A Brief Introduction to Symmetric Cryptography

KG

August 30, 2018

Contents

1 Introduction

In this note, we consider the following problem. Alice wants to send messages to Bob via some communications channel. Eve has access to the channel and she may eavesdrop on and possibly tamper with anything sent over the channel.

Alice does not want Eve to be able to eavesdrop on her messages. She wants to communicate *confidentially*. Also, when Bob receives a message that looks like it came from Alice, then Bob wants to be sure that Alice really sent the message and that Eve did not tamper with it. Alice and Bob want *integrity*.

Alice and Bob share a secret, called the *key*. Cryptography where the sender and the receiver (the honest users) have the same knowledge is called *symmetric cryptography*, where the word symmetry refers to the symmetry of knowledge.

We define what a *symmetric cryptosystem* is and what the *security requirements* are for such cryptosystems in Section 2.

To illustrate standard *attacks*, it is useful to study some historic cryptosystems and how those systems can be attacked. This is done in Section 3, which alternates between discussing historical ciphers and discussing interesting attacks that apply to the historical ciphers. (Note that Section 3 is no history of cryptography.)

Section 3 contains a few constructions that provide confidentiality against eavesdroppers. These constructions do not provide integrity, nor do they provide confidentiality if Eve is willing to tamper with ciphertexts.

The main tool for providing integrity is discussed in Section 4. How to combine the constructions provided in Sections 3 and 4 into cryptosystems providing both integrity and confidentiality even when Eve tampers with the ciphertexts is discussed in Section 5.

This text is intended for a reader that is familiar with mathematical language, basic algebra (groups, rings, fields, linear algebra and polynomials), elementary probability theory and elementary computer science (algorithms).

This text is very informal. Every concept and result mentioned in the text can be made precise, but the technical details are out of scope for this text.

While modern high-level constructions are discussed in this note, low-level constructions are out of scope. This explains why this note defines what a block cipher is and gives an informal explanation of what it means for a block cipher to be secure, but does not contain a single example of a modern block cipher.

Another topic that is out of scope is proving the security of the modern constructions discussed. We only include proofs for *information theoretically secure* constructions.

This text uses colour to indicate who is supposed to know what. Red denotes secret information (typically keys) known only by Alice and Bob. Green denotes information that Alice and Bob want to protect, typically messages. Blue denotes information that the eavesdropper will see.

2 Basic Definitions

We begin with the definition of a symmetric cryptosystem and what it means for a cryptosystem to be *secure*.

Definition 1. A *symmetric cryptosystem* consists of

- a set $\mathcal{K}$ of *keys*;
- a set $\mathcal{P}$ of *plaintexts*;
- a set $\mathcal{C}$ of *ciphertexts*;
- an *encryption algorithm* $\mathcal{E}$ that on input of a key and a plaintext outputs a ciphertext; and
- a *decryption algorithm* $\mathcal{D}$ that on input of a key and a ciphertext outputs either a plaintext or the special symbol $\perp$ (indicating an invalid ciphertext).

For any key k and any plaintext m, we have that

$$\mathcal{D}(k, \mathcal{E}(k, m)) = m.$$

The set $\mathcal{P}$ will usually be a set of finite sequences of letters from an *alphabet*.

We shall assume that the key Alice and Bob share has been chosen uniformly at random from the set of keys.

3 Confidentiality Against Eavesdroppers

In this section we shall consider the situation where Eve is eavesdropping on Alice and Bob. Eve's goal is to understand what Alice is saying to Bob.

We shall briefly discuss some historic cryptosystems. We do this to give a gentle introduction to the basic concepts in cryptograpy and provide some insight into important attack strategies.

The presentation in this section alternates between describing a cryptosystem and describing how to attack that cryptosystem, until we reach systems that will provide confidentiality.

Definition (informal) 2. A symmetric cryptosystem provides *confidentiality* if it is – without knowledge of the key – hard to learn anything at all about the decryption of a ciphertext from the ciphertext itself, except possibly the length of the decryption.

3.1 Shift Cipher

The shift cipher is also known as the Cæsar cipher.

Suppose first that we give our alphabet G some group structure. There is a natural bijection between the English alphabet $\{\text{A}, \text{B}, \text{C}, \ldots, \text{Z}\}$ and the group $\mathbb{Z}_{26}^{+}$, given by $0 \leftrightarrow \text{A}$, $1 \leftrightarrow \text{B}$, etc. We add F and G by applying the bijection to get 5 and 6, adding them to 11, and then applying the inverse bijection to get L.

The *plaintext* m is a sequence of letters $m_1 m_2 \ldots m_L$ from the alphabet. The *key* is an element k from G. We *encrypt* the message by adding the key to each letter, that is, the ith ciphertext letter is

$$c_i = m_i + k, \qquad 1 \leq i \leq L. \tag{1}$$

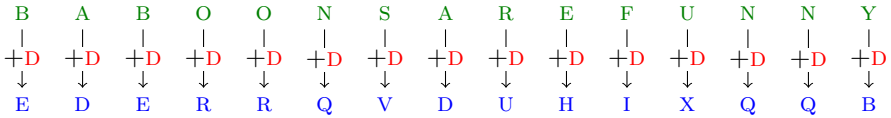

Figure 1: Example of shift cipher encryption with the key $k = \text{D}$ (D corresponds to the number 3).

The ciphertext c is the sequence of letters $c_1 c_2 \dots c_L$.

To *decrypt* a ciphertext $c = c_1 \dots c_L$, we subtract the key from each ciphertext letter, that is, the ith plaintext letter is

$$m_i = c_i - k, \qquad 1 \le i \le L.$$

E *Exercise (algorithms)* 1. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$ and $\mathcal{C}$ are, and implement the two algorithms $\mathcal{E}$ and $\mathcal{D}$. Show that $(\mathcal{K}, \mathcal{P}, \mathcal{C}, \mathcal{E}, \mathcal{D})$ is a symmetric cryptosystem.

E *Exercise* 2. How many different keys are there for the shift cipher when the alphabet has 26 elements?

3.1.1 Attack: Exhaustive Search

The easiest attack on the shift cipher is an *exhaustive search* for the key, or a *brute force attack*. The two assumptions required for this attack is that only one key will give a reasonable decryption, and that we will be able to recognize that decryption. Both of these assumptions are almost always true.

