`Surf127eps`: A key-exchange cryptosystem based on a Kummer surface

P. Gaudry, T. Houtmann and E. Thomé

Version 2. June 2007

The `Surf127eps` cryptosystem is a key-exchange cryptosystem based on the Kummer surface of a genus 2 hyperelliptic curve over the finite field with $p = 2^{127} - 735$ elements. The implementation follows the formulae described in [1].

1 Parameters of the Kummer surface

The Kummer surface that is used corresponds to a genus 2 curve that has complex multiplication by

$$K = \mathbb{Q}\left(i\sqrt{5+\sqrt{53}}\right).$$

This field has class number 4 and is non-Galois. The CM-ideal has degree 8 and is irreducible over $\mathbb{Q}$. Consider the prime integer

$$p = 2^{127} - 735.$$

This prime splits completely in K and the corresponding factors are good Weil numbers. Furthermore the Igusa invariants of one of the corresponding curves have quotient of squares of Theta constants which are $\mathbb{F}_p$-rational. With the terminology of [1], one can take (projectively):

$$\begin{aligned} a &= 1 \\ b &= 104737609498996807573042644460938049128 \\ c &= 149790188288476750369734729103725046598 \\ d &= 129048219867477581895670028479253045173\,. \end{aligned}$$

This data verifies the *Genericity Conditions* of [1]; also the underlying curve is known, so that there is no need to check the *Rationality Conditions*.

The CM theory provides the group orders for the Jacobians of the curve and its twists:

$$\begin{aligned} N &= 16 \times 1809251394333065553537167681402254284155645350006293173560781593047272845093 \\ \tilde{N} &= 256 \times 2551 \times 62039 \times 167974189 \times 36109087046045143171 \\ &\quad \times 11779981942828041718410273831589995910 1\,. \end{aligned}$$

The following point P on the Kummer surface has order $N/8$ and can therefore be used as a base point in a Diffie-Hellman key exchange.

$$P = (1, 1, 7, 904051916807198515906372082818304356 85).$$

Security. We consider here the elementary security of this cryptosystem, without taking into account the potential weaknesses of the implementation (there are some in this version) or in the inclusion in a larger protocol (authentication is not done, for instance). Let us then consider the best known algorithm for solving one discrete logarithm in the N-order Jacobian corresponding to this Kummer surface, namely Pollard's Rho algorithm or distributed variants of it. This takes about the square root of the largest prime factor of the group order: $\sqrt{N/16} \approx 2^{125}$. This is well beyond any feasible computation.

2 Some implementation details

2.1 Finite field arithmetic

Elements of the finite field are represented as a table of `unsigned long` of fixed length (2 or 4 words, depending on 32- or 64- architecture). In the current implementation, after each operation the returned element is normalized to an integer between 0 and $p-1$. This is probably not optimal. The reduction modulo p takes advantage of the particular form of p. Still, the assembly for AMD64 has been optimized, and the timings are much better than in version 1.

2.2 Encoding of keys

A secret key is an integer between 0 and $2^{256}-1$. We impose it to be a multiple of 2 in order to avoid subgroups attacks. This secret key is stored in big-endian form in 32 bytes.

The public key and the shared secret are points of the Kummer surface. Since these are projective coordinates, we can (or we must in the case of the share secret) make them affine by, for instance, putting the first coordinate to 1. There are 3 elements left to store, that all fill in 128 bits (actually 127), therefore it takes 48 bytes to store a public key or a shared secret.

3 Future work

Here is our "to do" list for the next version, that should be much closer to a decent production software. This should not deteriorate the speed of the algorithm, except maybe for the key validation part.

- Different representation of finite field elements.

  As said above, the current representation of field element is maybe too strict to allow maximal performance. We might change our choice of the base field in order to have *nails* (GMP language).

- Key validation.

  It is necessary to check that the public key of the other party is valid. One hope would be to avoid this completely by using an appropriate surface that corresponds to secure curve and twist. However this is not enough, since there is also a need to check that the point is indeed on the Kummer surface.

- Study and avoid degenerate cases.

  The current version does not take into account the potential exceptional cases that would create a non-point with all-zero coordinates. There is some theory to do here before fixing the implementation.

- Study the point compression.

  There is some redundancy in the 48 bytes used for representing a Kummer point. We can put all the information in 32 bytes, but then it is required to compute the roots of a polynomial of degree 4. We might implement that in the next version.

- Implement Montgomery's PRAC algorithm.

  The Lucas chain we use is the classical double and add which is not optimal. If we assume that resistance to side-channel attack is not an issue, we can use the PRAC algorithm to save some work.

- Use a random or a RM curve, obtained by point counting.

  Point counting on this size of genus 2 curve has not yet been done. It is not clear that this will be feasible in a near future without algorithmic improvements.

One possibility is to use a curve with real multiplication. Indeed, the point counting can take advantage of that particularity, and cryptographical sizes are easily in reach with current technology and algorithms. This approach is completely compatible with forcing small Kummer parameters in order to have several multiplications by small constant integers. Hence some speed-up is expected compared to the current system that uses a CM curve.

References

[1] P. Gaudry. Fast genus 2 arithmetic based on theta functions. Cryptology ePrint Archive: Report 2005/314, 2005.