CS3381-Cryptography

Lecture 3: One-time Pad, Perfect Secrecy, and Practical Security

September 8, 2014

1 One-time pad

Suppose we use the Vigenère cipher with longer and longer keys. The farthest we can go with this is to make the key as long as the plaintext, so instead of many repetitions of the short key you have a single occurrence of the long one. Here is an example with the plaintext "LAUNCHTHEATTACK" and key "NOWISTHETIMEFOR".

L	A	U	N	C	H	T	H	E	A	T	T	A	C	K
N	O	W	I	S	T	H	E	T	I	M	E	F	O	R
Y	O	Q	V	U	A	A	L	X	I	F	X	F	Q	B

This is called the *one-time pad,* or *Vernam cipher*.

The attack on Vigenère does not work at all here. With no repetition of the keyword, our method of inferring the key length is not applicable. If we know in advance that the key has the same length as the plaintext, the method for determining the key is of no use, since we would begin by partitioning the text into subtexts, each of which has only one symbol. We would conclude in every case that the plaintext is "EEEE...."!

In fact, by adjusting the key, we could get the same ciphertext with *any* plaintext of the same length; for example.

S	U	R	R	E	N	D	E	R	A	T	O	N	C	E
G	U	Z	E	Q	N	X	H	G	I	M	J	S	O	X
Y	O	Q	V	U	A	A	L	X	I	F	X	F	Q	B

Thus the ciphertext does not help us at all.

2 What's 'one-time' about the one-time pad?

If you use the key in a one-time pad to encrypt two different plaintexts, then the resulting ciphertexts differ in exactly the same places, and by exactly the same amount. (The same is true of the Vigenère cipher, which is a special case of the one-time pad.) If one of the two plaintexts is later recovered, then we obtain the key itself, and thus the second plaintext, and all other plaintexts that are encrypted using the same key. Thus *the key in a one-time pad can never be re-used.*

3 Perfect secrecy

Intuitively, the ciphertext obtained with a one-time pad gives away *no* information about the plaintext, except for its length. This property (which we have formulated only vaguely) is called *perfect secrecy.*

To formulate a precise definition of perfect secrecy, imagine the following scenario. We intercept a ciphertext c, and we suspect that it is either the encryption of 'ATTACKATTHEFIRSTOPPORTUNITY' or of 'SURRENDERASSOONASYOUAREABLE', but do not know which of these two plaintexts is the right one. The ciphertext leaks *some* information about the plaintext if it allows us to say which of these two plaintexts is more likely. We say that encryption scheme has *perfect secrecy* if for all plaintexts m_1, m_2, and all ciphertexts c, we have

$$\Pr_{k\in\mathcal{K}}[E(k, m_1) = c] = \Pr_{k\in\mathcal{K}}[E(k, m_2) = c].$$

Here the probabilities are assigned based on choosing the key k uniformly and at random from the set $\mathcal{K}$of all possible keys. (We assume here that the set $\mathcal{M}$ of all possible plaintexts consists of strings of some fixed length, and that c belongs to the corresponding set $\mathcal{M}'$ of possible ciphertexts.)

We will prove the following two facts:

1. The one-time pad has perfect secrecy.
2. If a cryptographic system has perfect secrecy, then $|\mathcal{K}| \geq |\mathcal{M}|$. That is, there are at least as many keys as possible plaintexts.

To prove item 1, note that for the one-time pad there is exactly one key k out of the $|\mathcal{K}|$ possible keys that encrypts m_1 to c, and similarly exactly one key k' such that $E(k', m_2) = c$. So

$$\Pr_{k\in\mathcal{K}}[E(k,m_1)=c] = \Pr_{k\in\mathcal{K}}[E(k,m_2)=c] = 1/|\mathcal{K}|,$$

which is what we wanted to prove.

