Principle of Information & Security

Siddharth Bhat

Contents

Chapter 1

Introduction

1.0.1 Impossiblity of Infosec problems

Common aspect across all infosec problems to date is that it is impossible to solve.

- Password schemes - It is impossible to design a good password scheme. The machine must know something about the password you need to give. Call it the password file (TODO: how to do monospace?)
- Password Length - everlasting is impossible. One can always brute force passwords. Infinite length passwords do not work.
- Secure communication over insecure channels
- Signing
- Digital cash

TODO: learn TIKZ

Secure communication over insecure channels

sender —¿ key —¿ receiver — v adversary

At time t0, everything that receiver knows, adversary knows (assuming no one-time pad). After that, everything the receiver receives, the adversary also knows as well. So, the adversary has all information that the receiver does.

It is impossible to do secure communication over insecure channels.

Signing

Digital signature is impossible - Unforgable digital signature should not exist.

1. Signature should be a function of the message for it to be useful as a signature. Otherwise, an attacker could intersect messages to find the signature. 2. Signature must be publically verifiable. 3. A trapdoor function can be reverse-engineered.

Digital cash

How do we detect counterfeit cash? Double spending is a problem. Cryptocurrencies used the exact same mathematical methods that are shared across crypto.

1.0.2 A Tom and Jerry analogy

Tom & Spike are both Jerry's opponents. So, Jerry is able to play Tom and spike against each other, and have them beat each other.

That is, pair adversaries against each other to have them screw with each other.

Password schemes, take 2

We needed infinite length passwords because an adversary will win if we have finite length password. However, there are other adversaries. For example, the adversary for algorithms is the person who provides inputs. Example, think of sorting networks or uses of bubble sort: Sorting networks are useful on small numbers of elements to sort. Bubble sort does not screw with cache coherence. However, these are both bad solutions *in general*.

When the worst case input giver is an adversary, and a person who is trying to crack our password is an adversary, we can have these two interfere.

To find out ‘y = f(x)‘, we wind up using the algorithmic adversary who provides hard problems for ‘f‘.

Structure of information matters. Example, linked list v/s balanced tree. The process of decryption can exploit structure of information.

eg: Natural number can be represented as a product of primes, and in the decimal notation.

Active adversary / noise

We cannot design error detection codes for any amount of error. Hence, if we think of adversary as error in the stream, we can think of secure communication on a channel with an active adversary as ECC.

So now, this problem is now an information theory problem.

The adversary must make a modification such that the bank cannot detect it. Coding theory tells us that such a modification is always possible. Infosec tells us that we can design schemes where this takes a long time.

Chapter 2

Lecture 2 - More philosophy - Amazing Advantages of Additional Adversity

Textbook is

- Introduction to Modern Cryptography

2.0.1 Ceasar Cipher

rotate letters by a certain amount.

crypto goes to FUBSWR.

2.1 Kerckhoff's Principle

Security of system depends on secrecy of the key and not on the obscurity of the algorithm.

2.1.1 Password Shadows

Password is $\{x\}$, we store $\{f(x)\}$.

It is possible to reverse-engineer f to discover x. So, we should not depend on f being secure.

2.1.2

2.1.3 Shift Cipher

We can brute force this, we can brute force keys.

Principles learnt from shift ciphers

- Key space needs to be large. for shift cipher, key space is 26.

p_i probability of letter in plaintext. q_i probability of letter in ciphertext.
$\exists delta, \forall xinLetter, p_i = q_{i+k}$
$pi \cdot p_{i+k} = p_i^2$ if we wind the right k. So, we need to find the right k.
So, large key space is not enough. We need to ensure that frequency is also fudged.

2.1.4 Monoalphabetic Substitution Cipher

Create a bijection $\{f : Letter \leftarrow Letter\}$. This has a large key space, $\{26!\}$.

Attack is based on frequency. $\{\forall x \in Letter, freq(x) = freq(f(x))\}$. So, one can match x with $f(x)$.

Again, we need to fudge frequency.

2.1.5 Polyalphabetic sustitution cipher

This needs a passphrase, for example, Cat.

Add passphrase to plaintext.

$crypto + catcat = \cdots$

Frequencies are not maintained, because different text is added each time to the same plaintext.

