The Factoring Dead: Preparing for the Cryptopocalypse

Javed Samuel — javed[at]isecpartners[dot]com

iSEC Partners, Inc
123 Mission Street, Suite 1020
San Francisco, CA 94105
https://www.isecpartners.com

March 20, 2014

Abstract

This paper will explain the latest breakthroughs in the academic cryptography community and look ahead at what practical issues could arise for popular cryptosystems. Specifically, we will focus on the recent major developments in discrete mathematics and their potential ability to undermine our trust in the most basic asymmetric primitives, including RSA. We will explain the basic theories behind RSA and the state-of-the-art in large numbering factoring, and how several recent papers may point the way to massive improvements in this area.

The paper will then switch to the practical aspects of the doomsday scenario, and will answer the question "What happens the day after RSA is broken?" We will point out the many obvious and hidden uses of RSA and related algorithms and outline how software engineers and security teams can operate in a post-RSA world. We will also discuss the results of our survey of popular products and software, and point out the ways in which individuals can prepare for the "zombie cryptopocalypse".

1 Introduction

Over the past few years, there have been numerous attacks on the current SSL infrastructure. These have ranged from BEAST [97], CRIME [88], Lucky 13 [2] [86], RC4 bias attacks [1] [91] and BREACH [42]. These attacks all show the fragility of the current SSL architecture as vulnerabilities have been found in a variety of features ranging from compression, timing and padding [90]. It is therefore urgent for the various stakeholders to begin the migration to more secure cryptosystems with stronger mathematical guarantees.

Typically it has taken a decade or more before cryptosystems or cryptographic functions move from the academic world to practice. This can be seen by the time taken for DES and MD5 to finally be transitioned out which are still used in numerous applications. In some cases these provably weak cryptosystems are still used today by some organizations and have not yet been deprecated. The delay in the transition to Elliptic Curve Cryptography (ECC) has been further exacerbated by the murky patent situation resulting in many organizations taking a wait-and-see approach. Even with the advent of NSA's Suite B recommendation, the uptake of ECC has been very slow and few organizations currently use ECC as their default cryptosystem.

The significant cost of migration to a new cryptosystem or algorithm tends to discourage practitioners from updating their organization's code or infrastructure. Furthermore, many systems are not designed for cryptographic agility and as a result any changes to cryptosystems usually results in significant development and testing time. The systems in the academic world tend to be easier to upgrade and test and they are usually not faced with the backwards compatibility problem. Furthermore, it can be challenging for Information Security practitioners in the real world to understand the ramifications of improvements in the theoretical cryptography world and why these academic

breakthroughs may matter. It is hoped that this paper describes some of the potential implications of continued improvements in some of the key mathematical problems used in today's cryptosystems.

2 Main

The paper will be split into two major sections, "The Math" and "The Impact". "The Math" discusses several cryptographic systems and their related mathematical concept. Some new developments which potentially can dramatically affect today's current cryptographic systems are discussed. "The Impact" section discusses some of the issues that software engineers and designers may face if some of the fundamental mathematical assumptions that are relied on in may cryptosystems are shattered.

3 The Math

Modern cryptographic system rely on several key mathematical assumptions. The security of a cryptosystem is usually based on the hardness of a corresponding mathematics problem. Any algorithmic advances of the underlying mathematical problems could potentially break today's cryptosystems. This section describes several cryptosystems and some major new advances in 2013 particularly in discrete logarithm research .

3.1 Asymmetric Cryptography Mathematics

This section discusses some of the fundamental mathematics used in today's cryptographic systems. The cryptosystems and their corresponding mathematical problem are the following:

- RSA Cryptosystem and Factoring
- Diffie-Helman Cryptosystems and Discrete Logarithms
- Elliptic Curve Cryptosystems and Elliptic Curve Discrete Logarithms

3.1.1 Factoring

The Factoring Problem can be described as follows: given a positive composite integer N, find an integer x, with $1 < x < N$, such that x divides N.

Factoring is the underlying, presumably computationally hard problem upon which several public-key cryptosystems are based, including RSA. The ability to factor an RSA modulus allows an attacker to compute the private key of a public/private key pair; thus, anyone who can factor the modulus can decrypt messages and forge signatures. The security of RSA is completely dependent on the factoring problem being difficult and the presence of no other types of attack. Unfortunately, it has not been proven that factoring is computationally difficult, and there remains a possibility that a polynomial time factoring method might be discovered in the future.

The time required for factoring is dependent on the size of the number. Hence the size of the modulus in RSA determines how secure an actual use of RSA is. The larger the RSA modulus, the longer it would take an attacker to factor, and thus the more resistant to attack the RSA modulus is. Currently, the best known algorithm is capable of factoring a 768-bit RSA modulus. The factoring problem and the discrete logarithm problem appear to be interrelated. Generally improvements in algorithms that solve the discrete logarithm problem have resulted in similar improvement in factoring algorithms.

Fermat's Little Theorem is highly useful in number theory for simplifying the computation of exponents in modular arithmetic . If a is an integer, p is a prime number and a is not divisible by p , then $a^{p-1} \equiv 1 \pmod{p}$.

A frequently used corollary of Fermat's Little Theorem is $a^p \equiv a \pmod{p}$. As can be seen, this is derived by multiplying both sides of the theorem by a . The restated form is helpful since it no longer restricts oneself to only integers a not divisible by p. Fermat's Little Theorem is at the heart of the RSA cryptosystem.

3.1.2 RSA

The RSA scheme of Rivest, Shamir, and Adleman is as follows. Key pairs are generated by the Gen algorithm:

$Gen(1^k)$:

1. Select random k-bit prime numbers p and q, and let $N = p \cdot q$
2. Select at random $e < \phi(N)$ so that $gcd(e, \phi(N)) = 1$
3. Solve $ed \equiv 1 (mod \phi(N))$ for d
4. Output $PK = (N, e)$; $SK = (N, d)$

The space of possible messages for a given public key (N, e) is all integers $m \in Z : 0 < m < N$), and the encryption operates as follows:

$$Enc_{(N,e)}(m): \text{Output } m^e \bmod N$$

Decryption for RSA is the same operation as encryption, using the secret key as the exponent instead of the public one as follows:

$$Dec(N, d)c): \text{Output } c^d \bmod N$$

If there is an efficient algorithm for factoring integers with high probability, then there also exists an efficient algorithm exists to recover RSA private keys with high probability. Therefore, if a polynomial time factoring algorithm is ever discovered, RSA will then be completely broken and no longer suitable for cryptographic purposes.

3.1.3 General Number Field Sieve (GNFS)

The General Number Field Sieve algorithm is the fastest known method for factoring large integers. It is a sub-exponential time algorithm which means that it can be solved in running times whose logarithms grow smaller than any given polynomial. The general number field sieve (GNFS) is an improvement to the simpler rational sieve or quadratic sieve. The quadratic or rational sieve algorithms search for smooth numbers (i.e. numbers with small prime factors) of order $\frac{n^{\frac{1}{2}}}{2}$ when factoring a large number n. The size of these values is exponential in the size of n. The general number field sieve, on the other hand, manages to search for smooth numbers that are **sub-exponential** in the size of n. Since these numbers are smaller, they are more likely to be smooth than the numbers used in the rational sieve or quadratic sieve. However, this adds to the complexity of the GNFS.

On December 12, 2009, Kleinjung et al factored the 768-bit, 232-digit number RSA-768 by the number field sieve. The number RSA-768 was taken from the now obsolete RSA Challenge list as a representative 768-bit RSA modulus. This result is currently the record for factoring general integers. It is likely that a 1024-bit RSA modulus will be factored within the next couple of years if it has not already been done yet. Additionally, if some of the techniques recently discovered by Joux's new discrete logarithm algorithm [59], can be applied to factoring even larger RSA-modulus could be factored in the near future.

