M⊕dern Cryptogr🔒phy

or: The Unofficial Notes on the Georgia Institute
of Technology's **CS6260**: *Applied Cryptography*

George Kudrayvtsev
george.k@gatech.edu

Last Updated: January 12, 2020

Preface

> *I read that Teddy Roosevelt once said, "Do what you can with what you have where you are." Of course, I doubt he was in the tub when he said that.*
>
> — Bill Watterson, *Calvin and Hobbes*

Before we begin to dive into all things cryptography, I'll enumerate a few things I do in this notebook to elaborate on concepts:

- An item that is **highlighted like this** is a "term;" this is some vocabulary that will be used and repeated regularly in subsequent sections. I try to cross-reference these any time they come up again to link back to its first defined usage; most mentions are available in the Index.
- An item that is **highlighted like this** is a "mathematical property;" such properties are often used in subsequent sections and their understanding is assumed there.
- An item in a maroon box, like...

Boxes: A Rigorous Approach

... this often represents fun and interesting asides or examples that pertain to the material being discussed. They are largely optional, but should be interesting to read and have value, even if it's not immediately rewarding.

- An item in a **blue box**, like...

Quick Maffs: Proving That the Box Exists

... this is a mathematical aside; I only write these if I need to dive deeper into a concept that's mentioned in lecture. This could be proofs, examples, or just a more thorough explanation of something

that might've been "assumed knowledge" in the text.

- An item in a green box, like...

> **Definition 0.1: Example**
>
> ... this is an important cryptographic definition. It will often be accompanied by a highlighted **term** and dive into it with some mathematical rigor.

I also sometimes include margin notes like the one here (which just links back here) that reference content sources so you can easily explore the concepts further. Linky

Introduction

> *Cryptography is hard.*
>
> — Anonymous

The purpose of a cryptographic scheme falls into three very distinct categories. A common metaphor used to explain these concepts is a legal document.

- **confidentiality** ensures content *secrecy*—that it can't be read without knowledge of some secret. In our example, this would be like writing the document in a language nobody except you and your recipient understand.
- **authenticity** guarantees content *authorship*—that its author can be irrefutably proven. In our example, this is like signing the original document in pen (assuming, of course, your signature was impossible to forge).
- **integrity** guarantees content *immutability*—that it has not been changed. In our example, this could be that you get an emailed copy of the signed document to ensure that its language cannot be changed post-signing.

Note that even though all three of these properties can go hand-in-hand, they are not mutually constitutive. You can have any of them without the others: you can just get a copy of an unsigned document sent to you in plain English to ensure its integrity later down the line.

Analysing any proposed protocol, handshake, or other cryptographic exchange through the lens of each of these principles will be enlightening. Not every scheme is intended to guarantee all three of them, and different methods are often combined to achieve more than one of these properties. In fact, cases in which only one of the three properties are necessary occur all the time. It's important to not make a cryptographic scheme more complicated than it needs to be to achieve a given purpose: complexity breeds bugs.

Trivia: **Cryptography's Common Cast of Characters**

It's really useful to anthropomorphize our discussion of the mathematical intricacies in cryptography. For that, we use a cast of characters whose names give us immediate insight into what we should expect from them.

- **Alice** and **Bob** are the most common sender-recipient pairing. They are generally acting in good faith and aren't trying to break the cryptographic scheme in question. If a third member is necessary, **Carol** will enter the fray (for consistency of the allusion to Lewis Carroll's *Alice in Wonderland* ☺).
- **Eve** and **Mallory** are typically the two members trying to break the scheme. Eve is a *passive* attacker (short for eavesdropper) that merely observes messages between Alice and Bob, whereas malicious Mallory is an *active* attacker who can capture, modify, and inject her own messages into exchanges between other members.

You can check out the Wikipedia article on the topic for more historic trivia and the full cast of characters.

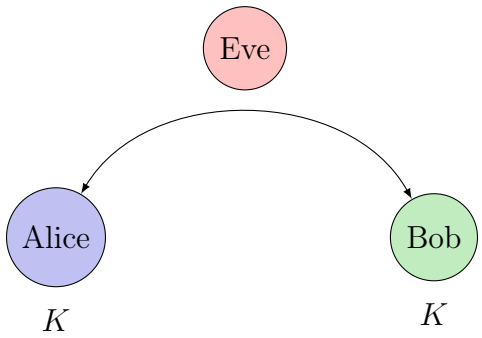
1.1 Overview

There are a number of different categories of cryptography, each of which applies in different settings.

$$\underbrace{\text{crypto}}_{\text{secret}}\underbrace{\text{graphy}}_{\text{writing}}$$

