Cryptology from Roman Days to Electronic Times

Henk van Tilborg

Eindhoven University of Technology

Cryptology goes back to Julius Caesar or even further. Most of these old systems look very amateurish, although a system like Vigenère remained secure for almost three centuries. During World War II, smartly designed mechanical and electro-mechanical cryptosystems were in use. Shortly later, completely electronical circuits replaced the old systems. Nowadays, cryptosystems are mostly implemented on chips.

Introduction

Starting in the late Seventies, completely different crypto-systems were proposed. They are ideally suited for communication systems that are completely controlled by computers. In these new cryptosystems, no secret agreements are necessary. They make use of mathematical methods, like elementary number theory and abstract algebra. It turns out that they also make it possible to add a digital signature to files. In many practical situations this is more important than privacy.

Classical systems

Historians tell us that Julius Caesar was the first to make use of a cryptosystem. He used it to keep his communication safe from his enemies in Rome. Caesar simply replaced every letter by the letter five places further in the alphabet (so his name becomes "Ozqnzx Hfjxfw"). Of course, the five (called the *key*) in this system can be replaced by any other value less than 26, but that still does not create enough possibilities to prevent breaking the system by exhaustive methods. The reader is invited to decrypt the Latin phrase 'wxtxrtxheuxwh'.

The recordholder in security, already mentioned in the abstract, was proposed in 1586 by *Vigenère* and consists of a suitable number of different Caesar ciphers applied periodically, for example with the key 'xootic' one gets the following encryption:

```
c r y p t o l o g y i s f a s c i n a t i n g
x o o t i c x o o t i c x o o t i c x o o t i
z f m i b q i c u r q u c o g v q p x h w g o
```

The basic difficulty for the *cryptanalist* with this system is the uncertainty about the length of the key word/phrase. It lasted until 1863 before the system was broken by statistical means. The reader who is interested in these old cryptosystems is referred to [3].

DES

A well-known designer's principle in cryptography is to repeatedly alternate between a substitution of symbols by other symbols and a permutation of their order. This is exactly what happens in the succesful *Data Encryption Standard* (DES). It was introduced in 1977 and implemented on a chip. DES (see Figure 1) operates on 64 bits at a time. In sixteen separate rounds they are permuted and partially replaced by other bits. The permutation used in each round is a cyclic

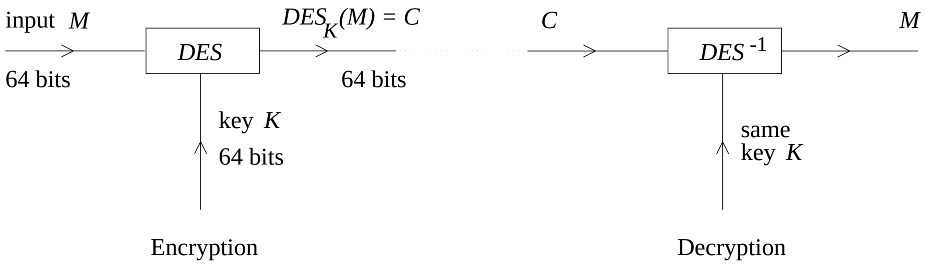

Figure 1. The Data Encryption Standard

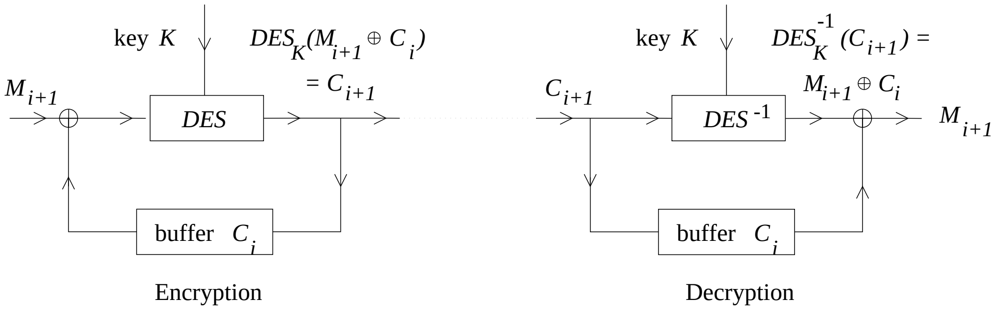

Figure 2. Cipherblock chain

shift over 32 positions, while the substitution in each round is determined by a key which is derived from the DES key K. The DES key consists of 64 bits of which 56 can be chosen at random. Some critics find this number too small to resist an exhaustive key search with today's technology.