Childers in 2012 was also able to factor a 1061-bit number using the Special Number Field Sieve (SNFS). He used the SNFS to determine the complete factorization of the Mersenne number $2^{1061} - 1$ using publicly available software. This algorithm consists of five basic steps:

- polynomial selection
- sieving for relations

- filtering of relations
- linear algebra
- square root

The current factoring records are as follows:

- 768-bit, 232-digit number from RSA's challenge list in 2009 which took 3300 CPU years
- 200 digit number in 2005 which took 170 CPU years
- Mersenne number $2^{1061} - 1$ by the Special Number Field Sieve in 2012 which took 400 CPU years
- 1024-bit, in progress though with currently known techniques this is likely more than 5 years away. However with new techniques it could be much sooner.

Some public GNFS implementations that are currently available. Other researchers have may also have their own proprietary implementations.

1. MSIEVE [56] is a C library implementing a suite of algorithms to factor large integers. It contains an implementation of the GNFS algorithms; the latter has helped complete some of the largest public factorization known.
2. CADO-NFS [10] is a complete implementation in C/C++ of the Number Field Sieve (NFS) algorithm for factoring integers. It consists in various programs corresponding to all the phases of the algorithm, and a general Perl script that runs them, possibly in parallel over a network of computers.
3. GGNFS [75] is a GPL'd implementation of the General Number Field Sieve (GNFS) for factoring integers. Active development is currently stalled. It has been used on SNFS numbers up to 180+ digits and general numbers up to 140.

3.2 Diffie-Hellman Exchange

Diffie–Hellman establishes a shared secret that can be used for secret communications by exchanging data over a public network. A general description of the protocol is as follows:

- Alice and Bob agree on a finite cyclic group G and a generating element g in G. (This is usually done long before the rest of the protocol; g is assumed to be known by all attackers.)
- Alice picks a random natural number a and sends g^a to Bob.
- Bob picks a random natural number b and sends g^b to Alice.
- Alice computes $(g^b)^a$
- Bob computes $(g^a)^b$
- Both Alice and Bob are now in possession of the group element g^{ab}, which can serve as the shared secret key. The values of $(g^b)^a$ and $(g^a)^b$ are the same because groups are power associative.

In order to decrypt a message m, sent as mg^{ab}, Bob (or Alice) must first compute $(g^{ab})^{-1}$.

When Alice sends Bob the encrypted message, mg^{ab}, Bob applies $(g^{ab})^{-1}$ and recovers $mg^{ab}(g^{ab})^{-1} = m(1) = m$.

The security of the Diffie-Hellman protocol relies on the **hardness of the discrete logarithm problem**.

3.2.1 Discrete Logarithm Problem

Suppose $h = g^x$ for some g in the finite field and secret integer x. The discrete logarithm problem is to find the element x, when only g and h are known.

The discrete logarithm problem is hard in general and is used as a hard problem in cryptography. The security of many widely used public key cryptosystems such as the well-known Diffie-Hellman key exchange algorithm and the El-Gamal Digital signature algorithm, depends on the assumption that for suitably chosen primes, discrete logs are hard to compute. As such, one of the most stimulating factors in research on the complexity of discrete logs is the fact that fast discrete logarithm algorithms could easily undermine these cryptosystems. Discrete logarithm algorithms can be broken down in two main categories:

- Generic algorithms (for any G) such as Pohlig-Hellman which shows that discrete logarithm can be computed by breaking up the groups into subgroups of prime order. Generic algorithms are exponential time algorithms.
- Specific algorithms (make use of group representation) such as index calculus algorithms. These are the main algorithms discussed in this paper. They leverage particular properties of the group and result in sub-exponential running time.

For the past 20 years, the fastest index calculus algorithms have had a run time of $L(\frac{1}{3})$ where L is:

$$L(a) = O(exp((\log q)^a(\log \log q)^{(1-a)}))$$

However in a major new development, Antoine Joux has discovered a new algorithm for discrete logarithms in small characteristics which has total complexity of $L(\frac{1}{4} + o(1))$. The order, or number of elements, of a finite field is of the form p^n, where p is a prime number called the characteristic of the field. Joux's new index algorithm only remains practical for characteristics less than 30, at which the number of equations that need to be solved become impractical with current mathematical techniques.

This result, as well as others, indicate that finite fields of small characteristic are not appropriate for pairing-based cryptography [66] which combines elements of two cryptographic groups to a third group to construct cryptographic systems. The security of pairing-based cryptosystems is based on the intractability of variations of the Diffie-Hellman problem in the respective groups. Pairing- based cryptography (PBC) can be used for identity-based encryption, keyword searchable encryption and other applications for which traditional public key cryptography may be unsuitable. PBC has been very attractive for cryptographers since 2000, when it was used to develop a one-round three-party key agreement protocol as an alternative to the two-round three-party Diffie-Hellman key exchange. However, being a novelty in the cryptographic world, its security has not been thoroughly studied. The Discrete Logarithm Problem has now been solved in a finite field of 2^{6120} elements (characteristic = 2). Similar methods would apply to characteristic 3 and other small characteristics as well. As a result, since finite fields with small characteristics which can now be solved with a faster sub-exponential algorithm should no longer be used in pairing-based cryptosystems.

3.2.2 New Developments in Discrete Logarithm Research in 2013

There have been some major developments in the world of discrete logarithm research in the past year. These changes may have a significant impact on the public-key cryptography field, affecting everyone from code makers and code breakers, to published standards and cryptography guidelines. Some of these new developments in the field of discrete logarithm research include the following:

- February 15, 2013: Robert Granger, Faruk Gologlu, Gary McGuire and Jens Zumbragel [44] found discrete logarithms in the finite field of $GF(2^{1971}) = GF((2^{27})^{73})$. This is over twice the bit-length of the best record for factoring or discrete logarithms in large characteristic fields.
- February 20, 2013: Antoine Joux [59] found a method to solve discrete logarithms in small characteristic fields which runs in time $L(\frac{1}{4})$.

- February 22, 2013: A. Joux [60] found discrete logarithms in the finite field $GF(2^{4080}) = GF((2^{16})^{255})$.
- April 6, 2013: Barbulescu, Cyril Bouvier, Jérémie Detrey [14] discuss details on solving the discrete logarithm problem in the 202-bit prime order subgroup of $F_{2^{809}}$ using the Function Field Sieve algorithm (FFS).
- April 11, 2013: R. Granger, et al. [49] announced a new record of solving the DLP in $GF(2^{6120}) = GF(((2^{24})^3)^{255})$.
- May 21, 2013: A. Joux [58] found discrete logarithms in the finite field $GF(2^{6168}) = GF((2^{257})^{24})$.
- June 18, 2013: Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, Emmanuel Thomé [15] published a quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. They published a revision of the paper in November 26 2013.
- October 18, 2013: Cheng, Wan and Zhaung [26] published a Traps to the BGJT-Algorithm for Discrete Logarithms.
- February 15, 2014: Granger, Kleinjung and Zumbrägel [50] published how to solve discrete logarithms in $F_{2^{4\cdot1223}}$ and $F_{2^{12\cdot367}}$.

3.2.3 Discrete Logarithm Algorithms

Discrete logarithm computations in the fields of the form F_{q^n} can be performed either with the function field sieve or the number field sieve. This is dependent on the size of q. The function field sieve and number field sieve are index calculus algorithms that consists of three main phases namely:

- Generation of multiplicative relations - Relation collection phase which produces many multiplicative relations between elements of the smoothness basis. This is modulo group order for discrete logarithm and modulo 2 for factoring.
- Linear algebra - Linear system solving phase based on the coefficient of the multiplicative relations. This produces a vector of consistent logarithms for all elements in the smoothness basis. This phase contains a large sparse system. The numbers of unknowns can range up to dozens of millions and the number of equations potentially very large. Specialized hardware and software are best to solve such systems.
- Computation of individual logarithms - The final descent phase has the input as an arbitrary element of the finite field and then expresses it as a product of elements of the smoothness basis.