1.1.1 Symmetric Cryptography

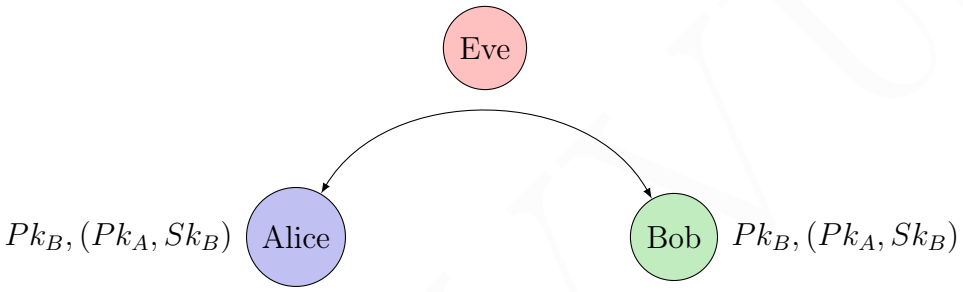
The notion of **symmetric key**s comes from the fact that both the sender and receiver of encrypted information share the same secret key, K. This secret is the only thing that separates a viable receiver from an attacker.



Symmetric key algorithms are often very efficient and supported by hardware, but their fatal flaw lies in **key distribution**. If two parties need to share a secret without anyone else knowing, how do they get it to each other without already having a secure channel?

1.1.2 Asymmetric Cryptography

Asymmetric cryptography is built to solve this problem. Here, secrets are only known to one party; instead, a key (Pk, Sk) is broken into two mathematically-linked components. There is a **public key** that is broadcasted to the world, and a **private key** that must is kept secret.



Asymmetric cryptography is often used to mutually establish a secret key (without revealing it!) for use with symmetric key algorithms, since those are faster. It's a little counterintuitive: two strangers can "meet" and "speak" publicly, yet walk away having established a mutual secret.

The big assumption (which we'll attack later) is that there's a trusted public repository that lists everyone's authentic public keys; without this assumption, we're back at the key distribution problem.

1.1.3 Cryptanalysis

An important part of cryptography is *evaluation*: how good is a given scheme? There are a number of approaches that can help us attack a "secure" protocol:

trial-and-error The most straightforward approach is to identify a possible attack vector and try it out. If it results in a successful attack, the protocol needs to be patched, and you can try finding another vector. If it doesn't seem possible to find an attack vector, that does not make the protocol secure. And yet, how do we find another vulnerability...?

provable security The best way to ensure that a protocol cannot be exploited is to prove it mathematically. It typically relies on proof by contradiction: if an attack is found, then some underlying mathematical assumptions must be

false. A good example of this is the factoring problem that underlies most asymmetric cryptography: we've demonstrated time and again the difficulty in factoring large prime numbers. If an efficient algorithm were to arise, a lot of things would go very wrong, very quickly.

Symmetric Cryptography

> *How long do you want these messages to remain secret? I want them to remain secret for as long as men are capable of evil.*
>
> — Neal Stephenson, *Cryptonomicon*

As mentioned in the Introduction, algorithms in symmetric cryptography rely on all members having a shared secret. Let's cover the basic notation we'll be using to describe our schemes and then dive into some.

2.1 Notation & Syntax

For consistency, we'll need common conventions when referring to cryptographic primitives in a scheme.

A well-described symmetric encryption scheme covers the following:

- a **message space**, denoted as the $\mathcal{M}$sgSp or $\mathcal{M}$ for short, describes the set of things which can be encrypted. This is typically unrestricted and (especially in the context of the digital world) includes anything that can be encoded as bytes.
- a **key generation algorithm**, $\mathcal{K}$, or the key space spanned by that algorithm, $\mathcal{K}$eySp, describes the set of possible keys for the scheme and how they are created. The space typically restricts its members to a specific length.
- a **encryption algorithm** and its corresponding **decryption algorithm** $(\mathcal{E}, \mathcal{D})$ describe how a message $m \in \mathcal{M}$ is converted to and from ciphertext. We will use the notation $\mathcal{E}(K, M)$ and $\mathcal{E}_K(M)$ interchangeably to indicate encrypting M using the key K (and similarly for $\mathcal{D}$).

A well-formed scheme *must* allow all valid messages to be en/decrypted. Formally, this means:

$$\forall m \in \mathcal{M}\text{sgSp},\ \forall k \in \mathcal{K}\text{eySp} : \mathcal{D}(k, \mathcal{E}(k, m)) = m$$

An encryption scheme defines the message space and three algorithms: $(\mathcal{M}\text{sgSp}, \mathcal{E}, \mathcal{D}, \mathcal{K})$. The key generation algorithm often just pulls a random n-bit string from the entire $\{0,1\}^n$ bit space. The encryption algorithm is often randomized (taking random input in addition to (K, M)) and stateful. We'll see deeper examples of all of these shortly.

