CS 549 - Cryptography

Ismael Lopez

1 Introduction

1.1 Cryptography and Modern Cryptography

Definition

The study of mathematical techniques for securing digial info, systems, and distributed computations against adversarial attacks.

History

Used to exclusively focus on ensuring private communication and be more like art (subjective). Now scope is broader and more like science.

1.2 The Setting of Private-Key Encryption

Aim was to secure messages using *codes*, or *ciphers*. In modern parlance, *codes* are called *encryption schemes*. The security of these schemes relied on secret *keys*. A *plaintext* message is encrypted using the *key* to form a *cipher* which is sent to the recipient. They then used the same *key* to decrypt the message. **Symmetric-key encryption** is when both parties use the same *key* to encrypt and decrypt a message. This is in contrast to **asymmetric -key encryption** where encryption and decryption use different *keys*. Another way of looking at that is by associating a **secret;public** pair. Anyone wanting to message a person must use public key to encrypt. Only those having secret key can decrypt said messages.

The syntax of encryption

Formally, a private-key encryption scheme is defined by specifying a *message space* $\mathcal{M}$ (which defines the set of all legal messages) along with three algorithms:

Key Generation Algorithm - *Gen*

- probabilistic algorithm that outputs a key k chosen according to some distribution

Encryption Algorithm - $Enc_k(m)$

- takes as input a key k and a message m and outputs a ciphertext c

Decryption Algorithm - $Dec_k(m)$

- Takes as input a key k and a ciphertext c and outputs a plaintext message m.

1.2.1 Requirements

Correctness - decryption results in same message, with high probability. That is, for every key k output by Gen, and every message $m \in \mathcal{M}$, it holds that

$$Dec_k(Enc_k(m)) = m$$

Security - cannot infer the true message from ciphertext

Almost always, Gen simply chooses a uniform key from the key space $\mathcal{K}$.

Kerckhoffs' Principle

The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience. That is, assume attacker knows all the details of the scheme. As long as k is secure, the scheme should be secure. In addition, it is desired for encryption schemes to be standardized so that compatiblity is ensured by default and users will utilize an an encryption scheme that has been publicly tried and tested.

1.3 Historical Ciphers and Their Cryptanalysis

1.3.1 Caesar's cipher

Encryption involved shifting the letters of the alphabet 3 places forward (e.g. **a→D**). Problems include that the method is $fixed$; there is no key. Once the method is known, anyone can decrypt.

1.3.2 Shift cipher and the Sufficient key-space principle

Like a *keyed* variant of Caesar's cipher. Here, the key k is a number between 0 and 25. Instead of shifting by 3 spaces as in CC, letters are shifted k places. The process of mapping a to a mod N is called *reduction modulo N*. Used to keep within key space $\mathcal{K}$. An attack that involves trying every possible key is called a **brute-force** or **exhaustive-search** attack. Observation: *any secure encryption scheme must have a key space that is sufficiently large to make a brute-force attack infeasible.*

1.3.3 Mono-alphabetic Substitution cipher (Permutation)

Similar to shift cipher, except the mapping from plaintext alphabet to cipher alphabet is now *arbitrary*, as long as it is one-to-one (which we need to that we can decrypt). The key space $\mathcal{K}$ now consists of all *permutations* of the alphabet. As such, $|\mathcal{K}| = 26! \approx 2^{88}$. As such, brute force is infeasible. However, it is $still not secure$. Using statistical patterns of the plaintext language, we can see optimize it by seeing which cipher characters come up often and map those to the most frequent letters in the language's alphabet.