3.2.4 Antoine Joux new index calculus algorithm for discrete logarithm [59] — February 2013

Joux describes a new index calculus algorithm to compute discrete logarithms. This work applies to small characteristic fields. The main contributions are a new method for generating multiplicative relations among elements of a small smoothness basis and a new descent strategy that allows expresses the logarithm of an arbitrary finite field element in terms of the logarithm of elements from the smoothness basis. For a small characteristic finite field of size $Q = p^n$, this algorithm achieves heuristic complexity $L_Q(\frac{1}{4} + o(1))$. Some key ideas from his paper include the following

- **Linear Change of variables**. Earlier work showed that a polynomial f that can be transformed into several such polynomials using a linear change of variables $f(X) --> f(aX)$, for any constant a. This appears to be true for a larger class of change of variables than initially expected. The general change of variables used is $x = \frac{aX+b}{cX+d}$. This magnifies a single polynomial to a much larger extent than previously assumed. The linear change of variables, the number of amplified copies of a single polynomial is close to the size of the finite field in which a is picked. This judicious change of variables allows multiplicative relations to be found easily.
- **Systematic polynomial splitting**. The earlier change variables ensures that there are many copies of one polynomial, therefore it is possible to start from a single polynomial f. A specific polynomial with particular

factors can be chosen by design. This is greatly preferable than considering many polynomials until a candidate is found. Over a small finite field F_q, a candidate polynomial with these characteristics is: $f(X) = X^q - X$. It can then be easily seen that this polynomial splits into linear factors, since any element of F_q is a root of f.

- **Field definition**. The image of $X_q - X$ by a homography is a polynomial which only contains the monomials X^{q+1}, X^q and 1. To obtain a multiplicative relation, it is thus desirable to find a finite field representation that transforms such a polynomial into a low degree polynomial. This can be done by choosing the finite field representation in a way similar to that of Coppersmith's algorithm.
- **Finite Field Embedding**. The small characteristic finite field F_{p^n}, is embedded within a field of the form $F_{q^{2k}}$, with $k \leq q$. Two specific low degree polynomials are chosen which result in particular commutative properties. These characteristics yield simple linear relations between the elements of the smoothness basis and allow a reduction in size. This method works because the algorithm is dedicated to small characteristic, and only leads to low degree extensions.
- **New Descent Algorithm**. The computation of individual discrete logarithm phase relies on a descent method which contains two main strategies. For elements with representations of high degree, the typical descent strategy is used, while for lower degrees a new strategy based on the resolution of multivariate systems of bi-linear equations. This new descent step is essential to reach the announced complexity.
- **Improved Complexity**. Complexity is $L(\frac{1}{4}+o(1))$ which is the first sub $L(\frac{1}{3})$ algorithm published for discrete logarithms.
- **New Records**. This algorithm was used to compute the current discrete logarithm record [57]. The author computed discrete logarithms in $GF(2^{6168}) = GF((2^{257})^{24})$ using less than 550 CPU hours with this new index calculus algorithm using his proprietary solver.

Some limitations of Joux's new discrete algorithm algorithm are the following:

- Must have small fixed characteristic (probably less than 30). The order, or number of elements, of a finite field is of the form p^n, where p is a prime number called the characteristic of the field, and n is a positive integer. For small characteristic finite fields, the coefficients of the polynomials are small integers. In a "binary finite field," for example, the coefficients are 0 and 1, and the characteristic is 2. Large characteristic finite fields are however used in practice for example in the digital signature algorithm (DSA). Small characteristic fields are not used in today's current cryptosystems but are used in academic circles. As the size of the characteristics increases, the number of equations that needs to be solved increased significantly and becomes computationally unfeasible. The characteristic must be smaller than the square root of the extension degree.
- An open question whether this idea weakens fields F_{2^k} where k is prime and $1000 < k < 2000$ which are used in discrete logarithm cryptography. Joux's algorithm applies to composite field extensions such as $F_{2^{mk}}$ where m and k satisfy some relationship. Obviously prime fields F_{2^k} can be embedded into $F_{2^{mk}}$ for a suitable m. However, it does seem not computationally feasible to use his algorithm for larger fields. Though, additional special properties of certain primes within that range $1000 < k < 2000$ could conceivably allow it to be solvable.
- Joux's algorithm uses polynomials that decompose more frequently than usual but then still uses the standard heuristic smoothness assumption because he has no known reason to say they deviate from normal behavior.
- It is not currently known how to compute arbitrary logarithms using only linear polynomials and hence Joux extends the basis of known logarithms to include polynomials of degree 2. It might be necessary to extend and include polynomials of larger degree based on the finite field. However, this would reduce the efficiency of the algorithm.
- The classical descent method used cannot be used in the general characteristic case and is modified to work better on low degree polynomials. It does support fixed characteristic fields for example 2 or 3, and more general fields may require another technique. Joux interchanges between the classical descent (for high to middle degree polynomials) and the new descent (for medium to low degree polynomial).

- The new descent method creates a bi-linear system with $(D - d) + d$ variables. The complexity of this is exponential in the small number of variables. A judicious choice of d ensures that the complexity of the algorithm is kept to $L(\frac{1}{4} + o(1))$.

Some current implications of Joux's new index calculus algorithm include the following:

- Pairing based cryptography which can use small characteristics such as (2 and 3) may no longer be secure. It is advised that such cryptosystems move away from using small characteristics.
- Joux's algorithm may be improved over time and be able to replace the function field sieve for all cases. The function field sieve currently can be used to solve small to medium p while the number field sieve is preferable for larger values of p. It seems unlikely that such techniques can be used for the number field sieve since they rely on the action of Frobenius in order to complete the reduction. As a result there is no obvious threat to most practical finite fields of large characteristics.

3.2.5 Gologiu, Granger, McGuire and Zumbragel [44] — February 2013

This paper discusses a variant of the Joux-Lercier Function Field Sieve [62] which results in complexities as low as $L_{q^n}(\frac{1}{3}, \frac{2}{3})$ for computing arbitrary logarithms and also a heuristic polynomial time algorithm for finding discrete logarithms of degree one elements. Some key ideas proposed by the authors include:

- Key substitution of $Y = X^{2^k}$ which ends setting $g_2(X) = X^{2^k}$. This results in automatically eliminating half of the factor base. This applies because any linear polynomial $(Y + a)$ is then equal to $(X + a^{2^{-k}})^{2^k}$, and hence $\log(Y + a) = 2^k.\log(X + a^{2^{-k}})$.
- The use of non-prime base fields induces extra automorphisms of the factor base which reduces its size further.
- There are other performance speedups from the $Y = X^{2^k}$ substitution. This includes the matrix-vector multiplication being able to done by only rotations and an increase in the probability of smoothness.

Some limitations and assumptions in the paper are:

- Only logarithms of degree one elements can be computed using the author's transformations. It does not appear that this can be generalized to any higher and more practical degrees.
- The complexity of solving the discrete logarithm is dependent on the values of n, k and d_1. This ranges from $L_Q(\frac{1}{3}, \frac{2}{3})$ to $L_Q(\frac{1}{3}, 0.961)$.

3.2.6 Barbulescu, Cyril Bouvier, Jérémie Detrey [14] — April 2013

This paper from April 2013 discusses details on solving the discrete logarithm problem in the 202-bit prime order subgroup of $F_{2^{809}}$ using the Function Field Sieve algorithm (FFS). This computation is the largest discrete logarithm computation so far in a binary field extension of prime degree. At some point it may turn out to be more efficient to solve discrete logs in $GF(2^p)$, where p is prime, by embedding them into fields $GF(2^{p*m})$. But for the moment, the best way to solve such problems appears to be the traditional use of the function field sieve. Some key ideas from this computation are the following:

- This choice of polynomials was driven solely by the efficiency of the relation collection. The result was that a polynomial $f(x)$ of degree 6 was the best choice and no special care was taken to ensure that $g(x)$ was a linear monic polynomial.
- The relation collection notions are very similar to the ones used when sieving relations for factoring integers using Number Field Sieve (NFS); irreducible polynomials of a given degree playing the role of prime numbers of a given number of bits.

