CENTRE *for* DOCTORAL TRAINING *in*
CYBER SECURITY

CDT Technical Paper

06/16

Evaluating software packages for attacks against elliptic curve cryptography

Nicholas Moore

Evaluating software packages for attacks against elliptic curve cryptography

Nicholas Moore

September 9, 2016

Abstract

Until the coming of quantum computers and post-quantum cryptography, elliptic curve cryptography seems to be a prime candidate for modern encryption systems. Therefore producing attacks that reduce the strength of security claims about them allows researchers to better understand the practical limits of elliptic curves for cryptography – and improving their performance may potentially allow attacks to be made against the current best practice to demonstrate weaknesses. In this paper, we will be looking at how some attacks on elliptic curve cryptography work and evaluate a number of programming languages to test their suitability for implementing an attack that uses a Gröbner basis calculation.

1 Background

1.1 Discrete Logarithm Problem

Given a finite field $\mathbb{F}_p$, the generator for the field g and an $X \in \mathbb{F}_p$, the discrete logarithm problem (DLP) is to find an integer x such that $X = g^x$. There are well-known and relatively efficient algorithms for calculating X given $\mathbb{F}_p$, g and x, however finding an algorithm to efficiently solve the DLP has proved to be much harder, especially when p is a large prime since it creates a substantial search space for potential solutions that cannot be easily split into smaller spaces.

The difficulty of the DLP and the efficiency of its inverse is used in asymmetric cryptographic protocols, where so-called "trapdoor" functions are used to create a public and private key pair, but finding the private key given only the public key appears to be infeasible. For DLP the most common use case is the Diffie-Hellman protocol (DHP), where the private key is some integer d and the public key is $D = g^d$.

The following is a simplified example of how the DHP could be used in a key agreement protocol. To send a message from Alice with keys $(a, A = g^a)$ to Bob with keys $(b, B = g^b)$, Alice would take Bob's public key B, raise

it by the power of her private key a to find the shared secret $K = B^a$ and encrypt her message with K. Bob can calculate the same K by raising Alice's public key A by the power of his private key b, which will result in the same secret $K = A^b$, and therefore can decrypt the message. This works due to the transitivity of exponentiation (i.e. $g^{ab} = g^{ba}$) and it is difficult for an attacker with access to A and B to defeat due to the apparent hardness of the DLP.

The most efficient algorithms for solving the DLP generally attempt to find and utilise some underlying structure in the finite field and reduce the problem down to a smaller field, where the search space for solutions is smaller and therefore computable. See [12] for a recent survey of some of these algorithms.

One such algorithm was proposed in [4], which targets a specialisation of the DHP, called the static Diffie-Hellman protocol (SDHP). This protocol provides access to an oracle that computes X^q for any $X \neq g$ in $\mathbb{F}_p$, where q is the private key of interest. The algorithm exploits this feature to construct a smaller sub-field within $\mathbb{F}_p$, which it enumerates using Shanks' 'baby-step giant-step' algorithm to find, with high probability, a correct candidate for the value of q.

1.2 Elliptic curves

An elliptic curve E is an algebraic curve, usually represented in the form

$$E : y^2 = x^3 + ax + b,$$

that is non-singular. It is defined over a finite field $\mathbb{F}_q$, where $a, b \in \mathbb{F}_q$.

The group $E(\mathbb{F}_q)$ is defined by finding all the points $P = \langle x, y \rangle$ such that x and y are elements of $\mathbb{F}_q$ and are rational solutions for the curve E.

Any elliptic curve $E(\mathbb{F}_q)$ will also have a "point at infinity", P_∞ (called O in some literature). This point acts as the additive identity - i.e. $P + P_\infty = P$. Since elliptic curves have a line of symmetry along the x-axis, for any given P, $-P$ has the same x co-ordinate but the y co-ordinate is negated, and $P - P = P_\infty$, which follows from the definition of additive identity.