1.3.4 Improved attack on the Shift cipher

Associate letters of English alphabet with 0-25. Let p_i, with $0 \leq p_i \leq 1$, denote the frequency of the ith letter in normal texts. Using known frequencies, we get

$$\sum_{i=0}^{25} p_i^2 \approx 0.065$$

Now, given some ciphertext and let q_i denote frequency of ith letter of the alphabet in this ciphertext (i.e. q_i is simply the number of occurences of the ith letter of the alphabet in the ciphertext divided by length of the ciphertext). If the key is k, then p_i should be roughly equal to $q_{i+k} \forall i$, because the ith letter is mapped to the $(i+k)$th letter. Thus, if we compute

$$I_k = \sum_{i=0}^{25} p_i \cdot q_{i+j}$$

for each value of $j \in \{0, ..., 25\}$, then we expect to find that $I_k \approx 0.065$ where k is the actual key. What this means is we can automate the attack until we find the closest I_j, for all j, to I_k.

1.3.5 Vigenère (Poly-Alpha. Shift) cipher

Previous ciphers were statistically attackable because the key defined a fixed mapping that was applied letter-by-letter to the plaintext. Here, the key defines a mapping that is applied on *blocks* of plaintext characters (e.g. $ab \rightarrow DZ$ and $ac \rightarrow TY$). This *smooths out* the frequency distribution of characters in the ciphertext, making it harder to statistically analyze. The $Vigenère$ cipher is a special case. The key is now a *string* of letters; encryption is done by shifting each plaintext character by the amount indicated by the next character of the key, wrapping around in the key when necessary. Note this is a normal shift cipher is $|k| = 1$. An example would be:

Plaintext:	a	t	t	a	c	k	s	o	o	n	o	k
Key (repeated):	c	a	f	e	c	a	f	e	c	a	f	e
Ciphertext:	V	E	Q	O	J	I	R	E	D	O	F	G

Attacking the $Vigenère$ **cipher**:

If the *length of the key* is known, then attacking the cipher is easy. Though a large enough key space is **necessary** for any secure cipher, it is far from being *sufficient*.

1.3.6 Kasiski's Method

Since words like 'the' are frequently used, identify repeated patterns of such length in the ciphertext to narrow down possibilities.

1.3.7 Index of Coincidence

Method similar to improved shift cipher attack. Analyze the distribution of letters and find the coincidences. Measure the gap between them for the approximate key length, or at least a multiple of it.

1.3.8 Ciphertext length and attacks

Previous mentioned methods require long enough cipher text to get an accurate distrubution of observed frequencies.

1.4 Principles of Modern Cryptography

Schemes need to be proven rigorously using formal definitions. Most cryptographic proofs rely on currently unproven *assumptions* about the algorithmic hardness of certain mathematicl problems.

1.4.1 Principle 1 - Formal Definitions

If you don't understand what you want to achieve, how can you possibly know when (or if) you have achieved it? Definitions can guide design, evaluate and analyze what is contructed, and show security. Security goal: *regardless of any information an attacker already has, a ciphertext should leak no additional information about the underlying plaintext.* There are different threat models, in order of increasing power of the attacker:

Ciphertext-only attack: Most basic attack. Adversary just observes a ciphertext and tries to determine information about the underlying plaintext. Adversary is not able to distinguish an encryption of $m^{(0)}$ from $m^{(1)}$.

Known-plaintext attack: Adversary is able to learn at least one plaintext/ciphertext pair generated using some key. Thus, their aim is now to deduce information about *another* ciphertext produced using the same key.

Chosen-plaintext attack: Adversary can obtain plaintext/ciphertext pairs *for plaintexts of its choice.*

Chosen-ciphertext attack: Adversary is additionally able to obtain (some info about) the *decryption* of ciphertexts of its choice (e.g. whether the decryption of some ciphertext chosen by the attacker yields a valid English message). Their aim, once again, is to learn information about the underlying plaintext of some *other* ciphertext (whose decryption they are unable to obtain directly).

1.4.2 Principle 2 - Precise Assumptions

Proofs of security typically rely on assumptions. Clear assumptions can be validated, used as comparisons between schemes (weaker assumptions mean weaker schemes), and used to ensure changes still lead to a secure scheme.

1.4.3 Principle 3 - Proofs of Security

They give an iron-clad gaurantee (relative to definition and assumptions) that no attacker will succeed.

