Cryptography and Security
Executive Summary

Serge Vaudenay

EPFL
Lausanne, Switzerland
http://lasec.epfl.ch

1 Ancient Cryptography

Originally, the goal of cryptography was to protect the privacy of information by means of encryption. Modern cryptography is founded on the principle that users of encryption would take an available algorithm on which they could easily configure a *secret key*. The algorithm could be public (for instance because it is patented or a standard) or a trade secret (for business reasons). However, the *Kerckhoffs principle* states that we should not assume that the adversary who tries to break the scheme ignores how the algorithm works. The adversary knows the algorithm. Also, security naturally degrades because technology improves, and so is the power of the adversary. The *Moore law* states an exponential grows of the computational ability with time, which requires to regularly increase the security parameters of cryptographic schemes, until the Moore law is not valid any more.

The *Vernam cipher* formalizes naturally in a *group*. A group is an algebraic notion. It is a set in which we can do an operation (for instance, an addition) which is invertible. Following the Vernam cipher, a message is a group element. A secret key is a group element which is selected uniformly at random. The encryption is the sum of the message and the key. It is required that the key is used only once, hence this is also called *one-time pad*. Since the message and the key live in the same domain, the key is de facto at least as long as the message. Except in some corner applications such as the *red phone*, this is not convenient at all.

We can formalize the notion of *perfect secrecy* based on information theory. This was done by *Shannon*. It ensures that an adversary, after seeing an encrypted messages, cannot learn any information about the message which was not known before. This notion can be reached by the Vernam cipher. Shannon proved that perfect secrecy requires the key to be at least as long as the message, so there is no better solution than the Vernam cipher which reaches this security goal. What is missing in this notion is the cost to retrieve the information: the *complexity*. Henceforth, we later focus on algorithm with imperfect secrecy but in which the complexity of an attack is not affordable for an adversary.

2 Diffie-Hellman Cryptography

The *Diffie-Hellman protocol* allows to establish a secret key between two participants who can only communicate over a public channel and who share no prior secret. It works on a *group*. This is an algebraic notion for a set with an invertible operation which can be an addition or a multiplication.

One typical group which is used is a subgroup of prime order q of $\mathbf{Z}_p^*$, where p is a prime number. This implies that q divides $p-1$. Given an integer n, $\mathbf{Z}_n$ is the set of integers between 0 and $n-1$ which is given with the addition and the multiplication modulo n. It forms a *ring*, which is another algebraic notion. The set $\mathbf{Z}_n^*$ is the set of all ring elements which have a multiplicative

inverse. In those structures, we have efficient algorithms to do computations. For instance, the *square-and-multiply algorithm* allows to compute exponentials efficiently and the *extended Euclid algorithm* allows to compute multiplicative inverses.

For the Diffie-Hellman protocol to be secure, we need the *decisional Diffie-Hellman problem* (DDH) to be hard on the group. This requires the group to have a prime order. This implies the hardness of the *computational Diffie-Hellman problem* (CDH), which implies the hardness of the *discrete logarithm problem* (DL). All those problems are described by *games* and the hardness implication is proven by *reduction techniques*.

$$\mathsf{DDH\ hard} \Longrightarrow \mathsf{CDH\ hard} \Longrightarrow \mathsf{DL\ hard}$$

In a group of order n, there are generic algorithms to solve the discrete logarithm problem in $\mathcal{O}(\sqrt{n})$ time (which is exponential in the size of n). In $\mathbf{Z}_p^*$, the best algorithm is GNFS (which is subexponential but non-polynomial in the size of n). It requires a huge precomputation time. Once precomputation is done, each discrete logarithm is much faster to compute. Hence, if many people use the same parameter p, this shall be taken into account.

The Diffie-Hellman protocol can be converted into the *ElGamal cryptosystem*. It requires messages to be group elements. However, it is not always easy to encode a message into a group element. We can study security by game reduction techniques. For instance, the key recovery problem (i.e., the problem of the adversary to find the secret key of a user) is equivalent to the DL problem. The message decryption problem (i.e., the problem of the adversary to decrypt a message sent to a user) is equivalent to the CDH problem. And the IND-CPA security (which is the standard security notion to be discussed in another chapter) is equivalent to the DDH problem.

The DL problem will be very easy once quantum computers will be available, due to the *Shor algorithm*. Hence, the other mentioned problems will fall too.

3 RSA Cryptography

When p and q are coprime, making rational operations in $\mathbf{Z}_{pq}$ can be seen as equivalent to computing in $\mathbf{Z}_p$ and $\mathbf{Z}_q$ in parallel, thanks to the *Chinese remainder theorem*. This finds numerous applications. The RSA cryptosystem relies on the $\mathbf{Z}_{pq}$ structure. It requires that each user who sets up its private/public key pair generates his own prime numbers p and q. To generate a prime number, we pick a number at random and test if it is a prime until we find one. Testing primality can efficiently be done. The *Miller-Rabin* primality test is as efficient as computing an exponential in $\mathbf{Z}_n$.

