1

Basics of cryptography

The Oxford English Dictionary gives the following definition of cryptography.

'A secret manner of writing, either by arbitrary characters, by using letters or characters in other than their ordinary sense, or by other methods intelligible only to those possessing the key; also anything written in this way. Generally, the art of writing or solving ciphers.'

Cryptography is an ancient art, and until relatively recently the above definition would have been quite adequate. However, in the last thirty years it has expanded to encompass much more than secret messages or *ciphers*.

For example cryptographic protocols for securely proving your identity online (perhaps to your bank's website) or signing binding digital contracts are now at least as important as ciphers.

As the scope of cryptography has broadened in recent years attempts have been made to lay more rigorous mathematical foundations for the subject. While cryptography has historically been seen as an art rather than a science this has always really depended on which side of the 'cryptographic fence' you belong. We distinguish between *cryptographers*, whose job it is to design cryptographic systems, and *cryptanalysts*, whose job it is to try to break them. Cryptanalysts have been using mathematics to break ciphers for more than a thousand years. Indeed Mary Queen of Scots fell victim to a mathematical cryptanalyst using statistical frequency analysis in 1586!

The development of computers from Babbage's early designs for his 'Difference Engines' to Turing's involvement in breaking the Enigma code owes much to cryptanalysts desire to automate their mathematically based methods for breaking ciphers. This continues with the National Security Agency (NSA) being one of the largest single users of computing power in the world.

One could argue that cryptographers have been less scientific when designing cryptosystems. They have often relied on intuition to guide their choice of cipher. A common mistake that is repeated throughout the history of

cryptography is that a 'complicated' cryptosystem must be secure. As we will see those cryptosystems which are currently believed to be most secure are really quite simple to describe.

The massive increase in the public use of cryptography, driven partly by the advent of the Internet, has led to a large amount of work attempting to put cryptography on a firm scientific footing. In many ways this has been extremely successful: for example it is now possible to agree (up to a point) on what it means to say that a cryptographic protocol is secure. However, we must caution against complacency: the inability to prove that certain computational problems are indeed 'difficult' means that almost every aspect of modern cryptography relies on extremely plausible, but nevertheless unproven, security assumptions. In this respect modern cryptography shares some unfortunate similarities with the cryptography of earlier times!

1.1 Cryptographic models

When discussing cryptographic protocols we necessarily consider abstract, idealised situations which hopefully capture the essential characteristics of the real-world situations we are attempting to model. In order to describe the various scenarios arising in modern cryptography it is useful to introduce a collection of now infamous characters with specific roles.

The players

Alice and *Bob* are the principal characters. Usually Alice wants to send a secret message to Bob. Bob may also want her to digitally sign the message so that she cannot deny sending it at a later date and he can be sure that the message is authentic. Generally Alice and Bob are the good guys, but even this cannot always be taken for granted. Sometimes they do not simply send messages. For example they might try to toss a coin down the telephone line!

Eve is the arch-villain of the piece, a passive eavesdropper who can listen in to all communications between Alice and Bob. She will happily read any message that is not securely encrypted. Although she is unable to modify messages in transit she may be able to convince Alice and Bob to exchange messages of her own choosing.

Fred is a forger who will attempt to forge Alice's signature on messages to Bob.

Mallory is an active malicious attacker. He can (and will) do anything that Eve is capable of. Even more worryingly for Alice and Bob he can also modify

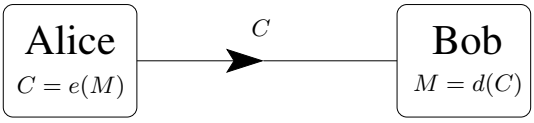

Fig. 1.1 Alice and Bob using a cryptosystem.

or even replace messages in transit. He is also sometimes known as the 'man in the middle'.

Peggy and *Victor* are the key players in identification schemes. In general Peggy (the prover) must convince Victor (the verifier) of her identity. While Victor must be careful that Peggy really is who she claims to be, Peggy must also make sure that she does not provide Victor with information that will allow him to impersonate her at a later stage.

Trent is a trusted central authority who plays different roles in different situations. One important responsibility he has is to act as a digital 'passport agency', issuing certificates to Alice and Bob which allow them to identify themselves convincingly to each other, hopefully enabling them to thwart Mallory.

Conveniently all of our characters have names starting with distinct letters of the alphabet so we will sometimes refer to them by these abbreviations.

