Symmetric Encryption: PRPs, CBC Mode, AES, and DES

Notes by Henry Corrigan-Gibbs

MIT - 6.5610
Lecture 4 (February 12, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Logistics

- Due Friday: Pset 1
- Due Friday: Post a project idea on Piazza

Outline

- Pseudorandom permutations (PRPs)
- CBC Mode
- AES
- DES

Pseudorandom permutations and block ciphers

Very often in practice, we build encryption schemes using pseudorandom functions, as we saw with counter mode.

An alternative approach is to start from a related primitive, called a pseudorandom permutation (PRP). In practical contexts, we call PRPs "block ciphers."

Why use a PRP instead of a PRF? We already saw how to construct encryption schemes from PRFs. So why discuss PRPs? The main reasons are historical and practical: the most widely used encryption schemes are built from PRPs.

Definition.

Like a PRF, a PRP is defined over a keyspace $\mathcal{K}$ and input space $\mathcal{X}$. Unlike a PRF, a PRP actually consists of *two* efficient algorithms P and P^{-1}, both with the type $P, P^{-1}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$.

When using a block cipher (PRP) with input space $\mathcal{X} = \{0,1\}^n$, we refer to the bitlength of the input space n as the *block size*.

A PRP is very similar to a PRF except that:

- for all keys $k \in \mathcal{K}$, $P(k,\cdot)$ maps distinct inputs to distinct outputs, and
- there is an efficient algorithm $P^{-1}(k,\cdot)$ that inverts $P(k,\cdot)$.

Neither of these properties is true for a PRF.

In more detail, we demand the following properties of a PRP:

Correctness. A PRP has a correctness definition, which is that for all keys $k \in \mathcal{K}$, P and P^{-1} must be inverses of each other: For all $k \in \mathcal{K}$ and all $x \in \mathcal{X}$: we require $P(k, P^{-1}(k,x)) = x$.

Security. The PRP security definition is almost exactly the same as that for a PRF, except that we require that oracle access to the PRP keyed with a random value be indistinguishable from oracle access to a truly random *permutation* on the input space.

Formally, we define PRP security using a game:

Definition 1 (PRP Security Game)**.** The game is parameterized by a PRP $P\colon \mathcal{K} \times \mathcal{X} \to \mathcal{X}$, and adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$.

- The challenger samples a key $k \xleftarrow{R} \mathcal{K}$.
- If $b = 0$, the challenger sets $\pi(\cdot) := P(k,\cdot)$.
- If $b = 1$, the challenger sets $\pi(\cdot) \xleftarrow{R} \mathsf{Perms}[\mathcal{X}]$.
- Then for $i = 1, 2, \ldots$ (polynomially many times):
  - The adversary $\mathcal{A}$ sends the challenger a value $x_i \in \mathcal{X}$.
  - The challenger responds with $y_i \leftarrow f(x_i) \in \mathcal{Y}$.

Here, Perms is the set of all permutations on $\mathcal{X}$.

- The adversary outputs a bit $\hat{b}$.

For $b \in \{0,1\}$, let W_b denote the probability that some adversary $\mathcal{A}$ outputs bit "1" in the PRP security game parameterized with bit b. Then define the PRP advantage of $\mathcal{A}$ at attacking P as:

$$\mathsf{PRPAdv}[\mathcal{A}, P] := |\Pr[W_0] - \Pr[W_1]|\,.$$

The Advanced Encryption Standard (AES) is the most widely used block cipher today. The Data Encryption Standard (DES) was the prior standard. Both of these were standardized by the U.S. government. These ciphers are extraordinarily widespread. Almost every computing device you touch today will have one or more AES implementations in its hardware or software. We will discuss the design of DES and AES in a bit.

Review: Security against chosen-plaintext attacks (CPA security)

IMPORTANT: CPA security is a relatively weak definition of security. In practice we use encryption systems that satisfy "authenticated encryption"—a much stronger definition of security. The 6.1600 lecture notes have lots of details on CPA security and authenticated encryption, so look there for more information. Suffice it to say CPA-secure encryption schemes are useful building blocks but are NOT the encryption schemes that you would ever use alone in practice to encrypt messages.