If there are few keys, we can decrypt with all possible keys in reasonable time. The correct key will be the one that gives a reasonable decryption.

E *Exercise* 3. Find all the possible decryptions of HGHUUT. How many are English words? What about the possible decryptions of MBQ?

E *Exercise (for groups)* 4. Choose a key for the Shift cipher at random and encrypt some message. Give the ciphertext to someone else in the group and have them decrypt it without knowing the key.

3.2 Affine Cipher

Now we give our alphabet R a ring structure, say like $\mathbb{Z}_{26}$. We add as before. We multiply F and G by applying the bijection to get 5 and 6, multiplying them to get 30 which is 4 modulo 26, and then applying the inverse bijection to get E.

The plaintext m is a sequence of letters $m_1 m_2 \dots m_L$ from the alphabet. The key is a pair (k_1, k_2) of ring elements, the first of which must be invertible. We encrypt the message letterwise using the formula

$$c_i = k_1 m_i + k_2, \qquad 1 \le i \le L. \tag{2}$$

The ciphertext c is the sequence of letters $c_1c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 \dots c_L$, we compute the ith plaintext letter using the formulas

$$m_i = k_1^{-1}(c_i - k_2), \qquad 1 \leq i \leq L.$$

E *Exercise* 5. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

E *Exercise* 6. How many different keys are there for the affine cipher when the alphabet has 26 elements?

3.2.1 Attack: Known Plaintext

Suppose Eve knows that Alice always begins her messages with HI. One ciphertext starts with the letters UB. When the attacker knows the plaintext corresponding to a piece of ciphertext, that is called *known plaintext*.

Eve knows that Alice used the affine cipher, which means that equation (2) was used to encrypt H to U and I to B. She gets the following two equations:

$$\begin{aligned} \mathrm{U} &= k_1\mathrm{H} + k_2, \\ \mathrm{B} &= k_1\mathrm{I} + k_2. \end{aligned} \tag{3}$$

This is a linear system of equations with two equations and two unknowns. As long as the difference $\mathrm{H} - \mathrm{I}$ is invertible in the ring (which it is), we can solve the system and recover the key (k_1, k_2).

E *Exercise* 7. Solve the linear system of equations given in (3) to find the key.

E *Exercise (for groups)* 8. Choose a key for the affine cipher at random and encrypt some message. Give the ciphertext along with some known plaintext to someone else in the group and have them decrypt it without knowing the key.

E *Exercise* 9. Develop a similar known plaintext attack for the Shift cipher from Section 3.1.

3.3 Substitution Cipher

The formulas (1) and (2) define bijections on the alphabet. We can generalize these schemes by using any bijection or *permutation* on our alphabet. Let our alphabet be a set S.

The plaintext m is a sequence of letters $m_1m_2 \dots m_L$ from the alphabet. The key is a permutation π on S. We encrypt the message letterwise using the formula

$$c_i = \pi(m_i), \qquad 1 \leq i \leq L. \tag{4}$$

The ciphertext c is the sequence of letters $c_1c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 \dots c_L$, we compute the ith plaintext letter using the formula

$$m_i = \pi^{-1}(c_i), \qquad 1 \leq i \leq L.$$

E *Exercise* 10. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

E *Exercise* 11. How many different keys are there for the substitution cipher when the alphabet has 26 elements?

E *Exercise* 12. Explain how we can recover part of the key (a *partial key*) from known plaintext, but not necessarily the full key.

3.3.1 Attack: Frequency Analysis

Known plaintext will reveal part of the key. But there are stronger attacks on the substitution cipher, based on the number of times the various ciphertext letters appear.

If the permutation takes the plaintext letter A to the ciphertext letter Z, the number of Z's in the ciphertext will be the same as the number of A's in the plaintext. This means that the relative frequencies of the ciphertext letters will be the same as the relative frequencies of the plaintext letters, up to permutation.

For most long English texts, the relative frequency of the various letters is constant. This means that for encryptions of long English texts, the relative frequencies of ciphertext letters is a simple permutation of the relative frequencies of letters in English text. It will be a simple matter of matching plaintext letters and ciphertext letters and thereby recovering the key and thus the plaintext.

For English texts of moderate length, the relative frequencies of the less common letters will vary a lot, and reliable matching of plaintext letters to ciphertext letters will be impossible. However, some letters, E in particular, are so common in English that they will usually be the most common letters, even for fairly short texts.

E *Exercise* 13. Gather a collection of English texts of varying topic and length. Compute the frequency distributions. Use these distributions to estimate how long a text must be before we can expect to identify with reasonable certainty (a) E, (b) the five most frequent letters, and (c) the ten most frequent letters.

This means that even though we cannot reliably match every plaintext letter to every ciphertext letter, we can match a few plaintext letters to a few ciphertext letters. This gives us a partial key and a partial decryption.

The next step is to pretend that this partial decryption is a crossword puzzle, and guess some plaintext words that fit with the partial decryption. We then treat these guesses as known plaintext and recover more of the key. This gives us a better partial decryption. If the new partial decryption does not make sense or is impossible, we guessed wrong. We backtrack and guess again.

If, on the other hand, the new partial decryption makes sense, we probably guessed right. Now we treat the new partial decryption as a crossword puzzle. We repeat this process of *guessing* and *verifying* until we have the complete decryption.

E *Exercise (for groups)* 14. Choose a key for the substitution cipher and encrypt some sufficiently long message. Give the ciphertext to someone else in the group and have them decrypt it without knowing the key. You may also give them a small amount of known plaintext.

3.4 Towards Block Ciphers

One approach to preventing frequency analysis is to use a permutation on pairs of letters. That is, our permutation acts on the set S of all pairs of letters, not the set of letters.

E *Exercise* 15. For a substitution cipher based on permutations on pairs, write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

Unfortunately, the frequencies of pairs are uneven, which means that frequency analysis still works, although it is less effective. A permutation on triples of letters would be better, but still not perfect.

Even better would be l-tuples. The number l is called the *block length*. Unfortunately, representing a random permutation over a large set is impractical. (Merely writing down a permutation requires at least $\log_2(|S|^l!) \approx |S|^l(\ln|S|^l - 1)/\ln 2$ binary digits.)

One idea would be to use not a random permutation, but instead use a random member of some family of permutations.