For item 2, suppose $|\mathcal{K}| < |\mathcal{M}|$. We'll show that the cipher does not have perfect secrecy. Choose any plaintext $m_1 \in \mathcal{M}$ and a key $k^* \in \mathcal{K}$. Set $c = E(k^*, m_1)$. Now decrypt c under all possible keys. The resulting set of plaintexts contains m_1, but since there are only $|\mathcal{K}| < |\mathcal{M}|$ distinct keys, there must be some element $m_2 \in \mathcal{M}$ that does not encrypt to c under any key. Thus

$$\Pr_{k\in\mathcal{K}}[E(k,m_1)=c] \geq 1/|\mathcal{K}| > 0 = \Pr_{k\in\mathcal{K}}[E(k,m_2)=c],$$

so we don't have perfect secrecy.

4 If you can't have perfect secrecy, what can you have?

We have seen that if you want perfect secrecy, you have to have a key space that is at least as large as the message space, and can never re-use a key. But the proof of this fact also shows that it might be ok to settle for less than perfect secrecy.

Let's go back to the scenario with the 'ATTACK' and 'SURRENDER' plaintexts, and suppose we are using a cipher in which the number of keys is strictly less than the number of possible plaintexts. The proof of item 2 that we gave above provides a strategy for an eavesdropper Eve to guess which of the two messages was sent: She computes the decryption of the intercepted ciphertext c under all possible keys, giving a set S of possible plaintexts. Assuming that the plaintext is one of these two messages, then at least one of m_1, m_2 will belong to S. If only one of the two messages belongs to S, then Eve will know what the plaintext was. If they both belong to S, then Eve flips a coin.

How likely is it that this strategy succeeds?

Suppose the sender is sending the attack message. A key k is picked at random, and the message is encrypted with k, giving the ciphertext c. There is at least one plaintext message that does not encrypt to c under any key (*i.e.,* at least one plaintext message that is not in S). The probability that this plaintext message is the surrender message is thus at least $1/|\mathcal{M}|$. In this case Eve's strategy is sure to succeed. The probability that the surrender message is in S is at most $1 - 1/|\mathcal{M}|$, and in this case Eve's strategy succeeds with probability $1/2$. So if

the attack message was chosen, Eve succeeds with probability

$$\frac{1}{2} \cdot (1 - 1/|\mathcal{M}|) + 1 \cdot 1/|\mathcal{M}| = \frac{1}{2} + \frac{1}{2|\mathcal{M}|}.$$

The probability is the same assuming that the surrender message was sent.

In practice, there are two problems with this approach. The first is that the *computational effort required is infeasible*. There are 26^{27} possible plaintexts, so there may be as many as $26^{27} - 1 > 10^{38}$ different keys to try to construct the set S. Second, the probability of success is only *negligibly* better than $1/2$, since $\frac{1}{2|\mathcal{M}|} < 10^{-38}$.

If we instead reduced the keyspace by a large factor–say, we used 20-letter characters rather that 27 letters—then the probability that the encryption of the surrender message is outside of S is now very high, and the attack is almost certain to succeed, but we still have the problem of the computational difficulty of mounting it.

For a system to be secure, there should be no *feasible* algorithm that gives more than a *negligible* advantage over random guessing.

Notions like 'feasible' and 'negligible' seem rather fuzzy, and are dependent on the current state of technology. It is probably safe to say that an algorithm requiring $2^{80} > 10^{24}$ steps is infeasible, and that a probability of 2^{-80} is negligible. Computer scientists often use the following definition as a stand-in for feasibility of an algorithm: A feasible algorithm is one that has a running time that grows as a polynomial function of its input size (*e.g.,* running time N^2 on inputs of length N, rather than exponential run times like 2^N.) This connects cryptography with computational complexity theory. Unfortunately, we do not know how to *prove* that a given problem has no polynomial-time algorithms (this is one of the biggest unsolved problems in mathematics!) There are some candidate problems that are widely believed to have no polynomial-time algorithm, and right now the best we can do is prove that if there were efficient attacks on a cryptosystem, then there would be efficient algorithms to solve these problems. Such a result is taken as evidence that the system is secure. We will see this approach later when we study cryptographic systems whose security rests on the difficulty of factoring an integer into its prime factors.