Definition 2 (CPA Security game)**.** The game is parameterized by an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ over message space $\mathcal{M}$, key space $\mathcal{K}$, and a bit $b \in \{0,1\}$:

- The challenger samples $k \xleftarrow{R} \mathcal{K}$.
- As many times as the adversary wants:
  - The adversary sends the challenger messages $m_{i0}, m_{i1} \in \mathcal{M}$. (We require $|m_0^*| = |m_1^*|$.)
  - The challenger replies with $c_i \leftarrow \mathsf{Enc}(k, m_{ib})$.
- The adversary outputs a value $b' \in \{0,1\}$.

Standard encryption systems do not hide the length of the message being encrypted. So, if the message space $\mathcal{M}$ contains messages of different lengths, our security definition requires the adversary to distinguish the encryption of two messages of the *same* length.

For an adversary $\mathcal{A}$ and $b \in \{0,1\}$, let W_b be the event that the adversary outputs "1" in the above game. Then we say that an encryption scheme $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ is **CPA-secure** if for all "efficient" adversaries $\mathcal{A}$, we have

$$\mathsf{CPAAdv}[\mathcal{A}, \mathcal{E}] := |\Pr[W_0] - \Pr[W_1]| \leq \text{"negligible."}$$

Encrypting using a PRP: Switching Lemma

There many different ways to construct CPA-secure encryption schemes using a PRP. The simplest way is via the "Switching Lemma,"

which just says that we can use a PRP as a PRF, provided that the block size is large enough. If we treat the PRP as a PRF, then we can encrypt with the PRP using counter-mode encryption, as we have already seen. This is the approach at the heart of the AES-GCM encryption scheme, which is one of the most widely used symmetric-key encryption schemes today.

The important idea is

See Boneh-Shoup, Theorem 4.4, for the full statement and proof.

Lemma 3 (PRP Switching Lemma)**.** *Let* $P\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$. *Then for every adversary* $\mathcal{A}$ *making at most* T *queries to its challenger,*

$$\mathsf{PRFAdv}[\mathcal{A}, P] \leq \mathsf{PRPAdv}[\mathcal{A}, P] + \frac{T^2}{2^n}.$$

In other words: Say that there exists some efficient adversary $\mathcal{A}$ breaking P as a PRF. Then provided that $T^2 \ll 2^n$, the same adversary can break P as a PRP with roughly the same advantage. If P is a secure PRP, then no such adversary can exist and we can conclude that P is secure as a PRF.

Notice that as soon as $T^2 \approx 2^n$, the Switching Lemma becomes vacuous: it guarantees nothing about whether a secure PRP is a secure PRF. There is a good reason for this: if an adversary can query the PRP/PRF at $T \approx \sqrt{2^n}$ points, it can efficiently distinguish a PRP from a PRF.

To do so, the adversary evaluates the PRP/PRF at $\sqrt{2^n}$ distinct random points. If the adversary every sees a collision—two distinct inputs giving the same output—the adversary outputs "PRF." Otherwise, it outputs "PRP." By the Birthday Paradox, after seeing $\sqrt{2^n}$ input/output pairs, with a PRF, you are likely to see collisions. With a PRP you are not.

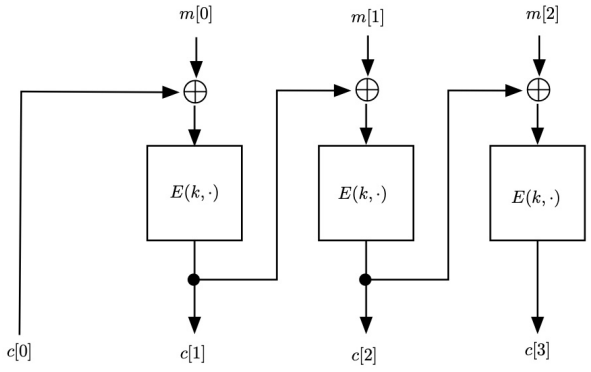
The importance of block size. AES supports up to 256-bit encryption keys, but its block size is fixed as $n = 128$ bits. As a consequence, when using AES as a PRF, as modern systems do in many applications, it is important to remember that it is insecure to invoke AES anywhere near 2^{64} times using the same key.