- All the special-q of degree from 24 to 27 (inclusive) were sieved, producing a bit more than 52 million relations (possibly non-unique).
- The filtering step was performed using the implementation described in Bouvier 2012 [22].It was run on the two sets of relations produced by the earlier relation collection step. The goal of the filtering step is to construct a matrix which is the smallest and the sparsest possible.

Some limitations and assumptions of this computation are the following:

- A skewness of 2 was used for relation collection. It is not clear whether this is the most optimal skewness to be used.
- A factor base bound of 23 was used in the relation collection. The two cases a large prime bound of 27 or large prime bound of 28 are not exhaustive and future work is possible in that area.
- For the actual computation, relations collected with both values of the large prime bound were considered to produce the matrix. This is of course not "fair," in the sense that if the computation were to be run again, they would have only one of the two relation sets.
- The question of where to stop sieving is not so easy to answer in advance. Though with the data that they collected during this research, it can give some hints for future choices for the best stopping point.

3.2.7 Barbulescu, Gaudry, Joux, Thomé [15] — June 2013

This paper first published in June 2013 [16] and then revised in November 2013 [15] discusses a quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. This is a major new development and shows that in certain circumstances the discrete logarithm cannot be considered a hard problem. The main result gives a quasi-polynomial heuristic complexity for the DLP in finite field of small characteristic. The complexity is of type $n^{O(\log n)}$ where n is the bit-size of the cardinality of the finite field. Though it remains super-polynomial in the size of the input, it is a major improvement compared to the sub-exponential algorithms.

Some key ideas from this computation are the following:

- The relation collection and linear algebra phases remains the same as Joux's 2013 algorithm [59].
- The algorithmic building blocks are elementary. They avoid the use of Gröbner basis algorithms [92] which complicates the linear algebra.
- There exists of a polynomials representation of the appropriate form of the finite field.
- The smoothness probabilities of some non-uniformly distributed polynomials are similar to the probabilities for uniformly random polynomials of the same degree.
- There exists linear independence of some finite field elements related to the action of projective linear group $PGL_2(F_q)$.

Some limitations and constraints from this new quasi-polynomial algorithm are:

- The descent process is limited in certain cases. Because of the arity of the descent tree, the breadth eventually exceeds the number of polynomials below some degree bound. It makes no sense, therefore, to use the descent procedure beyond this point, as the recovery of discrete logarithms of all these polynomials is better achieved as a precomputation.
- This technique does not use Gröbner bases which hinders the practical efficiency of the algorithm due to the $O(q^2D)$ We note that one of the key factors which hinders the practical efficiency of this algorithm is the $O(q^2D)$ arity of the descent tree. Joux's earlier algorithm leverages Gröbner bases to obtain a $O(q)$ arity algorithm.

- Only a heuristic proof was provided for this quasi-polynomial algorithm. This proof was related to the certain properties of matrices which may have a more concrete algebraic proof.

Some implications from this new quasi-polynomial algorithm are:

- The algorithm is asymptotically faster than the Function Field Sieve algorithm in almost all the range previously covered by this algorithm. If the characteristic of the base field is not so small compared to the extension degree, the complexity of the algorithm does not keep its nice quasi-polynomial form. However, in almost the whole range of applicability of the Function Field Sieve algorithm, the algorithm is asymptotically better than FFS.
- The complexity of the discrete logarithm algorithm is in $(\log Q)^{O(\log\log Q)}$. This is smaller than any expression of the form $L_Q(\alpha)$ for $\alpha > 0$. Hence, although the complexity is still super-polynomial, it is much closer to polynomial time complexity than the classical sub-exponential complexities which are involved in integer factorization or in previously known discrete logarithm algorithms.

This new research has some interesting possible implications in discrete logarithm field. It will spur greater interest in the field and could result in more generic algorithms. While it is likely that new techniques are necessary to attack practical discrete logarithm field, some of the ideas in today's research could prove invaluable.

- Specific primes fields F_{2^k} may be solvable using similar modified techniques. These primes need to have particular properties that allow reduction in the number of equations that need to be solved. It is not clear how easy it would be prove that a particular finite field is secure even if for now the general prime field will remain computationally difficult to solve.
- A limit of $L(\frac{1}{3})$ stood for close to twenty years which has now been broken in certain cases by making relatively simple modifications to the algorithms. This implies that further research may yield additional improvements which could result in a quasi-polynomial algorithm for discrete logarithms in certain cases. Discrete logarithm currently appears easy for finite fields of particular properties. The first two phases before the individual logarithm phase can be completed in polynomial time.
- The recent computation of $GF(2^{6168}) = GF((2^{257})^{24})$ is that super-singular curves (of characteristic 1 or 2) defined over $GF(2^{257})$ cannot be used securely for pairing-based cryptography. This is because such a curve can be solved in a computationally feasible sub- exponential algorithm. In addition, it is likely that additional curves may be solvable in the future as well.
- Judicious polynomial selection could also provide some benefit in the number field sieve case [61] For the algorithm to be efficient, these polynomials must have small coefficients and there are currently several techniques that are available to choose them.
- Finding an analogue to "action of Frobenius" for more general cases [51]. Joux uses the "action of Frobenius" to reduce the size of the factor basis which allows the systems of equations to be solvable.

3.2.8 Cheng, Wan, Zhaung [26]— October 2013

This paper fixes the BGJT algorithm in non-Kummer cases, without altering the quasi-polynomial time complexity. Their paper addresses various traps that exist in the BGJT algorithm. These traps appear in the descent path in particular circumstances and essentially block the descent resulting in the BGJT algorithm failing in those cases. It is of particular concern if the polynomial has many small degree factors.

Some key ideas from this computation are the following:

- They include some extra requirements to the polynomials h_0 and h_1 which help in the case of non-Kummer extensions.

- They create a trap-avoiding descent strategy which eliminates some of the edge cases faced by the earlier descent strategy. Furthermore, these modifications does not increase the algorithmic complexity of the algorithm.

3.2.9 Discrete Logarithm Computation and Factoring Implementations

Classical CPUs tend to be used for the collection of relations, and GPUs for the linear algebra step. Magma [52] tends to be used by several of the researchers to perform the required linear algebra computation. MPFQ [40] can be used to perform manipulation in finite fields. CADO [10] is a complete implementation in C/C++ of the Number Field Sieve (NFS) algorithm for factoring integers. It consists in various programs corresponding to all the phases of the algorithm, and a general Perl script that run them, possibly in parallel over a network of computers. CADO also recently added an FFS implementation [41] which can serve as a reference point for computing discrete logarithms in small characteristic finite fields and for for dimensioning key-sizes for DLP-based cryptosystems. Joux currently uses a non-public proprietary implementation for his new index calculus algorithm. He leverages some external libraries for certain aspects of his algorithm and uses Magma in certain computations.

3.2.10 Similarities between Discrete Logarithm and Factoring

Historically, it has been the case that an algorithmic advance in either problem, factoring or discrete logs, was then applied to the other. This suggests that the degrees of difficulty of both problems are closely linked, and that any breakthrough, either positive or negative, will affect both problems equally. Mutual advances between discrete logarithm and factoring over the past decades include:

- Théories des nombres (1922) and On Factoring Large Numbers (1931) talk about solving Discrete Logs and Factoring respectively.
- 1975 Pollard's Rho in Factoring -> 1978 Pollard's Rho in Discrete Log.
- 1984 Quadratic Sieve Factoring -> 1987 improvements in Discrete Log Index Calculus Algorithms.
- 1993/4 Discrete Log Number & Function Field Sieves -> 1994 General Number Field Sieve for Factoring.

Factoring and discrete logarithm algorithms are closely related. They have very similar steps and hence improvement in one algorithm tends to lead to a similar improvement in the corresponding problem. Both algorithms have a polynomial selection phase and a relation sieve process. There is a significant linear algebra step. The final step is to solve for the exact number that is to be factored or the specific discrete logarithm.

