

Introduction to cryptology (GBIN8U16)



Finite fields, block ciphers

Pierre Karpman

`pierre.karpman@univ-grenoble-alpes.fr`

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

2019-01-30

\mathbb{F}_2 primer

Symmetric cryptography

BC: First definitions

Symmetric encryption schemes

Bits as field elements

- ▶ Digital processing of information \leadsto dealing with bits
- ▶ Error-correcting codes, crypto \leadsto need analysis \leadsto maths
- ▶ \Rightarrow bits (no structure) \mapsto field elements (math object)

- ▶ “Natural” match: $\{0, 1\} \cong \mathbb{F}_2 \equiv \mathbb{Z}/2\mathbb{Z} \equiv$ “(natural) integers modulo 2”
- ▶ \mathbb{F}_2 : two elements (0, 1), two operations (+, \times)

What's \mathbb{F}_2 like?

- ▶ Addition \equiv exclusive or (XOR (\oplus))
- ▶ Multiplication \equiv logical and (\wedge)
- ▶ \Rightarrow “Boolean” arithmetic

- ▶ Fact: any Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed using only \oplus and \wedge
- ▶ Fact 2: ditto, $g : \{0, 1\}^n \rightarrow \{0, 1\}^m$
- ▶ Fact 3: ditto, using NAND ($\neg \circ \wedge$)

One bit is nice, but...

- ▶ We rather need bit strings $\{0, 1\}^n$ than single bits
- ▶ Now two “natural” matches:
 - ▶ \mathbb{F}_2^n (vectors over \mathbb{F}_2)
 - ▶ Can add two vectors
 - ▶ Cannot multiply “internally” (but there’s a dot/scalar product)
 - ▶ $\mathbb{Z}/2^n\mathbb{Z}$ (natural integers modulo 2^n)
 - ▶ Can add, multiply
 - ▶ Not all elements are invertible (e.g. 2) \Rightarrow only a ring

Exercise: How do you implement operations in \mathbb{F}_2^{64} , $\mathbb{Z}/2^{64}\mathbb{Z}$ in C?

A third way

- ▶ Also possible: \mathbb{F}_{2^n} : an *extension* field
 - ▶ Addition “like in \mathbb{F}_2 ”
 - ▶ Plus an internal multiplication
 - ▶ All elements (except zero) are invertible
- ▶ Not for today!

Why are these useful?

- ▶ Allows to perform operations on inputs
 - ▶ E.g. adding two messages together
- ▶ Vector spaces \Rightarrow linear algebra (matrices)
 - ▶ Powerful tools to solve “easy” problems
 - ▶ (Intuition: crypto shouldn't be linear)
- ▶ Fields \Rightarrow polynomials
 - ▶ With one or more variable
 - ▶ \Rightarrow Can compute differentials

\mathbb{F}_2 primer

Symmetric cryptography

BC: First definitions

Symmetric encryption schemes

Context for the next few weeks*

- ▶ Two parties A , B
- ▶ Who share a *secret key* k
- ▶ And wish to communicate securely (e.g. need for authenticity and/or confidentiality)

Remarks:

- ▶ The secret key is assumed to be unknown by the adversaries (but one may “attack” to find it)
- ▶ We are not concerned (yet) with how A and B manage to share k

* Except for the *hash functions* part

\mathbb{F}_2 primer

Symmetric cryptography

BC: First definitions

Symmetric encryption schemes

Block ciphers as a figure

↷ on the board

A main alternative: stream ciphers, still as a figure

~> still on the board

Block ciphers: “simple” binary mappings

Block cipher

A block cipher is a mapping $\mathcal{E} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}'$ 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 $n \in \{64, 128, 256\}$

\Rightarrow BCs are *families of permutations* over binary domains

Block ciphers: for what?

Ultimate goal: symmetric encryption (and more!)

- ▶ 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?