The DES system is implemented as a component in many information security systems, although the key length is no longer considered secure enough. In order to avoid that the same plaintext will be encrypted each time (under the same key) into the same ciphertext, one can use methods like *cipherblock chaining* (see Figure 2), in which the last ciphertext is XOR-ed componentwise with the new plaintext (at the receiving end the opposite has to be done)

On many smartcard applications DES is not only used for encryption purposes, but also for *authentication* purposes. When a smart card is inserted into a card reader they want to verify that the other is genuine. On the card of a certain person, say Ann, an identity number I_{Ann} is stored that is presented to the reader. Also a secret key K_{Ann} is stored. A genuine card reader can compute the secret key K_{Ann} by means of Ann's identity number (the alternative that each card reader has to store the secret keys of all possible cards is for obvious reasons too unattractive).

Ann's card checks the card readers authenticity by generating a random string M of 64 bits and sending this to the card reader. The card reader has to send back $DES_{K_{\text{Ann}}}(M)$. Ann's smart card checks this calculation and if the outcome is correct it will trust the card reader. In Figure 3, this authenticity checking protocol is depicted

In the same way as above, the card reader can check

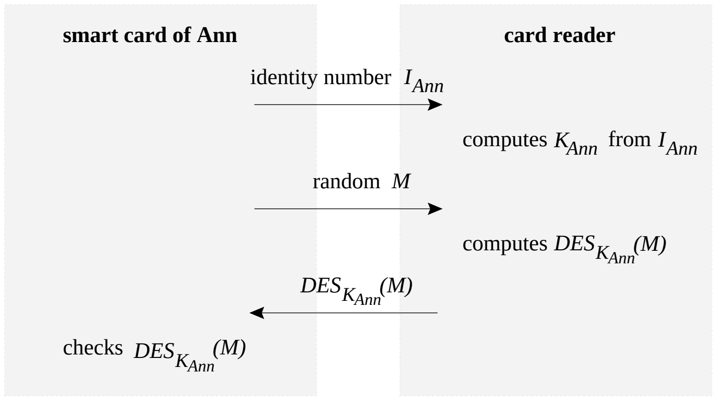
smart card of Ann
card reader
identity number I_{Ann}
computes K_{Ann} from I_{Ann}
random M
computes $DES_{K_{Ann}}(M)$
$DES_{K_{Ann}}(M)$
checks $DES_{K_{Ann}}(M)$

Figure 3. A protocol for checking the authenticity of the card reader

that the smart card which presented Ann's identity number I_{Ann} indeed knows Ann's secret key K_{Ann} and thus must be Ann's card.

Public key cryptography

All the systems explained above are of the so-called *conventional* type, which means that sender and receiver must have agreed upon a common secret key. In modern applications, such as large computer controlled communication networks, there are simply too many users to establish distinct keys between every pair. Similarly they may be too far apart or they do not know each other in advance.

In 1976, W. Diffie and M. E. Hellman [1] introduced an entirely new concept: *public key cryptography*. Here, a user, say U, makes two algorithms, one (to be called D_U) remains secret, the other (E_U) has to be made public. Of course some properties must hold. For instance, a cryptanalist should not be able to determine the secret algorithm D_U from the public algorithm E_U.

If Ann wants to send a secret message m to Bob, she can look up Bob's public encryption algorithm E_{Bob} and send $c = E_{\text{Bob}}(m)$. Bob can retrieve m from c by computing $D_{\text{Bob}}(c)$. Here, we need that $D_{\text{Bob}}(E_{\text{Bob}}(m)) = m$ for all possible messages m.

Much more is possible [4]. In the next section, we show a particular public key cryptosystem used to add a digital signature to an electronic document.

A digital signature scheme

In many applications secrecy is not the issue, but the *integrity* and *authenticity* of the data are. Assume that Ann is sending a long file to Bob. She wants to append a small tail to the file that allows Bob to check that the received file has not been changed and that it indeed comes from Ann.