In factoring, the polynomial selection takes some time depending on the quality of the polynomial chosen. In the polynomial selection step two co-prime, irreducible polynomials, $f_1; f_2 \in Z[X]$, are chosen such that they share a common root m modulo N. The running time of the subsequent steps depends on the quality of the chosen polynomial pair, many pairs should be considered and the best one chosen to result in the lowest running time of the overall algorithm. In Joux's improved discrete logarithm algorithm, he has chosen his polynomial as a constant, based on specific properties of the group.

The next phase in factoring is the relation collection where the aim of this step is to find sufficiently many co-prime pairs $(a; b) \in ZxN$ such that $F_i(a; b)$ is L_i-smooth for $i = 1; 2$, where L_1 and L_2 are parameters. There are several methods for finding relations, in practice one uses line sieving and lattice sieving. Sieving also applies to discrete logarithm algorithms and is in the index calculus algorithm where knowledge about the logarithms of some elements of the group is built up and which generates a significant portion of the group. The logarithms computed in the relation collection step can then be used solve easily for the required logarithm.

Both factoring and discrete logarithm have a computationally intensive linear algebra step which is difficult to parallelize. The computational complexity of the linear algebra is dependent on the earlier phases. For example, special matrix properties help to reduce the algorithmic difficulty of the linear algebra step. The linear algebra phase is more

expensive in discrete logarithm algorithms as compared to factoring algorithms. It requires significant memory and memory bandwidth, in addition to a lot of CPU time.

The most notable difference is that the final step is significantly more difficult for discrete logarithm. The descent is extremely painful for discrete logs, but the analogous step, the square root, takes minutes for factoring. The descent for the discrete logarithm has seen some major advances in the past year.

3.3 Elliptic Curve Cryptography

An elliptic curve E over R real numbers is defined by a Weierstrass equation:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Two other simplified Wierstrass equations that can be defined are the following:

$$y^2 = x^3 + ax + b$$
$$y^2 + xy = x^3 + ax^2b$$

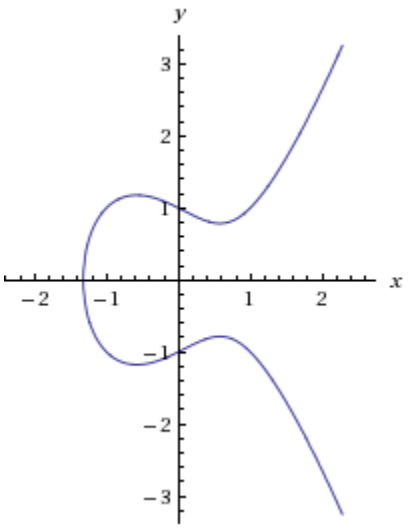

Figure 1: Example of elliptic curve $y^2 = x^3 - x + 1$ using WolframAlpha

Elliptic curves have several interesting properties. One of these is horizontal symmetry. Any point on the curve can be reflected over the x-axis and remain the same curve. A more interesting property is that any non-vertical line will intersect the curve in at most three places.

Elliptic curves over real numbers are generally not used in cryptographic schemes due to the following factors:

- Calculations prove to be slow
- Inaccurate due to rounding error
- Requires an infinite field

Elliptic curves over finite fields come in two flavors:

- Supersingular: Let E be an elliptic curve over F_q. The curve E is said supersingular if $E[q] = O$. A supersingular elliptic curve is one that has the maximum number of symmetries and is the weaker of the two curves.
- Ordinary: In all other cases, an elliptic curve is considered ordinary. This is the general case of elliptic curves and is used in almost all cryptosystems.

Supersingular curves differ from ordinary curves in many ways, and this has practical implications for algorithms that work with elliptic curves over finite fields, such as algorithms for counting points, generating codes, computing endomorphism rings, and calculating discrete logarithms. In practice, ordinary elliptic curves are used in cryptographic applications, however some pairing based cryptographic schemes use supersingular elliptic curves. The recent improvements made by Joux et al. in discrete logarithm algorithms are not applicable to either ordinary or supersingular elliptic curves.

Cryptographic schemes require fast and accurate arithmetic and use one of the following fields:

- Prime Field F_p where p is a prime. Prime curves are best for software applications as they do not need complicated bit-fiddling operations as binary curves do.
- Binary Field F_2m, where m is a positive integer. Binary curves are best for hardware applications since they require fewer logic gates to create a cryptosystem compared to prime curves.

3.3.1 Elliptic Curve Arithmetic

Elliptic curves are used to construct the public key cryptography system. ECC is secure due to the hardness of the elliptic curve discrete logarithm problem (ECDLP). The private key d is selected from $[1, n-1]$ where n is an integer. Then the public key Q is computed by dP where P and Q are points on the elliptic curve. The key pair (d, Q) can be used for a variety of cryptosystems including signature and encryption/decryption.

The intuitive approach is to do the following $dP = P + P...P_{dtimes}$. This requires $d - 1$ times point addition over the elliptic curve. The Double-and-Add algorithm can be used to minimize the number of operations.

For example to compute $17P$, the intuitive approach requires 16 point additions. The Double-and-Add Algorithm would require 4 point doublings and one point addition since $17P = 2(2(2(2P))) + P$.

3.3.2 Elliptic Curve Discrete Logarithm Problem

Given an elliptic curve E defined over a finite field F_q, a point $P \in E(F_q)$ of order n, and a point $Q \in E$, find the integer $d \in [0, n-1]$ such that $Q = dP$. The fastest algorithm to solve ECDLP is Pollard's rho algorithm which is an exponential time algorithm. Since there are sub- exponential time algorithms for factoring (GNFS) and discrete logarithm problem (FFS and Joux's new index calculus algorithm [59]) and only exponential-time algorithms for ECDLP, the NIST recommended key size is significantly smaller for ECC.

An elliptic curve cryptosystem can be defined by picking a prime number as a maximum, a curve equation, and a public point on the curve. A private key is a number d and a public key is the public point dotted with itself d times. Computing the private key from the public key in this kind of cryptosystem is called the elliptic curve discrete logarithm function. This turns out to be the trapdoor function.

The elliptic curve discrete logarithm is the mathematical problem that is the basis for elliptic curve cryptography. The ECDLP has been studied intensively in the academic community for more than thirty years and the current best algorithm to solve the problem is still the naive approach using Pollard's Rho algorithm [11] which is a fully exponential algorithm. Basically, unlike factoring or the traditional discrete logarithm problem, there does not currently exist a shortcut that simplifies the elliptic curve discrete logarithm problem.This means that for numbers of the same size, solving elliptic curve discrete logarithms is significantly harder than factoring or the traditional discrete logarithm problem. Since a more computationally intensive problem means a stronger cryptographic system, it follows that elliptic curve cryptosystems are harder to break than RSA and Diffie-Hellman.

With ECC, one can use smaller keys to get the same levels of security. Small keys are important, especially in a world where more and more cryptography is done on less powerful devices like mobile phones or embedded devices like smart cards. While multiplying two prime numbers together is easier than factoring the product into its component parts, when the prime numbers start to get very long, even just the multiplication step can take some time on a low

Table 1: NIST Recommended key sizes in bits

Symmetric Algorithm	RSA and DH	ECC
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

powered device. While you could likely continue to keep RSA secure by increasing the key length, that comes with a cost of slower cryptographic performance on the client. ECC appears to offer a better tradeoff: high security with short, fast keys as compared to RSA.

There are various factors that affect ECC security [17]. Some of these factors include the following:

- How easy is it to complete a transfer that converts ECDLP into a linear algebraic group DLP? This includes additive and multiplicative transfers. The minimum possible multiplicative-transfer degree for a particular elliptic-curve group is called the embedding degree of that group.
- What is the complex-multiplication field discriminant D of the elliptic curve? Currently there is no evidence of serious problems with either small |D| or large |D|, but the security story is more complicated for small |D| and there are some specialized attacks. Therefore large |D| curves are recommended.
- How rigid is the curve? Rigidity is a feature of a curve-generation process, limiting the number of curves that can be generated by the process. In a fully rigid elliptic curve, the curve-generation process is completely explained. In manipulatable elliptic curve, the curve-generation process has a large unexplained input, giving the curve generator a large space of curves to choose from. The NIST curves contain a unexplained seed and are considered manipulatable.

