Cryptographic Attack Models

Based on *Cryptography Engineering* sections 2.6, 2.7
By Ferguson, Schnier, and Kohno

Lecture notes of Alexander Wood
awood@jjay.cuny.edu

John Jay College of Criminal Justice

Kerckhoffs’ Principle

Recall **Kerckhoffs’ Principle**, formulated by the Dutch mathematician in the 1880s.

This principle states that *we should be able to publish all of the information about how the cryptosystem works without compromising its security.*

Cryptanalytic Attack Models

First we will go over attack models which aim to recover either the *plaintext* or the **decryption key**.

- Ciphertext-Only Model
- Known-Plaintext Model
- Chosen-Plaintext Model
- Adaptive-Chosen-Plaintext Model
- Chosen-Ciphertext Model

Ciphertext-Only Model

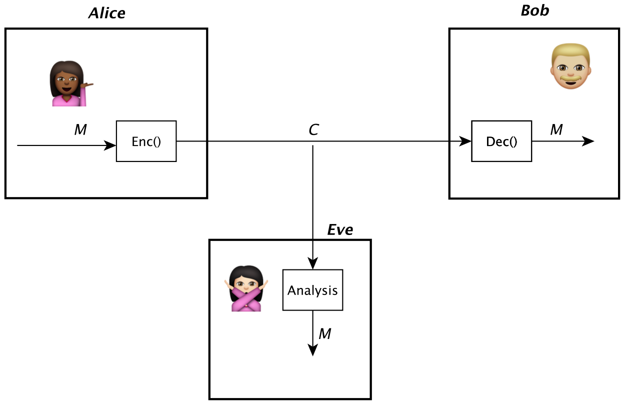
This model is the most basic model, and is what people usually mean when they discuss breaking encryption. In this model, Eve is eavesdropping on communication between Alice and Bob, and all she sees as the attacker is the ciphertexts.

- *Given*: Ciphertexts $C_1, C_2, \ldots, C_k$
- *Deduce:* $M_1, M_2, \ldots, M_k$ or $Enc()$.

She attempts to use properties of these plaintexts to deduce either the contents of the messages, the keys, or both.

Ciphertext-Only Model

Eve tries to determine they key(s) or the plaintext(s) by analyzing intercepted ciphertexts.



Ciphertext-Only Model

We saw how to break several encryption systems using the ciphertext-only model.

- We broke the **Caesar Shift** using a brute-force attack in the ciphertext-only model.
- We broke the **substitution cipher** using a frequency analysis attack in the ciphertext-only model.

Ciphertext-Only Model

This is generally considered the most difficult type of attack because the attacker has the least amount of information to work with.

Often, we are in situations where the attacker has access to more information than just the ciphertexts.

Known-Plaintext Model

A **known-plaintext attack** is carried out when Eve knows both the plaintext and the ciphertext.

- *Given:* Plaintexts and their associated ciphertexts
- *Deduce:* Decryption key and/or a method of deducing plaintexts from ciphertexts
  Now, Eve not only as access to the ciphertext of several messages, but also their corresponding plaintexts. She must deduce the key(s) to the algorithm, or a method of obtaining the plaintexts from ciphertexts.

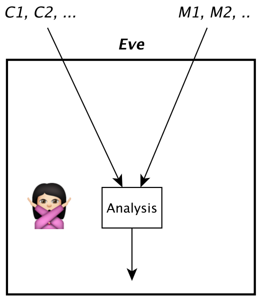
Known-Plaintext Model

Why do we consider this model? What we need to ask is, when could Eve have access to plaintext?

Occasionally there are messages which are easy to predict. Consider the Bombe machine's attack on the Enigma in WWII. The plaintexts “Wetterbericht” (weather report) and “Heil Hitler” were essentially guaranteed to appear in some messages. The Bombe machine carried out a **known-plaintext attack**.

Known-Plaintext Model

Eve uses the knowledge of some plaintext-ciphertext pairs in order to deduce the key, which she can use to decrypt new ciphertexts.



Chosen-Plaintext Model

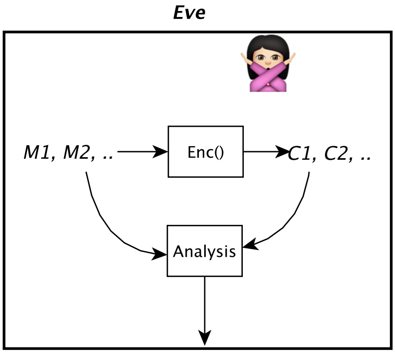
This is a more powerful model. Eve has access to plaintexts and their corresponding ciphertexts, as before. However in a chosen-plaintext attack, Eve gets to choose which plaintexts to encrypt!
This is a relevant attack which any good cryptosystem should be able to withstand. For example, say Eve sends Alice and email which she knows she will encrypt and forward to Bob.

Chosen-Plaintext Model: The Game

Cryptographer Victor Shoup has presented models for security proofs as **sequences of games.** In the Chosen-Plaintext Game,

1. Eve chooses plaintext messages $M_1, M_2, \ldots, M_n$ to encrypt.
2. Eve is given access to an **encryption oracle**, which returns to her the corresponding ciphertexts $C_1, C_2, \ldots, C_n$.
3. Eve deduces the decryption key using a chosen-plaintext attack.