One that's:

- ▶ “Efficient”
 - ▶ Fast (e.g. a few *cycles per byte* on modern high-end CPUs)
 - ▶ \wedge/\vee Compact (small code, circuit size)
 - ▶ \wedge/\vee Easy to implement “securely” (e.g. to prevent side-channel attacks)
 - ▶ Etc.
- ▶ “Secure”
 - ▶ Large security parameters (key, block size)
 - ▶ \wedge No (known) dedicated attacks.

What's a secure 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

⇒ need a better way to formulate expectations

Ideal block cipher

Let $\text{Perm}(\mathcal{M})$ be the set of the $(\#\mathcal{M})!$ permutations of \mathcal{M} ; an *ideal block cipher* $\mathcal{E} : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{M}$ is s.t. $\forall k \in \mathcal{K}$, $\mathcal{E}(k, \cdot) \stackrel{\$}{\leftarrow} \text{Perm}(\mathcal{M})$

- ▶ “Maximally random”
 - ▶ All keys yield truly independent permutations
 - ▶ Quite costly to implement
 - ▶ Say $\mathcal{M} = \{0, 1\}^{32} \rightsquigarrow 2^{32}! < (2^{32})^{2^{32}}$ permutations
 - ▶ So about $32 \times 2^{32} = 2^{37}$ bits to describe one (\rightsquigarrow key size)
- \Rightarrow Not very practical

(S)PRP security

Good enough if \mathcal{E} is a “good” pseudo-random permutation (PRP):

- ▶ An adversary has access to an oracle \mathbb{O}
- ▶ In one world, $\mathbb{O} \stackrel{\$}{\leftarrow} \text{Perm}(\mathcal{M})$
- ▶ In another, $k \stackrel{\$}{\leftarrow} \mathcal{K}$, $\mathbb{O} = \mathcal{E}(k, \cdot)$
- ▶ The adversary cannot tell in which world he lives
- ▶ (“Strong/Super” variant: give oracle access to \mathbb{O}^{\pm})

\Rightarrow *Stronger* requirement than key recovery (is implied by it, converse is not true)

(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 usually don't do that)
- ▶ \mathcal{E} is \mathbb{F}_2 -linear (say), or even “close to”
- ▶ Etc.

⇒ Don't have to explicitly define all the “bad cases”

Plus:

- ▶ Can't do better than a random permutation anyways
- ▶ If it looks like one, either it's fine, or BCs are useless

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 (cheap tape), RAM (costly))
- ▶ Data (D) (“how many oracle queries”)
 - ▶ Query type (to \mathcal{E} , to \mathcal{E}^{-1} , *adaptive* or not, etc.)
- ▶ Success probability (p)

Generic attack examples

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

- ▶ 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 $T = 2^\kappa$; $M = O(1)$, $D = O(1)$, $p = 1$

We have “small” secrets \Rightarrow attacks always possible =
computational security

A “single” measure

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

Adv^{PRP}

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

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

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

“Good PRPs”

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

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

(As long as $q \geq D = O(1)$)

- ▶ Matched by a generic attack (i.e. key guessing)
- ▶ Equality if \mathcal{E} is ideal
- ▶ Anything that's (sensibly) better is a *dedicated* attack

Parameters choice

Even an ideal block cipher is useless if its key is too small

- ▶ If $\kappa = 32$, $t = 2^\kappa = 2^{32}$ is small
- ▶ But when do you know κ 's large enough?
- ▶ Look at the time/energy/infrastructure to count up to 2^κ

Some examples

- ▶ $\approx 40 \rightsquigarrow$ breakable w/ a small Raspberry Pi cluster
- ▶ $\approx 60 \rightsquigarrow$ breakable w/ a large CPU/GPU cluster
 - ▶ Already done (equivalently) several times in the academia:
 - ▶ Ex. RSA-768 (Kleinjung et al., 2010), 2000 core-years ($\equiv 2^{67}$ bit operations)
 - ▶ Ex. DL-768 (Kleinjung et al., 2016), 5300 core-years
 - ▶ Ex. SHA-1 collision (Stevens et al., and me!, 2017), 6500 core-years + 100 GPU-year ($\equiv 2^{63}$ hash computations)
- ▶ $\approx 80 \rightsquigarrow$ breakable w/ an ASIC cluster (cf. Bitcoin mining)

Parameters choice (cont.)