With RSA, key recovery is equivalent to the *factoring problem*. Some equivalent problems are the problem of computing orders, computing $\varphi(n)$, or any multiple of it.[1] Decryption is called the *RSA problem*. Factoring will be very easy on quantum computers, with the Shor algorithm. To factor RSA numbers on a classical computer, the best algorithm is the *NFS algorithm*. To factor numbers which have an exceptionally small factor, *ECM* is the best method. Both have subexponential complexity in the size of n.

4 Elliptic Curve Cryptography

Fields are rings in which any non-zero element is invertible. In this course, we mostly focus on the *prime field* $\mathbf{Z}_p$, where p is a prime number. We may use as well a *binary field* $\mathsf{GF}(2^k)$, which has 2^k elements, and which can be seen as the set of all polynomials with coefficients equal to 0 or 1 and degree bounded by $k-1$. Addition is done modulo 2 and multiplication is done modulo a fixed irreducible polynomial of degree k to specify.

An *elliptic curve* is defined over a field by an equation which has some parameters. It is a set of points. Except the *point at infinity* which is special, each point has two coordinates, x and y,

[1] φ is the *Euler totient function*. $\varphi(n)$ is the order of $\mathbf{Z}_n^*$.

which are field elements. Since x and y satisfy a relation, we can "compress" a point by specifying only x and one bit of y. An elliptic curve has a group structure in which point addition is defined by some rational formulas. Scalar multiplication (i.e., multiplying a point by an integer) follows by using the *double-and-add* algorithm. An elliptic curve has a cardinality which is close to the cardinality of the field.

The *j-invariant* of the curve characterizes the algebraic structure of the elliptic curve. Two curves with same j-invariant are either isomorphic or *twist* of each other. Actually, two twists are simply isomorphic if we define them over a bigger field.

The DL problem is believed to be hard in most of curves. However, there exist very special curves on which is it easy. Hence, it is not recommended to use special curves unless we know exactly what we are doing. It is better to use pseudorandom curves. There exist standards which propose concrete curves which were generated at random. Most of standards propose the very same curves under different names.

Cryptographic schemes are adapted to elliptic curves. ECDH is the elliptic-curve Diffie-Hellman protocol. ECIES is a hybrid encryption scheme inspired from the ElGamal cryptosystem. ECDSA is a digital signature scheme based on elliptic curves. Contrarily to $\mathbf{Z}_p^*$ in which GNFS computes discrete logarithms in subexponential time, only generic methods (with exponential complexity) seem to apply to elliptic curves. Hence, they can provide the same security with much smaller parameters and there is a significant complexity advantage in using them.

With some special curves, we can construct *pairing*. These are bilinear functions which allow to construct new cryptographic schemes. For instance, identity-based cryptography is possible using pairing.

5 Symmetric Encryption

Stream ciphers essentially consist of using the Vernam cipher. A stream cipher generates a keystream from a key and a *nonce* (a number to be used only once). *Block ciphers* allow to encrypt a block of a fixed length (typically 128 bits) into a block of same length. There exist several families of block cipher designs, the most popular being the ones based of the Feistel scheme (like DES) and substitution-permutation networks (like AES).

Block ciphers can be transformed into a variable-length encryption scheme by using a *mode of operation*. The most naïve mode of operation, ECB, which encrypts the blocks separately in the same manner, is very weak in many applications and should not be used. The most popular modes of operation are CBC, CTR, OFB. CBC needs the plaintext to have a length multiple of the block length. CTR and OFB actually make a stream cipher from a block cipher and need a nonce. CTR can encrypt arbitrary parts of the message in parallel while CBC and OFB need to treat it sequentially.

Breaking a symmetric encryption scheme is typically done by a *key recovery*. This can be done by bruteforce, which has a cost proportional to 2^ℓ, where ℓ is the bitlength of the key. An encryption scheme is secure when bruteforce exhaustive search is the best attack strategy we can find. To derive any concrete conclusion on the cost of this attack method, one should keep in mind the Moore law which says that the computational power of hardware increases with time. The key length should be adjusted accordingly.

The cost of attacks may decrease if the goal is to break at least one instance in a set of targets. This is the case of inversion attacks on password hash systems. To mitigate multi-target attacks, we use *salt* as an additional input. A salt is not secret but changes for each potential target.