1.2 A basic scenario: cryptosystems

The first situation we consider is the most obvious: Alice and Bob wish to communicate secretly. We assume that it is Alice who sends a message to Bob.

The fundamental cryptographic protocol they use is a *cryptosystem* or *cipher*. Formally Alice has a *message* or *plaintext* M which she *encrypts* using an encryption function $e(\cdot)$. This produces a *cryptogram* or *ciphertext*

$$C = e(M).$$

She sends this to Bob who *decrypts* it using a a decryption function $d(\cdot)$ to recover the message

$$d(C) = d(e(M)) = M.$$

The above description explains how Alice and Bob wish to communicate but does not consider the possible attackers or adversaries they may face. We first need to consider what an adversary (say Eve the eavesdropper) is hoping to achieve.

Eve's primary goal is to read as many of Alice's messages as possible.

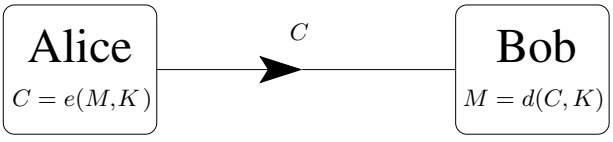

Fig. 1.2 Alice and Bob using a symmetric cryptosystem.

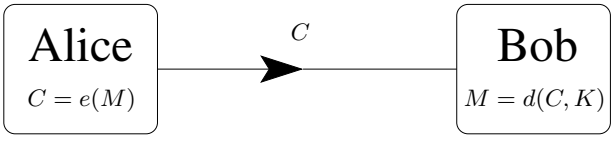

Fig. 1.3 Alice and Bob using a public key cryptosystem.

We assume that Eve knows the form of the cryptosystem Alice and Bob are using, that is she knows the functions $d(\cdot)$ and $e(\cdot)$. Since she is eavesdropping we can also assume that she observes the ciphertext C.

At this point Alice and Bob should be worried. We seem to be assuming that Eve knows everything that Bob knows. In which case she can simply decrypt the ciphertext and recover the message!

This reasoning implies that for a cryptosystem to be secure against Eve there must be a secret which is known to Bob but not to Eve. Such a secret is called a *key*.

But what about Alice, does she need to know Bob's secret key? Until the late twentieth century most cryptographers would have assumed that Alice must also know Bob's secret key. Cryptosystems for which this is true are said to be *symmetric*.

The realisation that cryptosystems need not be symmetric was the single most important breakthrough in modern cryptography. Cryptosystems in which Alice does not know Bob's secret key are known as *public key cryptosystems*.

Given our assumption that Eve knows the encryption and decryption functions but does not know Bob's secret key what type of attack might she mount?

The first possibility is that the only other information Eve has is the ciphertext itself. An attack based on this information is called a *ciphertext only* attack (since Eve knows C but not M). (See Figure 1.4.)

To assume that this is all that Eve knows would be extremely foolish. History tells us that many cryptosystems have been broken by cryptanalysts who either had access to the plaintext of several messages or were able to make inspired guesses as to what the plaintext might be.

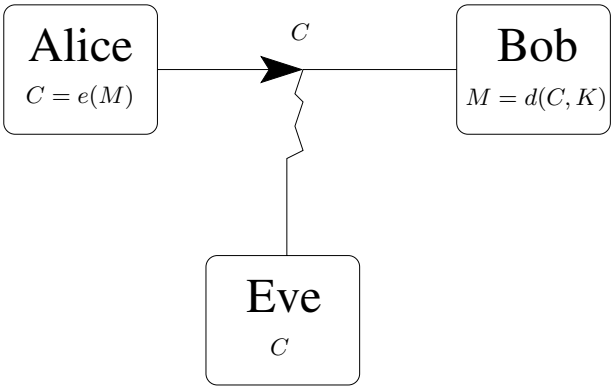

Fig. 1.4 Eve performs a ciphertext only attack.

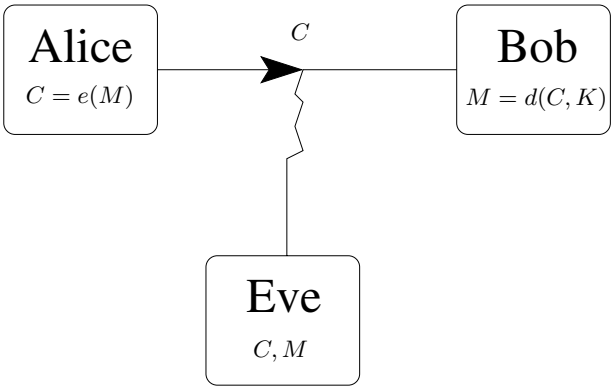

Fig. 1.5 Eve performs a known plaintext attack.