Encrypting with a PRP: Cipher-block chaining (CBC) mode

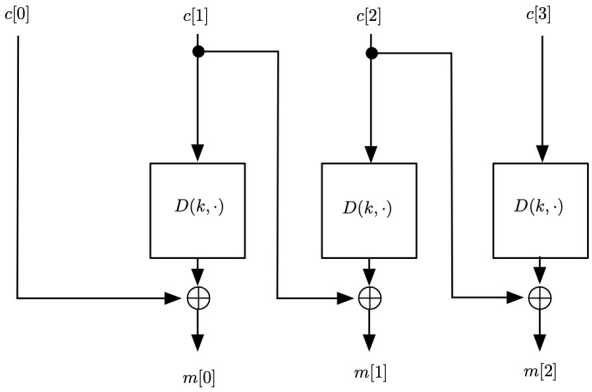
A "mode of operation" is a method for using a block cipher to build an encryption scheme for longer messages. There are many modes of operation out there, but only a few widely used ones. Counter mode we have already seen.

A second mode of operation, which is historically important but should never be used in new systems, is cipher-block chaining (CBC) mode, depicted in Fig. 1. This mode is also randomized: the first

ciphertext block $c[0]$, is a random element in the domain of the block cipher (i.e., a nonce).



(a) encryption



(b) decryption

Figure 1: CBC-mode encryption. This figure appears in the Boneh-Shoup textbook.

As with counter-mode, it is possible to prove a statement showing that CBC-mode encryption is CPA-secure if the underlying pseudo-random permutation $P\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ is secure. The statement asserts that, for every CPA adversary $\mathcal{A}$ seeing T encryptions of ℓ-block messages there is a PRP adversary $\mathcal{B}$ running in roughly the same time such that:

$$\mathsf{CPAAdv}[A,\mathcal{E}] \leq 2 \cdot \mathsf{PRPAdv}[\mathcal{B},P] + \frac{2T^2\ell^2}{2^n}.$$

Many legacy systems use CBC-mode encryption, while modern systems do not. Why?

- CBC mode is not parallelizable.
- CBC mode requires evaluating the PRP in both the forward and inverse directions. This can require more code and more

hardware in implementations.

- CBC ciphertexts must be a multiple of the block size, which is annoying for short messages and requires additional delicate padding.

Attacking CBC mode with small block size ("Sweet32 attack"). Some protocols for legacy reasons still support CBC mode with block ciphers using an $n = 64$-bit block. The security statement we have above becomes vacuous as soon as an attacker can observe 2^{32} encrypted blocks of traffic.

Indeed, there is a concrete attack: after seeing enough encrypted blocks, it is likely that two ciphertext blocks are equal: $c_i = c_j$. Since $P(k, \cdot)$ is a permutation, this means that:

$$\begin{aligned} c_i &= c_j \\ P(k, m_i \oplus c_{i-1}) &= P(k, m_j \oplus c_{j-1}) \\ m_i \oplus c_{i-1} &= m_j \oplus c_{j-1} \\ m_i \oplus m_j &= c_{i-1} \oplus c_{j-1}. \end{aligned}$$

So a collision reveals the XOR of two message blocks!

Exploiting this in practice may not be easy, is theoretically possible given a few hundred GBs of encrypted traffic [1]. One moral is that using a large-enough block size is important. A second moral is that it's important to pay attention to the concrete security guarantees that a cryptographic construction has when you are setting parameters.