1.4.4 Provable Security and Real-World Security

Provable security of a scheme does not necessarily imply security of that scheme in the real world because the definitions and assumptions may not capture an adversary's true abiltiies. So the idea is: attackers focus attention on the definition or assumptions, while cryptographers continually refine these to match the real world.

2 Perfectly Secret Encryption

Generating randomness

Book assumes bits are random. In practice, modern *random-number generation* proceeds in two steps: 1. a 'pool' of high-entropy data is collected (entropy meaning unpredictability) 2. this data is processed to yield a sequence of nearly independent and unbiased bits.

2.1 Definitions

Recall an encryption scheme is defined by Gen, Enc, and Dec, with a finite *message space* $\mathcal{M}$ with $|\mathcal{M}| > 1$.

m	a plaintext message
M	Random Variable (denoting value of message)
$\mathcal{M}$	set of all possible **messages**
k	a key
K	Random Variable (denoting the key)
$\mathcal{K}$	set of all possible **keys**
c	a ciphertext
C	Random Variable (denoting the ciphertext)
$\mathcal{C}$	set of all possible **cyphertexts**

We now allow Enc to be probabilistic (meaning it might output a different ciphertext when run multiple times). We write $c \leftarrow Enc_k(m)$ to denote the possibly probabilistic process by which message $m \in \mathcal{M}$ is encrypted using key $k \in \mathcal{K}$ to give ciphertext $c \in \mathcal{C}$. If Enc is **deterministic**, we write it as $c := Enc_k(m)$. Also use $x \leftarrow S$ to denote uniform selection of x from set S. Correctness means $m := Dec_k(c)$.

2.1.1 Perfect Secrecy

DEFINITION 2.3

Encryption scheme (Gen, Enc, Dec) *with message space* $\mathcal{M}$ *is* `perfectly secret` *if for every probability distribution over* $\mathcal{M}$*, every message* $m \in \mathcal{M}$*, and every ciphertext* $c \in \mathcal{C}$ *for which* $\Pr[C = c] > 0$:

$$\Pr[M = m|C = c] = \Pr[M = m]$$

LEMMA 2.4

An encryption scheme (Gen, Enc, Dec) *with message space* $\mathcal{M}$ *is* `perfectly secret` *iff* $\Pr[Enc_K(m) = c] = \Pr[Enc_K(m') = c]$ *holds for every* $m, m' \in \mathcal{M}$ *and every* $c \in \mathcal{C}$.

2.1.2 Perfect (adversarial) Indistinguishability

An encryption scheme is *perfectly indistinguishable* if *no* adversary $\mathcal{A}$ can succeed in determining which of m or m' was used with a probability greater than 1/2. The experiment $PrivK^{eav}_{\mathcal{A},\Pi}$ is defined as:

1. *The adversary* $\mathcal{A}$ *outputs a pair of messages* $m_0, m_1 \in \mathcal{M}$.
2. *A key* k *is generated using* Gen*, and a uniform bit* $b \in \{0, 1\}$ *is chosen. Ciphertext* $c \leftarrow Enc_k(m_b)$ *is computed and given to* $\mathcal{A}$*. We refer to* c *as the* `challenge ciphertext`.
3. $\mathcal{A}$ *outputs a bit* b'.
4. *The output of the experiment is defined to be* 1 *if* $b' = b$*, and* 0 *otherwise. We write* $PrivK^{eav}_{\mathcal{A},\Pi} = 1$ *if the output of the experiment is* 1 *and in this case we say that* $\mathcal{A}$ `succeeds`.

DEFINITION 2.5

Encryption scheme Π = (Gen, Enc, Dec) *with message space* $\mathcal{M}$ *is* `perfectly distinguishable` *if for every* $\mathcal{A}$ *it holds that*

$$\Pr[PrivK^{eav}_{\mathcal{A},\Pi}] = \frac{1}{2}$$

The following lemma says Def 2.5 is equivalent to Def 2.3.