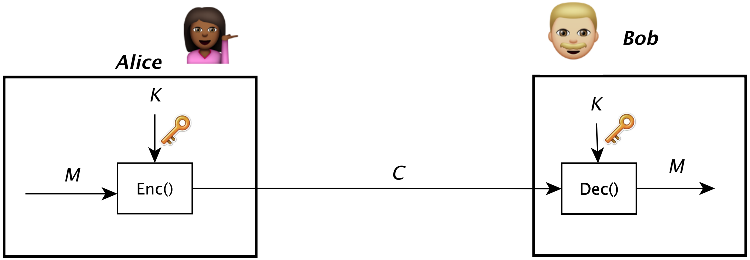
Chosen Plaintext Model



Eve chooses which plaintexts to encrypt.

Chosen-Plaintext Model: Private Key

In the private key model, a known-plaintext attack is less common because the only information Eve can eavesdrop on is the ciphertexts. She does not have any method of encrypting messages herself.

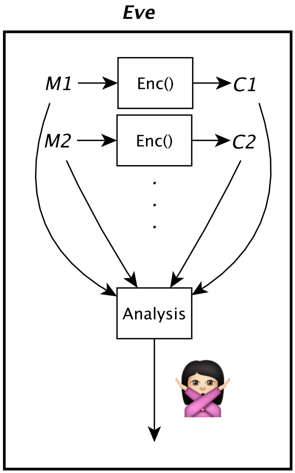


Chosen Plaintext Model: Public Key Cryptosystems

What happens if a public-key cryptosystem is vulnerable to a chosen-plaintext attack?

If the encryption key is a **public key**, then vulnerability to a chosen-plaintext attack comprimises the security of the cryptosystem. Eve can access the plaintext-ciphertext pair of any plaintext she chooses.

Adaptive-Chosen-Plaintext Model



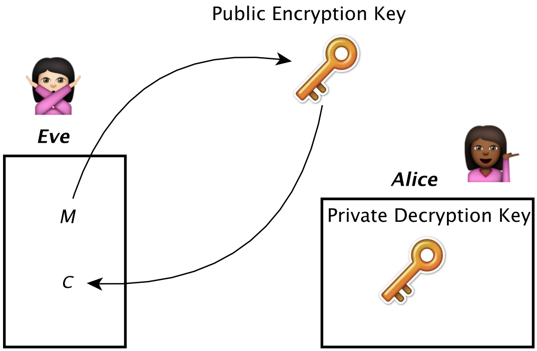
This is similar to the previous model. However, before, Eve had to choose all of the plaintexts she would like to encrypt up front. Now, Eve can adapt the plaintexts she encrypts based off of previous encryptions.

Adaptive-Chosen-Plaintext Model: The Game

1. For as long as Eve chooses, she carries out the following:
   1.1 Eve chooses a message M.
   1.2 Eve receives its encryption C.
   1.3 Eve analyzes the plaintext-ciphertext pairs she has collected. She uses this knowledge to prepare her next message.
2. Eve deduces the decryption key.

Adaptive-Chosen-Plaintext Model: Public Key

In the public-key model it is sometimes imperative that the cryptosystem can withstand an adaptive chosen plaintext attack.



For instance, Alice may have a public key which any person can use to encrypt and send a message – which only she is able to decrypt using the private key. In this case, Eve and encrypt any plaintext and view its ciphertext.

Chosen-Ciphertext Model

This should really be called a *chosen-ciphertext-and-plaintext* model but that is too long. In this model, Eve can choose both plaintext values to encrypt as well as ciphertext values to decrypt.

This is a very powerful form of security.

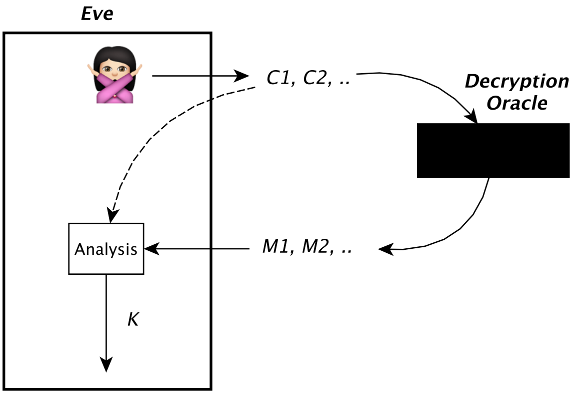
Chosen-Ciphertext Model

- *Given:* Plaintext messages and their corresponding encryptions; ciphertext messages and their corresponding decryptions
- *Deduce:* K, the decryption key

Eve can choose any ciphertexts to be decrypted, and is able to access their decrypted plaintexts.

Chosen-Ciphertext Model: Black Box Decryption

Eve is given what is called **black-box access** to a **Decryption Oracle.** She does not know the decryption key, but she is able to send ciphertexts of her choosing through this oracle and receive the corresponding plaintexts in return.



