A Residue Approach of the Finite Fields Arithmetics

Jean Claude Bajard
LIRMM, CNRS UMR 5506, Univ. Montpellier 2

***Abstract*— Finite fields arithmetic is one of the challenges in current computer arithmetic. It occurs, in particular, in cryptography where the needs increase with the evolution of the technologies and also of the attacks. Through our research, we have proposed different systems based on residues representations. Different kinds of finite fields are concerned with. For each of them, some specificities of the representations are exploited to ensure the efficiency, as well as for the performances, than for the robustness to side channel attacks. In this paper, we deal with three similar approaches: the first one is dedicated to prime field using residue number systems, a seconde one concerns extension finite fields of characteristic two, the last one discusses of medium characteristic finite fields. The main interest of these systems is their inherent modularity, well suited for circuit implementations.**

I. Introduction to residue systems

The idea of residue representations comes from the Chinese Remainder Theorem. A first approach seems due to a chinese mathematician Sun Tsŭ (around the fourth century). A generalization was proposed by another chinese mathematician Shu Shu Chiu Chang in 1247 [11]. Then, this notion was introduced in computer science during the fifties [13], [8], [14].

The main feature of these residue representations is to define a number by its residues modulo a set of relatively prime numbers. It is very similar to the representation of a polynomial of degree n by its values on $n+1$ points. The translation from a classical position representation (for example in radix 2 or in radix 10) to a residue system is done by the evaluation of the residues (equiv. to the evaluation of the polynomial on each point). The inverse translation is obtained by summations which looks like the Lagrange interpolation formulas for obtaining the coefficients of a polynomial. This reconstruction from the residue can also be done, using an intermediate representation called mixed radix, which is, in this case, similar to the Newton interpolation.

As number and polynomials are very close, a common approach can be available for the different kinds of finite fields (i.e. prime finite field, characteristic two extensions, and medium characteristic extensions). In this paper we present the adaptation of the residues representations, focussing our study on the finite field multiplication whose complexity is critical in most applications, like in cryptography. To end we give some examples where the properties of residue representations are interesting.

II. Finite field arithmetics

In this section, we introduce the notion of Residue Systems for representing the elements of a finite field. We consider two cases: prime fields and finite extension fields.

A. Prime finite field

In a prime finite field $GF(p)$, elements can be considered as integer with an arithmetic modulo p. We can deal with the classical Residue Number Systems (RNS), which is an interesting way to represent the integers.

A RNS is defined by a set of coprime numbers $(m_1, m_2, \ldots, m_n)$, called RNS base. Then, a number X can be represented by its residues $(x_1, x_2, \ldots, x_n)$ with $x_i = |X|_{m_i} = X \bmod m_i$.

We generally assume that $0 \leq X < M = \prod_{i=1}^{n} m_i$. In RNS, addition and multiplication are made modulo M independently on each residue, which means that a full parallelism is possible. The elements of the RNS base $(m_1, ..., m_n)$ are chosen such that, $m_i = 2^s - c_i$ where c_i is very small, and s is the number of bits of the basic cells. This property ensures that the reduction part on each m_i can be neglected [6]. We consider two numbers A, and B which are represented in a such system by:

$$A_{rs} = (a_1, \ldots, a_n) \quad \text{and} \quad B_{rs} = (b_1, \ldots, b_n)$$

We note $\otimes$ the operation which could be an addition or a multiplication, then $R = A \otimes B \bmod M$ can be directly evaluated in the Residue System:

$$R_{rs} = (|a_1 \otimes b_1|_{m_1}, \ldots, |a_n \otimes b_n|_{m_n}) \quad (1)$$

We can remark that, if $A \otimes B < M$ then R_{rs} represents the exact result.

Now, if we consider the arithmetic in a finite field $GF(p)$ with p a prime number, then the Residue System must satisfy $p < M$ for representing the elements of $GF(p)$ and we must be able to perform a multiplication: $|A \otimes B|_p$, which requires a modular reduction introducing some other constraints. This operation is presented in section III.

