Abstract — This paper describes circuits for executing the most complex operations of public-key cryptography and gives estimations of their execution time within field programmable devices. The following operations are considered: mod $n$ exponentiation, mod $p$ division, mod $f(x)$ multiplication of polynomials, mod $f(x)$ division of polynomials and point multiplication over an elliptic curve.

Keywords — public-key cryptography, finite fields, arithmetic, programmable devices, FPGA

I. INTRODUCTION
The designer of systems including cryptographic algorithms — ciphering / deciphering, digital signature, authentication - is often faced with the following apparent contradiction: on the one hand, in many cases cryptographic algorithms are used within real time systems, so that their response time must be short; on the other hand, the security is related to the algorithm complexity. In order to make compatible those apparently contradictory characteristics, a possible solution is the use of specific hardware, that is, circuits specifically designed for executing those complex algorithms: they implement the particular computation primitives of the algorithms and take profit of their inherent parallelism. Among the technologies at hand for developing specific circuits are the field programmable devices, for example the Field Programmable Gate Arrays (FPGA). They constitute an attractive option for small production quantities as their non-recurrent engineering costs are much lower than those corresponding to Application Specific Integrated Circuits (ASIC). Furthermore, in order to reduce their size, and so the unit cost, an interesting possibility is to reconfigure them at run time so that the same programmable device can execute different predefined functions.

This paper describes circuits for executing the most complex operations of public-key cryptography and gives estimations of their execution time within field programmable devices. It is organized in the following way: section II briefly describes the main public-key cryptographic algorithms and deduces a list of complex computation primitives that should be implemented in hardware. Section III to VII propose generic algorithms\(^1\) and circuits for executing the mod $n$ exponentiation, the mod $p$ division, the mod $f(x)$ multiplication of polynomials, the mod $f(x)$ division of polynomials and the point multiplication over an elliptic curve, respectively. The adjective “generic” alludes to the fact that particular characteristics of the underlying algebraic structure, for instance special values of $n$, $p$ or $f(x)$, are not taken into account (except the case $p = 2$). Actually, a lot of improvements can be obtained if particular values of $p$ and $f(x)$ are chosen, but their description falls beyond the scope of this paper.

II. MAIN ARITHMETIC OPERATIONS
The most time-consuming operations correspond to public-key cryptography, that is, encryption / decryption schemes using different keys for ciphering (public key) and deciphering (private key). Among the most used are the RSA and the Discrete Logarithm systems.

In the first case (RSA, Adleman et al., 1978), two primes $p$ and $q$ are chosen. The public key is a pair $(n,e)$ of naturals where $n = p.q$, $e$ belongs to the interval $0 < e < (p-1)(q-1)$ and $e$ is relatively prime with $(p-1)(q-1)$. The private key is $d = e^{-1} \mod (p-1)(q-1)$. It can be shown that $x^d = x \mod n$, for any natural $x$. The encryption / decryption algorithm is the following: giving a message $mes$ represented under the form of a natural belonging to the interval $0 < mes < n$, compute the ciphered text $c = mes^e \mod n$. In order to decrypt $c$, compute $c^d \mod n$. Observe that knowing the public key $(n,e)$, the computation of the private key amounts to decompose $n$ under the form $n = p.q$ and then calculate $d = e^{-1} \mod (p-1)(q-1)$. Nowadays, the factorization problem is intractable for key sizes greater 1024 bits.

In the second case (Discrete Logarithm), a finite group $(G,*,1)$ is defined and some element $g$ of $G$ is chosen. Let $n$ be the order of $g$. Thus, the set $\{1, g, g^2, ..., g^{n-1}\}$ is a cyclic subgroup of $G$. The private key is a natural $x$ belonging to the interval $0 < x < n$, and the public key is the element $y$ of the cyclic subgroup defined by $y = g^x$. The message $mes$ must be represented under the form of an element of $G$. The encryption algorithm is the following: randomly choose a natural $k$ belonging to $0 < k < n$, compute $c_1 = g^k$ and $c_2 = mes^y$. The ciphered text is made up of $c_1$ and $c_2$. In order to decrypt the message, compute $c_2^{(c_1)^{-1}}$. Observe that knowing the public key $y$, the computation of the private key $x$ amounts to calculate $log_G y$, presumably a very hard problem.

In the basic version of the Discrete Logarithm scheme (ElGamal, 1985), $G$ is the set of natural $\{1,2, ..., p-1\}$, where $p$ is a prime, so that all operations are per-