For any given element $P \in E(\mathbb{F}_q)$, any element in $E(\mathbb{F}_q)$ can be created by multiplying P by some integer i. Given some $Q \in E(\mathbb{F}_q)$, finding x such that $Q = xP$ is called the elliptic curve discrete logarithm problem (ECDLP) [13].

When used by asymmetric cryptographic protocols such as DHP, the curve (which can be specified by a prime power q, two elements of $\mathbb{F}_q$ a and b, and a curve element P) will normally be pre-chosen, often making use of curves standardised by organisations such as NIST [1]. The key pair will be some integer d modulo q, which is the private key, and the curve element $D = dP$, which is the public key.

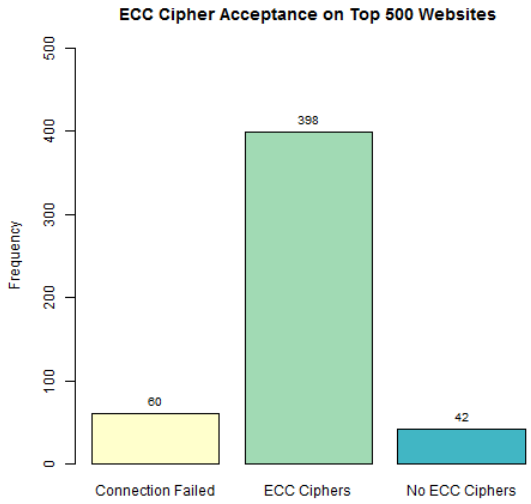

Figure 1: Acceptance rates of ECC ciphers

1.2.1 Usage of elliptic curves in cryptography

Elliptic curve cryptography (ECC) has seen wide-scale adoption on the internet. IETF/NIST proposals to use ECC across number of existing internet protocols, including TLS [16], SSH [11], and IPsec [14] have been taken up across the board. Although fewer hardware implementations of ECC operations are currently available, ECC does offer a number of benefits, such as requiring a smaller key for the same level of security as offered by larger keys under RSA.

For example, we found in a small sample of 500 commonly accessed websites [2], 398 (79.6%) of them accept HTTPS clients that use ECC encryption for key exchange and 18 (3.6%) of them will also use ECC digital signatures. Figure 1 shows the results of the queries. The column labelled Connection Failed indicates that the website either does not support HTTPS or we could not connect to it during the testing, whilst No ECC Ciphers indicates that a HTTPS connection was initiated, but no ECC cipher could be agreed on during the handshake (indicating the website does not support ECC).

Thus attempting to find faster methods of solving ECDLP has the potential to impact real world use cases. However, in order to reduce the potential for unexpected structure within the finite field $\mathbb{F}_q$, when used for cryptography q is usually chosen to be either an odd prime or 2^q with q being a large prime, which is called a binary curve. The attacks detailed below all target composite values of q, limiting their effectiveness against ECC.

1.3 Semaev's summation polynomials

In [17], Semaev proposed a series of summation polynomials that, when solved for certain values, can be used to perform index calculus on elliptic curves. These functions are named f_n, where n is the number of variables in the polynomial, and are defined recursively, with fixed f_2 and f_3 for Koblitz curves.

The basic idea behind the summation polynomials is that, for the values $x_1, ..., x_n$ that make $f_n(X_1, ..., X_n) = 0 \mod p$, the summation over the corresponding elliptic curve points $(x_1, y_1), ..., (x_n, y_n)$ equals P_∞.

The most efficient method currently known of finding the values $x_1, ..., x_n$ is to calculate the Gröbner basis of f_n, which is to polynomial systems as Gaussian elimination is to linear algebra. These values are expected to be smaller than $q^{1/n+\delta}$ for some small $\delta > 0$. Because of this fact, we can therefore use this summation polynomial over an elliptic curve in $\mathbb{F}_{q^n}$, but expect to find x co-ordinates in $\mathbb{F}_q$, which reduces the search space for results.

If we have some target curve point $R = (x_R, y_R)$, finding a basis for $f_{n+1}(x_1, ..., x_n, x_R)$ allows us to find n curve points that compose to equal R. This is because together the points must sum to P_∞ - the additive identity, therefore subtracting one or more points from the equation must result in an equation that sums to those points.