Block ciphers used in practice

NIST publishes standards for block ciphers. There have been three widely used ones:

	Key size	Block size
DES (1975)	56 bits	64 bits
3DES	168 bits	64 bits
AES (1998)	128, 192, or 256 bits	128 bits

We will see one attack that exploits small block sizes. A 128-bit block size is the minimum.

Even-Mansour Cipher and AES

Before describing AES, let's look at a simpler cipher design by Even and Mansour.

Even-Mansour Cipher

On security parameter n, the Even-Mansour cipher has a $2n$-bit key and an n-bit block size.

The cipher makes use of a public invertible permutation $\Pi\colon \{0,1\}^n \to \{0,1\}^n$, as in ChaCha20, which we can model as a truly random permutation.

The cipher is then just defined as:

$$P_{\mathsf{EM}}((k_0, k_1), x) := k_1 \oplus \Pi(x \oplus k_0).$$

You might notice that this design looks quite similar to the overall structure of the ChaCha20 PRF.

Provided that we model Π as a truly random permutation, we can prove that P_{EM} is a secure PRP. That is, for every adversary $\mathcal{A}$ making at most T queries to Π and at most T queries to the PRP challenger,

$$\mathsf{PRPAdv}[\mathcal{A}, P_{\mathsf{EM}}] \leq \frac{2T^2}{2^n}.$$

AES

AES is an "iterated Even-Mansour cipher." AES operates on a key of size $\{128, 192, 256\}$. The block size is a fixed 128 bits.

Let the block size be n. AES first derives a number of "round keys" $k_0, \ldots, k_r \in \{0,1\}^n$ from the input key k. Each round key is a linear function of the input key k.

Unlike DES, the government uses AES to protect classified data. They use 128+ bits for SECRET data, and 192+ bits for TOP SECRET data.

Foreign governments who do not trust AES and design their own ciphers apparently often use weak homebrewed ciphers.

After that, the AES cipher on input $x \in \{0,1\}^n$ just alternates between XORing a round key into the state and applying the permutation Π:

- $\mathsf{st} \leftarrow x \oplus k_0$
- For $i = 1, \ldots, r$:
  - $\mathsf{st} \leftarrow \Pi(x) \oplus k_i$.
- Output st.

When using 128-bit keys, the number of rounds is $r = 10$.

The one detail we omit is that true AES uses a slightly different permutation Π in the last round. This apparently enables some performance optimizations in hardware.

Instantiate permutation – final AES design

Now to get to the full AES construction, we just need to instantiate the public permutation $\Pi\colon \{0,1\}^n \to \{0,1\}^n$. It modifies the input in three steps:

- `SubBytes`. Use a lookup table $S\colon \{0,1\}^8 \to \{0,1\}^8$ hardcoded into the design to replace each of the 16 input bytes with a different one:

  $$b_1 \| b_2 \| \cdots \| b_{16} \quad \longmapsto \quad S(b_1) \| S(b_2) \| \cdots \| S(b_{16}).$$

  This is a non-linear operation.
- `ShiftRows`. View the 16-byte block as a 4×4 matrix. Perform a cyclic shift on this matrix: shift the 0th row 0 cells to the right, the first row 1 cell to the right, the second row 2 cells to the right, and the third row 3 cells to the right.
- `MixColumns`. View the 16-byte block as a 4×4 matrix. Multiply it by a fixed matrix.

The matrix multiplication in the `MixColumns` step is in $\mathrm{GF}(2^8)$. If you don't know what that means, it doesn't matter.

Cache attacks

Notice that a naïve implementation of the SubBytes operation in AES requires secret-dependent memory accesses. This is one reason why many implementers frown on using software AES implementations—they are difficult to get right.

The issue is in implementing the S-boxes, or other table lookups, in software. *Cache-timing attacks* are one serious pitfall. To explain: because of caching, if the AES routine makes consecutive lookups to $S[1]$ then $S[1]$, these will complete faster than if it looks up $S[1]$ then $S[187]$. The difference in timing—even though it is small—leaks information about the internal state of the cipher. This can be enough to perform devastating attacks.

ChaCha20 does not have this problem.