2.2 One-Time Pads

A **one-time pad** (or OTP) is a very basic and simple way to ensure absolutely perfect encryption, and it hinges on a fundamental binary operator that will be at the heart of many of our symmetric encryption schemes in the future: **exclusive-or**.

Messages are encrypted with an equally-long sequence of random bits using XOR. With our notation, one-time pads can be described as such:

- the key space is all n-bit strings: $\mathcal{K}\text{eySp} = \{0,1\}^n$
- the message space is the same: $\mathcal{M}\text{sgSp} = \{0,1\}^n$
- encryption and decryption are just XOR:

$$\mathcal{E}(K, M) = M \oplus K$$
$$\mathcal{D}(K, C) = C \oplus K$$

Can we be a little more specific with this notion of "perfect encryption"? Intuitively, a secure encryption scheme should reveal nothing to adversaries who have access to the ciphertext. Formally, this notion is called being Shannon-secure (and is also referred to as **perfect security**): the probability of a ciphertext occurring should be equal for any two messages.

> Definition 2.1: **Shannon-secure**
>
> An encryption scheme, $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, is **Shannon-secure** if:
>
> $$\forall m_1, m_2 \in \mathcal{M}\text{sgSp}, \forall C : \\ \Pr[\mathcal{E}(K, m_1) = C] = \Pr[\mathcal{E}(K, m_2) = C] \tag{2.1}$$
>
> That is, the probability of a ciphertext C must be equally-likely for any two messages that are run through $\mathcal{E}$.

Are one-time pads Shannon-secure? Yes, thanks to XOR.

2.2.1 The Beauty of XOR

XOR is the only primitive binary operator that outputs 1s and 0s with the same frequency, and that's what makes it the perfect operation for achieving unpredictable ciphertexts.

Given a single bit, what's the probability of the input bit?

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Table 2.1: The truth table for XOR.

Suppose you have some $c = 0$ (where $c \in \{0,1\}^1$); what was the input bit m? Well it could've been 1 and been XOR'd with 1 OR it could've been 0 and been XOR'd with 0... Knowing c gives us no new information about the input: our guess is still as good as random chance ($\frac{1}{2} = 50\%$).

Now suppose you know that $c = 1$; are your odds any better? In this case, m could've been 1 and been XOR'd with 0 OR it could've been 0 and XOR'd with 1... Again, we can't do better than random chance.

By the very definition of being Shannon-secure, if we (as the attacker) can't do better than random chance when given a ciphertext, the scheme is perfectly secure.

2.2.2 Proving Security

So we did a bit of a hand-wavy proof to show that XOR is Shannon-secure, but let's be a little more formal in proving that OTPs are perfect as well.

Theorem 2.1. *One-time pads are a perfect-security encryption scheme.*

Proof. We start by fixing an arbitrary n-bit ciphertext: $C \in \{0,1\}^n$. We also choose a fixed n-bit message, $m \in \mathcal{M}\text{sgSp}$. Then, what's the probability that a randomly-generated key $k \in \mathcal{K}\text{eySp}$ will encrypt that message to be that ciphertext? Namely, what is

$$\Pr[\mathcal{E}(K, m) = C]$$

for our **fixed** m and C? In other words, how many keys can turn m into C?

Well, by the definition of the OTP, we know that this can only be true for a single key: $K = m \oplus C$. Well, since every bit counts, and the probability of a single bit in

the key being "right" is $1/2$:

$$\begin{aligned}\Pr[\mathcal{E}(K,m)=C] &= \Pr[K = m \oplus C] \\ &= \underbrace{\frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2}\cdot\ldots}_{n \text{ times}} \\ &= \frac{1}{2^n}\end{aligned}$$

Note that this is true $\forall m \in \mathcal{M}\text{sgSp}$, which fulfills the requirement for perfect security! Every message is equally likely to result in a particular ciphertext. □

The problem with OTPs is that keys can only be used once. If we're going to go through the trouble of securely distributing OTPs,[1] we could just exchange the messages themselves at that point in time...

Let's look at what happens when we use the same key across two messages. From the scheme itself, we know that $C_i = K \oplus M_i$. Well if we also have $C_j = K \oplus M_j$, then:

$$\begin{aligned}C_i \oplus C_j &= (K \oplus M_i) \oplus (K \oplus M_j) \\ &= (K \oplus K) \oplus (M_i \oplus M_j) && \text{XOR is associative} \\ &= M_i \oplus M_j && a \oplus a = 0\end{aligned}$$

Though this may seem like insignificant information, it actually can reveal quite a bit about the inputs, and eventually the entire key if it's reused enough times.