A more realistic attack is a *known plaintext attack*. In this case Eve also knows the message M that is encrypted. (See Figure 1.5.)

An even more dangerous attack is when Eve manages to choose the message that Alice encrypts. This is known as a *chosen plaintext attack* and is the strongest attack that Eve can perform. (See Figure 1.6.)

On the face of it we now seem to be overestimating Eve's capabilities to influence Alice and Bob's communications. However, in practice it is reasonable to suppose that Eve can conduct a chosen plaintext attack. For instance she may be a 'friend' of Alice and so be able to influence the messages Alice chooses to send. Another important possibility is that Alice and Bob use a public key

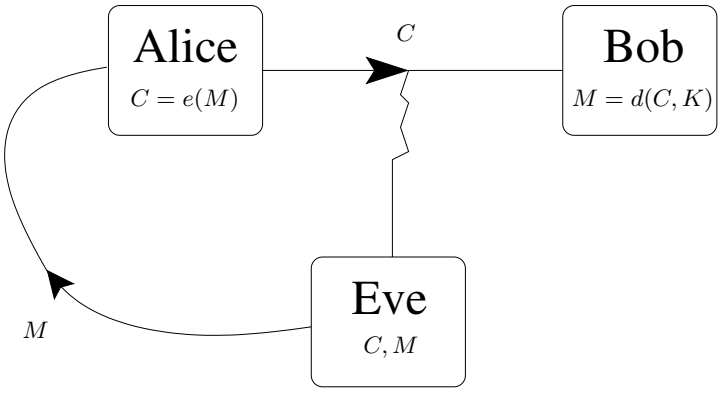

Fig. 1.6 Eve performs a chosen plaintext attack.

Fig. 1.7 Alice and Bob using a cryptosystem attacked by Mallory.

cryptosystem and so Eve can encrypt any message she likes since encryption does not depend on a secret key.

Certainly any cryptosystem that cannot withstand a chosen plaintext attack would not be considered secure.

From now on we will assume that any adversary has access to as many chosen pairs of messages and corresponding cryptograms as they can possibly make use of.

There is a different and possibly even worse scenario than Eve conducting a chosen plaintext attack. Namely Mallory, the malicious attacker, might interfere with the cryptosystem, modifying and even replacing messages in transit. (See Figure 1.7.)

The problems posed by Mallory are rather different. For example, he may pretend to be Bob to Alice and Alice to Bob and then convince them to divulge secrets to him! We will see more of him in Chapter 9.

We now need to decide two things.

(1) What can Eve do with the message-cryptogram pairs she obtains in a chosen message attack?
(2) What outcome should Alice and Bob be happy with?

There are two very different approaches to cryptographic security, depending essentially on how we answer these questions.

Historically the first rigorous approach to security was due to Shannon (1949a). In his model Eve is allowed unlimited computational power and Alice and Bob then try to limit the 'information' Eve can obtain about future messages (and Bob's secret key) given her message-cryptogram pairs. He was able to show that there are cryptosystems that are perfectly secure in this model. However, he also showed that any such cryptosystem will have some rather unfortunate drawbacks, principally the key must be as long as the message that is sent.

Modern cryptography is based on a complexity theoretic approach. It starts with the assumption that Eve has limited computational resources and attempts to build a theory of security that ensures Eve is extremely unlikely to be able to read or obtain any useful information about future messages.

We briefly outline the two approaches below.

1.3 Classical cryptography

Consider the following situation. Alice wishes to send Bob n messages. Each message is either a zero or a one. Sometime earlier Alice and Bob met and flipped an unbiased coin n times. They both recorded the sequence of random coin tosses as a string $K \in \{\mathrm{H}, \mathrm{T}\}^n$ and kept this secret from Eve.

Alice encrypts her messages $M_1, M_2, \ldots, M_n$ as follows.

$$C_i = e(M_i) = \begin{cases} M_i, & \text{if } K_i = \mathrm{H}, \\ M_i \oplus 1, & \text{if } K_i = \mathrm{T}. \end{cases}$$

(Here $\oplus$ denotes 'exclusive or' (XOR), so $0 \oplus 0 = 1 \oplus 1 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$.)

