Notes for Cryptography

Jing LI
pkuyplijing@gmail.com

September 12, 2016

Sincere gratitude to Professor Dan Boneh
for offering such a wonderful course.

Contents

Chapter 1

Introduction

1.1 Course Overview

The main objectives of this course are

- Learn how cryptographic primitives work;
- Learn how to use them correctly and reason about security.

By the end of the course we will be able to reason about the security of cryptography constructions and break ones that are not secure.

Cryptography is used everywhere.

Secure communication Web traffic: https. Wireless traffic: 802.11i WPA2, GSM, Bluetooth.

Encrypting files on disk EFS, TrueCrypt

Content protection(e.g. DVD, blue ray) CSS(Content Scrambling System), AACS

User authentication

Secure communication is concerned about a laptop "Alice" trying to communicate with a web server "Bob" [1]. The protocol used is HTTP, or to say SSL(Secure Sockets Layer)/TLS(Transport Layer Security). The goal is to make sure that as data travels across the network, an attacker can neither eavesdrop on or tamper the data. It consists of two parts:

- Handshake protocol: establish a shared secret key using public-key cryptography;
- Record layer: transmit data using this shared secret key, with confidentiality and integrity ensured.

[1] Alice and Bob are two nicknames that will be used in the course

Storing encrypted files on disk is actually logically the same as protecting communication: encrypting files on disk is essentially securing the communication between "Alice today" and "Alice tomorrow".

The building block of secure communication is symmetric encryption. A message m is encrypted by a cipher using a shared key k into a cipher text $c = E(k, m)$, and this cipher text is deciphered back to the message: $m = D(k, c)$. The encryption algorithm E and the decryption algorithm D are publicly known. Proprietary ciphers should not be used for security reasons.

We will discuss two use cases of symmetric encryption. A single use key is used to encrypt only one message, while a multi use key is used to encrypt multiple messages.

Cryptography is a tremendous tool serving as the basis for many security mechanisms. Nonetheless, it is not the solution to all security problems. Software bugs can cause security problems unsolvable using cryptography. Neither can cryptography prevent us from social engineering attacks. Moreover, cryptography is not reliable unless implemented and used properly. It is not something we try to invent ourselves.

1.2 What is Cryptography

As stated in the previous section, the core of cryptography consists of two steps: secret key establishment and secure communication using the secret key. But cryptography does much more than that. We will provide a few examples here.

Digital Signature

Digital signature is the analog of the signature in the physical world. In the digital world, signatures cannot be the same for different documents. Rather, digital signature is a function of the content beings signed. Simply copying the signature on one document onto another one will result in failure of verification.

Anonymous Communication

Anonymous communication over the Internet can be established with a commonly used mechanism called "mix net". Bob has no idea whom he is talking to when Alice contacts him anonymously, yet he can still send responses to Alice.

Anonymous Digital Cash

The use of digital cash is concerned with two problems: how to spend a "digital coin" without anyone knowing who I am, and how to prevent double spending. There seems to be a paradox between anonymity and security. Roughly speaking, anonymous digital cash is used in such a way that when used once, no one knows who used it, whilst once used twice, the identity of the user gets exposed.

Protocols

In a election, we may hope to design a system that reveals the winner of the election without exposing individual voting choices. In a private auction, it is ideal that only the winner and the 2^{nd} highest bid is outputted by the auction center. These are actually examples of a more general problem called secure multi-party computation. It aims at calculating a target function $f(x_i)$ without exposing individual inputs x_i.

"Crypto Magic"

Privately outsourcing computation: send an encrypted query to Google, and Google can compute on the encrypted query and send back results without knowing content of the query.

Zero knowledge proof: convince someone else that we have solved a problem without exposing the solution to him.

Cryptography is a rigorous science. Every concept that we will describe is going to follow three steps:

1. Precisely specify the threat model;
2. Propose a construction;
3. Prove that breaking the construction under the threat model is equivalent to solving an underlying hard problem.

1.3 History of Cryptography

We will provide a few historical ciphers, all of which are now badly broken.

Substitution Cipher

Cipher a piece of text by substituting each letter with another letter, i.e. encrypt messages by shuffling the alphabet. An example of substitution cipher is the Caesar cipher. It shifts the alphabet by 3 positions, i.e. $a \to d, b \to e$, etc.

