Crypto Engineering '21

✦

Block ciphers

Pierre Karpman
pierre.karpman@univ-grenoble-alpes.fr
https://membres-ljk.imag.fr/Pierre.Karpman/tea.html

2021–09–24

The “symmetric” part of this course (with me)

- 6 CMs; 2 TDs; $2*(2*2) = 8$ TPs
- About symmetric encryption, authentication, hashing
- Goal 1: understanding the models $\rightsquigarrow$ What can we/do we try to achieve?
- Goal 2: looking a bit at some design(s): the why and hows
- Goal 3: getting a few ideas of what can go terribly wrong :(

The practical part of the “asymmetric” part of this course (with me)

- 1*(2*2) = 4 TPs
- Kangaroos for memory-efficient discrete logarithms computation

~Today

BC: First definitions

Symmetric encryption schemes

BC: Evolutions

But first...

- Cryptography: we want to hide stuff (e.g., messages to be sent over an insecure channel)
- Symmetric: we only do that assuming a preexisting shared secret

- A major question: when is the hiding "good enough"?
  - "HELLO" ↦ "HULLO": not great
  - "HELLO" ↦ "ZNPQE": maybe better
  - "HELLO" ↦ "ZNPQE"; "HELLO" ↦ "ZNPQE"; "HELLO" ↦ "ZNPQE"...: (Okay, those same 5 letters at the start of your messages probably always mean "hello")

The problem with deterministic encryption

Figure: XKCD #257

Note that $257 = 2^{2^3} + 1$ is a "Fermat" prime

So...

- Encryption MUST be non-deterministic
- Also (a bit harder to see): messages MUST (pretty much always) be authenticated to prevent tampering if the adversary is *active* (even if only "confidentiality" is a concern)

Now our main concerns:

- How do we formalise what we want to achieve?
- How do we actually build schemes that work?

BC: First definitions

Symmetric encryption schemes

BC: Evolutions

Block ciphers: for what?

Ultimate goal: symmetric encryption (and more!)

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

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

Block ciphers: do that *one-to-one* (for a fixed key) for plaintexts $\in \{0,1\}^n$

- (Very) small example: 32 randomly shuffled cards = 5-bit block cipher
- Typical block sizes = "what's easy to implement"
- Mostly useless in isolation (e.g. they're deterministic) but very useful when plugged into suitable higher-level schemes

Block ciphers as a figure

$\rightsquigarrow$ on the board

Block ciphers: “simple” binary mappings

Block cipher

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

⇒ BCs are *families of permutations* over binary domains

- Exception: *Format Preserving Encryption* (FPE)

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"
  - ???

What's a secure block cipher?

What do you think?

What's a secure block cipher?

Expected behaviour:

- Given *oracle access* to $\mathcal{E}(k, \cdot)$, with a secret $k \twoheadleftarrow \mathcal{K}$, 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

The plan

- Define security relatively to the expected behaviour of an ideal cipher *of the same size*, *given some model for the adversary*
- Choose key, block sizes s.t. generic attacks are out of reach

Ideal block ciphers

Ideal block cipher

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

- “Maximally random”
- All keys yield truly random and independent permutations
- Quite costly to implement
  - Say $\mathcal{M} = \{0,1\}^{32} \rightsquigarrow (2^{32})^{2^{31}} < 2^{32}! < (2^{32})^{2^{32}}$ permutations
  - So about $32 \times 2^{32} = 2^{37}$ *bits* to describe one ($\leftsquigarrow$ key size)

$\Rightarrow$ Not very practical

IBC (cont.)

Why is an ideal block cipher ideal?

- The idea: for all fixed k the full knowledge of $\mathcal{E}(\mathcal{K}\backslash k, \mathcal{M})$ and $\mathcal{E}(k, \mathcal{S})$ gives *no information* on $\mathcal{E}(k, \overline{\mathcal{S}})$ except that it is disjoint from $\overline{\mathcal{E}(k, \mathcal{S})}$

(S)PRP security

Most of the time, good enough if $\mathcal{E}$ is a “good” *pseudo-random permutation* (PRP):

- An adversary has access to an oracle $\mathbb{O}$
- In one world, $\mathbb{O} \twoheadleftarrow \mathrm{Perm}(\mathcal{M})$
- In another, $k \twoheadleftarrow \mathcal{K}$, $\mathbb{O} = \mathcal{E}(k, \cdot)$
- It is “hard” for the adversary to 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.