The Hill cipher is an example of such a family of permutations, namely the permutations described by invertible matrices. We give our alphabet R a ring structure, say like $\mathbb{Z}_{26}$. We denote a l-tuple of letters as $\mathbf{m} \in R^l$. An invertible $l \times l$ matrix K acts upon such l-tuples in the obvious fashion, and we denote this action by $K\mathbf{m}$.

The plaintext m is a sequence of l-tuples of letters $\mathbf{m}_1\mathbf{m}_2\ldots\mathbf{m}_L$. The key is an invertible $l \times l$ matrix K. We encrypt the message using the formula

$$\mathbf{c}_i = K\mathbf{m}_i, \qquad 1 \le i \le L.$$

The ciphertext c is the sequence of l-tuples $\mathbf{c}_1\mathbf{c}_2\ldots\mathbf{c}_L$.

To decrypt a ciphertext $c = \mathbf{c}_1\ldots\mathbf{c}_L$, we compute the ith plaintext tuple using the formula

$$\mathbf{m}_i = K^{-1}\mathbf{c}_i, \qquad 1 \le i \le L.$$

E *Exercise* 16. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

E *Exercise* 17. How many different keys are there for the Hill cipher with block length 2 when the alphabet has 29 elements?

E *Exercise* 18. How many blocks of ciphertext-plaintext correspondences do you need to recover K with reasonable probability, when the block length is 2 and the alphabet has 29 elements? (For the purposes of this exercise only, you may assume that the known plaintext consists of random letters from the alphabet.)

E *Exercise (for groups)* 19. Choose a key for the Hill cipher and encrypt some message. Give the ciphertext along with some known plaintext to someone else in the group and have them decrypt it without knowing the key.

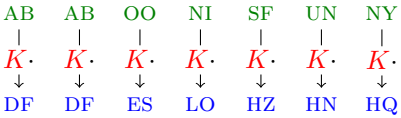

Figure 2: Example of Hill cipher encryption with the key $K = \begin{pmatrix} \text{B} & \text{D} \\ \text{A} & \text{F} \end{pmatrix}$.

A second example of a family of permutations is the Pohlig-Hellman exponentiation cipher. This time, we do not consider tuples of letters, but rather a very large alphabet. We give our large alphabet G the structure of a cyclic group of order n.

The plaintext m is a sequence of group elements $m_1 m_2 \dots m_L$. The key is an integer k between 0 and n that is relatively prime to n. We encrypt the message using the formula

$$c_i = k m_i, \qquad 1 \le i \le L.$$

The ciphertext c is the sequence of group elements $c_1 c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 c_2 \dots c_L$, we compute the i plaintext tuple using the formula

$$m_i = k^{-1} c_i, \qquad 1 \le i \le L,$$

where the inverse k^{-1} of k is computed modulo n.

Note that the expression km_i where k is an integer and m_i is a group element is very different from the expression km_i when k and m_i are elements in a ring. The former notation is short for $m_i + m_i + \dots + m_i$, where the sum contains k terms.

E *Exercise* 20. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

It is generally believed that if the group G is carefully chosen, it is hard to find the key, even with known or chosen plaintext.

3.4.1 Attack: Distinguishing

The permutations used in the Hill cipher (linear, invertible maps) are very different from most permutations. For any two l-tuples $\mathbf{m}, \mathbf{m}'$ of letters from the alphabet, an invertible matrix K satisfies the equation

$$K\mathbf{m} + K\mathbf{m}' = K(\mathbf{m} + \mathbf{m}').$$

The same observation holds for the permutations used in the Pohlig-Hellman cipher. For any two elements $m, m' \in G$, we get that

$$km + km' = k(m + m').$$

Most permutations would not satisfy these equations. This means that the permutations do not look like randomly chosen permutations. It is easy to *distinguish* the Hill cipher and Pohlig-Hellman cipher permutations from random permutations.

We can use this property to make simple deductions about plaintext based only on ciphertext properties.

E *Exercise* 21. Consider the Hill cipher. Suppose c, c' and c'' are ciphertexts such that $\mathbf{c}_i + \mathbf{c}'_i = \mathbf{c}''_i$ for one or more indexes i. What can you say about the corresponding plaintexts?

3.5 Block Ciphers

We shall now work with a set S, typically the set of l-tuples of letters from our alphabet.

Definition 3. A *block cipher* is a pair of maps $\pi, \pi^{-1} : \mathcal{K} \times S \to S$ such that for all $k \in \mathcal{K}$ and $s \in S$ we have that

$$\pi(k, \pi^{-1}(k, s)) = s \text{ and } \pi^{-1}(k, \pi(k, s)) = s.$$

In other words, a block cipher is a family of permutations on a set S indexed by a key set $\mathcal{K}$.

E *Exercise* 22. The Hill cipher is based on a block cipher. Identify the block cipher by explaining what the key set $\mathcal{K}$, the set S and the functions π, π^{-1} are.

E *Exercise* 23. The Pohlig-Hellman cipher is based on a block cipher. Identify the block cipher by explaining what the key set $\mathcal{K}$, the set S and the functions π, π^{-1} are.

Despite the name, a block cipher by itself is not a cryptosystem. But we construct a cryptosystem based on a block cipher.

The plaintext m is a sequence of elements $m_1 m_2 \dots m_L$ from the set S. The key is an element k in $\mathcal{K}$. We encrypt the message elementwise using the formula

$$c_i = \pi(k, m_i), \qquad 1 \leq i \leq L.$$

The ciphertext c is the sequence of set elements $c_1 c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 \dots c_L$, we compute the ith plaintext element using the formula

$$m_i = \pi^{-1}(k, c_i), \qquad 1 \leq i \leq L.$$

E *Exercise* 24. The above is an informal description of a block cipher used in *electronic code book (ECB) mode*. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

Our hope is that this scheme will be both practical and as good as the substitution cipher when used on l-tuples. But as we have seen, the Hill cipher is easy to attack.

Definition (informal) 4. A block cipher is *secure* if it is hard to distinguish a pair of randomly chosen inverse permutations (π, π^{-1}) from the pair of inverse permutations $(\pi(k, \cdot), \pi^{-1}(k, \cdot))$, where k has been chosen uniformly at random from $\mathcal{K}$.