A drawback to the summation polynomials is the amount of memory it takes to represent them. For f_6 the naive representation in Magma will not fit in 256GB of memory. As such, their utility when used directly against curves over large fields is limited. Instead, research has focused on finding elliptic curves that can be easily reduced to a smaller curve where the polynomials can be used.

2 Attacks against ECDLP

2.1 On the Static Diffie-Hellman Problem on Elliptic Curves over Extension Fields

One proposed attack in [10] against small elliptic curves was made that followed [4] in attacking the Static DHP. That is, it assumes the existence of an oracle that will calculate dX for any X in $E(\mathbb{F}_{q^n})$, where d is the private key of interest, q is prime or a prime power, and n is a small integer (≤ 4). The aim is to find dR for a given R, without directly querying the oracle with R.

The algorithm constructs a factor base F – a series of elements in $E(\mathbb{F}_{q^n})$ where the x-coordinate is taken from $\mathbb{F}_q$. It then constructs Semaev's equation f_{n+1} with x_R in the last position, and finds its Gröbner basis. Because Semaev's equations produce $x_i \in \mathbb{F}_q$, we should be able to query the factor base to find the corresponding points P_i in $E(\mathbb{F}_{q^n})$. We can then combine

the factor base elements $P_1, ..., P_n$ to calculate R. Calling the oracle on each of these $P_1, ..., P_n$ to produce $dP_1, ..., dP_n$ and combining them in the same way will give you the desired dR.

2.2 On the discrete logarithm problem in elliptic curves

Another attack proposal on ECDLP was made by Gaudry in [9], improved by Diem in [7], and with improved complexity analysis and applicability to prime powers in [15], which also makes use of Semaev's summation polynomials and reduces the effective field size from $\mathbb{F}_{q^n}$ to $\mathbb{F}_q$.

The basic idea is similar – the aim is find points $P_1, ..., P_n$ that can be combined to calculate R. However, rather than build a factor base from those points in $E(\mathbb{F}_{q^n})$ with x coordinates in $\mathbb{F}_q$ – it uses a technique called Weil restriction to map the elliptic curve $E(\mathbb{F}_{q^n})$ to a more complex "hyper-elliptic" curve in $\mathbb{F}_{q^m}$, where $m|n$. This produces larger polynomial systems to be solved, but makes the search space quicker to enumerate and allows Sameav's summation polynomials to be used when $m \leq 4$.

Finding the Gröbner basis of Semaev's summation polynomial f_{m+1} over the mapped curve in $\mathbb{F}_{q^m}$ produces results in $\mathbb{F}_q$ that map directly to a solution on the elliptic curve in $\mathbb{F}_{q^n}$.

3 Implementation of Gröbner basis algorithm

As we have seen, several proposed attacks against elliptic curves construct Semaev's summation polynomials and calculate their Gröbner basis. In experimentation, we have found that this Gröbner basis calculation tends to be the most computationally expensive operation in these algorithms, and therefore finding a more efficient implementation would improve these attacks, potentially making them able to target larger elliptic curves. We had access to the following candidates for evaluation:

- FGb[8]: closed source implementation, provides C/C++ bindings as well as a Maple extension,
- Magma[3]: closed source implementation, accessible only through Magma,
- Singular[5]: open source implementation, provides Sage[6] bindings as well as command line execution.

In the following sections, I will contrast and compare the different implementations, their debug output, and their documentation.

3.1 FGb

FGb is the original implementation of Faugère's F4 and F5 algorithms for calculating Gröbner bases, written and maintained by the author. The

library is written in C, making use of their own linear algebra library, and also has bindings so it can be called directly from the Maple programming language. It supports multi-threading, but the number of threads used must be specified prior to execution being started and will not be dynamically altered based on workload. One potential limitation to FGb is that the documentation states that when working over a finite field $\mathbb{F}_p$, p must be prime – it does not support finite fields with prime power orders.