- Step 1 - Given length, we break the cipher.
- Step 2 - Length is susceptible to brute force attack.

Breaking given length

Assume the length of passphrase is known, say, k.

Let ciphertext be $c_0 c_1 c_2 c_3 c_4$... Let us look at ciphertext at lengths of 3.

This will give us a *shift cipher*, since the text is all shifted by the *same* letter in the passphrase. Now, we can perform the frequency attack.

If we screw up the partition, then the frequency spectra will be gibberish. zsh:1: command not found: :w

What we learnt by breaking

This was also broken. So, they learnt that ”security is hard, forget it!”. Or, complication does not imply security.

2.1.6 What is an Unbreakable cipher? Or, shannon enters the scene

A preamble, the thought process

- We need to specify what it means to have a good cipher. Where do we stop? We need a formal spec. (Definition of security).
- Precise assumptions involved must be known. (Hardness assumption).
- The truth of security, and the trade-offs involved (Shannon’s Proof)

Chapter 3

Information Theory

3.0.1 shannon's perfect secrecy (1949)

Shannon framed a secrecy theory based on information theory. If no information is revealed to the other person, it is secure.

Cipher = $< Gen, Enc, Dec, M >$.

M is the message space.

$Gen :: KeyLength- > Key$. Set of all keys Gen can output ($Image(Gen)$) is called the key space. Key space (K)is asked to be finite.

$Enc :: M- > K- > C$. $C = ciphertext$. $Image(Enc)$ is called the ciphertext space. Enc takes a message and a key, and returns a ciphertext.

$Dec :: C- > K- > M$. such that $\forall(m : M)(k : K)Dec(k, Enc(m, k)) = m$.

A cipher scheme is secure iff: $\forall p =$ probability distributions over the message space (we don't know the exact probability distribution over plaintext). $\forall m \in M, \forall c \in C, P(C = c > 0) => P[M = m] = P[M = m|C = c]$.

What we know about the message before looking at the ciphertext is the same as what we know about the message ⅁℧≈∖ we know the ciphertext. We make sure that we do not take degenerate c (c that does not ever occur) to prevent nastiness in conditional probability.

3.1 Shannon and secure channel capacity of systems

Sender———* — — — — insercure channel secure channel(1Mbps) (1 Gbps) — v — Receiver¡— —*

If I have perfect security, what is the bandwidth of the ⪷≂⋊⋖∼↷∼≈⋗?

It must be between 1Mbps and 1Gpbs + 1Mbps. (minimum is 1Mbps).

If we have only an insecure channel, then set secure channel capacity to 0.

Shannon ı∖⋊⪸ that the secure channel capacity of the ⪷≂⋊⋖∼↷∼≈⋗ is 1Mbps (that of the secure channel).

also, if the secure channel has bandwidth 0, then it is impossible to have security.

3.2 Proof

3.2.1 First equivalence

A cipher is perfectly secret iff $\forall m \in M, \forall c \in C, forallprobabilitydistributionsoverM, P[C = c|M = m] = P[C = c]$

Proof

TODO: how to get aligned text.

$P[C = c|M = m] = P[C = c]$ $P[C = c|M = m] * P[M = m]/P[C = c] = P[C = c] * P[M = m]/P[C = c]$

Reminder: $P[A|B] = \frac{P[B|A]*P[A]}{P[B]}$

$P[M = m|C = c] = P[M = m]$

Qed. (TODO: how to get box)

3.2.2 Second equivalence

A cipher is perfectly secret iff $\forall m_0, m_1 \in M, P[C = c|M = m_0] = P[C = c|M = m_1]$.

Proof (=¿ directiion)

If the cipher is perfectly secure, $P[C = c|M = m] = P[C = c]$ (from first equivalence).

$P[C = c|M = m_0] = P[C = c]$. $P[C = c|M = m_1] = P[C = c]$.

Hence, $P[C = c|M = m_0] = P[C = c|M = m_1]$ *Qed.*

Proof (¡= directiion)

Given $\forall m_0, m_1 \in M, P[C = c|M = m_0] = P[C = c|M = m_1] = p$ $P[C = c] = \sum_{m \in M} P[C = c|M = m] * P[M = m]$ $P[C = c] = \sum_{m \in M} p * P[M = m]$ $P[C = c] = p \sum_{m \in M} P[M = m]$ Since we are summing over probability space, $P[C = c] = p * 1$ $P[C = c] = P[C = c|M = m]$ for any m. Qed.