Note that whoever is trying to distinguish is only allowed to see the functions evaluated at various points. He is never allowed to see the permutation π or the key k.

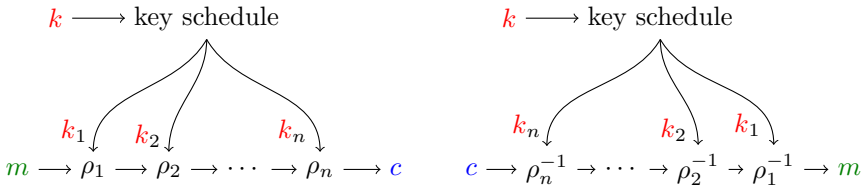

Figure 3: Typical high-level block cipher design.

The idea is that if it is hard to distinguish the block cipher's permutations from "average" permutations, we may as well use the block cipher with a random key instead of a random permutation. If there is an attack on the block cipher cryptosystem that does not work on the substitution cipher, that will be one way to distinguish the block cipher.

We note that for a block cipher to be secure, the key set must be very large. Otherwise, one can recognize the block cipher permutations with high probability by enumerating all the keys and observing how the corresponding permutation affects one or two elements of the set.

3.5.1 Sketch: Feistel Ciphers

How to construct secure block ciphers is out of scope of this note. However, we shall very briefly discuss one popular design for block ciphers, the *Feistel cipher*.

Block ciphers are typically built by repeatedly applying one or more simple block ciphers called *rounds*. A single round will be very easy to break, but the composition of sufficiently many rounds may be hard to break. Given the rounds $\rho_1, \rho_2, \ldots, \rho_n$ with corresponding inverse rounds, we get a block cipher π by composition:

$$\pi(k, m) = \rho_n(k_n, \cdots \rho_2(k_2, \rho_1(k_1, m)) \cdots) \qquad \text{and}$$
$$\pi^{-1}(k, m) = \rho_1^{-1}(k, \cdots \rho_{n-1}^{-1}(k_{n-1}, \rho_n^{-1}(k_n, m)) \cdots)$$

The *round keys* $k_1, k_2, \ldots, k_n$ are derived from the key k. Using the same key for each round is problematic. Using independent keys such that $k = (k_1, k_2, \ldots, k_n)$ leads to impractically large keys. Instead, a *key schedule* is usually used to derive round keys from the block cipher key. This high-level design is shown in Figure 3. We note that for good ciphers, the key schedule and the rounds are highly dependent on eachother.

One convenient way to design rounds is the *Feistel round*. The construction assumes that $S = G \times G$ for some finite group G. It uses a *round function* $F : \mathcal{K} \times G \to G$ to construct the the permutation

$$\rho(k, (L, R)) = (R, L + F(k, R)).$$

It is easy to see that the inverse permutation is

$$\rho^{-1}(k, (L', R')) = (R' - F(k, L'), L').$$

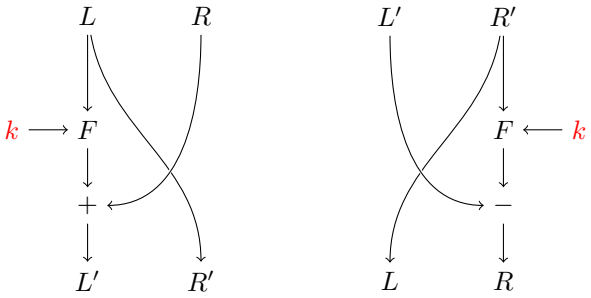

Figure 4: The Feistel round and its inverse.

Choosing a suitable round function is a hard problem, especially when the goal is to find a round function that can be computed very quickly and that does not require many rounds. Again, this problem is out of scope for this note.

3.5.2 Practical: Padding

The plaintext set for the cryptosystem from Exercise 24 is the set of finite sequences of elements from S, where each set element is typically an l-tuple of letters from the alphabet. In other words, the plaintext set is the set of letter sequences whose length is divisible by l.

But when l is large, it is unreasonable to expect message lengths to be a multiple of l. We usually need to encrypt arbitrary sequences of letters. Since we need to decrypt correctly, we cannot just append some fixed letter until the sequence length is a multiple of l.

We extend a cryptosystem to accept sequences of any length by constructing a suitable injective function, a so-called *padding scheme*.

Definition 5. Let $\mathcal{P}$ and $\mathcal{P}'$ be sets. A *padding scheme for $\mathcal{P}$ and $\mathcal{P}'$* consists of two functions $\iota : \mathcal{P} \to \mathcal{P}'$ and $\lambda : \mathcal{P}' \to \mathcal{P} \cup \{\bot\}$ satisfying

$$\lambda(\iota(m)) = m \text{ for all } m \in \mathcal{P}.$$

E *Exercise* 25. Suppose you have a cryptosystem $(\mathcal{K}, \mathcal{P}', \mathcal{C}, \mathcal{E}', \mathcal{D}')$ and a padding scheme (ι, λ) for $\mathcal{P}$ and $\mathcal{P}'$. Based on the padding scheme and the cryptosystem, construct a new cryptosystem $(\mathcal{K}, \mathcal{P}, \mathcal{C}, \mathcal{E}, \mathcal{D})$. Show that it is indeed a cryptosystem.

Typically, the alphabet will be $\{0, 1\}$ and our set is $S = \{0, 1\}^l$, bit strings of length l. The plaintext set $\mathcal{P}'$ will then be bit strings of length divisible by l.

One padding scheme is the following: We first add one 1-bit, then we add 0-bits until the total length is divisible by l. If the block size l is 8, the bit string 10 10 1 will become 10 10 11 00. If the block size l is 5, the bit string 01 01 0 becomes 01 01 01 00 00.

To remove the padding, we remove up to $l-1$ trailing 0-bits and exactly one 1-bit. If the block size l is 8, the string 01 01 01 01 01 becomes the bit string 01 01 01 01 0. If the

block size l is 5, the string 01 01 0 becomes 01 0, while the string 10 10 10 00 00 cannot be decoded and therefore becomes $\perp$.

If decoding fails as in the last example, we have a *padding error*. Padding errors have subtle effects on the security of cryptographic protocols.

3.5.3 Attack: Block Repetitions

We make two observations about ECB mode (the cryptosystem from Exercise 24):

- Any repetition among plaintext blocks will cause corresponding repetitions among ciphertext blocks.
- If we encrypt the same message twice, we get the same ciphertext.

