A Diffie-Hellman implementation using the method of Galbraith, Lin and Scott

Michael Scott

School of Computing
Dublin City University
Ballymun, Dublin 9, Ireland.
mike@computing.dcu.ie

Abstract. Herein we very briefly document an experimental implementation of the Diffie-Hellman algorithm using an elliptic curve over the quadratic extension, and also exploiting an endomorphism to implement the Gallant-Lambert-Vanstone idea - see Galbraith, Lin and Scott (eprint report http://eprint.iacr.org/2008/194) for more details. The implementation is for x86-64 processors only and uses the MIRACL library. This version used Twisted Inverted Edwards Coordinates.

1 Implementation

This implementation is quite fast, but very experimental. It currently has some shortcomings.

- Its vulnerable to side-channel attack
- Its not particularly small
- It is not at all portable – x86-64 architecture only

A future implementation may address some of these issues.

One potential advantage of the GLS method, is that the base field can be "half-sized", in this case just 127-bits. The elliptic curve is defined as

$$x^2 + y^2 = x^2y^2 + 42 \bmod p$$

We use $p = 2^{127} - 1$, a Mersenne prime. However we do not work directly on this curve, but rather in a 252-bit prime-order group on a quadratic twist over the field $\mathbb{F}_{p^2}$, which lifts us to an (approximate) AES-128 equivalent level of security. The number of points on the quadratic twist is four times a prime, the best that can be achieved using Edwards coordinates. See the full paper for details. The base field arithmetic is implemented in x86-64 assembly language. As each field element can now be stored in just two 64-bit registers, this part of the implementation is quite small and fast. This implementation benefits from (and ruthlessly exploits) the particularly simple form of the modulus. Addition and doubling of elliptic curve points uses the formulae from the paper "Twisted Edwards Curves", by Bernstein, Birkner, Joye, Lange and Peters (http://eprint.iacr.org/2008/013).

In this version point compression is used, which requires the calculation of a square root in the quadratic extension. This is simplified somewhat by the fact that square roots in the base field can be calculated as

$$\sqrt{x} = x^{2^{125}}$$

Exploiting an endomorphism the well-known method of Gallant, Lambert and Vanstone applies. Therefore the point multiplication required by ECDH is actually carried out using a double-point multiplication of $a_0.P + a_1.\psi(P)$, where ψ is the endomorphism. This double multiplication uses a standard interleaving algorithm with precomputation, with a fractional windows technique, in order to limit the number of point additions in the left-to-right double-and-add algorithm.

This version speeds up the `keypair` calculation considerably using a Comb algorithm and the storage of 64 precomputed points on the elliptic curve.