3.3.3 Elliptic Curve Factoring Problem

Researchers use the the Elliptic Curve method [102] or also known as the Lenstra elliptic curve factorization to compute a large multiple of a point on a random elliptic curve modulo the number to be factored. For comparison to the earlier factoring records, the current elliptic curve factoring records are as follows:

- 79 digits, found by Sam Wagstaff on August 12, 2012.
- 75 digits, found by Sam Wagstaff on August 2, 2012.
- 73 digits, found by J. Bos, T. Kleinjung, A. Lenstra, P. Montgomery on March 6, 2010.
- 68 digits, found by yoyo@home/M. Thompson on December 28, 2009.

3.3.4 ECC Patent Challenges

The general idea of ECC is not patented, however there are a number of patents regarding the efficient implementation of ECC. Most of the these patents are owned by Certicom which is now owned by Blackberry. Some of these patents include:

- Efficient $GF(2^n)$ multiplication in normal basis representation.
- Technique of validating key exchange messages to prevent a man-in-the-middle attack.
- Technique for compressing elliptic curve point representations.

In order to clear the way for the implementation of elliptic curves to protect US and allied government information, the National Security Agency purchased from Certicom a license that covers all of their intellectual property in a restricted field of use. The license would be limited to implementations that were for national security uses and certified under FIPS 140-2 or were approved by NSA. Commercial vendors may receive a license from the NSA provided their products fit within the field of use of NSA's license. Alternatively, commercial vendors may contact Certicom for a license for the same 26 patents.

The patent issue for elliptic curve cryptosystems is the opposite of that for RSA and Diffie-Hellman, where the cryptosystems themselves have patents, but efficient implementation techniques do not have patents. This has proven to a significant barrier for ECC and has limited widespread adoption. However, with the advances in discrete logarithm algorithms and possibly factoring algorithms, it is perhaps time for a quicker migration to Elliptic Curve Cryptography.

3.4 Why do we care?

However, there is no obvious technique from Joux's new discrete logarithm account that can be used for factoring. The public colloquium and publications seem to indicate that NSA/NIST may also already be very concerned about this and might recommend a push to the Elliptic Curve Cryptography (ECC) algorithms as quickly as possible. The fundamental mathematics used in ECC still requires exponential time algorithms which provides significantly greater security guarantees.

A general solution for an even faster sub-exponential discrete logarithm algorithm or factoring algorithm will result in significantly larger key sizes being required. This would be mean that RSA key sizes may have to be more than 15K bits and dramatically slow down performance. Key sizes for other cryptographic schemes such as El-Gamal, DSA and unsigned Diffie-Hellman would also have to be considerably higher. These cryptosystems will all be completely broken if a general purpose polynomial algorithm is discovered and in that case provide little if any security. While the odds of such a general purpose algorithm is remote, the consequences would be extremely significant. This has become more pressing given Joux's new improved discrete logarithm algorithm, though it is not yet an imminent threat since there still remains some work to do be done.

The current main alternative cryptosystems are the respective ECC algorithms such as ECDH, ECIES and ECDSA which are discussed in more detail later in this paper. While the fundamental mathematics used by ECC has been studied for less time than factoring or discrete logarithms, there currently exist only exponential time algorithms to solve the ECC discrete logarithm problems. Therefore, ECC has much better security guarantees than traditional cryptosystems which rely on less computational difficult mathematical problems.

4 The Impact

In this section we address some of the issues that software engineers and designers may face if some of the fundamental mathematical assumptions that are relied on in many cryptosystems are shattered.

4.1 What happens if RSA/DH fails NOW?

RSA and Diffie-Hellman are the de-facto cryptosystems used in practice today including almost all banking and financial systems and in the public key infrastructure for all certificates. Any failure of RSA and Diffie-Hellman would lead to chaos in the entire technology world. Electronic commerce would come to screeching halt as most people scramble to find an alternatives. Currently we are not prepared for a failure of RSA/DH.

4.1.1 RSA vs ECC

ECC is an alternative cryptosystem that has been around since the 1980's and can also be used for key exchange (ECDH) and for signatures (ECDSA). The mathematics used by RSA is somewhat simpler than those involved for elliptic curves. As a result many engineers feel that they understand RSA more than elliptic curves. RSA relies on the hardness of factorization, which has been studied for 2500 years, whereas ECC is dependent on computing discrete logarithm on elliptic curves. Both cryptosystems have had their patent issues but RSA patents have expired since 2000 while some ECC patents are still valid.

There has been a significant effort by both academic researchers and institutional organizations to push for the adoption of elliptic curves in cryptography. However, given RSA's dominant position it seems likely that this will remain the de-facto standard unless factoring is close to being broken. The perceived mathematical complexity and the potential legal risks related to ECC patents continue to hinder wide acceptance of elliptic curves even with their greater mathematical hardness guarantee. ECC also face compatibility issues earlier since different curves could be used, however this is no longer an issue since NIST has specified 15 standard curves [79] [80].

4.1.2 What are different ECC algorithms?

- ECDH(E): The elliptic curve Diffie-Hellman scheme is a key agreement scheme based on ECC. It is designed to provide a variety of security goals depending on its application — goals it can provide include unilateral implicit key authentication, mutual implicit key authentication, known-key security, and forward secrecy — depending on factors such as whether or not public keys are exchanged in an authentic manner, and whether key pairs are ephemeral or static.
- ECIES: The Elliptic Curve Integrated Encryption Scheme (ECIES) is a public-key encryption scheme based on ECC. It is designed to be semantically secure in the presence of an adversary capable of launching chosen-plaintext and chosen-ciphertext attacks.
- ECDSA: The Elliptic Curve Digital Signature Algorithm (ECDSA) is a signature scheme with appendix based on ECC. It is designed to be existentially unforgeable, even in the presence of an adversary capable of launching chosen-message attacks.
- ECMQV: The elliptic curve MQV scheme is a key agreement scheme based on ECC. It is designed to provide a variety of security goals depending on its application — goals it can provide include mutual implicit key authentication, known-key security, and forward secrecy.
- ECWKTS: ECWKTS is a wrapped key transport scheme uses a combination of a key wrap scheme and a key agreement scheme. The key agreement used can be either ECDH or ECMQV, but in either case it must be a 1-pass variant. In a 1-pass variant of a key agreement scheme, during the key deployment phase, entity U must obtain authentic copies of all of the keys of V, in addition to the usual key deployment operations.

ECC is not only the candidate to replace RSA and DH cryptosystems. Some other alternatives include the following cryptosystems:

- NTRU: NTRU is a patented public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. The security of NTRU is based on the hardness of some lattice problems, namely the shortest and closest vector problems. It consists of two algorithms: NTRUEncrypt [53], which is used for encryption, and NTRUSign, which is used for digital signatures. Unlike other popular public-key cryptosystems, it is resistant to attacks using Shor's algorithm and its performance has been shown to be significantly better. NTRU is patented and the patents will be enforced [54] for commercial uses.
- McEliece cryptosystem: The algorithm is based on the hardness of decoding a general linear code (which is known to be NP-hard [69] [24]. The biggest challenge with the McEliece cryptosystem is that very large key sizes are necessary and render it impractical, for example 80 bits of security requires a key size of 500KB.

Various modifications have been suggested to improve the efficiency of the McEliece cryptosystem in order to have more practical key sizes [12].

4.1.3 Overview of Suite B

The US National Security Agency (NSA) recommends a set of inter-operable cryptographic algorithms in its Suite B standard.

