Séminaires & Congrès
11, 2005, p. 41–73

MATHEMATICAL BACKGROUND OF PUBLIC KEY CRYPTOGRAPHY

by

Gerhard Frey & Tanja Lange

***Abstract.* —** The two main systems used for public key cryptography are RSA and protocols based on the discrete logarithm problem in some cyclic group. We focus on the latter problem and state cryptographic protocols and mathematical background material.

***Résumé* (Éléments mathématiques de la cryptographie à clef publique). —** Les deux systèmes principaux de cryptographie à clef publique sont RSA et le calcul de logarithmes discrets dans un groupe cyclique. Nous nous intéressons aux logarithmes discrets et présentons les faits mathématiques qu'il faut connaître pour apprendre la cryptographie mathématique.

1. Data Security and Arithmetic

Cryptography is, in the true sense of the word, a classic discipline: we find it in Mesopotamia and Caesar used it. Typically, the historical examples involve secret services and military. Information is exchanged amongst a limited community in which each member is to be trusted. Like Caesar's chiffre these systems were entirely symmetric. Thus, the communicating parties needed to have a common key which is used to de- and encrypt. The key exchange posed a problem (and gives a marvellous plot for spy-novels) but the number of people involved was rather bounded. This has changed dramatically because of electronic communication in public networks. Since

2000 *Mathematics Subject Classification.* — 11T71.
***Key words and phrases.* —** Elliptic curve cryptography, mathematics of public key cryptography, hyperelliptic curves.

The authors would like to thank the organizers of the conference for generous support, an interesting program and last but not least for a very inspiring and pleasant atmosphere.
The second author acknowledges financial support by STORK `http:\\www.stork.org`. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

each pair of participants needs a secret key, a network of n users needs $n(n-1)/2$ keys. Besides the storage problem, one cannot arrange a key exchange for each pair of participants for the huge number of users in today's networks. The solution to this problem came in 1976 with the ground breaking paper by Diffie and Hellman [**16**]. They propose *public key cryptosystems*. This way, parties can agree on a joint secret key over an insecure channel. This key is then used with modern symmetric ciphers like AES [**13**]. The concept of public key cryptography relies heavily on *one way functions*. We give an informal definition:

Definition 1.1. — Let $\mathcal{A}$ and $\mathcal{B}$ be two sets and f a map from $\mathcal{A}$ to $\mathcal{B}$. f is a *one way function* if one can "easily calculate" $f(a)$ but for "essentially all" elements $b \in \mathrm{Im}(f)$ it is "computationally infeasible" to find an $a \in \mathcal{A}$ such that $f(a) = b$.

In a *public key cryptosystem*, each member A of the network has *two* keys: a *private key* s_A produced by himself, never leaving the private secure environment and a *public key* p_A published in a directory. p_A is related to s_A by a (publicly known) one way function. In a protocol, A uses both keys (and the public key of the partner B if necessary). One has to ensure that the function to derive p_A from s_A is one way, and the protocols have to be designed in a manner that there is no usable leakage of information about s_A, s_B from the publicly accessible values.

Today, messages are stored and transmitted as numbers. This makes it possible to apply *Arithmetic* to construct candidates for one way functions, to bring them in such a shape that computation is fast, and to analyze possible attacks.

We shall concentrate on systems based on the *Discrete Logarithm (DL)*. For a general overview of applied cryptography including protocols see [**42**]. In this exposition we can only outline the methods and mathematical facts used for designing secure and efficient DL-Systems. Much more details both for the mathematical background, the basic algorithms and their efficient implementation and the realisation of DL-systems in hardware can be found in [**4**].

2. Abstract DL-Systems

To give mathematical sound definitions we first describe DL-systems in an abstract setting. We give the minimal requirements needed for key exchange and signatures. For the remainder of this section we assume that $\mathcal{A} \subset \mathbb{N}$ [(1)] and that $\mathcal{B} \subset \mathrm{End}_{\mathrm{set}}(\mathcal{A})$, the set of endomorphism of $\mathcal{A}$. Hence, for any $a \in \mathcal{A}$ and any $b \in \mathcal{B}$ we have $b(a) \in \mathcal{A}$.

(1) This is also important for practical application as one can represent a natural number as a string of bits on a computer.