Both of these observations allow an eavesdropper to learn something about the message from the ciphertext, and thereby break confidentiality. Both observations are independent of which block cipher we use. The problem is not with the concept of block cipher, but with how we use the block cipher.

E *Exercise* 26. Suppose a message has been encrypted with ECB mode using a block cipher with block length 4, and that you know that the message is either

SELL THE HOUSE NOW DO NOT SELL THE CABIN

or

SELL THE HOUSE AND EVERYTHING ELSE NOW.

(Ignore spaces and punctuation.) Explain how you can decide which message is the decryption by only looking at the ciphertext.

3.6 A Correct Use of a Block Cipher

We now allow our block cipher to operate on a group G instead of a set. (Note that our permutations will still act on the set of group elements. Since most permutations on G do not respect the group operation, neither should the block cipher permutations.)

The plaintext m is a sequence of elements $m_1 m_2 \dots m_L$ from the group G. The key is an element k in $\mathcal{K}$. We encrypt the message elementwise by first choosing a random group element c_0 and then using the formula

$$c_i = \pi(k, m_i + c_{i-1}), \qquad 1 \le i \le L.$$

The ciphertext c is the sequence of set elements $c_0 c_1 c_2 \dots c_L$.

To decrypt a ciphertext $c = c_0 c_1 \dots c_L$, we compute the ith plaintext element using the formula

$$m_i = \pi^{-1}(k, c_i) - c_{i-1}, \qquad 1 \le i \le L.$$

E *Exercise* 27. The above is an informal description of a block cipher used in *cipherblock chaining (CBC) mode*. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

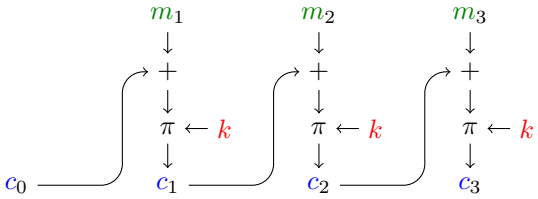

Figure 5: Cipherblock chaining (CBC) mode encryption diagram using the block cipher (π, π^{-1}). To get the decryption diagram, reverse the direction of the vertical arrows, replace $+$ by $-$ and π by π^{-1}.

What happens when we encrypt is that we start at a random group element. This random element is added to the first message block, which is then permuted, resulting in essentially a random-looking group element. This element is added to the second message block, which is again permuted, resulting in essentially a second random-looking group element. This process continues, producing a ciphertext that consists of a sequence of random-looking group elements.

Remark. The initial random group element c_0 is often called an *initialization vector*.

It is possible to prove a precise variant of the following statement. Its proof is out of scope for this note.

Security claim (informal) 1. *A secure block cipher used in CBC mode provides confidentiality against eavesdroppers.*

E *Exercise* 28. Consider the attacks discussed in previous sections. Explain why they fail against a secure block cipher used in CBC mode.

E *Exercise* 29. CBC mode is insecure if the initialization vector c_0 is predictable. Suppose the group G is $\mathbb{Z}^+_{2^{128}}$. We play the following game.

1. You get an encryption $c^* = c_0^* c_1^*$ of a plaintext $m^* \in \mathbb{Z}^+_{2^{128}}$.
2. You choose $m \in \mathbb{Z}^+_{2^{128}}$.
3. You are given an encryption c of $m + \Delta$, $\Delta \in \{0, 1\}$, where the initial random element c_0 was not chosen at random, but rather as c_1^*. That is, $c = c_0 c_1$ with $c_0 = c_1^*$.

Show how you can play this game and be able to determine Δ by choosing m carefully.

3.7 Vigenère Cipher

Frequency analysis worked very well against the substitution cipher from Section 3.3. One approach to preventing frequency analysis might be to encrypt different plaintext letters with different substitution ciphers.

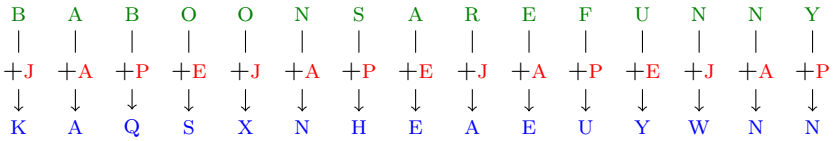

Figure 6: Example of Vigenère cipher encryption with the key $k = \text{JAPE}$.

The idea is that the frequencies produced by the encryption will be the average of the frequencies produced by the different substitution ciphers, which should tend towards a uniform distribution, thereby preventing frequency analysis.

Again, we let our alphabet be a group G, such as $\mathbb{Z}_{26}^{+}$.

The plaintext m is a sequence of elements $m_1 m_2 \dots m_L$ from the group G. The key is a sequence of elements $k_1, k_2, k_3, \dots, k_l$ from G. We encrypt the message elementwise with the formula

$$c_i = m_i + k_j, \text{ where } 1 \leq i \leq L,\ 1 \leq j \leq l \text{ and } j \equiv i \pmod{l}.$$

The ciphertext c is the sequence of elements $c_1 c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 c_2 \dots c_L$, we compute the ith plaintext element using the formula

$$m_i = c_i - k_j, \text{ where } 1 \leq i \leq L,\ 1 \leq j \leq l \text{ and } j \equiv i \pmod{l}.$$

E *Exercise* 30. The above is an informal description. Write down carefully what the three sets $\mathcal{K}, \mathcal{P}, \mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

E *Exercise (for groups)* 31. Choose a key for the Vigenère cipher and encrypt some message. Give the ciphertext along with sufficient known plaintext to someone else in the group and have them decrypt it without knowing the key.

3.7.1 Attack: Frequency Analysis II

There is an easy attack against the Vigenère cipher if we have known plaintext. If we subtract the known plaintext from the corresponding ciphertext, we get the key repeated over and over. However, there are stronger attacks based on frequency analysis.

We begin with an English text and create a subsequence of letters by starting at the ith letter and then adding every lth letter. It so happens that such subsequences tend to have the same frequency distribution as the entire text.