Two caveats:

1 Careful about multiuser security

- ▶ If a single user changes keys *a lot* and breaking one is enough
- ▶ If targeting one random user among many
- ▶ A mix of the two (best!)
- ▶ \leadsto have to account for that

2 Should we care about quantum computers??

- ▶ Would gain a $\sqrt{\cdot}$ factor
- ▶ “128-bit classical” \Rightarrow “64-bit quantum”
- ▶ (But a direct comparison is not so meaningful, actually)

In case of doubt, 256 bits?

Parameters choice (cont.)

What about block size?

- ▶ Security not (directly) related to computational power
- ▶ Dictated by the volume encrypted with a single key (cf. next)

In the end, it's always a cost/security tradeoff

(If you need a conventional BC with ridiculously large params, SHACAL-2, w/ $n = 256$, $\kappa = 512$ is a good choice!)



\mathbb{F}_2 primer

Symmetric cryptography

BC: First definitions

Symmetric encryption schemes

Block ciphers are not enough

What block ciphers do:

- ▶ One-to-one encryption of fixed-size messages

What do we want:

- ▶ One-to-many encryption of variable-size messages
- ▶ Why?
 - ▶ Variable-size → kind of obvious?
 - ▶ One-to-many → necessary for *semantic security* → cannot tell if two ciphertexts are of the same message or not

Enter modes of operation

- ▶ A *mode of operation* transforms a block cipher into a *symmetric encryption scheme*
- ▶ $\approx \mathcal{E} \rightsquigarrow \text{Enc} : \{0, 1\}^\kappa \times \{0, 1\}^r \times \{0, 1\}^* \rightarrow \{0, 1\}^*$
- ▶ For all $k \in \{0, 1\}^\kappa$, $r \in \{0, 1\}^r$, $\text{Enc}(k, r, \cdot)$ is invertible
- ▶ $\{0, 1\}^r$, $r \geq 0$ is used to make encryption non-deterministic
- ▶ A mode is “good” if it gives “good encryption schemes” when used with “good BCs”
- ▶ So what’s a good encryption scheme?

IND-CPA for Symmetric encryption

IND-CPA for Enc: An adversary cannot distinguish $\text{Enc}(k, m_0)$ from $\text{Enc}(k, m_1)$ for an unknown key k and equal-length messages m_0, m_1 when given oracle access to an $\text{Enc}(k, \cdot)$ oracle:

- 1 The Challenger chooses a key $k \xleftarrow{\$} \{0, 1\}^\kappa$
- 2 The Adversary may repeatedly submit queries x_i to the Challenger
- 3 The Challenger answers a query with $\text{Enc}(k, r_i, x_i)$
- 4 The Adversary now submits m_0, m_1 of equal length
- 5 The Challenger draws $b \xleftarrow{\$} \{0, 1\}$, answers with $\text{Enc}(k, r', m_b)$
- 6 The Adversary tries to guess b
 - ▶ The choice of r_i, r' is defined by the mode (made explicit here, may be omitted)

- ▶ A random adversary succeeds w/ prob. $1/2 \rightarrow$ the correct success measure is the *advantage* over this
 - ▶ Advantage (one possible definition): $|\Pr[\text{Adversary answers } 1 : b = 0] - \Pr[\text{Adversary answers } 1 : b = 1]|$
 - ▶ (Same as for PRP security)
- ▶ An adversary may always succeed w/ advantage 1 given enough resources
 - ▶ Find the key spending time $t \leq 2^k$ and a few oracle queries
- ▶ What matters is the “best possible” advantage in function of the attack complexity

First (non-) mode example: ECB

- ▶ ECB: just concatenate independent calls to \mathcal{E}

Electronic Code Book mode

$m_0 || m_1 || \dots \mapsto \mathcal{E}(k, m_0) || \mathcal{E}(k, m_1) || \dots$