2.1. Key Exchange. — Assume that the elements of $\mathcal{B}$ commute: for all $a \in \mathcal{A}$ and $b_1, b_2 \in \mathcal{B}$ we have

$$b_1(b_2(a)) = b_2(b_1(a)).$$

Then we can use $\mathcal{A}, \mathcal{B}$ for a key exchange system in the following way:

We fix a (publicly known) base point $P_0 \in \mathcal{A}$. Each participant S_i chooses an $s_i \in \mathcal{B}$ and publishes $p_i := s_i(P_0)$. Then $s_i(p_j) = s_j(p_i)$ is the shared secret of S_i and S_j.

The security depends (not only) on the complexity to find for any randomly chosen $a \in \mathcal{A}$ and $a_1, a_2 \in \mathcal{B} \circ \{a\}$ *all* elements $b \in \mathcal{B}$ with $b(a) = a_1$ modulo $\mathrm{Fix}_{\mathcal{B}}(a_2) = \{b \in \mathcal{B} : b(a_2) = a_2\}$.

The efficiency depends on the "size" of elements in $\mathcal{A}, \mathcal{B}$ and on the complexity of evaluating $b \in \mathcal{B}$.

2.2. Signature Scheme of El Gamal-Type. — In addition we assume that there are three more structures:

(1) $h : \mathbb{N} \to \mathcal{B}$, a cryptographic hash function[(2)]

(2) $\mu : \mathcal{A} \times \mathcal{A} \to \mathcal{C}$ a map into a set $\mathcal{C}$ in which equality of elements can be checked fast

(3) $\nu : \mathcal{B} \times \mathcal{B} \to \mathcal{D} \subset \mathrm{Hom}_{\mathrm{set}}(\mathcal{A}, \mathcal{C})$

with $\nu(b_1, b_2)(a) = \mu(b_1(a), b_2(a))$ for all $a \in \mathcal{A}, b_i \in \mathcal{B}$.

Signature. — Let a base point $P_0 \in \mathcal{A}$ be given (or introduced as part as the public key). Like before, each participant S_i has his private key $s_i(P_0)$ and publishes his public key p_i.

To sign a message m, the signer S_i chooses a random element $k \in \mathcal{B}$ and computes $\phi := \nu(h(m) \circ s_i, h(k(P_0)) \circ k) \in \mathcal{D}$ using the knowledge of his private key s_i. Then he sends $(\phi, m, k(P_0))$ as the signature of the message m.

Verification. — The verifier V looks up $s_i(P_0)$, computes

$$\mu\Big(h(m)(s_i(P_0)), h(k(P_0))(k(P_0))\Big),$$

and compares it to $\phi(P_0)$.

The signature is valid if the results are equal.

2.3. The Most Popular Realization. — In practice we often encounter the following situation: Let p be a prime and consider an injective map $(\mathbb{Z}/p, +) \xrightarrow{f} \mathbb{N}$. Let $\mathcal{A} = \mathrm{Im}(\mathbb{Z}/p)$ be the image of f. $\mathcal{A}$ becomes a group with the composition $\oplus$ by the rule:

$$a_1 \oplus a_2 := f(f^{-1}(a_1) + f^{-1}(a_2)).$$

(2) We require h to be one way and collision resistant.

Note that in general $\oplus$ does not coincide with the usual addition in $\mathbb{N}$. For an element $P \in \mathcal{A}$ we define

$$kP = \underbrace{P \oplus P \oplus \cdots \oplus P}_{k \text{ times}}.$$

We require $\oplus$ to be computable in $\mathcal{A}$, *i.e.* without going back to $\mathbb{Z}/p$. Then $\mathcal{A}$ with the operation $\oplus$ is called a *group with numeration*.

We show how this matches with our previous definitions.

Choose $f(0 + p\mathbb{Z}) \neq P_0 \in \mathcal{A}$. The set $\mathcal{B} = \mathrm{Aut}_{\mathbb{Z}}(\mathcal{A}) \cong (\mathbb{Z}/p)^*$ is identified with $\{1, \ldots, p-1\}$ via $b(P) := bP$. We let $\mathcal{C} = \mathcal{A}$, $\mu =$ operation $\oplus$ in $\mathcal{A}$, $\nu =$ addition of endomorphisms, and $h =$ a hash function from $\mathbb{N}$ to $\{1, \ldots, p-1\}$.

Signature scheme. — We translate the abstract scheme to this situation: S chooses randomly and secretly, his *private key* $s \in \{1, \ldots, p-1\}$ and publishes his *public key* $P_S := sP_0$. This key pair is used for many messages.

To sign a message m, S chooses a random number k, which is only used for this one message, and computes

$$r := h(m)s + h(kP_0)k \mod p.$$

The signed message consists of (m, kP_0, r).