$\Rightarrow$ 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 ($\leftarrow$ "true" most of the time but not always)

(S)PRP: it's not everything

- Sometimes a PRP is not enough and one needs a stronger/different model such as the *ideal block cipher* model
- For instance when the adversary has access to the key ($\rightsquigarrow$ considering a uniform choice doesn't make sense anymore)
- Example: when using block ciphers to build compression functions (cf. the hash function lecture)

Complexity issues

We still need to define what means “hard” ⇒ cost measures:

- Time (T) (“how much computation”)
    - Sequential? Parallelizable?
- 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 $\mathcal{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$

- Can find an unknown key with $T = 2^\kappa$, $M = \mathrm{O}(\kappa)$, $D = \mathrm{O}(\kappa)$, $p = 1$
- Can find an unknown key with $T = 1$, $M = 0$, $D = 0$, $p = 2^{-\kappa}$
- In general, can find an unknown key with $T = t \leq 2^\kappa$, $M = \mathrm{O}(\kappa)$, $D = \mathrm{O}(\kappa)$, $p = t/2^\kappa$

We have "small" secrets ⇒ attacks always possible = *computational security*

A “single” measure

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

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

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

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

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

(OBTW) The same, for families of functions

The PRP definition easily adapts to (not necessarily invertible) functions $\mathcal{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}'$:

Adv$^{\mathrm{PRF}}$

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

$$\max_{A_{q,t}} \left| \Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} \twoheadleftarrow \mathsf{Func}(\mathcal{M}, \mathcal{M}')] \right.$$
$$\left. - \Pr[A_{q,t}^{\mathbb{O}}() = 1 : \mathbb{O} = \mathcal{F}(k, \cdot), k \twoheadleftarrow \mathcal{K}] \right|$$

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

“Good PRPs”

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

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

(As long as $q \geq D \approx \lceil \kappa/n \rceil$)

- 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 a good PRP is useless if its keyspace 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.)

What about 128?

Objective: run a function 2^{128} times within 34 years ($\approx 2^{30}$ seconds), assuming:

- Hardware at 2^{50} iterations/s (that's pretty good)
- Trivially parallelizable
- 1000 W per device, no overhead (that's pretty good)

$\Rightarrow$

- $2^{128-50-30} \approx 2^{48}$ machines needed
- $\approx$ 280 000 000 GW 'round the clock
    - $\approx$ 170 000 000 EPR nuclear power plants

Looks hard enough

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!)
    - $\rightsquigarrow$ 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 the *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!)

BC: First definitions

Symmetric encryption schemes

BC: Evolutions

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*
- $\approx \mathcal{E} \rightsquigarrow \mathsf{Enc} : \{0,1\}^\kappa \times \{0,1\}^r \times \{0,1\}^* \to \{0,1\}^*$
- For all $k \in \{0,1\}^\kappa$, $r \in \{0,1\}^r$, $\mathsf{Enc}(k, r, \cdot)$ 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 $\mathsf{Enc}(k, m_0)$ from $\mathsf{Enc}(k, m_1)$ for an unknown key k and equal-length messages m_0, m_1 when given oracle access to an $\mathsf{Enc}(k, \cdot)$ oracle:

1. The Challenger chooses a key $k \twoheadleftarrow \{0,1\}^\kappa$
2. The Adversary may repeatedly submit queries x_i to the Challenger
3. The Challenger answers a query with $\mathsf{Enc}(k, r_i, x_i)$
4. The Adversary now submits m_0, m_1 of equal length
5. The Challenger draws $b \twoheadleftarrow \{0,1\}$, answers with $\mathsf{Enc}(k, r', m_b)$
6. The Adversary tries to guess b

- The choice of r_i, r' is defined by the mode (made explicit here, may be omitted)

IND-CPA comments

- A random adversary succeeds w/ prob. $1/2 \rightarrow$ the correct success measure is the *advantage* over this
  - (Same as for e.g. PRP security)
- An adversary may always succeed w/ advantage 1 given enough ressources $\rightsquigarrow$ only computational security (again)
  - Find the key spending time $t \leq 2^\kappa$ 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

$m_1 || m_2 || \ldots \mapsto \mathcal{E}(k, m_1) || \mathcal{E}(k, m_2) || \ldots$

- 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

$r \times m_1||m_2||\ldots \mapsto$
$c_0 := r||c_1 := \mathcal{E}(k, m_1 \oplus c_0)||c2 := \mathcal{E}(k, m_2 \oplus c_1)||\ldots$

- Output block i (ciphtertext) added (XORed) w/ input block $i+1$ (plaintext)
- For first (m_1) block: use random IV r
- Okay security in theory $\leadsto$ okay security in practice *if used properly*

CBC IVs

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

- Consider an IND-CPA adversary who asks an oracle query 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 $c_b = \text{CBC-ENC}(m_b)$, $b \twoheadleftarrow \{0, 1\}$
- If $c_b = c$, guess $b = 0$, else $b = 1$