The debug output produced when executing FGb from the C interface is limited, even when run with verbose logging. At each step of the F4 algorithm, it will output only the degree and the dimensions of the matrix, without giving any indication of the sparsity of that matrix or where the program is spending the majority of its time other than doing a separate timer for symbolic and linear algebra operations.

The documentation available for FGb is good, featuring both an in-depth technical manual that explains how the C interface is used, providing a short sample program to see a full implementation, as well as a simplified introduction to the Maple interface. However, there is little to no detail on how to improve performance, beyond increasing the number of threads available. However, even this is not fool-proof and if the number of threads is larger than necessary, it may result in decreased performance.

3.2 Magma

Magma uses its own implementation of the Gröbner basis algorithm, using a dense variant of Faugère's F4 algorithm [18]. This implementation allows Magma to make significant efficiency gains when working with systems of polynomials where each variable appears in most polynomials. Since the systems we are working with feature small number of complex polynomials, this seems to be likely.

Unlike FGb, the CPU implementation only supports single threaded execution – however, Magma also supports using the GPU for Gröbner basis calculations through NVIDIA's CUDA interface, for highly parallel operation. Since no other languages supports this option, we will not use it when evaluating the implementations – but the experimental results in [18] have shown that using CUDA may result in better performance.

The debugging output Magma produces when calculating Gröbner bases has a high level of granular control, and at the most verbose level it provides a detailed insight into what calculations are being done as well as the basic matrix dimensions, sparsity and degree.

Magma's documentation for their Gröbner basis algorithm is thorough, but not always easily found. Different versions of Magma documentation have different levels of information, some of which is outdated, but others still seem to be relevant. For example, a previous version of the manual makes suggestions regarding the addition of certain polynomials to the sys-

tem for efficiency. This recommendation no longer appears in the current version of the documentation – but the performance improvement is still notable.

3.3 Singular

Singular is an open source library that includes an implementation of Gröbner basis, using Faugère's F4 algorithm. It can be called directly, but Sage (an open source programming language) provides an easier interface to program against, without a significant performance reduction.

Unfortunately, the level of debugging provided by Singular is not currently sufficient to fully evaluate the implementation. As far as we can tell, Singular will only output the degree of each step, without providing any information about the size of the matrices used. As Singular is open source, it would be conceivable that this information could be added, but it was beyond the scope of this project to do so. However, during ad-hoc testing, Singular appeared to be an order of magnitude slower than FGb, so excluding it from the results is not a massive drawback.

Singular's documentation is generally high quality, although it also provides very little guidance on improving performance.

4 Results

To evaluate each of these Gröbner basis implementations for the purpose of attacking elliptic curves, we use the following methodology:

1. Generate a random elliptic curve and Semaev's summation polynomials in Magma using Diem-Gaudry's ECDLP algorithm, producing a system of polynomials
2. Convert those polynomials into a FGb C program
3. Calculate the Gröbner basis with both Magma and FGb, recording any debugging output and the result
4. Compare the Gröbner bases to ensure they produced the same normalised basis

The measurements that are compared in table 1 are the mean and maximum size of the matrices used, and the CPU time taken in seconds (as reported by the program). All measurements were performed on a E5-2680 Intel® Xeon® CPU with 8 cores (32 hyper-threaded cores) clocked at 3.1GHz, with 256GB of RAM and no GPU processing available. The parameters give the size of the finite field $\mathbb{F}_{2^n}$ that the target elliptic curve is over, and the size of the smaller sub-field $\mathbb{F}_{2^m}$ that the Gröbner basis is calculated

		FGb			Magma			Speed-up
n	m	Mean	Max	Time	Mean	Max	Time	Ratio
12	3	2.44×10^7	1.44×10^8	2.16	8.24×10^5	5.27×10^6	0.26	8.31×
15	3	1.15×10^9	1.16×10^{10}	210	5.7×10^7	5.64×10^8	8.11	25.9×
18	3	1.83×10^{10}	1.42×10^{11}	13500	3.02×10^8	3.88×10^9	329	41×

Table 1: Matrix sizes and timing info for different implementations of Gröbner basis algorithm on Sameav's summation polynomial, all values reported to 3 significant figures

over. Each test was performed three times (with the results being the mean of those runs), in order to avoid biasing due to particular parameters being extreme cases. The final column shows the ratio of the time taken by FGb over the time taken by Magma. A positive number represents Magma being faster than FGb, a negative number shows the opposite.

