'
Can ECB mode be IND-CPA?

Discussion time.
Indistinguishibility under Adaptively-Chosen Ciphertext Attacks (IND-CCA)

An adaptive-chosen plaintext attack against indistinguishability is a probabilistic algorithm with access to an encryption and decryption oracle $O^{K,K^{-1}}$, that works in stages:

1\textsuperscript{st} computational stage: The attacker can perform arbitrary computations and query $O^{K,K^{-1}}$ on blocks of his choice (indicating for each, if encryption or decryption is desired).

proposal and challenge: The adversary submits (same length) messages $m_0$ and $m_1$ to a challenger, which selects one of two messages $m_b$, $b \in \{0, 1\}$ at random and returns $C = Enc_K(m_b)$.

2\textsuperscript{nd} computational stage: Like the 1\textsuperscript{st} stage, but the adversary cannot query for $(\text{decrypt}, c_j)$, where $c_j$ is some block of $C$.

Guess: The adversary outputs a guess bit $b'$ \textit{and wins} if $b' = b$. 
The adversary is not allowed, in the second stage, to query for the decryption of blocks which are part of the challenge $C = \text{Enc}(K, m_b)$. 

---

**IND-CCA(2)**

**An instance of the IND-CCA(2) game**

- Adversary
- Challenger

Proposal: $m_0, m_1$

Challenge: $C = \text{Enc}(K, m_b)$

- $(p_1, \text{decrypt}), (p_2, \text{encrypt}), \ldots, (p_k, ?)$
- $c_1, c_2, c_3, \ldots, c_k$
- $(p_{k+1}, \text{encrypt}), (p_{k+2}, \text{decrypt}), \ldots, p_n$
- $c_{k+1}, c_{k+2}, \ldots, c_n$

Output: $O^{K, K^{-1}}(\cdot)$

Decision: $b'$
Security Definition for Symmetric encryption

Let \(\text{atk}\) stand for either the string “cpa” or “cca.” Let \(\text{Exp}^{\text{ind-\text{atk}}-\text{b}}(k)\) stand for an instance of the IND-\text{atk} game (either IND-CPA or IND-CCA), where \(b\) is the bit used by the challenger, and \(k\) is the security parameter—i.e., the entropy of the keyspace.

The advantage of the adversary (as a probabilistic algorithm), is by definition:

\[
\text{Adv}^{\text{ind-\text{cpa}}}(k) = Pr[\text{Exp}^{\text{ind-\text{cpa}}-1}(k) = 1] - Pr[\text{Exp}^{\text{ind-\text{cpa}}-0}(k) = 1].
\]

We say that a symmetric encryption scheme \(SE\) is secure if, there does not exist a PTM \(\mathcal{A}\), and polynomials \(t(\cdot), \ell(\cdot), p(\cdot)\): exists such that:

\begin{itemize}
  \item takes less than \(t(k)\) steps to run.
  \item makes queries totaling fewer than \(\ell(k)\) bits.
  \item has advantage higher than \(1/p(k)\).
\end{itemize}