Chosen-Ciphertext Model: The Game

This is the *offline* version, where Eve must choose all plaintext and ciphertext messages in advance.

1. Eve chooses $M_1, \ldots, M_n$ and $\hat{C}_1, \ldots, \hat{C}_k$.
2. Eve is given $C_i = D_K(M_i)$ for $1 \leq i \leq n$ and $\hat{M}_j = E_{\hat{k}}(\hat{C}_j)$ for $1 \leq j \leq k$.
3. Eve deduces K, the decryption key.

Adaptive Chosen-Ciphertext Model

There is also a corresponding *adaptive*, or *online*, chosen-ciphertext model. In this case, Eve chooses a plaintext or ciphertext message, receives either its encryption or decryption, and can use this information to selectively pick her next plaintext or ciphertext.

This is an even stronger model of security.

Other Attack Models

The attacks described above attempt to recover the decryption key or the plaintext. However, it is possible to mount a different type of attack. Attacks could be designed to:

- Recover partial information about a plaintext (length, partial decryptions, etc)
- Forge a message
- Many many more!

Distinguishing Attacks

A **distinguishing attack** finds a difference between an ideal encryption scheme, where no information could possibly leaked in any case ever, and the encryption scheme we are using.

This covers all possible attacks including ones not yet discovered.

What is a perfect encryption scheme?

Unconditional Security

We call an algorithm unconditionally secure if no amount of time and/or computational power would be enough for any adversary to ever recover the plaintext. The only unconditionally secure cryptosystem is the **one-time pad** which we will see next week.

The one-time pad is not a feasible cryptosystem for most applications, hence the need for other cryptosystems.

Security

Because unconditional security is not feasible in the real world, we must instead create cryptosystems which are resistant to feasible real-world attacks.

Brute-Force Attacks

All other cryptosystems could be broken by a ciphertext-only attack. The attacker would simply have to try every possible key until she found the correct one. This is called a **brute-force attack.**

A cryptosystem is called **computationally secure** if it is unfeasible to break the system with current and foreseeable future computational resources.

This is not generally a very feasible attack on a cryptosystem. Why not?

Other Attacks

- The attacker can base attacks based on how fast encryption and decryption operations take – known as **timing attacks**.
- An attack which uses this sort of additional leakage of information is called a **side-channel attack**.

The Birthday Paradox

The next attack we look at, called a **birthday attack**, is named after the **birthday paradox**.

How many people do you need in a room before there is a 50% chance that two of them share a birthday?

The Birthday Paradox

To calculate this, instead of calculating the probability of two people sharing a birthday we calculate the probability that a pair of people *do not* share a birthday.

The probability that two people have different birthdays is given by

$$1 - \frac{1}{365} = \frac{364}{365} = .99726$$

The Birthday Paradox

What is the probability that 4 people share a birthday? Well, given four people, there are $\binom{4}{2} = 6$ possible pairs of people. So the probability that every two out of four people have different birthday is given by

$$\left(\frac{364}{365}\right)^6 = 0.983\cdots$$

The Birthday Paradox

Somewhat counterintuitively it only takes 23 people before the probability that all possible pairs of them do not share a birthday drops below 50%. Note that $\binom{23}{2} = 253$ and

$$\left(\frac{364}{365}\right)^{23} = 0.499\cdots$$

So a slightly less than 50% chance that any pair of people do *not* share a birthday, which translates to slightly greater than 50% chance that some pair of people share a birthday.

Birthday Attacks

Birthday attacks use the fact that **collisions**, or duplicate values, occur more often than expected.

Example: Birthday Attacks

Suppose there is a financial transaction system which uses a new 64-bit key for each transaction. There are 2^{64} different keys, or over eighteen billion billion possible keys. After seeing only 2^{32} (four billion) transactions the attacker can expect that some pair of transactions likely used the same key.

The attacker can use two messages with the same key to forge a message which will still be authenticated, using its **message authentication code (MAC)**. This will be discussed further later in the course.

Birthday Attacks

If an element can take N values then we expect a collision after choosing approximately $\sqrt{N}$ random elements.

When using encryption we often use bit-string representations of data. If there are 2^k possible bit strings of lenght k, then we need approximately $2^{k/2}$ elements before we can expect a collision. This is called the **birthday bound**.

Meet-In-The-Middle Attacks

This is another type of a known-plaintext attack. Now, Eve will build her own table of keys.

In the previous example of the 64-bit financial transaction system, Eve carries out this attack by first choosing 2^{23} 64-bit keys at random. She then computes the MAC on the “Are you ready to receive the transaction?” message that is sent at the beginning of each round. If the MAC appears in her table then it is highly likely that the authentication key for that transaction is the same one the attacker used to compute that table entry, so Eve can recover this key.

We will discuss this attack further later in the course.

Man-in-the-Middle Attacks

These are a very common and useful form of attack. Eve now will insert herself in the middle of the conversation between Bob and Alice, and impersonate both of them in order to gain access to information they were trying to send each other.

We will discuss this in more detail later in the course.

References

These slides are based off of Sections 2.6 and 2.7 of *Cryptography Engineering* by Ferguson, Schneier, and Kohno.