The numbers clearly show that Magma not only produces two orders of magnitude smaller matrices during the Gröbner basis calculation, it also does so faster than FGb for these polynomial systems, with a super-linear speed-up as the size of the finite field increases linearly.

5 Conclusion

Overall, the results have shown that Magma's implementation of the Gröbner basis is significantly more efficient when used in combination with Diem-Gaudry's elliptic curve index calculus attack. This seems to be due to its use of the specialised dense variant, which appears to be effective when used on the polynomial systems produced by Sameav's summation polynomials over a hyper-elliptic curve. This variant not only produces smaller matrices, it also reduces the complexity of the linear algebra operations, making significant performance gains.

Further investigation could explore whether Magma's GPU implementation of the Gröbner basis algorithm would provide any additional performance improvement, or how other attacks on elliptic curves that make use of the Gröbner basis algorithm are affected by their choice of implementation. As mapping a curve in a field to a sub-field curve seems to inevitably increase the complexity of the polynomial systems, it seems likely that this is the case, but this work was certainly not sufficient to show it.

A more in-depth work might attempt to investigate how Sameav's summation polynomials can be more efficiently represented and used. Reducing the memory complexity of f_n could allow direct attacks on elliptic curves that are in use today, as well as improving the potential of attacks that use processes such as Weil descent.

References

[1] *Digital Signature Standard (DSS) (FIPS 186-4)*. National Institute of Standards and Technology (NIST), July 2013. URL `http://dx.doi.org/10.6028/NIST.FIPS.184-4`.

[2] The Moz Top 500, August 2016. URL `https://moz.com/top500`.

[3] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4): 235–265, 1997. ISSN 0747-7171. doi: 10.1006/jsco.1996.0125. URL `http://dx.doi.org/10.1006/jsco.1996.0125`. Computational algebra and number theory (London, 1993).

[4] Daniel RL Brown and Robert P Gallant. The Static Diffie-Hellman Problem. *IACR Cryptology ePrint Archive*, 2004:306, 2004.

[5] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-0-2 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2015.

[6] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.2)*, 2016. `http://www.sagemath.org`.

[7] Claus Diem. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147(01):75–104, 2011.

[8] Jean-Charles Faugère. FGb: A Library for Computing Gröbner Bases. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg. doi: 10.1007/978-3-642-15582-6_17.

[9] Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, 44(12):1690–1702, 2009.

[10] Robert Granger. On the static diffie-hellman problem on elliptic curves over extension fields. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 283–302. Springer, 2010.

[11] K Igoe. Suite B Cryptographic Suites for Secure Shell (SSH). RFC 6239, RFC Editor, May 2011. URL `https://tools.ietf.org/html/rfc6239`.

[12] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. In *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer, 2014.

[13] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[14] L Law and J Solinas. Suite B Cryptographic Suites for IPsec. RFC 6379, RFC Editor, October 2011. URL `https://tools.ietf.org/html/rfc6379`.

[15] Christophe Petit and Jean-Jacques Quisquater. On polynomial systems arising from a weil descent. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 451–466. Springer, 2012.

[16] M Salter and R Housley. Suite B Profile for Transport Layer Security (TLS). RFC 6460, RFC Editor, January 2012. URL `https://tools.ietf.org/html/rfc6460`.

[17] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. *IACR Cryptology ePrint Archive*, 2004:31, 2004.

[18] Allan Steel. Direct solution of the (11,9,8)-minrank problem by the block wiedemann algorithm in magma with a Tesla GPU. In *Proceedings of the 2015 International Workshop on Parallel Symbolic Computation*, PASCO '15, pages 2–6, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3599-7. doi: 10.1145/2790282.2791392. URL `http://doi.acm.org/10.1145/2790282.2791392`.