LEMMA 2.6

Encryption scheme Π *is perfectly secret if and only if it is perfectly indistinguishable.*

2.1.3 Lecture Definitions (may include repeats)

Definition A

(Idea: For any plaintext, we have an equivalent probability of generating a given ciphertext) For all $m \in \mathcal{M}$ and $c \in \mathcal{C}$, we have:

$$\Pr[C = c|M = m] = \Pr[C = c]$$

Definition B

(Idea: The probability of generating a ciphertext given as message m_0 is the same as generating it as message m_1) For any message $m_0, m_1 \in \mathcal{M}$ and $c \in \mathcal{C}$, we have:

$$\Pr[C = c|M = m_0] = \Pr[C = c|M = m_1]$$

Definition C

For all adversary $\mathcal{A}$, its advantage in the security game is 0.

2.2 The One-Time Pad

CONSTRUCTION 2.8

Fix an integer $l > 0$. The message space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ are all equal to $\{0, 1\}^l$ (the set of all binary strings of length l).

Gen: the key-generation algorithm chooses a key from $\mathcal{K} = \{0, 1\}^l$ according to the uniform distribution.

Enc: given a key $k \in \{0, 1\}^l$ and a message $m \in \{0, 1\}^l$, the encryption algorithm outputs the ciphertext $c := k \oplus m$.

Dec: given a key $k \in \{0, 1\}^l$ and a ciphertext $c \in \{0, 1\}^l$, the decryption algorithm outputs the message $m := k \oplus c$.

THEOREM 2.9

The one-time pad encryption scheme is perfectly secret.

Drawback to OTP the messages may be large and the key needs to match that, not to mention it must be generated *for each message.* Used ≈ 100 years ago. Only secure if used once.

2.3 Limitations of Perfect Secrecy

Stated in Theorem 2.10. Basically, keys need to be as long as messages and that is not great.

THEOREM 2.10

If (Gen, Enc, Dec) *is a perfectly secret encryption scheme with message space* $\mathcal{M}$ *and key space* $\mathcal{K}$*, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

2.4 Shannon's Theorem

The characterization of perfectly secret encryption schemes says that under certain conditions, Gen must choose the key *uniformly* from $\mathcal{K}$.

Theorem:

Let (Gen, Enc, Dec) *be an encryption scheme with message space* $\mathcal{M}$*, for which* $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$*. The scheme is perfectly secret iff*:

1. *Every key* $k \in \mathcal{K}$ *is chosen with (equal) probability* $1/|\mathcal{K}|$ *by algorithm* Gen.
2. *For every message* $m \in \mathcal{M}$ *and every ciphertext* $c \in \mathcal{C}$*, there exists a unique key* $k \in \mathcal{K}$ *s.t.* $Enc_k(m)$ *yields* c.

3 Private-Key Encryption

3.1 Computational Security

Security definitions that take into account computational limits, and allow for a small probability of failure, are called *computational* to distinguish them from notions that are *information-theoritic* (like perfect secrecy). Relaxations include *security is only gauranteed against* `efficient` *adversaries that run for some feasible amountof time* and *adversaries can potentially succeed with some* `very small probability`. Perfect seems unnecessarily strong. Computational means it's okay to leak some information with a tiny probability to eavesdroppers with bounded computational resources.