- ▶ No security
 - ▶ Exercise: give a simple attack on ECB for the IND-CPA security notion w/ advantage 1, low complexity

Second (actual) mode example: CBC

- ▶ Cipher Block Chaining: Chain blocks together (duh)

Cipher Block Chaining mode

$$r \times m_0 \| m_1 \| \dots \mapsto c_0 := \mathcal{E}(k, m_0 \oplus r) \| c_1 := \mathcal{E}(k, m_1 \oplus c_0) \| \dots$$

- ▶ Output block i (ciphertext) added (XORed) w/ input block $i + 1$ (plaintext)
- ▶ For first (m_0) block: use random IV r
- ▶ Okay security in theory \rightsquigarrow okay security in practice *if used properly*

CBC has bad IND-CPA security if the IVs are not random

- ▶ Consider an IND-CPA adversary who asks an oracle query $\text{CBC-ENC}(m)$, gets $r, c = \mathcal{E}(k, m \oplus r)$ (where \mathcal{E} is the cipher used in CBC-ENC)
- ▶ Assume the adversary knows that for the next IV r' , $\Pr[r' = x]$ is large
- ▶ Sends two challenges $m_0 = m \oplus r \oplus x$, $m_1 = m_0 \oplus 1$
- ▶ Gets $c_b = \text{CBC-ENC}(m_b)$, $b \stackrel{\$}{\leftarrow} \{0, 1\}$
- ▶ If $c_b = c$, guess $b = 0$, else $b = 1$

Generic CBC collision attack

Even with random IVs, CBC has some drawbacks

An observation:

- ▶ In CBC, inputs to \mathcal{E} are of the form $x \oplus y$ where x is a message block and y an IV or a ciphertext block
- ▶ If $x \oplus y = x' \oplus y'$, then $\mathcal{E}(k, x \oplus y) = \mathcal{E}(k, x' \oplus y')$

A consequence:

- ▶ If $c_i = \mathcal{E}(k, m_i \oplus c_{i-1}) = c'_j = \mathcal{E}(k, m'_j \oplus c'_{j-1})$, then $c_{i-1} \oplus c'_{j-1} = m_i \oplus m'_j$
- ▶ \leadsto knowing identical ciphertext blocks reveals information about the message blocks
- ▶ \Rightarrow breaks IND-CPA security
- ▶ Regardless of the security of \mathcal{E} !

CBC collisions: how likely?

How soon does a collision happen?

- ▶ Assumption: the distribution of the $(x \oplus y)$ is \approx uniform
 - ▶ If y is an IV it has to be (close to) uniformly random, otherwise we have an attack (two slides ago)
 - ▶ If $y = \mathcal{E}(k, z)$ is a ciphertext block, ditto for y knowing z , otherwise we have an attack on \mathcal{E}
- ▶ \Rightarrow A collision occurs w.h.p. after $\sqrt{\#\{0, 1\}^n} = 2^{n/2}$ blocks are observed (with identical key k) \leftarrow *The birthday bound*
- ▶ (Slightly more precisely, w/ prob. $\approx q^2/2^n, q \leq 2^{n/2}$ after q blocks)

Some CBC recap

A decent mode, but

- ▶ Must use random IVs
- ▶ Must change key *much* before encrypting $2^{n/2}$ blocks when using an n -bit block cipher
- ▶ And this *regardless of the key size κ*
- ▶ This is a common restriction for modes of operation (cf. next slide)

Another classical mode: CTR

Counter mode

$$m_0 \| m_1 \| \dots \mapsto \mathcal{E}(k, s++) \oplus m_0 \| \mathcal{E}(k, s++) \oplus m_1 \| \dots$$

- ▶ This uses a global state s for the *counter*, with C-like semantics for $s++$
- ▶ Encrypts a public counter \rightsquigarrow pseudo-random keystream \rightsquigarrow (perfect) one-time-pad approximation (i.e. a *stream cipher*)
- ▶ Like CBC, must change key *much* before encrypting $2^{n/2}$ blocks when using an n -bit block cipher