There are 26! possible keys for a substitution cipher, which is a fairly large key space. However it can be easily broken with letter frequencies. Being vulnerable to the worst possible attack, namely a cipher text only attack, it should not be the choice of any serious secure communication.

Vigener Cipher

A cipher dating back to 16^{th}-century Rome. Given a key word, for example "crypto". Repeat the word enough times until reaching the length of the message, then add the long key with the message numerically to obtain the cipher text. It can also be broken with letter frequencies.

Rotor Machines

A cipher invented during the electric era, i.e. the 19^{th} century. The most famous one was the Enigma.

Data Encryption Standard

A post-WWII cipher with a key space of size 2^{56}, which was large enough then but is too small today.

1.4 Discrete Probability

The whole theory of cryptography is built upon the knowledge of discrete probability. This section serves as a quick recap of discrete probability.

Discrete probability is always defined on a finite set called the **universe**. In this course we will always use the set $U = \{0,1\}^n$, i.e. the set of all n-bit binary strings.

Definition 1. *A **probability distribution** P is a function from U to [0,1] such that* $\sum_{x \in U} P(x) = 1$.

Definition 2. *A subset of the universe $A \subseteq U$ is called an **event**. The probability of the event is* $Pr(A) = \sum_{x \in A} P(x) \in [0,1]$.

Definition 3. *A **random variable** is a function $X : U \to V$. X take values in V and defines a distribution on V.*

Definition 4. *Events A and B are **independent** if $Pr(A \wedge B) = Pr(A) \cdot Pr(B)$.*

An important property of XOR will be frequently used in this course.

Theorem 1. ***(Property of XOR)** Suppose we have a random variable Y and a uniform variable X independent from Y on $\{0,1\}^n$, then the XOR of X and Y $Z := X \oplus Y$ is a uniform variable on $\{0,1\}^n$.*

The birthday paradox is an interesting topic that we will talk about in detail later.

Theorem 2. ***(The Birthday Paradox)** Let $r_i \in U (i = 1, \ldots, n)$ be independent identically distributed variables. If $n \geq 1.2 \times |U|^{1/2}$, then*

$$Pr(\exists i \neq j \; s.t. \; r_i = r_j) \geq \frac{1}{2}.$$

Chapter 2

Stream Ciphers

First we would like to provide a precise definition for symmetric ciphers.

Definition 5. ***(Symmetric Cipher)*** *A* ***cipher*** *defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, in which $\mathcal{K}$ is the set of all possible keys, $\mathcal{M}$ is the set of all possible messages and $\mathcal{C}$ is the set of all possible cipher texts, is a pair of efficient algorithms (E, D) in which the encryption algorithm $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and the decryption algorithm $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$ satisfy the consistence equation*

$$D(k, E(k, m)) = m, \forall k \in \mathcal{K}, m \in \mathcal{M}.$$

In practice, E is often randomized, while D is always deterministic.

2.1 OTP and Perfect Secrecy

Now we will introduce our first example of a secure cipher, namely **the one time pad**. In this case, we have $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$. A key is simply a bit string as long as the message to be encrypted. The cipher text is the XOR of the key and the message, i.e.

$$c := E(k, m) = k \oplus m.$$

To decrypt a cipher text, we simply compute the XOR again, i.e.

$$m := E(k, c) = k \oplus c.$$

Since we have $D(k, E(k, m)) = k \oplus (k \oplus m) = (k \oplus k) \oplus m = 0 \oplus m = m$[1], obviously the consistence equation is satisfied. Similarly we have $k = m \oplus c$.

Now let's explain why OTP is a "good" cipher.

Definition 6. ***(Perfect Secrecy)*** *A cipher (E, D) is said to have* ***perfect secrecy*** *if $\forall m_0, m_1 \in \mathcal{M}$ with equal length and $\forall c \in \mathcal{C}$, we have*

$$Pr(E(k, m_0) = c) = Pr(E(k, m_1) = c),$$

[1] XOR is addition mod 2, thus is associative. Also provable using $a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b$.

in which k is uniform in $\mathcal{K}$: $k \xleftarrow{R} \mathcal{K}$.

This definition means that an attacker cannot tell if the message is m_0, m_1 or any other message with equal length given only the cipher text. Thus nothing can be learnt about the message from the cipher text, and the cipher is safe from CT-only attack.

Since $k = c \oplus m$ for OTP, we have $Pr(E(k, m) = c) = \frac{1}{|\mathcal{K}|}, \forall k$. Hence OTP has perfect secrecy. Nonetheless, the key has to be as long as the message, which makes OTP impractical in actual use.