The reason it's all p is because of transitivity. $M_0 = M_1$, $M_1 = M_2$, hence everything is equal.

3.3 Is there a scheme that exists that is perfectly secure?

3.3.1 One time pad

$Gen = k < -0, 1^n.P[K = k] = \frac{1}{2^n}$. $Encrypt(m, k) = kXORm.m \in 0, 1^n$. $Decrypt(c, k) = kXORc.c \in 0, 1^n$.

Perfect security of one time pad: proof (Vernam cipher)

We will show this by using the phrasing: A cipher is perfectly secret iff $\forall m_0, m_1 \in M, P[C = c|M = m_0] = P[C = c|M = m_1]$.

$P[C = c|M = m_0] = P[C = kXORm_0] = P[K = cXORm_0] = \frac{1}{2^k}$ $P[C = c|M = m_1] = P[C = kXORm_1] = P[K = cXORm_1] = \frac{1}{2^k}$

We pick the key independent of the message, so it doesn't matter what the message is.

Limitations of one-time-pad

.

Since we need to send the key securely, we will need to send the key over the slow secure channel. We can send the message over the insecure channel. However, to decrypt the *nth* insecure bit, we need the *nth* secure bit. So, for this, we might as well send message over the secure channel.

However, if both the secure channel and the message are available at different times, then one-time-pad is useful. We can send the key over the secure channel, and use it later to decrypt a message sent over the insecure channel.

Next, if sender = receiver, then it makes sense to have one-time-pad. The channel of internal transfer should be very fast (eg. memory transfer is fast, versus network transfer is slow).

3.4 Every perfectly security scheme is isomorphic to one-time-pad.

3.4.1 Theorem

$\forall$perfectly secret cipher, $|K| \geq |M|$.

Note on bit sizes

It is possible that $|K| \geq |M|$, but $nbits(K) \leq nbits(M)$. That is, the number of bits needed to store the space can be smaller than the space (low entropy).

3.4.2 Proof

Suppose for contradiction $|K| < |M|$.

One ciphertext c can be decrypted into at most $|K|$ messages. In the message space, there must be one message $M*$ that is not part of the decryption of c (since $|K| < |M|$).

$P[M = m * |C = c] = 0$ since if $c = c$, $m*$ cannot occur. However $P[M = m*]$ is non-zero.

Hence, $P[M = m * |C = c] = P[M = m*]$.

Shannon further proves that the entropy of the key space must be greater than the entropy of our message space. Hence, we will need to send as much data over the secure channel as long as the key, usually.

3.5 Tangent: Entropy, Expectation, random kannan

3.5.1 Expectation

$E[x] = \sum_x x \cdot p(X = x)$

3.5.2 Entropy

There can be many indexing schemes to store data. $M = m_0, m_1, \cdots, m_n$. We can use $log(n)$ bits to store the index.

What is the expected number of bits to store a message space with n messages? Say m_i occurs with probability p_i.

$E[\#bitstostoreM] = \sum p_i \cdot ixlength(m_i)$

The ones where p_i is high, we want $ixlnegth_i$ to be small for an efficient compression scheme. Index the thing that occurs most often with the least bits.

We can represent the $\lessdot\gtrdot\ltimes\approx$ m_i in terms of p_i (how often it occurs in the space). We can make messages that occur more frequently with strings of smaller length.

Eg: for a message with $p = \frac{1}{2}$, use 1 bit to represent ix. Eg: for a message with $p = \frac{1}{4}$, use 2 bits to represent ix. Eg: for a message with $p = \frac{1}{n}$, use n bits to represent ix. Eg: for a message with $p = k$, use $\frac{1}{k}$ bits to represent ix.

$E\#bitstostoreM = \sum_i p_i \log(\frac{1}{p_i})$. $E\#bitstostoreM = -\sum_i p_i \log(p_i)$. Entropy $= -\sum_i p_i \log(p_i)$.

3.6 Looking at the future (next lecture)

Can we actually fully utilise the insecure channel by relaxing our definitions of secure?