Speeding up Discrete Logarithm and Elliptic Curve Based Cryptography over $GF(2^m)$ on General Purpose Processors using Lookup Table Based Finite Field Arithmetic Techniques

Athar Mahboob

DHA Suffa University, Karachi, Pakistan
Email: atharmahboob@yahoo.com

Keywords: binary finite field arithmetic, abstract algebra computations, elliptic curve cryptography

A number of public key cryptographic technques depending on the intractability of the discrete logarithm problem in certain finite algebraic groups, such as the Diffie-Hellman Key Agreeement Protocol, El-Gamal Encryption and Signature schemes and their Elliptic Curves based analogs, utilize the binary finite field $GF(2^m)$ as the under-lying mathemtical structure in which to perform the group operations. In general, to ensure adequate security level the paramater sizes used in these schemes are large, ranging from a few hundred to a few thousand bits, when represented on computers. General purpose processors coupled with high-level programming languages are the pre-dominant computing platform used today from high-performance servers to personal digital assistants. As such these platforms are targets for software implementation of these cryptographic algorithms. Abstraction, modularisation, portability and modifiability offered by software development in standard programming languages are some of the most important benefits of a cryptographic software implementation. However, the restriction of processing in word-sized chunks at the lowest level of machine hardware combined with the unavailability of a machine level $GF(2^m)$ multiply instruction on general purpose processors are the most important limitations. Efficient finite field arithmetic is essential for fast implementation of these cryptographic algorithms in software environments. Software implementations thus present unique opportunities and challenges to the implementor. Whereas $GF(p)$ software implementations on general purpose processors are able to benefit from the machine level integer multiply instruction that takes one or, at most, just a few clock cycles to compute a word-sized multiplication, which may be then used as a building block for the $GF(p)$ multiplication for any parameter size. For $GF(2^m)$ multiplication, unfortunately, one has to fall back on the awkward bit by bit processing, resulting in much slower performance of the critical $GF(2^m)$ arithmetic operations in software implementations. Same is true for the squaring and inversion arithmetic operations in $GF(2^m)$.

In this paper we present methods for performing binary finite field arithmetic in Polynomial Basis using lookup tables (LUT) which are highly applicable to software environments. The Lookup Table based $GF(2^m)$ arithmetic techniques we have developed are applicable to multiplication, squaring and inversion, the three fundamental arithmetic operations. Finite field multiplication, squaring and inversion are the most important arithmetic operations in the binary finite field $GF(2^m)$. They form the fundamental operations is required for many cryptographic techniques based on the discrete logarithm problem (DLP) in the multiplicative group of a finite field or additive group of points on an elliptic curve defined over a finite field. The lookup table based arithmetic techniques we present utilize the polynomial basis finite field representation which is conceptually simpler when compared with other representations. Furthermore, unlike some older LUT techniques, the LUTs in our techniques are calculated only once, and for all. The LUTs are of a constant size, independent of the field extension m. Our LUT based techniques are valid for any m, whereas many older LUT techniques worked only for composite m. We show by performing complexity analysis that the our techniques result in the lowest number of word-level operations in software environments on contemporary computing platforms. For performing fast inversion we use the Fermat's Little Theorem (FLT) and our fast LUT based $GF(2^m)$ multipliers and squarers. According to Fermats Little Theorem in a finite field with q elements $a^q = a$. This can be rewritten by dividing both sides by a^2 as $a^{q-2} = a^{-1}$. For the binary finite field $GF(2^m)$, $q = 2^m$, therefore we can write $a^{-1} = a^{2^m-2}$. Due to the addition chain first proposed by the Itoh-Tsujii this particular exponentiation can be carried out in at most $2 \cdot log_2(m-1)$ multiplications and $m-1$ squarings. As we show this yields a very competitve $GF(2^m)$ inversion scheme when compared with Extended Euclidean Algorithm based inversion schemes.

Our $GF(2^m)$ arithmetic methods result in significant performance gains for discrete logarithm based cryptosystems in software environments, particularly elliptic curve cryptosystems. Our method results in performance gains over methods reported earlier in the literature on platforms where fast memory lookups can be done. We present also demonstrate significant performance gains for ECC using our proposed arithmetic methods.