Unfortunately, it can be proved that a cipher with perfect secrecy must satisfy $|\mathcal{K}| > |\mathcal{M}|$, which means the length of the key is at least as long as that of the message. So such ciphers are actually hard to use in practice.

2.2 Stream Ciphers

In this section we will try to make OTP practical. The idea is to use a pseudo-random key instead of a random key.

Definition 7. ***(PRG)*** *A **pseudo-random generator** is a function $G : \{0,1\}^s \rightarrow \{0,1\}^n$, in which $n \gg s$.*

$\{0,1\}^s$ is called the **seed space**. A PRG must be efficiently computable by a deterministic algorithm. A **stream cipher** just substitutes the random key in OTP with a pseudo-random key generated by a PRG, i.e. with a random seed $k \in \{0,1\}^s$, we have

$$c = E(k, m) = m \oplus G(k)$$
$$m = D(k, m) = c \oplus G(k)$$

Obviously a stream cipher cannot have perfect secrecy because the seed is much shorter than the message. Thus we need a different notion of security, and the security of a stream cipher depends on the PRG used.

If an attacker is able to compute all bits of $G(k)$ according to the first i bits, then the PRG is not safe from CT-only attacks. For example, an email following the SMTP protocol always starts with "from:", thus the attacker can obtain a prefix of $G(k)$ according to the cipher text. We should always use **unpredictable PRGs** as defined below.

Definition 8. ***(Predictable PRG)*** *A PRG G is said to be **predictable** if $\exists$ efficient algorithm A and $i \in [1, n-1]$ s.t.*

$$Pr\left(A\left(G(k)|_{1,\ldots,i}\right) = G(k)|_{i+1}\right) \geq \frac{1}{2} + \epsilon$$

for some non-negligible ϵ.

Definition 9. ***(Unpredictable PRG)*** *A PRG is **unpredictable** if it is not predictable.*

An example of predictable PRG is the famous **linear congruential generator**. A seed $r[0]$ is chosen randomly. Then we calculate

$$r[i] = (a \cdot r[i-1] + b) \bmod p$$

in which a, b, p are parameters. In each iteration a few bits of $r[i]$ is outputted. This PRG is easy to predict.

A variant is the $random()$ in glibc:

$$r[i] = (r[i-3] + r[i-31])\%2^{32}$$

and output $r[i] \gg 1$. $random()$ should never be used for cryptography.

A theoretical notion of negligibility is to view ϵ as a function rather than a scalar. $\epsilon : Z^+ \to R^+$ is non-negligible if $\exists d$ s.t. $\epsilon(\lambda) \geq \frac{1}{\lambda^d}$ is infinitely often, i.e. ϵ is larger than $\frac{1}{\text{polynomial of } \lambda}$ for many λ. On the contrary, ϵ is negligible if $\forall d, \exists \lambda_d$ s.t. $\epsilon(\lambda) \leq \frac{1}{\lambda^d}, \forall \lambda \geq \lambda_d$. For instance, $\frac{1}{2^\lambda}$ is negligible, whilst $\frac{1}{\lambda^{1000}}$ is non-negligible. As a trick example,

$$\epsilon(\lambda) = \begin{cases} \frac{1}{2^\lambda} & \text{for odd } \lambda \\ \frac{1}{\lambda^{1000}} & \text{for even } \lambda \end{cases}$$

is non-negligible.

2.3 Attacks on OTP and Stream Ciphers

2.3.1 Two Time Pad

A pad should not be used more than once, otherwise the encrypted messages shall be decrypted easily. Suppose two messages m_1, m_2 are encrypted using the same pad $PRG(k)$:

$$c_1 \leftarrow m_1 \oplus PRG(k)$$
$$c_2 \leftarrow m_2 \oplus PRG(k)$$

If an eavesdropper has intercepted the two cipher texts. By calculating their XOR:

$$\begin{aligned} c_1 \oplus c_2 &= (m_1 \oplus PRG(k)) \oplus (m_2 \oplus PRG(k)) \\ &= m_1 \oplus (PRG(k) \oplus PRG(k)) \oplus m_2 \\ &= m_1 \oplus 0 \oplus m_2 = m_1 \oplus m_2 \end{aligned}$$

The English language and the ASCII encoding has enough redundancy so that m_1 and m_2 can be easily decoded from $m_1 \oplus m_2$. Examples of the use of two time pads are not rare in the real world. In network traffic, if the same key is used for different sessions, as was the case in MS-PPTP and 802.11b WEP, the encryption will be unsafe. Instead, a new key should be used for each session, which is implemented by TLS. And typically it is not wise to use a stream cipher for disk encryption, because once a file is changed, an attacker can easily know where the change happened.