B. Finite extension field

An extension of a finite field is a field of the form $GF(p^k)$ with p a prime and k an integer called the degree. A such a field can be defined by $I(X)$ an irreducible $GF(p)$ polynomial of degree k. In a certain way, $GF(p^k)$ is seen as $GF(p)(X)/I(X)$. Hence, elements of $GF(p^k)$ can be considered as $GF(p)$ polynomials of degree lower than k.

1) In medium characteristic: When the characteristic p is larger than k, we can consider a set of k different points $(e_1, ..., e_k)$ in $GF(p)$ for defining an element of $GF(p^k)$, we call this set the Lagrange Base. Then, an element of $GF(p^k)$ which can be defined as a polynomial of degree lower than k, due to the Lagrange interpolation formulas, can be represented by its values at these k points. An element $A(X) = \alpha_0 + \alpha_1 X + ... + \alpha_{k-1} X^{k-1}$ of $GF(p)$ can be represented by $(a_1, ..., a_k)$ with $a_i = |A(e_i)|_p$. We remark that:

$$a_i = |A(e_i)|_p = |A(X) \bmod (X - e_i)|_p .$$

Thus, if we note, $m_i(X) = (X - e_i)$, we obtain a residue representation similar to RNS ([5]) with the same operations properties.

For representing correctly the product of two polynomials of degree k, we will see in section III that we need to increase the Lagrange base to $2k$ elements, for that reason the condition needed for an arithmetic over $GF(p^k)$ is $p > 2k$ to assume the existence of these points.

2) In characteristic two: If the characteristic is smaller than the degree: $GF(p^n)$ with $p < n$, which is always the case in characteristic 2, then we consider a set $(m_1(X), ..., m_k(X))$ of coprime $GF(p)$ polynomials which defines a residue base. Thus, an element $A(X)$ of $GF(p^n)$ is defined by $(a_1, ..., a_k)$ the set of its residues over the base: $a_i(X) = |A(X)|_{m_i(X)}$.

If the $m_i(X)$ have the same degree d: $k \times d > n$, then the $a_(X)$ are $GF(p)$ polynomials of degree lower than d. We can remark that, sparser are the $m_i(X)$, faster are the calculus. For this reason, we had proposed in [4], to choose trinomials for the residue base.

III. Modular multiplication in a residue system

We give in this section a way to perform the arithmetic in a residue system over a finite field. The addition is trivial in the case of the finite extension fields, we apply directly the residue addition. For prime finite fields a reduction $\bmod p$ can be needed, but it can be associated to one needed for a product. Now, for the multiplication, a reduction can be needed in most the cases. We present how reduce the result of a product in a residue system.

A. Introduction to Montgomery reduction

The most used reduction algorithm is due to Peter Montgomery [12]. It is currently considered as the most efficient method. This approach evaluates first a value $q = -(Ap^{-1}) \bmod (\beta^s)$ (β is the radix

used or X in the polynomial case) such that $(A + qp)$is a multiple of (β^s). Then, it constructs $R = (A+qp)/\beta^s$, such that: $R \equiv A \times \beta^{-s} \pmod{p}$ and $R < 2p$ if $A < p\beta^s$. We note, Montg$(A, \beta^s, p) = R$.

This approach does not reduce exactly a value A (or $A(X)$) modulo p (or $I(X)$), it computes a reduced values which is equivalent to $A \times \beta^{-s}$ modulo p (or $I(X)$).

For avoiding an increase of the factor β^{-s}, we use the Montgomery notation: $A' = |A \times \beta^s|_p$, thus Montg$(A' \times B', \beta^s, p) \equiv A \times B \times \beta^s \pmod{p}$.

B. Montgomery reduction in a residue system

The Montgomery algorithm can be easily adapted to residue systems [1]. In this case, it is necessary to extend the base for two reasons: for representing the exact value of the product, and to be able to multiply by the inverse of the product of the elements of the primary base.

The initial residue base is such that $p < M$ (or degree of $M(X) \geq k$), and the auxiliary base verifies $p < M'$ (or degree of $M'(X) \geq k$), M' and M are coprimes.