Alice then sends the cryptograms $C_1, \ldots, C_n$ to Bob, one at a time.

Bob can decrypt easily, since he also knows the sequence of coin tosses, as follows

$$M_i = d(C_i) = \begin{cases} C_i, & \text{if } K_i = \mathrm{H}, \\ C_i \oplus 1, & \text{if } K_i = \mathrm{T}. \end{cases}$$

So encryption and decryption are straightforward for Alice and Bob. But what about Eve? Suppose she knows both the first $n-1$ cryptograms and also the corresponding messages. Then she has $n-1$ message-cryptogram pairs

$$(C_1, M_1), (C_2, M_2), \ldots, (C_{n-1}, M_{n-1}).$$

If Eve is then shown the final cryptogram C_n what can she deduce about M_n?

Well since K_n was a random coin toss there is a 50% chance that $C_n = M_n$ and a 50% chance that $C_n = M_n \oplus 1$. Since K_n was independent of the other key bits then knowledge of these will not help. So what can Eve do?

Suppose for the moment that the messages that Alice sent were also the result of another series of independent coin tosses, that is they were also a random sequence of zeros and ones. In this case Eve could try to guess the message M_n by tossing a coin herself: at least she would have a 50% chance of guessing correctly. In fact this is the best she can hope for!

But what if the messages were not random? Messages usually contain useful (non-random) information. In this case Eve may know something about how likely different messages are. For instance she may know that Alice is far more likely to send a one rather than a zero. If Eve knows this then she could guess that $M_n = 1$ and would be correct most of the time. However, she could have guessed this *before* she saw the final cryptogram C_n. Eve has gained *no new information* about the message by seeing the cryptogram. This is the basic idea of perfect secrecy in Shannon's model of cryptography.

- The cryptogram should reveal no new information about the message.

This theory will be developed in more detail in Chapter 5.

1.4 Modern cryptography

Modern cryptography starts from a rather different position. It is founded on complexity theory: that is the theory of how easy or difficult problems are to solve computationally.

Modern cryptographic security can informally be summarised by the following statement.

- It should not matter whether a cryptogram reveals information about the message. What matters is whether this information can be efficiently extracted by an adversary.

Obviously this point of view would be futile if we were faced with an adversary with unbounded computational resources. So we make the following assumption.

- Eve's computational resources are limited.

If we limit Eve's computational resources then we must also limit those of Alice and Bob. Yet we still require them to be able to encrypt and decrypt messages easily. This leads to a second assumption.

- There exist functions which are 'easy' to compute and yet 'hard' to invert. These are called *one-way functions*.

Given this assumption it is possible to construct cryptosystems in which there is a 'complexity theoretic gap' between the 'easy' procedures of decryption and encryption for Alice and Bob; and the 'hard' task of extracting information from a cryptogram faced by Eve.

To discuss this theory in detail we need to first cover the basics of complexity theory.

2

Complexity theory

2.1 What is complexity theory?

Computers have revolutionised many areas of life. For example, the human genome project, computational chemistry, air-traffic control and the Internet have all benefited from the ability of modern computers to solve computational problems which are far beyond the reach of humans. With the continual improvements in computing power it would be easy to believe that any computational problem we might wish to solve will soon be within reach. Unfortunately this does not appear to be true. Although almost every 'real' computational problem can, in theory, be solved by computer, in many cases the only known algorithms are completely impractical. Consider the following computational problem.

Example 2.1 *The Travelling Salesman Problem.*

Problem: given a list of n cities, $c_1, c_2, \ldots, c_n$ and an $n \times n$ symmetric matrix D of distances, such that

$$D_{ij} = \text{distance from city } c_i \text{ to city } c_j,$$

determine an optimal shortest tour visiting each of the cities exactly once.

An obvious naive algorithm is: 'try all possible tours in turn and choose the shortest one'. Such an algorithm will in theory work, in the sense that it will eventually find the correct answer. Unfortunately it will take a very long time to finish! If we use this method then we would need to check $n!$ tours, since there are $n!$ ways to order the n cities. More efficient algorithms for this problem exist, but a common trait they all share is that if we have n cities then, in the worst case, they may need to perform at least 2^n operations. To put this in perspective suppose we had $n = 300$, a not unreasonably large number of cities to visit.