To check the authenticity of the message one looks up S's public key and computes

$$R = rP_0, T = h(m)P_S, \quad H = h(kP_0)kP_0.$$

and checks whether

$$R = T \oplus H.$$

The security considerations *for the crypto primitive* boil down to estimating the complexity of computing *Discrete Logarithms:*

The *Discrete Logarithm Problem (DLP)* is as follows: For a given cyclic group with numeration $\mathcal{A}$ and for randomly chosen $P, Q \in \mathcal{A}$ compute $k \in \mathbb{N}$ with $Q = kP$.

We need to construct groups with numerations of large prime order p, which are secure and efficient. Note, that these aims can be contradictory. One requires that the time *or* space needed (probabilistically) to compute discrete logarithms is *exponential* in $\log(p)$. But time *and* space needed to write down the elements and to execute a group composition must be polynomial in $\log(p)$.

2.4. Generic Attacks. — We have motivated that for some protocols it is useful to use the algebraic structure "group". However, every additional structure opens the door to attacks. Assuming no special properties of $\mathcal{A}$, *i.e.* dealing with a so-called *black-box group* allows "generic" attacks. Shoup [**55**] proved that such a black-box group has security at least $\sqrt{|\mathcal{A}|}$. We present two algorithms having this complexity.

To solve the DLP on input $Q = kP$, both aim at retrieving an equality between multiples of P and Q. From $m_1Q = m_2P$ one obtains $k \equiv m_2/m_1 \bmod p$. Since these algorithms are inevitable we say that a group is suitable for cryptographic

applications, if only these algorithms (or ones with similar running-time) apply. As one is able to find such suitable instances, one should avoid using groups with more powerful attacks unless they offer special advantages like easier implementation or faster algorithms, but a careful security analysis is needed.

Shanks' Baby-Step-Giant-Step Method. — This method is a deterministic algorithm to solve the DLP, first proposed by Shanks [**54**].

– *Baby step:* For $i = 0, \dots, m \leqslant \sqrt{p}$ compute $(i \cdot P, i)$. These values are stored in a list ordered by the first argument.

– *Giant step:* For $j = 0, \dots, m \leqslant \sqrt{p}$ compute $(Q - jm \cdot P, j)$.

Then one compares the two lists looking for matching pairs. (In practice only one list is stored and each result of the giant step is compared to this.) If $i \cdot P = Q - jm \cdot P$ then $k = i + jm$ and we have solved the DLP. This algorithm has *complexity* $O(\sqrt{p})$ but there is a *disadvantage* – it needs $O(\sqrt{p})$ space.

Pollard's ρ-Algorithm. — Pollard's algorithm [**48**] is a probabilistic algorithm in the sense that the output is always correct but the computations involve random choices and thus the complexity analysis involves probability assumptions.

The principle behind this algorithm is that for randomly drawn elements of G the expected number of draws before an element is drawn twice is $\sqrt{\pi p/2}$ due to the birthday paradox. To get information out of this we use a controlled random walk, which we now present in the simplest version: The result x_i of the i-th step should depend only on x_{i-1}. So partition the group "randomly" into three sets T_j of size $\approx p/3$ and take

$$\begin{aligned} x_i &= P + x_{i-1} \quad \text{if } x_{i-1} \in T_1, \\ x_i &= Q + x_{i-1} \quad \text{if } x_{i-1} \in T_2, \\ x_i &= 2x_{i-1} \quad \text{if } x_{i-1} \in T_3. \end{aligned}$$

There are efficient methods to detect collisions. Like Shanks' method this algorithm has complexity $O(\sqrt{p})$ but requires far less memory.[(3)]

Security hierarchy. — To have a more precise statement on the complexity of algorithms we measure it by the function

$$L_p(\alpha, c) := \exp(c(\log p)^{\alpha}(\log\log p)^{1-\alpha})$$

with $0 \leqslant \alpha \leqslant 1$ and $c > 0$.

The *best case* for a cryptosystem is $\alpha = 1$ – then one has *exponential complexity*, this means that the complexity of solving the DLP is exponential in the binary length of the group size $\log p$. The *worst case* is when $\alpha = 0$ – then the system only has *polynomial complexity*. For $0 < \alpha < 1$ the complexity is called *subexponential*.

[(3)] Using such generic low storage methods the current "world record" w.r.t. Certicom challenge was solved: Compute DL in an 109-bit elliptic curve over a prime field.