The Suite B standard specifies a mode of operation in which only a specific set of secure cryptographic algorithms are used. The Suite B standard specifies:

- The encryption algorithm (AES) (with keys sizes of 128 and 256 bits)
- The key exchange algorithm (Elliptic Curve Diffie-Hellman, also known as ECDH) (using the curves with 256 and 384-bit prime moduli)
- The digital signature algorithm (Elliptic Curve Digital Signature Algorithm, also known as ECDSA) (using the curves with 256 and 384-bit prime moduli)
- The hashing algorithms (using SHA-256 and SHA-384)

4.2 What options do I have on major platforms?

ECC support has been added to many operating systems, programming languages and popular applications. In many cases, it may not be the default cryptographic algorithm used and there could be compatibility issues with more widespread usage of ECC.

4.2.1 Operating Systems

All the major operating system have had ECC support for a number of years.

- OSX/iOS
  - TLS/SecureTransport
    - This was first included in the following library: libsecurity_ssl36800 [3].
    - OSX/iOS TLS has support for RFC 4492 and supports ECDSA and ECDH
    - ECC support is available OSX 10.6.0 [8].
    - ECC support has been approximately available from iOS 3 based on release of OSX support
  - CDSA/CSSM:
    - This has been available OSX only but was deprecated in 10.7 [4].
    - This supports Fast Elliptic Encryption (FEE). FEE uses special primes and fast finite field and modular operations to reduce processor cycles, leading to less power consumption and heat dissipation [28].
  - OpenSSL
    - This has been deprecated on OSX 10.7 and is also not included in iOS.
    - Applications should bundle with OpenSSL if they need to use it.
  - CMS

    * ECC support was first included in the following library libsecurity_smime-36873 [6] [7].
    * ECC has been supported from OSX 10.6.0 [8].
    * It currently supports ECDSA and ECDH.

- Windows
  - CNG
    * ECC support is available on Vista and .NET 3.5 [72].
    * This supports both ECDSA and ECDH. [74]
  - TLS
    * ECC support is available from Vista [71].
    * This has support for ECDSA and ECDH.
  - Windows also has Suite B support [73] . Suite B support was added to IPsec in Windows Vista Service Pack 1, in Windows Server 2008, and in Windows 7. Support has been extended to the Suite B algorithms for the following areas:
    * Main mode
    * Quick mode
    * Authentication settings
- Android
  - Cryptographic Providers:
    * The default provider is a subset of the bouncy castle library which has been available from 4.0 [46].
    * Android currently supports ECDSA and ECDH [32].
    * A popular third party cryptographic provider which provided ECC support earlier on Android.
    * Spongy Castle provides the full bouncy castle library [99].
    * Spongy Castle supports ECDSA, ECDH, ECDHC, ECIES, and ECNR [65].
  - TLS:
    * This is provided by default through OpenSSL. ECC support is available from android-3.2.4_r1 225 [47].
    * Android TLS supports RFC 4492 [21].
    * Android TLS supports ECDSA and ECDH [23].

    Some popular third party providers are the following:
    * CyaSSL which is available from Android 2.4.6 [101].

      CyaSSL supports ECDSA, ECDH and NTRU [100].
    * NSS is available from 3.11 [77]. NSS supports ECDSA and ECDH [76].
  - Blackberry
    * Blackberry has had ECC support for a long time. There appears to be support from as far back as 3.6 [18].

* Unlike most of the companies they do not have any ECC patent issues since their acquisition of Certicom.
* Blackberry's ECC support includes ECDSA, ECDH, ECMQV and ECNR [19].

4.2.2 Programming Languages

- Python:

  Python currently provides ECC support only via third party libraries. This is done through PyECC which is a Python module wrapped around the libseccure library which itself is based off of code developed originally for the seccure utility [13] [87]. pyECC currently supports ECDSA, ECDH and ECIES.

- C: ECC support is only provided through third party libraries such as:
  - OpenSSL
  - NSS
  - GnuTLS
- Java

  ECC Cryptographic support is included in:
  - Java SE6 with third party libraries [84]. This includes support for ECDSA and ECDH [83].
  - Native Java SE7 support [85]. This includes support for ECDSA and ECDH.
  - Third-party support is also available from BouncyCastle. [64]. This includes support for ECDSA, ECDH, ECDHC, ECIES, and ECNR [65].

  ECC support is provided from JSSE SE6 with the EC cryptographic provider. Java TLS includes support for ECDH and ECDSA.

- Ruby:
  - OpenSSL wrapper included in stdlib
  - ECC support is available from 1.8.7 [89].
  - Ruby's ECC implementation supports ECDSA and ECDH.
  - TLS supports everything the underlying OpenSSL library does.

4.2.3 Third Party Libraries

- OpenSSL

  ECC support is available from Version 0.9.8 [82].

  OpenSSL's ECC implementation supports ECDSA and ECDH [81].

- GnuTLS ECC support is available from 2.99.2 [68].

  GnuTLS's ECC implementation supports ECDSA and ECDH [43].

- NSS

  ECC support is available from 3.11 [77]. NSS's ECC implementation supports ECDHA and ECDSA [76]

4.2.4 Popular Applications

- GnuPG - ECC support is included but is not visible without expert switch [63].
- OpenSSH - This has included support for ECC from version 5.7, however it is not the default [39].
- PuTTY - ECC is not supported by PuTTY but is included as a wishlist item [55].
- TOR - TOR uses Curve25519 for ntor handshake. It is also possible to use CC for client connections by leveraging TLS. However, the use of alternative cipher suites is discouraged. RSA is used for the public key cipher [67].
- OpenVPN - OpenVPN uses OpenSSL which includes ECC support [82] [81].
- Filezilla
  - FTPS - The FTPS client uses GnuTLS and provides ECC support [36].

    The FTPS server uses OpenSSL and provides ECC support [35].
  - - SFTP This uses PuTTY which does not support ECC keys yet [37].
- IPSEC - Cisco, Shiva and Nortel gateways support ECDH exchange [25].
- Chrome - Chrome uses NSS Library and provides ECC support [48].
- Firefox - Firefox uses NSS library and has supported ECC from version 2 [78].
- Internet Explorer - Internet Explorer uses the Windows library and includes ECC support in all versions from Vista and above [72].
- Outlook - Outlook is capable of using ECC keys in Vista and above using the Windows libraries.
- BlackBerry Mail - Blackberry uses ECC to encrypt data received while locked [20].

4.2.5 Code Signing

- Windows Code Signing - The default is RSA. ECC is supported through CSPs but no ECC CSP provided by default [72].
- Android Code Signing - Both DSA and RSA are currently supported [45].
- iOS code Signing - This uses CMS [9]. It supports ECDH and ECDSA [6] [7] [5].

4.2.6 Transport security

The only version of SSL/TLS which mentions ECC in its RFC is TLSv1.2. All the other versions do not explicitly mention ECC in their RFC's.

- SSLv2 and earlier are not recommended for use in today's systems and do not have support for ECC.
- SSLv3 [38] does not mention ECC in its RFC's.
- TLSv1.0 [29] does not mention ECC in its RFC.
- TLSv1.1 [30] does not mention ECC in its RFC.
- TLSv1.2 [31] is the first specification to include ECC algorithms. However the only mandatory cipher suite is TLS_RSA_WITH_AES_128_CBC_SHA, which uses RSA for key exchange.

RFC 4492 [21] specifies an extension to TLS that includes EC cipher suites. However, this extension is not required to implement any TLS version. It specifies how ECDH(E) may be used for key exchange and how ECDSA may be used for signing. Most implementations do support EC cipher suites even if not explicitly required. It is unclear how well-tested or deployed EC cipher suites are in practice.

TLS certificate signing - In TLSv1.1 and earlier the server's certificate must have been signed by a CA with the same kind of certificate (e.g. If the server uses RSA to sign ephemeral keys, it must be signed via RSA). In TLS 1.2, this restriction is removed. A certificate containing a key for one signature algorithm may be signed using a different signature algorithm (for instance, an RSA key signed with a DSA key). This implies that the DH_DSS, DH_RSA, ECDH_ECDSA, and ECDH_RSA key exchange algorithms do not restrict the algorithm used to sign the certificate. This is important for migration purposes as a server can use a ECDSA capable certificate without it needing to be signed with ECDSA [31].