The Montgomery multiplication algorithm adapted to residue systems is decomposed in the following steps, where B and C are two elements of the finite field:

1) $A \leftarrow B \times C$ (residue calculus in base M and M'),
2) $q \leftarrow -(Ap^{-1}) \bmod M$ (residue calculus in base M),
3) Extension of the representation of q from the base M to the base M',
4) $R \leftarrow (A + qp) \times M^{-1}$ (residue calculus in base M'),
5) Extension of the representation of R from the base M' to the base M.

The value A is represented in the two bases. This suggests that the elements of the finite field B and C must be defined in the two bases. Indeed, the residue product of two elements must be performed in the two bases before the modular reduction.

The extension of the representation comes from the Lagrange interpolation.

If $(a_1, ..., a_k)$ is the residue representation in the base M, then

$$A = \sum_{i=1}^{k} \left| a_i \times \left[\frac{M}{m_i} \right]_{m_i}^{-1} \right|_{m_i} \times \frac{M}{m_i} - \alpha M$$

The factor α can be in certain cases neglected, or computed [1]. If we note m_i the elements of the first base and m'_j these of the second, the extension is obtained with:

$$A = \left| \sum_{i=1}^{k} \left| a_i \times \left[\frac{M}{m_i} \right]_{m_i}^{-1} \right|_{m_i} \times \left| \frac{M}{m_i} \right|_{m'_j} - \alpha \left| M \right|_{m'_j} \right|_{m'_j}$$

Another approach consists in the Newton interpolation where A is correctly reconstructed [4].

The residue multiplication is very efficient, for a parallel implementation, its cost is the one of a multiplication on a residue. The reduction part is costly. It is quadratic for RNS [1], and Lagrange representation [5]. We obtain a sub-quadratic complexity, $O(k^{1.6})$ for trinomials [4]. But, if we take into account that, for base extensions, most of the operations are multiplications by a constant, this cost can be considerably smaller.

IV. Applications to cryptography

Many cryptographic protocols are based on mathematical results on algebraic curves. The most popular are based on the arithmetic over elliptic curves and the pairings.

A. Elliptic curve formulas

An elliptic curve $E(X)$ over a finite field $GF(p)$ (or $GF(p^k)$, $p \neq 2, 3$), can be defined by an equation of the following form $Y^2Z = X^3 + aXZ^2 + bZ^3$ (for $p = 2, 3$ there is equivalent equations). We note that it exists different other curves or formulations used in cryptography: Hessian, Jacobi etc...[2].

A point $P \in E(X)$ of order n defines an Abelean subgroup. The addition of two points is given by formulas obtained from the intersection of the curve with the straight line defined by the added points (or the tangent to the curve for a doubling). To

avoid any inversion, which is very costly, most of the time we use homogeneous (or jacobean, or chudnovski...) coordinates. Considering these projective coordinates, the addition formulas contain only additions and multiplications over the finite field. We have seen that, using a residue systems, one operation in a finite field is done by a residue operation followed by a reduction.

The operations in residue systems are very efficient except the reduction which is costly. It is the opposite of the classical representation where operations, like the multiplications, can be costly but the reduction, for specific choices of p or $I(X)$, is particularly easy. This knowledge makes natural the idea of modifying the formulas to minimize the number of reductions, even if the number of residue operations increase. Clearly, for a formula like $A \times B + C \times D$, only one reduction is needed. This work which was done in [2], gives promising results. It shows that for 512 bits values and more over a prime field, Residues Systems are more efficient than classical representations. This first approach could probably be improved, and apply to finite extension fields. Furthermore, a similar work can be done for hyper-elliptic curves arithmetic.

B. Arithmetic of the pairings

Summarizing, we define a pairing as following: G_1 and G_2 two additive abelean groups of cardinal n and G_3 a cyclic multiplicative group of cardinal n. A pairing is a function $e : G_1 \times G_2 \rightarrow G_3$ which verifies the following properties: bilinearity, non-degeneracy. If we consider the pairings defined on elliptic curves over a finite field $GF(p)$, we have more precisely $G_1 \subset E(GF(p))$, $G_2 \subset E(GF(p^k))$ and $G_3 \subset GF(p^k)$ with E an elliptic curve and where k is the smallest integer such that n divides $p^k - 1$, k is called the embedded degree of the curve. In [7] the authors propose a taxonomy of the pairings over elliptic curves with small embedding degree and large prime-order subgroup which are the most implemented pairings for cryptographic protocols [9] [10].