The first step is to compress the file f to a tail, say t, by means of a mapping h. So $t = h(f)$. The mapping h is called a *hash function*. It should be cryptographically secure. This implies in particular that it is not possible for somebody else to find a second file f' with $h(f) = h(f')$. In this way, nobody else can replace the file f by f', while keeping the same tail t. (In this way integrity of the data will be guaranteed.)

The hash value t does not prove to Bob that the message came from Ann. In [2], it is shown how this can be realized. To this end, we assume that the tail t above is represented by an integer in between 0 and a large prime p. All further calculations will be performed *modulo* some *modulus*. This means that any upcoming number will replaced by the value of its (non-negative) remainder after division by the modulus. Before we can explain the system, we need to explain the mathematical tools that will be needed. It turns out that the *ElGamal digital signature scheme* relies in an essential way on the following two properties.

1. Exponentiation modulo a prime p is quite doable, even for large values of p. Indeed, it takes at most $2^2 \ln p$ multiplications, as the following example shows (note how the binary representation 101101011 of the exponent 363 has been used: a 0 stands for squaring, a 1 for squaring followed by a multiplication by m):

$$m^{363} = ((((((((((((m^2)^2)m)^2)m)^2)^2)m)^2)^2)m)^2)m$$

2. Taking logarithms modulo p, i.e. solving m from

$$g^m \equiv c (\operatorname{mod} p)$$

with known g and c is computationally infeasible for large p. In other words, there is an exponential relation between the complexity of exponentiation and that of taking logarithms.

In ElGamal's signature scheme p and g are system parameters shared by all participants, e.g. p is a 100 digits long prime and g is something like 2 or 3. Each user U chooses a random exponent m_U, computes $c_U = g^{m_U} (\operatorname{mod} p)$ and makes the value of c_U public.

When Ann wants to append a tail to file f to serve as her digital signature on f, she proceeds as follows:

1. Ann computes the hash value $h(f)$ of the file f,
2. Ann chooses a random number r, $1 \le r \le p-2$, coprime with $p-1$,
3. Ann computes $R \equiv g^r (\operatorname{mod} p)$ and $S \equiv (h(f) + m_{\text{Ann}} R)/r (\operatorname{mod} p-1)$.
4. Ann appends R and S as her signature to the file f.

Advertentie Philips

How can anybody, say Bob, check the authenticity and integrity of the file f? This is depicted below:

1. Bob computes the hash value $h(f)$ of the file f,
2. Bob looks up Ann's public c_{Ann},
3. Bob checks if $g^{h(f)} c_{\mathrm{Ann}}^{\;R} \equiv R^{S} (\bmod p)$.

It is a matter of simple substitution to verify that the relation above should hold. Of importance is to notice that only Ann could have made the signature R and S, because only she knows the secret m_{Ann}. The reader may wonder why the random number r is introduced when Ann wants to add a digital signature to a file f. The reason is that Ann wants to use her secret exponent m_{Ann} over and over again. The value of m_{Ann} is in the current scheme, each time it is used, hidden by a (different) random r.

Conclusion

Cryptography has evolved from the secret world of military applications to a discipline that finds its applications whenever parties want to communicate over open networks in a secure way. The methods have changed from ingeniously designed systems, via complicated mechanical devices to algorithms relying on advanced mathematical theories. □

References

[1] Diffie, W. and M.E. Hellman, *New Direction in Cryptography*, IEEE Trans. Inf. Theory, IT-22, p. 644-654, 1976.

[2] ElGamal, T., *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory, IT-31, p. 469--472, 1985.

[3] Kahn, D., *The codebreakers, the story of secret writing*, Macmillan Company, New York, 1967.

[4] Van Tilborg, H.C.A., *An introduction to cryptology*, Kluwer Academic Publishers, Boston, 1988.

Pasfoto

Tilborg

Henk van Tilborg obtained his PhD in 1976 at the Eindhoven University of Technology. He has spent sabbatical periods at AT&T Bell Laboratories, Jet Propulsion Laboratories, California Institute of Technology, IBM Almaden Research Center, and the University of Pretoria. Since 1992 he is a full professor in Coding Theory and Cryptology at EUT.