Generic CBC collision attack

Even with random IVs, CBC can be attacked

An observation:

- For a fixed k, $\mathcal{E}(k,\cdot)$ is a permutation so $\mathcal{E}(k,x) = \mathcal{E}(k,y) \Leftrightarrow x = y$
- 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
- So $\mathcal{E}(k, x \oplus y) = \mathcal{E}(k, x' \oplus y') \Leftrightarrow x \oplus y = x' \oplus y'$

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 $m_i \oplus c_{i-1} = m'_j \oplus c'_{j-1}$, and then $c_{i-1} \oplus c'_{j-1} = m_i \oplus m'_j$
- $\rightsquigarrow$ knowing identical ciphertext blocks reveals information about the message blocks
- $\Rightarrow$ breaks IND-CPA security
- Regardless of the security of $\mathcal{E}$ (i.e. even if it is ideal)!

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) ← *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 uniformly 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
- Only “birthday bound” security: this is a common restriction for modes of operation (cf. next slide)

Another classical mode: CTR

Counter mode

$m_1||m_2||\ldots \mapsto \mathcal{E}(k, \mathtt{s}\texttt{++}) \oplus m_1||\mathcal{E}(k, \mathtt{s}\texttt{++}) \oplus m_2||\ldots$

- This uses a global state $\mathtt{s}$ for the *counter*, with C-like semantics for $\mathtt{s}\texttt{++}$
- 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

Security reduction

- For good modes such as CBC, CTR, one can prove statements of the form: “if [the mode] is instantiated with a ‘good PRP’, then this gives a ‘good IND-CPA encryption scheme’ ”
- This is an example of *security reduction* (here of the encryption scheme to the block cipher)
- Quite common & useful in crypto $\rightsquigarrow$ modular designs are nice

Security reduction: very quick CTR illustration

The main steps to prove the IND-CPA security of the CTR mode are:

- If the keystream is generated from an ideal function (in counter mode), the advantage is zero
- So the IND-CPA advantage is $\leq$ the *PRF* advantage of $\mathcal{E}$
- One then uses the "PRP-PRF switching lemma" which states that $\mathbf{Adv}_{\mathcal{E}}^{\mathrm{PRF}}(q, t) \lessapprox \mathbf{Adv}_{\mathcal{E}}^{\mathrm{PRP}}(q, t) + q^2/2^n$
  - The idea: the only way to distinguish an ideal permutation from an ideal function is with collisions, and the PRP advantage of $\mathcal{E}$ tells here how close it is from an ideal permutation

BC: First definitions

Symmetric encryption schemes

BC: Evolutions

Block cipher evolutions

Block ciphers are very versatile, $\rightsquigarrow$

- Symmetric encryption
- Authentication
- Hashing
- (More exotic constructions)

But not the only candidate primitives for the above

Two possible variations:

- Add one parameter (*tweakable* block ciphers)
- Remove one parameter (*permutations*)

Tweakable block ciphers

Tweakable block cipher

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

The *tweak* t:

- Acts like a key in how it parameterizes a permutation
- Is *public* (known to any adversary)
- Could even be chosen by anyone

Why TBCs?

Tweakable block ciphers are nice:

- Simplify the design/proofs of higher-level constructions: they're an expressive abstraction
- Typically useful in authenticated-encryption modes (e.g. ΘCB)
- Help a lot in getting beyond-birthday-bound (BBB) security

An intuition of usefulness:

- Never reuse a tweak ⇒ always use independent permutations
- Becomes quite harder to attack/distinguish

TBC constructions

Tweakable block ciphers may be built either:

- As high-level constructions, typically from a regular BC
  - Example: $\widetilde{\mathcal{E}}(k, t, \cdot) = \mathcal{E}(k \oplus t, \cdot)$ (adequate if $\mathcal{E}$ is secure against XOR related-key attacks)
- As dedicated designs (like a regular BC)
  - Example: KIASU-BC

Permutations

Permutation

A permutation is an invertible mapping $\mathcal{P} : \mathcal{M} \to \mathcal{M}$

- No key anymore!
  - One consequence: no notion similar to PRP to formalize sec.
- Easy to build as $\mathcal{E}(0, \cdot)$

Rationale:

- In BCs, it may be wasteful to process the key and plaintext separately
- Inverting a permutation is often not necessary in constructions; usages like $\mathcal{P}(k \| m)$ are okay

Permutation uses

Hash functions:

- SHA-3 (Keccak)
- JH
- Grøstl
- Etc.

Authenticated encryption:

- River/Lake/Sea/Ocean/Lunar Keyak
- Ascon
- Etc.