Cryptology complementary

✧

Block ciphers (1)

Pierre Karpman

pierre.karpman@univ-grenoble-alpes.fr

https://www-ljk.imag.fr/membres/Pierre.Karpman/tea.html

2018–02–15

Block ciphers: “simple” binary mappings

Block ciphers

A block cipher is a mapping $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}'$ s.t. $\forall k \in \mathcal{K}$, $\mathcal{E}(k, \cdot)$ is invertible

In practice, most of the time:

- Keys $\mathcal{K} = \{0,1\}^\kappa$, with $\kappa \in \{$~~64~~, ~~80~~, ~~96~~, ~~112~~, 128, 192, 256$\}$
- Plaintexts/ciphertexts $\mathcal{M} = \mathcal{M}' = \{0,1\}^n$, with $n \in \{64, 128, 256\}$

Note

Block cipher inputs are *bits*, not vectors; field, ring elements

Block ciphers: for what?

Ultimate goal: symmetric encryption

- plaintext + key $\mapsto$ ciphertext
- ciphertext + key $\mapsto$ plaintext
- ciphertext $\mapsto$???

With *arbitrary* plaintexts $\in \{0,1\}^*$

Block ciphers: do that for plaintexts $\in \{0,1\}^n$

- (Very) small example: 32 randomly shuffled cards = 5-bit block cipher
- Typical block sizes n = "what's easy to implement"

What's a good block cipher

Expected behaviour:

- Given *oracle access* to $\mathcal{E}(k, \cdot)$, with a secret $k \xleftarrow{\$} \mathcal{K}$, it is "hard" to find k
- (Same with oracle access to $\mathcal{E}^{\pm}(k, \cdot) := \{\mathcal{E}(k, \cdot), \mathcal{E}^{-1}(k, \cdot)\}$)
- Given $c = \mathcal{E}(k, m)$, it is "hard" to find m (when k's unknown)
- Given m, it is "hard" to find $c = \mathcal{E}(k, m)$ (idem)

But that's not enough!

We need more

Define $\mathcal{E}_k : x_L || x_R \mapsto x_L || \mathcal{E}'_k(x_R)$ for some $\mathcal{E}'$

- If $\mathcal{E}'$ verifies all props. from the previous slide, then so does $\mathcal{E}$
- But $\mathcal{E}$ is obviously not so nice
- $\Rightarrow$ need a better way to formulate expectations

(S)PRP security

- Let $\mathrm{Perm}(\mathcal{M})$ be the set of the $(\#\mathcal{M})!$ permutations of $\mathcal{M}$
- Ideally, $\forall k$, $\mathcal{E}(k, \cdot) \xleftarrow{\$} \mathrm{Perm}(\mathcal{M})$
- In practice, good enough if $\mathcal{E}$ is a "good" pseudo-random permutation (PRP):
  - An adversary has access to an oracle $\mathfrak{O}$
  - In one world, $\mathfrak{O} \xleftarrow{\$} \mathrm{Perm}(\mathcal{M})$
  - In another, $k \xleftarrow{\$} \mathcal{K}$, $\mathfrak{O} = \mathcal{E}(k, \cdot)$
  - The adversary cannot tell in which world he lives

(S)PRP security: why it makes sense

It's easy to distinguish the two worlds if:

- It's easy to recover the key of $\mathcal{E}(k, \cdot)$ (try and see)
- It's easy to predict what $\mathcal{E}(k, m)$ will be (ditto)
- $\mathcal{E}_k : x_L \| x_R \mapsto x_L \| \mathcal{E}'_k(x_R)$ (random permutations don't to that (often))
- $\mathcal{E}$ is $\mathbb{F}_2$-linear (say), or even "close to"

$\Rightarrow$ Don't have to explicitly define all the "bad cases"

Complexity issues

We still need to define what means “hard” $\Rightarrow$ complexity measures:

- Time (T) (“how much computation”)
- Memory (M) (“how much storage”)
  - Memory type (sequential access, RAM)
- Data (D) (“how many oracle queries”)
  - Query type (to $\mathcal{E}$, to $\mathcal{E}^{-1}$, etc.)
- Success probability (p)

Generic attack examples

Take $\mathcal{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$

- Can guess an unknown key with $T = 2^\kappa$, $M = O(1)$, $D = O(1)$, $p = 1$
- Can guess an unknown key with $T = 1$, $M = O(1)$, $D = 0$, $p = 2^{-\kappa}$
- Given $\mathcal{E}(k, m)$, can guess m with $T = 1$; $M = O(1)$, $D = 0$, $p = 2^{-\kappa}$
- Given $\mathcal{E}(k, m)$, can guess m with $T = 1$; $M = O(1)$, $D = 0$, $p = 2^{-n}$
- Given $\mathcal{E}(k, m)$, can guess m with $T = 2^\kappa$; $M = O(1)$, $D = O(1)$, $p = 1$

A "single" measure

Define *advantage* functions associated w/ the security properties. For instance:

$\mathbf{Adv}^{\mathrm{PRP}}$

$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{PRP}}(q,t) =$

$$\max_{A_{q,t}} \left| \Pr[A_{q,t}^{\mathfrak{O}}() = 1 : \mathfrak{O} \xleftarrow{\$} \mathrm{Perm}(\mathcal{M})] - \Pr[A_{q,t}^{\mathfrak{O}}() = 1 : \mathfrak{O} = \mathcal{E}(k,\cdot), k \xleftarrow{\$} \mathcal{K}] \right|$$

$A_{q,t}^{\mathfrak{O}}$: An algorithm running in time $\leq t$, making $\leq q$ queries to $\mathfrak{O}$

“Good PRPs”

There is no definition of what a good PRP $\mathcal{E}$ is, but one can expect that:

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{PRP}}(q, t) \approx t/2^{\kappa}$$

(As long as $q \geq \mathrm{O}(1)$)