Theorem 2.2. *If a scheme is Shannon-secure, then the key space cannot be smaller than the message space. That is,*

$$|\mathcal{K}eySp| \geq |\mathcal{M}sgSp|$$

Proof. We are given an encryption scheme $\mathcal{E}$ that is supposedly perfectly-secure. So we start by fixing a ciphertext with a specific key, ($K_1 \in \mathcal{K}\text{eySp}$ and plaintext message, $m_1 \in \mathcal{M}\text{sgSp}$):

$$C = \mathcal{E}(K_1, m_1)$$

We know for a fact, then, that at least one key exists that can craft C; thus if we pick a key $K \in \mathcal{K}\text{eySp}$ *at random*, there's a non-zero probability that we'd get C again:

$$\Pr[\mathcal{E}(K, m_1) = C] > 0$$

[1] One could envision a literal physical pad in which each page contained a unique bitstring; if two people shared a copy of these pads, they could communicate securely until the bits were exhausted (or someone else found the pad). Of course, if either of them lost track of where they were in the pad, everything would be gibberish from then-on...

Suppose then there is a message $m_2 \in \mathcal{M}\text{sgSp}$ which we can *never* get from decrypting C:

$$\Pr[\mathcal{D}(K, C) = m_2] = 0 \qquad \forall K \in \mathcal{K}\text{eySp}$$

By the correctness requirement of a valid encryption scheme, if a message can never be decrypted from a ciphertext, neither should that ciphertext result from an encryption of the message:

$$\Pr[\mathcal{E}(K, M) = C] = 0 \qquad \forall K \in \mathcal{K}\text{eySp}$$

However, that violates Shannon-secrecy, in which the probability of a ciphertext resulting from the encryption of *any* two messages is equal; that's not the case here:

$$\Pr[\mathcal{E}(K, M_1) = C] \neq \Pr[\mathcal{E}(K, M_2) = C]$$

Thus, our assumption is wrong: m_2 cannot exist! Meaning there *must* be some $K_2 \in \mathcal{K}\text{eySp}$ that decrypts C: $\mathcal{D}(K_2, C) = M_2$. Thus, it must be the case that there are as many keys as there are messages:

$$|\mathcal{K}\text{eySp}| \geq |\mathcal{M}\text{sgSp}|$$

□

Ideally, we'd like to encrypt long messages using short keys, yet this theorem shows that we cannot be perfectly-secure if we do so. Does that indicate the end of this chapter? Thankfully not. If we operate under the assumption that our adversaries are computationally-bounded, it's okay to relax the security requirement and make breaking our encryption schemes very, *very* unlikely. Though we won't have *perfect* secrecy, we can still do extremely well.

We will create cryptographic schemes that are computationally-secure under **Kerckhoff's principle**, which effectively states that *everything* about a scheme should be publicly-available except for the secret key(s).

2.3 Block Ciphers

These are the building blocks of symmetric cryptography: a **block cipher** is a tool for encrypting short strings. Well-known examples include AES and DES.

Definition 2.2: **Block Cipher**

Formally, a block cipher is a function family that maps from a k-bit key and

an n-bit input string to an n-bit output string:

$$\mathcal{E} : \{0,1\}^k \times \{0,1\}^n \mapsto \{0,1\}^n$$

Additionally, $\forall K \in \{0,1\}^k$, $\mathcal{E}_K(\cdot)$ is a permutation on $\{0,1\}^n$. This means its inverse is well-defined; we denote it either as $\mathcal{E}_K^{-1}(\cdot)$ or the much more intuitive $\mathcal{D}_K(\cdot)$:

$$\forall M, C \in \{0,1\}^n : \quad \begin{aligned} \mathcal{E}_K(\mathcal{D}_K(C)) &= C \\ \mathcal{D}_K(\mathcal{E}_K(M)) &= M \end{aligned}$$

In a similar vein, ciphertexts are unique, so $\forall C \in \{0,1\}^n$, there exists a *single* M such that $C = \mathcal{E}_K(M)$.

Math Review: Functions

A function is **one-to-one** if every input value maps to a unique output value. In other words, it's when no two inputs map to the same output.

A function is **onto** if all of the elements in the range have a corresponding input. That is, $\forall y \, \exists x$ such that $f(x) = y$.

A function is **bijective** if it is both one-to-one and onto; it's a **permutation** if it maps from a set onto itself. In our case, the set in question is the set of all n-length bitstrings: $\{0,1\}^n$.

2.3.1 Modes of Operation

Index of Terms