3.1.1 The Concrete Approach

A scheme is (t, ϵ)-`secure` *if any adversary running for time at most* t *succeeds in breaking the scheme with probability at most* ϵ. Modern private-key encryption schemes are generally assumed to give almost optimal security when the key k has length n (so $|\mathcal{K}| = 2^n$), an adversary running for time t succeeds in breaking the scheme with a probability at most $ct/2^n$ for some fixed constant c

3.1.2 The Asymptotic Approach

This approach introduces an integer-valued *security parameter* n that parameterizes both cryptographic schemes as well as all involved parties. Running time of adversary as well as its success probability now a function of n. Thus, we equate 'efficient adversaries' with randomized algorithms running in time *polynomial* to n (i.e. there is a polynomial p s.t. the adversary runs for time $p(n)$). We also equate the notion of 'small probability of success' with success probabilities *smaller than any inverse polynomial of* n. Such probabilities are called *negligible.* Let **PPT** stand for *probabilistic polynomial time.* Thus, a formal definition for **asymptotic security** is:

A scheme is `secure` *if any PPT adversary succeeds in breaking the scheme with at most negligible probability.*

Asymptotic Approach in Detail

An algorithm A runs in polynomial time if there exists a polynomial p s.t. for every input $x \in \{0, 1\}^*$, the computation of $A(x)$ terminats within at most $p(|x|)$ steps. Since we measure runtime in terms of length of input, we sometimes provide algorithms with the security paramter written in unary (i.e. 1^n, or a string of 1s). We can see n as the key length.

DEFINITION 3.4

A function f *from the natrual numbers to the non-negative real numbers is* `negligible` *if for every positive polynomial* p *there is an* N *s.t.* $\forall$ *integers* $n > N$ *it holds that* $f(n) < \frac{1}{p(n)}$. We denote an arbitrary negligible function by `negl`. Negligible functions obey closure properties (of addition and polynomial multiplication.)

3.2 Defining Computationally Secure Encryption

DEFINITION 3.7

A `private-key encryption scheme` is a tuple of probabilistic polynomial-time (PPT) algorithms (Gen,Enc,Dec) such that:

1. The `key-generation algorithm Gen` takes as input 1^n (security param) and outputs a key k; we write $k \leftarrow \texttt{Gen}(1^n)$ (emphasizing Gen is a randomized algorithm). We assume without loss of generality that any key k output by $\texttt{Gen}(1^n)$ satisfies $|k| \geq n$.
2. The `encryption algorithm Enc` takes as input a key k and a plaintext message $m \in \{0, 1\}^*$, and outputs a ciphertext c. Since Enc may be randomized, we write this as $c \leftarrow \texttt{Enc}_k(m)$.
3. The `decryption algorithm Dec` takes as input a key k and a ciphertext c, and outputs a message m or an error. We assume that Dec is deterministic, and so write $m := \texttt{Dec}_k(c)$ (assuming here that Dec does not return an error). We denote a generic error by the symbol $\perp$.

It is required that for every n, every key k output by $\texttt{Gen}(1^n)$, and every $m \in \{0, 1\}^*$, it holds that $\texttt{Dec}(\texttt{Enc}_k(m)) = m$. If (Gen, Enc, Dec) is such that for k output by $\texttt{Gen}(1^n)$, algorithm $\texttt{Enc}_k$ is only defined for messages

$m \in \{0,1\}^{l(n)}$, then we say that $(\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ is a `fixed-length private-key encryption scheme for messages of length` $l(n)$.

3.2.1 The Basic Definition of Security

New adversarial indistinguishability experiment $\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\Pi}(n)$:

1. The adversary $\mathcal{A}$ is given input 1^n, and outputs a pair of messages m_0, m_1 with $|m_0| = |m_0|$.
2. A key k is generated by running $\texttt{Gen}(1^n)$, and a uniform bit $b \in \{0,1\}$ is chosen. Ciphertext $c \leftarrow \texttt{Enc}_k(m_b)$ is computed and given to $\mathcal{A}$. We refer to c as the `challenge ciphertext`.
3. $\mathcal{A}$ outputs a bit b'.
4. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise. If $\texttt{PrivK}^{\texttt{eav}}_{\mathcal{A},\Pi}(n) = 1$, we say that $\mathcal{A}$ `succeeds`.

4 PRFs, Stream Cipher, Block Cipher

5 Block Ciphers

6 CCA and MACs

7 Hash Functions

7.1 Applications

Can be used to gaurantee data preservation for evidence in court cases.