2.3.2 No Integrity

In general, OTP and stream ciphers provide no integrity at all. In other words, they are **malleable**. An attacker can modify the cipher text without being noticed by the receiver. In particular, since $(m \oplus k) \oplus p = (m \oplus p) \oplus k$, an attacker can impose a specific effect on the cipher text (i.e. XOR with a specific p).

2.4 PRG Security

Let $G : \mathcal{K} \to \{0,1\}^n$ be a PRG. We will define what it means when we say that the output of G on a random key is indistinguishable from the output of a truly random sampler on $\{0,1\}^n$.

Definition 10. ***(Statistical Test)*** *A* ***statistical test*** *is an algorithm A on $\{0,1\}^n$ that outputs 0 or 1.*

As a few examples:

- $A(x) = 1$ iff $|\#0(x) - \#1(x)| \leq 10\sqrt{n}$
- $A(x) = 1$ iff $|\#00(x) - \frac{n}{4}| \leq 10\sqrt{n}$
- $A(x) = 1$ iff $max - run - of - 0(x) \leq 10 \log n$[2]

Definition 11. ***(Advantage)*** *Let $G : \mathcal{K} \to \{0,1\}^n$ be a PRG and A be a statistical test. The* ***advantage*** *of A relative to G is defined as*

$$Adv_{PRG}[A,G] = \left| \Pr_{k \xleftarrow{R} \mathcal{K}} [A(G(k)) = 1] - \Pr_{r \xleftarrow{R} \{0,1\}^n} [A(r) = 1] \right|.$$

An advantage close to 1 means that A can distinguish the output of G from a random choice, whilst an advantage close to 0 means that A cannot tell the difference. For example, a dummy statistical test that always outputs 0 has 0 advantage for any G, meaning it cannot distinguish any PRG from random choice.

Definition 12. ***(Secure PRG)*** *A PRG G is called a* ***secure PRG*** *if $\forall$ efficient statistical test A, $Adv_{PRG}[A,G]$ is negligible.*

Note that the requirement of efficiency of the statistical tests is necessary for this definition to be satisfied. Providing a secure PRG will result in a proof of $P \neq NP$, thus we don't know yet if there exist any secure PRGs.

Theorem 3. *A secure PRG is unpredictable.*

[2] The expectation of the maximum run of 0 is roughly $\log n$.

Proof. We will prove its contrapositive, i.e. a predictable PRG is insecure. Recall that a PRG being predictable means the existence of an efficient algorithm A s.t.

$$Pr\left(A\left(G(k)|_{1,\ldots,i}\right) = G(k)|_{i+1}\right) \geq \frac{1}{2} + \epsilon$$

for a non-negligible ϵ. We define a statistical test $B(X)$ that outputs 1 iff $A(X|_{1,\ldots,i}) = X_{i+1}$. For a random choice, the $(i+1)^{th}$ bit is irrelevant from the first i bits, thus $Pr(B(r) = 1) = \frac{1}{2}$. However $Pr(B(G(k)) = 1) \geq \frac{1}{2} + \epsilon$, hence $Adv_{PRG}[B, G] \geq \epsilon$, which is not negligible and verifies the insecurity of G. □

Actually the converse also holds.

Theorem 4. *An unpredictable PRG is secure.*

This theorem means that if next-bit predictors cannot distinguish G from random choice, then no statistical test can.

Finally let's generalize the definition of computationally indistinguishability.

Definition 13. *Two distributions P_1 and P_2 over $\{0,1\}^n$ are **computationally indistinguishable** if $\forall$ efficient statistical test A,*

$$\left| \Pr_{x \leftarrow P_1}[A(x) = 1] - \Pr_{x \leftarrow P_2}[A(x) = 1] \right|$$

is negligible. Their relation is denoted as

$$P_1 \approx_p P_2.$$

According to this definition, a PRG is secure if $\{k \xleftarrow{R} \mathcal{K} : G(k)\} \approx_p uniform(\{0,1\}^n)$.

2.5 Semantic Security