E *Exercise* 32. In Exercise 13, you gathered a collection of English texts. Extract subsequences as above and compute the frequency distributions. Compare these distributions to the frequency distribution of the whole text, and estimate how long subsequences must be before we can expect to identify with reasonable certainty (a) E, (b) the five most frequent letters, and (c) the ten most frequent letters.

What happens to such subsequences when we encrypt the text with the Vigenère cipher using a key of length l? The letters in the subsequence are encrypted by adding the same letter from the key to it. In other words, the subsequence is encrypted using a shift cipher.

Earlier, we attacked the shift cipher by exhaustive search, but recognizing subsequences of English text is more difficult than recognizing English text. A better approach is to use frequency analysis. We know that E will likely be the most common letter in the plaintext subsequence, which corresponds to the most common letter in the ciphertext subsequence.

To recover a key of length l, all we have to do is run l frequency analysis attacks against the shift cipher.

E *Exercise (for groups)* 33. Choose a key for the Vigenère cipher and encrypt some sufficiently long message. Give the ciphertext along with key length to someone else in the group and have them decrypt it without knowing the key.

There is one minor problem: The key length may vary and we do not know it. The simplest approach is to try every possible length, beginning with $l = 1$. When we have the wrong key length, the attack will fail to produce a sensible decryption.

If everything has to be done by hand, there are faster ways to determine the key length. The oldest method relies on repetitions in the plaintext affecting the ciphertext. A newer method uses the so-called *index of coincidence*.

E *Exercise (for groups)* 34. Choose a key for the Vigenère cipher and encrypt some sufficiently long message. Give the ciphertext to someone else in the group and have them use the index of coincidence to determine the key length.

3.8 One Time Pad

Suppose the key for the Vigenère cipher is completely random, that is, each key letter is sampled from the uniform distribution and each letter is independent of the other letters. What happens if the key is at least as long as the message and used for only one message? When used like this, the cipher is known as the *one time pad*.

Again, our alphabet is a group G, such as $\mathbb{Z}_{26}^{+}$.

The (single) plaintext m is a sequence of L group elements $m_1 m_2 \dots m_L \in G^L$. The key k is a sequence of L group elements $k_1 k_2 \dots k_L \in G^L$. We encrypt the message elementwise using the formula

$$c_i = m_i + k_i, \text{ where } 1 \leq i \leq L.$$

The ciphertext c is the sequence of L group elements $c_1 c_2 \dots c_L$.

To decrypt a ciphertext $c = c_1 c_2 \dots c_L$, we compute the ith plaintext element using the formula

$$m_i = c_i - k_i, \text{ where } 1 \leq i \leq L.$$

E *Exercise* 35. The above is an informal description. Write down carefully what the three sets $\mathcal{K}, \mathcal{P}, \mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

We shall prove that Eve's knowledge about the message after she saw the ciphertext, is the same as the knowledge she had before she saw the ciphertext.

Even before Alice has sent her ciphertext, Eve has some information about what Alice's plaintext will be. We can model this information as a random variable M, which means that from Eve's point of view, we assign a probability for Alice's plaintext being a specific message m. We denote this probability by $\Pr[M = m]$.

Likewise, we model Eve's information about the ciphertext as a random variable C. Again, from Eve's point of view, we can now assign a probability to the likelihood of seeing a given ciphertext. We denote that probability by $\Pr[C = c]$. We can also consider, from Eve's point of view, the conditional probability of seeing a particular ciphertext, given a particular plaintext. We denote that conditional probability by $\Pr[C = c \mid M = m]$.

Once Eve has seen the ciphertext, she has perfect knowledge of it, so there is no longer any uncertainty. Now, however, her information about the message may have changed. We denote the probability of Alice's message being a specific message conditioned on the observed ciphertext by $\Pr[M = m \mid C = c]$.

Theorem 2. *If the above scheme has been used to encrypt exactly one message, then*

$$\Pr[M = m_1 m_2 \dots m_L \mid C = c_1 c_2 \dots c_L] = \Pr[M = m_1 m_2 \dots m_L].$$

Proof. The assumption on the key means that from Eve's point of view before seeing the ciphertext, the key letters are uniformly and independently distributed, which is expressed as the statement

$$\Pr[K = k_1 k_2 \dots k_L] = |G|^{-L}.$$

Again, K is a random variable describing Eve's knowledge about the key.

We first compute the probabilities

$$\begin{aligned}\Pr[C = c_1 c_2 \dots c_L \mid M = m_1 m_2 \dots m_L] \\ = \Pr[K = (c_1 - m_1)(c_2 - m_2) \dots (c_L - m_L)] = |G|^{-L}\end{aligned}$$

and

$$\begin{aligned}\Pr[C = c] &= \sum_m \Pr[C = c \mid M = m] \Pr[M = m] \\ &= |G|^{-L} \sum_m \Pr[M = m] = |G|^{-L}.\end{aligned}$$

Then we compute the a posteriori probability

$$\begin{aligned}\Pr[M = m \mid C = c] &= \frac{\Pr[M = m \wedge C = c]}{\Pr[C = c]} \\ &= \frac{\Pr[C = c \mid M = m] \Pr[M = m]}{\Pr[C = c]} \\ &= \Pr[M = m],\end{aligned}$$

which completes the proof. □

The a posteriori probability is the same as the a priori probability, which means that Eve has learned nothing new by observing the ciphertext. We have proved that the one time pad provides confidentiality against eavesdroppers.

Security claim 3. *If the key is used to encrypt only once, the one time pad provides confidentiality against eavesdroppers.*

Note that this claim is *unconditional*, unlike the corresponding claim for CBC mode in Section 3.6, which is conditional on the use of a secure block cipher. This is good. Unfortunately, we must note that the one time pad is impractical in almost every application.

E *Exercise (for groups)* 36. The one time pad is not secure if the key is used more than once. Choose a key of sufficient length for the one time pad and encrypt two different messages using the same key. Give the ciphertext along with part of one message to someone else in the group and have them decrypt it without knowing the key. To ease decryption, the messages should consist of mostly long words.

3.9 Stream Ciphers

There are many possible definitions of stream ciphers, but we shall consider only the notion of *synchronious* or *additive* stream ciphers. The idea is that a *key stream generator* expands a key and an *initialization vector* into something that looks like a key for the one time pad, which is then used to encrypt the message.

Definition 6. A *key stream generator* is a function $f : \mathcal{K} \times \mathcal{I} \to G^N$.