Finally, there are non-trivial attack strategies. For instance, the *time-memory tradeoff* precomputes tables of size $2^{\frac{2}{3}\ell}$ which can be used to attack one target with complexity $2^{\frac{2}{3}\ell}$. In another instance, the *meet-in-the-middle* attack breaks double-encryption at the cost of breaking single encryption. This is why people use triple-encryption.

6 Integrity and Authentication

Hash functions process a variable-length document and output a bitstring of fixed length (e.g. 256 bits). A hash function is the Swiss-army knife of cryptography. It is used to extract of kind of unique fingerprint from documents, if it is collision-resistant. It can also used for key derivation or for commitment. There are hash functions based on the Merkle-Damgård scheme such as functions in the SHA-2 family (like SHA256). There are others based on sponges like SHA-3.

In communication, message authentication and message integrity are enforced by the same cryptographic scheme: the *message authentication code* (MAC). A popular MAC is the HMAC algorithm which is based on a Merkle-Damgård hash function. There is also the CBCMAC construction based on a block cipher in CBC mode, but it should not be used alone due to some attacks. Either it should be encrypted, or it should be modified like in CMAC. The last MAC family is based on *universal hashing* and a stream cipher. Note that universal hashing is a combinatorial notion which is not related with the previously mentioned cryptographic hashing.

Some block ciphers can be used in *authenticated modes of operation* like CCM or GCM. This way, we have encryption, authentication, and integrity protection, all in the same algorithm.

Bruteforce attacks work like for symmetric encryption, except when it comes to look for collisions on hash function. For this, we rely on the *birthday paradox* which says that the cost to find a collision on a hash function hashing onto ℓ bits is proportional to $2^{\frac{1}{2}\ell}$.

7 Public-Key Cryptography

Diffie and Hellman invented public-key cryptography, with the notions of trapdoor permutation, public-key cryptosystem, key agreement, and digital signature. Signatures are typically used to sign public-key certificates.

Rivest, Shamir, and Adleman proposed RSA, which can be used for encryption and signature. The RSA standard for that is PKCS#1. The latest algorithms in this standard are RSA-OAEP for encryption and RSA-PSS for signature.

ElGamal cryptography is based on the Diffie-Hellman key agreement protocol. Besides the ElGamal cryptosystem (and its elliptic-curve variant), there is an ElGamal signature scheme and many variants, including the standards DSA and ECDSA. In all ElGamal signature variants, the randomness in the signature scheme is critical, as bad randomness often leads to key-recovery attacks.

For cryptographic schemes, parameters such as key lengths should be selected in a consistent manner. For this, as a rule of the thumb, the hash length should be twice the symmetric key length. The group order length should also be twice the symmetric key length, for Diffie-Hellman-based cryptography. The modulus size in RSA and in DSA should be of similar length, following tables to compare with symmetric key lengths. Roughly, a 128-bit symmetric key and a 2048-bit RSA modulus could be compared.

The standard security notion for a public-key cryptosystem is the IND-CCA notion, which says that we cannot distinguish the encryption of two given plaintexts, even with access to a decryption machine, except in a trivial way. The standard security notion for digital signature is the EF-CMA notion, which says that we cannot forge a signed message, even with access to a signing machine, except in a trivial way.

Both RSA-based and Diffie-Hellman-based cryptography will be broken by quantum computers. To get ready for replacements, some new algorithms are under standardization: *post-quantum* algorithms. So far, most of proposals are based on lattice-related problems.

8 Trust Establishment

There exist several techniques for access control protocols. Authentication by "what you know" is typically based on providing a password or a PIN code. Authentication by "what you possess"

can be a challenge-response protocol with a device holding a secret key. Authentication by "what you are" is based on biometry. Strong authentication use at least two of these methods.

Passwords can be used in password-authenticated key exchange (PAKE) which is a key agreement protocol enriched with password-based authentication. Without authentication, key agreement protocols are vulnerable to man-in-the-middle attacks. The difficulty of PAKE is to authenticate exchanges without releasing any information which can be used later on to do a bruteforce attack on the password in an offline manner.

To achieve secure communication, we use symmetric cryptography. However, we need to set up a session symmetric key. For that, we need a setup assumption. We can assume an authenticated public key and use public-key cryptography. We can assume a pre-establish password and use PAKE. We can assume a narrowband authenticated channel and use protocols based on, for instance, the comparison of two numbers by a human operator. We can also assume a third party. It can be a Kerberos server (which escrows all keys), a certificate authority (to sign certificates in a public-key infrastructure), or a server in an identity-based infrastructure.

Case Studies

The lecture is illustrated with a few case studies which are briefly presented:

- WiFi security;
- block chains (bitcoin);
- mobile telephony (GSM, 3G);
- Signal;
- NFC creditcard payment (EMV PayPass);
- Bluetooth (pairing);
- the biometric passport (ICAO MRTD);
- TLS.