The construction of the pairing implies values of $GF(p)$ and $GF(p^k)$ into the formulas. An approach with Residue Systems, similar to the one made on ECC is interesting [3]. Most of the time for algorithmic reasons, k is chosen as a small power of 2 and 3. But with residue arithmetics, we can pass over this restriction. With pairings, we can also imagine two levels of residue systems: one over $GF(p)$ and one over $GF(p^k)$.

C. Conclusions

We have shown in this paper, that residue systems are available for each kind of finite fields. The arithmetic over these representations offers efficient residue operations. The modular reduction which is costly, is not necessary after each operation. Hence, for applications using finite fields, we can reformulate the different algebraic expressions used for minimizing the number of reductions. Thus, residue systems becomes efficient. Some results are already available for elliptic curves defined over prime finite fields. Further works must be done on elliptic curves over finite extension fields and hyper-elliptic curves.

REFERENCES

[1] Bajard, J.C., Didier, L.S., Kornerup, P.: Modular multiplication and base extension in residue number systems. 15th IEEE Symposium on Computer Arithmetic, 2001 Vail Colorado USA pp. 59–65

[2] Bajard, J.C., Duquesne, S., Ercegovac M. and Meloni N.: Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography, in Advanced Signal Processing Algorithms, Architectures, and Implementations XVI, SPIE 2006, San Diego, USA.

[3] Bajard, J.C. and ElMrabet N.: Pairing in cryptography: an arithmetic point of view, Advanced Signal Processing Algorithms, Architectures, and Implementations XVII, part of the SPIE Optics & Photonics 2007 Symposium. August 2007 San Diego, USA.

[4] J.C. Bajard, L. Imbert, and G. A. Jullien: Parallel Montgomery Multiplication in $GF(2^k)$ using Trinomial Residue Arithmetic, 17th IEEE symposium on Computer Arithmetic, 2005, Cape Cod, MA, USA.pp. 164-171

[5] J.C. Bajard, L. Imbert et Ch. Negre, Arithmetic Operations in Finite Fields of Medium Prime Characteristic Using the Lagrange Representation, journal IEEE Transactions on Computers, September 2006 (Vol. 55, No. 9) p p. 1167-1177

[6] Bajard, J.C., Meloni, N., Plantard, T.: Efficient RNS bases for Cryptography, IMACS'05, Applied Mathematics and Simulation, (2005).

[7] Freeman D., Sott M. and Teske E.: A Taxonomy of Pairing-Friendly Elliptic Curves, http://eprint.iacr.org/2006/372.pdf

[8] Garner, H.L.: The residue number system, IRE Transactions on Electronic Computers, EL **8:6** (1959) 140–147.

[9] Granger R., Page D., and Smart N.: High security pairing-based cryptography revisited. In Algorithmic Number Theory Symposium ANTS-VII, volume 4076 of Lecture Notes in Computer Science, pages 480?494. Springer, 2006.

[10] Koblitz N. and Menezes A.: Pairing-based cryptography at high security levels. In Proceedings of Cryptography and Coding: 10th IMA International Conference, volume 3796 of Lecture Notes in Computer Science, pages 13?36. Springer, 2005.

[11] Knuth, D.: Seminumerical Algorithms. The Art of Computer Programming, vol. 2, Addison-Wesley (1981).

[12] Montgomery, P.L.: Modular multiplication without trial division, Math. Comp. **44:170** (1985) 519–521.

[13] Svoboda, A. and Valach, M.: Operational Circuits, Stroje na Zpracovani Informaci, Sbornik III, Nakl. CSAV, Prague, 1955, pp.247-295.

[14] Szabo, N.S., Tanaka, R.I.: Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill (1967).