DES

Reference note: This discussion is just a restatement of the descriptions of DES and AES in the Boneh-Shoup textbook. Consult their book for details.

With explosion of potential commercial applications of cryptography, the predecessor to NIST published the "Data Encryption Standard" (DES) in 1975. The government never approved it for use in classified applications, as the 56-bit key length was too short even on the day the standard was published.

Diffie and Hellman at the time pointed out that 56-bit keys were unacceptably short. Their analysis was based on projections about cheap computation would get and how quickly. They presciently proposed that using 128-bit keys would be prudent.

As far as I know, the best known attack on DES today is Matsui's linear cryptanalysis (1993), which recovers the key from 2^{47} input-output pairs in a known-plaintext attack.

As with AES, we can view the design of DES as building a block cipher from a smaller idealized primitive. In AES, the idealized primitive was a public random permutation. In DES, the idealized primitive is a pseudorandom function.

Feistel Network: Building a PRP from a PRF.

The core of DES is a slick idea, proposed by Horst Feistel, for building a PRP out of a PRF. The neat property of the Feistel network is that it is invertible: it turns a random function—not invertible—into an efficiently invertible random permutation.

The Feistel network builds a PRP out of a pseudorandom function by applying a simple transformation many times (over many "rounds"). One round of Feistel network, when instantiated with a PRF $F\colon \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, is a keyed permutation

$$P\colon \mathcal{K} \times \{0,1\}^{2n} \to \{0,1\}^{2n},$$

defined as:

$$\pi_F(k,(x,y)) := (y, x \oplus F(k,y)).$$

The initial and final bit shuffling in DES have no apparent effect on cipher's security. Boneh-Shoup speculates that the initial and final permutations were to slow down DES implementations in software relative to hardware. This is consistent with the widespread (and somewhat confirmed) view that DES was designed to be breakable by governments but good enough for commercial use.

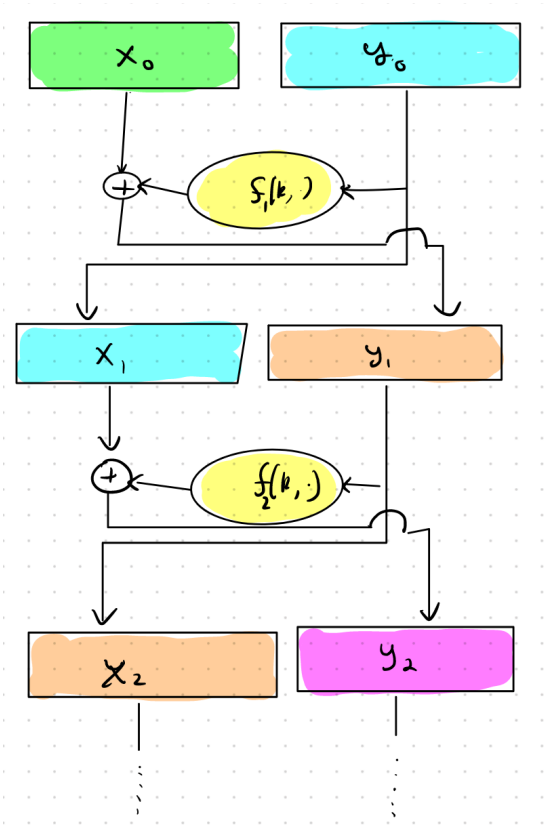
One round of the Feistel network is not enough to construct a PRP from a PRF, but many rounds are. The ℓ-round Feistel network is the construction that applies π_F many times with independent keys.

$$P_\ell((k_1,\ldots,k_\ell),(x,y)) := \pi_F(k_\ell, \pi_F(k_2, \pi_F(k_1,(x,y))) \cdots)$$

In particular, Luby and Rackoff proved that if F is a secure pseudorandom function, then the three-round Feistel network P_3 instantiated with F is a secure PRP. In particular, for every efficient adversary $\mathcal{A}$ attacking P_3 as a PRP, there is an efficient adversary $\mathcal{B}$ as a PRF such that:

$$\mathsf{PRPAdv}[\mathcal{A}, P_3] \leq 3 \cdot \mathsf{PRFAdv}[\mathcal{B}, F] + \frac{Q^2}{N} + \frac{Q^2}{2N^2}.$$

Figure 2: Feistel network



Instantiate the round functions – final DES construction

DES uses a 56-bit keyspace ($|\mathcal{K}| = 2^{56}$) and a 64-bit block size ($2n = 64$).