Definition (informal) 7. A key stream generator is *secure* if it is hard to distinguish the function values $f(k, iv_1), f(k, iv_2), \ldots, f(k, iv_n)$ from random values when k has been chosen uniformly at random from $\mathcal{K}$ and the values $iv_1, iv_2, \ldots, iv_n$ have been chosen uniformly at random from $\mathcal{I}$.

Again, our alphabet is a group G, such as $\mathbb{Z}_{26}^+$.

The plaintext m is a sequence of group elements $m_1 m_2 \ldots m_L$ of length $L \leq N$. The key k is an element in $\mathcal{K}$. We encrypt the message by first choosing iv uniformly at random from $\mathcal{I}$, then computing the L first elements $z_1 z_2 \ldots z_L$ of $f(k, iv) = z_1 z_2 \ldots z_N$. We encrypt the plaintext elementwise using the formula

$$w_i = m_i + z_i, \text{ where } 1 \leq i \leq L.$$

The ciphertext c is the pair $(iv, w_1 w_2 \ldots w_L)$.

To decrypt a ciphertext $c = (iv, w_1 w_2 \ldots w_L)$, we first compute the L first elements $z_1 z_2 \ldots z_L$ of $f(k, iv)$ and then compute the ith plaintext element using the formula

$$m_i = w_i - z_i, \text{ where } 1 \leq i \leq L.$$

E *Exercise* 37. The above is an informal description of a stream cipher based on a key stream generator. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{C}$ and the two algorithms $\mathcal{E}$ and $\mathcal{D}$ are. Show that they constitute a symmetric cryptosystems.

It is possible to prove a precise variant of the following statement. Its proof is out of scope for this note.

Security claim (informal) 4. *A stream cipher using a secure key stream generator provides confidentiality against eavesdroppers.*

3.9.1 Key Stream Generators from Block Ciphers

Let $\pi, \pi^{-1} : \mathcal{K} \times G \to G$ be a block cipher. Suppose the set $\mathcal{I} \times \{1, 2, \ldots, N\}$ is a subset of the set of group elements of G. We shall use this block cipher to construct two key stream generators, $f_{\mathrm{OFB}/\pi} : \mathcal{K} \times G \to G^N$ and $f_{\mathrm{CTR}/\pi} : \mathcal{K} \times \mathcal{I} \to G^N$.

For any $iv \in G$ and $k \in \mathcal{K}$, let

$$z_1 = \pi(k, iv) \text{ and } z_i = \pi(k, z_{i-1}), \text{ where } 2 \leq i \leq N.$$

Then $f_{\mathrm{OFB}/\pi}(k, iv) = z_1 z_2 \ldots z_N$.

For any $iv \in \mathcal{I}$ and $k \in \mathcal{K}$, let

$$z_i = \pi(k, (iv, i)), \text{ where } 1 \leq i \leq N.$$

Then $f_{\mathrm{CTR}/\pi}(k, iv) = z_1 z_2 \ldots z_N$.

It is possible to prove a precise variant of the following statement. Its proof is out of scope for this note.

Security claim (informal) 5. *Output feedback mode $f_{\mathrm{OFB}/\pi}$ and counter mode $f_{\mathrm{CTR}/\pi}$ using a secure block cipher are secure key stream generators.*

From a practical point of view, counter mode is very easy to parallelize and can therefore be made very fast. Output feedback mode is inherently unparallelizable.

4 Integrity

In this section we shall consider the situation where Eve controls the communications channel between Alice and Bob. Eve's goal is to tamper with the messages sent by Alice so that Bob receives a different message, *without noticing*. For the moment, we shall not care about confidentiality.

The main tool we shall use is the *message authentication code*, where Alice adds an *authentication tag* to her message that allows Bob the verify that the message is unchanged.

Definition 8. A *message authentication code* is a function $\mu : \mathcal{K} \times \mathcal{P} \to \mathcal{T}$.

Definition (informal) 9. A message authentication code is *secure* if it is hard to guess the function value $\mu(k, m)$ for any m, even after seeing the values $\mu(k, m_1), \mu(k, m_2), \ldots, \mu(k, m$ for any $m_1, m_2, \ldots, m_n \in \mathcal{P}$. Here, k has been chosen uniformly at random from $\mathcal{K}$, and $m \neq m_i$ for $i = 1, 2, \ldots, n$.

When Alice wants to send a message m to Bob, she sends the pair $(m, \mu(k, m))$. When Bob receives the pair (m', t), he checks that $\mu(k, m') = t$. If it is, he accepts that the message came from Alice. Otherwise, he discards the message.

Security claim (informal) 6. *A secure message authentication code used as above provides integrity.*

Note that nothing prevents Eve from *replaying* old messages by sending them to Bob. Defending against such attacks is out of scope for this note.

One of the most popular MAC constructions – HMAC – is based on so-called *hash functions*, which are out of scope for this note. We shall discuss two constructions of message authentication codes.

4.1 Polynomial evaluation MACs

We begin our discussion with a *one-time* polynomial-evaluation MAC. This MAC is insecure if it is used on more than one message. Such a MAC is impractical, so we shall also discuss how a block cipher can be used to make a more practical variant.

Our alphabet is a finite field $\mathbb{F}$, say a field with a prime p number of elements.

The plaintext m is a sequence of field elements $m_1 m_2 \dots m_L$. The key (k_1, k_2) is a pair of field elements. The function μ_{OTPE} is computed as

$$\mu_{\text{OTPE}}(k_1, k_2, m) = k_2 + \sum_{i=1}^{L} m_i k_1^i + k_1^{L+1}. \tag{5}$$

E *Exercise* 38. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{T}$ and the function μ_{OTPE} are. Show that μ_{OTPE} is a message authentication code.

We shall first prove the following statement.