The DES cipher on input $x \in \{0,1\}^{2n}$ is essentially just a 16-round Feistel network. (The only difference is that DES applies a fixed permutation of the input bits before and after the Feistel network.) The Luby-Rackoff result does not say anything about the security of DES since the functions that DES applies at each round are not pseudorandom functions at all: the keys are short enough to brute force, for one.

The round function. The last step is to instantiate the PRF used in the Feistel network with a concrete function.

To do so, DES first generates 16 "round keys" $k_1, \ldots, k_{16} \in \{0,1\}^{48}$, each derived from a subset of the bits of the 56-bit DES key k.

The round function $F(k_i, \cdot)$:

- computes the initial state as a linear function of the round key and its input,
- splits the state into six-bit chunks and applies a different *non-linear* function (called an S-box for "substitution") to each using a lookup table, and
- permutes the bits of the state.

The only non-linear part of the cipher is the S-boxes. It turns out that if you pick the S-boxes at random, the cipher becomes very weak. The DES S-boxes are constructed to have a bunch of nice statistical properties that prevent attacks. (For example, no output bit is not close to a linear function of the input bits.)

Instantiating the ideal DES construction with these 16 keyed round functions gives the final construction.

Lessons?

- Many rounds of a simple operation.
- Avoiding all of the known attacks takes a lot of care. Choosing parameters at random does not work.

Linear cryptanalysis

Note: Again, these notes are mostly a rephrasing of the content in Boneh-Shoup. Look there for the details.

The cryptanalysis of block ciphers is an art on its own. I will try to sketch the idea behind one sort of attack (based on the description of Matsui's attack as summarized in the Boneh-Shoup book), which may give you the flavor of how these attacks work.

Let P be a PRP in which inputs, outputs, and keys are all n-bit strings. Further, let say that you find that there is some bias in the relationship between the key bits, input bits, and output bits.

A *linear relation* on the PRP P exists if there are sets of bit positions $B_x, B_y, B_k \subseteq [n]$ such that

$$\Pr\left[x[B_x] \oplus y[B_y] = k[B_k] \ : \ \begin{array}{c} x \xleftarrow{R} \{0,1\}^n \\ y \leftarrow P(k,x) \end{array}\right] \geq \frac{1}{2} + \epsilon,$$

where ϵ is noticeable.

In a truly random permutation, $P(k,x)$ is independent of x, so the bias $\epsilon = 0$. In a concrete PRF, the bias can be non-zero.

The idea of the attack is to gather a very large number of input-output pairs: $(x_1, y_1), \ldots, (x_T, y_T)$.

Then if we look at all of the input/output XORs, the linear relation tells us that the resulting values will be slightly biased towards the value of the B_kth bit of the key:

$$(x_1[B_x] \oplus y_1[B_y]), \ldots, (x_T[B_x] \oplus y_T[B_y]).$$

The idea is then to take the majority of these T values and use that value as our guess at the XOR of a subset of the key bits $k[B_k]$.

The Chernoff bound tells us that our guess will be right with probability at least $1 - \exp(T\epsilon^2/2)$. So if we have a linear relation with bias ϵ, we get a bit of information about the key with probability well over $1/2$ after seeing something like $T = 4/\epsilon^2$ input/output pairs.

In the case of DES, one linear relation depends on 12 bits of the secret key. If you have additional linear relations, you can use these to recover the full key. See Boneh-Shoup 4.3.1 for details.

References

[1] Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In *ACM CCS*, 2016.