Theorem 7. *Let m, m' be two messages of length at most L. Let $t, t' \in \mathbb{F}$. The probability that $\mu_{\text{OTPE}}(k_1, k_2, m') = t'$ given that $\mu_{\text{OTPE}}(k_1, k_2, m) = t$ is at most $(L+1)/p$.*

Proof. For simplicity, we shall assume that both messages have length exactly L. We want to compute the probability

$$\Pr[\mu_{\text{OTPE}}(k_1, k_2, m') = t' \mid \mu_{\text{OTPE}}(k_1, k_2, m) = t].$$

We can do that by computing how many pairs (k_1, k_2) satisfy

$$t = k_2 + k_1^{L+1} + \sum_{i=1}^{L} m_i k_1^i \tag{6}$$

and how many also satisfy

$$t' = k_2 + k_1^{L+1} + \sum_{i=1}^{L} m'_i k_1^i. \tag{7}$$

It is clear that for every value of k_1, there is exactly one value of k_2 that satisfies (6), so there are exactly p pairs that we need to consider.

Combining the two equations, we get that k_1 must satisfy the equation

$$t' - t = \sum_{i=1}^{L} (m'_i - m_i) k_1^i.$$

A solution to this equation is a zero of a polynomial equation of degree at most L, which means that there are at most L solutions.

The conclusion is that out of p possible keys (k_1, k_2) satisfying (6), there are at most L pairs that also satisfy (7). It now follows that we have a bound on the probability.

When the messages have different length, the exact same argument applies, but the polynomial we consider has degree at most $L + 1$. □

Note that this means that when Alice sends a single message m to Bob with the tag $t = \mu_{\text{OTPE}}(k_1, k_2, m)$, and Bob receives the message $m' \neq m$ and the tag t', the probability that Bob accepts that message as coming from Alice is at most $(L + 1)/p$. This proves the following claim.

Security claim 8. *If the key is used to create a MAC tag only once, the one-time polynomial evaluation MAC is a secure message authentication code.*

E *Exercise* 39. Show that the scheme is not secure if we

a. replace k_1^i with k_1^{i-1} in the sum in (5);

b. remove the term k_2 from (5); or

c. remove the term k_1^{L+1} from (5).

The above construction is just a one-time MAC, which is usually impractical. However, if we use a block cipher $\pi, \pi^{-1} : \mathcal{K}' \times S \to S$ with $\mathbb{F} \subseteq S$, we can construct an alternative polynomial-evaluation MAC that can be used more than once, using

$$\mu_{\text{PE}/\pi}(k_1, k_2, m) = \pi(k_2, k_1^{L+1} + \sum_{i=1}^{L} m_i k_1^i).$$

E *Exercise* 40. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{T}$ and the function $\mu_{\text{PE}/\pi}$ are. Show that $\mu_{\text{PE}/\pi}$ is a message authentication code.

4.2 Block-cipher-based MACs

One simple MAC based on a block cipher $\pi, \pi^{-1} : \mathcal{K} \times G \to G$ is CBC-MAC, which is somewhat similar to Cipherblock Chaining mode from Section 3.6.

Fix some element $t_0 \in G$. The plaintext m is a sequence of group elements $m_1 m_2 \dots m_L$. The key k is an element of $\mathcal{K}$. Let

$$t_i = \pi(k, t_{i-1} + m_i), \text{ where } 1 \leq i \leq L.$$

Then $\mu_{\text{CBC}}(k, m) = t_L$.

E *Exercise* 41. The above is an informal description. Write down carefully what the three sets $\mathcal{K}$, $\mathcal{P}$, $\mathcal{T}$ and the function μ_{CBC} are. Show that μ_{CBC} is a message authentication code.

This MAC is only secure when restricted to messages of fixed length. If messages of different lengths are allowed, it is not secure.

E *Exercise* 42. Find an attack against this MAC when messages of different length are allowed.

There are many secure variants of this MAC. One variant is based on a block cipher $\pi, \pi^{-1} : \mathcal{K} \times \mathbb{F} \to \mathbb{F}$ over a field $\mathbb{F}$. The idea is to modify the final plaintext block with an unpredictable value.

Fix a non-zero element $g \in \mathbb{F}$. The plaintext m is a sequence of field elements $m_1 m_2 \dots m_L$. The key k is an element of $\mathcal{K}$. Let $t_0 = 0$, $h = \pi(k, 0)$ and

$$t_i = \pi(k, t_{i-1} + m_i), \text{ where } 1 \leq i \leq L - 1.$$

Then $\mu_{\text{CBC}'}(k, m) = \pi(k, hg + t_{L-1} + m_L)$.

Security claim (informal) 9. *The modified CBC-MAC using a secure block cipher is a secure MAC.*

5 Confidentiality and Integrity

Finally, we consider the situation where Eve controls the communications channel between Alice and Bob. Eve's goal is both to read Alice's messages to Bob and tamper with them.

We shall combine the secure cryptosystems we saw in Section 3 with the message authentication codes we saw in Section 4 into cryptosystems that provide both confidentiality and integrity.

Definition (informal) 10. A symmetric cryptosystem provides *integrity* if it is hard to create a ciphertext that decrypts to anything other than $\perp$ without knowledge of the key.

First we consider a general principle in cryptography: *Never use the same key for two different things.* This means that we should use different keys for the cryptosystem and the MAC when we combine them. That leaves us with three obvious ways of combining a cryptosystem with a MAC:

Encrypt-then-MAC First encrypt the message, then MAC the ciphertext:

$$w = \mathcal{E}(k_e, m);\ t = \mu(k_m, w);\ c = (w, t).$$

MAC-then-encrypt MAC the message, then encrypt the message and the tag:

$$t = \mu(k_m, m);\ c = \mathcal{E}(k_e, (m, t)).$$

Encrypt-and-MAC Encrypt the message and attach a MAC tag of the message:

$$w = \mathcal{E}(k_e, m);\ t = \mu(k_m, m);\ c = (w, t).$$

In all three cases, the decryption algorithm verifies the MAC and if the verification fails, the output of the decryption algorithm is $\perp$.

Encrypt-and-MAC is in general insecure. If you encrypt the same message twice, you will always get the same tag, something that Eve will notice. Therefore, it fails confidentiality.

MAC-then-encrypt is often secure, but there are special cases where it is not secure. In particular, such schemes will often fail when combined with padding schemes.

Encrypt-then-MAC is always secure, which means that this is the best choice.

Security claim (informal) 10. *A cryptosystem that provides confidentiality against eavesdroppers and a secure message authentication code combined using Encrypt-then-MAC provides both confidentiality and integrity.*

Note that nothing prevents Eve from *replaying* old ciphertexts by sending them to Bob. Defending against such attacks is out of scope for this note.