4.2.7 The PKI in a post-RSA world

- Buying and using ECDSA certificates - While ECC code root certificates may be available, purchasing ECC certificates is still in the very initial stages. There would significant work required in the transition form RSA to EC certificates. Some examples include
  - Thawte Root Certificate [98] - This root CA is not used today. It is intended for use in the future for SSL and Code Signing services needing an ECC encryption algorithm. This root should be included in root stores. Signature Algorithm: SHA384 With ECC
  - Verisign/Symantec Root Certificate [93] - Symantec has an ECC root for 5 years and just begun offering commercial certificate from this year. The root certificate is an ECC root that will be used in the future to as the root of Trust for Class 1, 2 and 3 certificates ECC certificates and should be included in root stores (ECDSAWithSHA384).

    Symantec's SSL certificates will now offer the choice of three different encryption algorithms - RSA, DSA [95], and ECC [96]. Further information about Symantec's new ECC offering can be found here [94].
  - Entrust ECC Certificates [34] - No global root certificate currently available today. It will use a Public ECC-256 Root. Best-use case will eventually be SSL certificate scenarios requiring NIST Suite B compliance, or improved performance.
  - Comodo [27] also has an ECC Root certificate. Comodo Root signing key pairs are ECC 384 bit. It does not currently offer certificates from that Root CA as yet.

4.3 What do you do now?

It is critical that all the various players in the information security field move towards widespread adoption of ECC within the next few years. This helps should RSA or Diffie-Hellman be broken, the transition to a secure cryptosystem can be done without widespread panic.

4.3.1 If you are an OS or language vendor

- Make ECC easy to use. NaCl's box() [33] and unbox() are good examples of easy to understand abstractions `crypto_box` is `curve25519xsalsa20poly1305`, a combination of `Curve25519`, `Salsa20`, and `Poly1305`.
- Update documentation to dissuade developers from using only RSA. Documentation should include how to use ECC primitives.

- Get aggressive about compatibility testing. RSA/DH has been the de-facto standard for decades and any change to alternative cryptosystems is likely to result in some incompatibility issues. These compatibility issues should be resolved before RSA/DH is broken.
- Eat your own dogfood. Ensure that ECC is used internally to minimize integration issues.

4.3.2 If you are a browser vendor

- TLS 1.2 needs to be a Priority 1 (P1) feature. Currently, Internet Explorer 11, Chrome 29 support and Safari 6.0.5 all support ECC. All browsers need to support ECC and eventually it should become the default cryptographic algorithm used by each browser.
- Push at CA/B Forum for standardized process for cross-signed certificates. A cross-certificate is a digital certificate issued by one Certificate Authority (CA) that is used to sign the public key for the root certificate of another Certificate Authority. Cross-certificates provide a means to create a chain of trust from a single, trusted, root CA to multiple other CAs.

4.3.3 If you are a software maker

- You need to support TLS 1.2 on all endpoints. TLS 1.2 has been defined from 2008 will be supported by all major applications in the future. It is essential to ensure that TLS 1.2 support is implemented in any new product.
- Build systems with pluggable cryptographic primitives. This should include versioning to allow for new cryptographic algorithms to be used easily in the future without re-implementing their entire system.
- Ensure that ECC is used for any new cryptosystems.
- Retrofit old mechanisms, via wrapping for example an ECC signed binary inside of a legacy RSA signature. This ensures that even if RSA is compromised, the full signature is still valid.

4.3.4 If you are a certificate authority.

- Make it easy to buy an ECC certificate. The process to purchase an ECC certificate should be simplified.
- Change documentation to include ECC certificate signing request (CSR) instructions. Currently all the documentation assume the purchase of an RSA certificate.
- Encourage the CA/Browser Forum to promulgate standards pushing the use of ECC certificates.

4.3.5 If you own ECC patents

- License the ECC patents openly to any implementation of Suite B and other curves, regardless of use.

4.3.6 If you are just a normal company.

- Use ECC certificates where possible. Certificate authorities should make it easier to use ECC certificates. Transition to ECC certificate as soon as possible.
- Continue to ask vendors about support for TLS 1.2 and ECC support. This should be high priority on the feature list.

- Turn on ECDHE perfect forward secrecy (PFS) from now.
- Survey you exposure so when the "cryptopocalypse" comes you have a clear idea of your exposure.

5 Conclusion

Current cryptosystems depend on discrete logarithm and factoring which has seen some major new developments in the past year. The academic community is continuing research in the area and there will be further improvements. We need to move to stronger cryptosystems that leverage more difficult mathematical problems such as ECC. Other options include lattice encryption (NTRU) [70].

Companies need to sponsor research into new and practical cryptographic algorithms since this is an essential part of our Internet economy. Improve cryptographic agility and make it possible to change the underlying cryptographic algorithms without large re-factoring or a new implementation of the entire code base. There is a significant amount of work that needs to be done, so please get started now.

References

[1] N. AlFardan, D. Bernstein, K. Paterson, B. Poettering, and J. Schuldt. Biases in the rc4 keystream. http://www.isg.rhul.ac.uk/tls/biases.pdf, August 2013. 1

[2] N. J. AlFardan and K. G. Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. http://www.isg.rhul.ac.uk/tls/TLStiming.pdf, February 2013. 1

[3] Apple. Ssl cipher suite definitions. http://www.opensource.apple.com/source/libsecurity_ssl/libsecurity_ssl-36800/lib/CipherSuite.h, January 2002. 17

[4] Apple. Apple cryptographic service provider functional specification. https://developer.apple.com/library/mac/documentation/Security/Reference/SecAppleCryptoSpec/Apple_Cryptographic_Service_Provider_Functional_Specification.pdf, March 2005. 17

[5] Apple. Cipher listing. http://www.opensource.apple.com/source/libsecurity_smime/libsecurity_smime-32850/lib/secoid.c, July 2006. 20

[6] Apple. Cms public key crypto. http://www.opensource.apple.com/source/libsecurity_smime/libsecurity_smime-36873/lib/cmspubkey.c, January 2008. 18, 20

[7] Apple. Cms signerinfo methods. http://www.opensource.apple.com/source/libsecurity_smime/libsecurity_smime-36873/lib/cmssiginfo.c, January 2008. 18, 20

[8] Apple. Mac os x 10.6 source. http://www.opensource.apple.com/release/mac-os-x-106/, June 2009. 17, 18

[9] Apple. Signing operation supervisor and controller. http://www.opensource.apple.com/source/libsecurity_codesigning/libsecurity_codesigning-55037.15/lib/signer.cpp, January 2010. 20

[10] S. Bai, C. Bouvier, A. Filbois, P. Gaudry, L. Imbert, A. Kruppa, F. Morain, E. Thomé, and P. Zimmermann. Complete implementation in c/c++ of the number field sieve (nfs) algorithm for factoring integers. http://cado-nfs.gforge.inria.fr/, November 2013. 4, 11

[11] S. Bai and R. P. Brent. On the efficiency of pollard's rho method for discrete logarithms. http://maths-people.anu.edu.au/ brent/pd/rpb231.pdf, June 2008. 13

[12] M. Baldi, M. Bianchi, and F. Chiaraluce. Security and complexity of the mceliece cryptosystem based on qc-ldpc codes. http://arxiv.org/pdf/1109.5827.pdf, January 2013. 17

[13] T. Ballance. Python module for performing elliptical curve cryptography. https://pypi.python.org/pypi/PyECC, June 2009. 19

[14] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. T. andMarion Videau, and P. Zimmermann. Discrete logarithm in $gf(2^{809})$ with ffs. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.7280&rep=rep1&type=pdf, April 2013. 6, 8

[15] R. Barbulescu, P. Gaudr, A. Joux, and E. Thomé. Quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. http://arxiv.org/pdf/1306.4244v2.pdf, June 2013. 6, 9

[16] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. http://hal.inria.fr/docs/00/83/54/46/PDF/quasi.pdf, June 2013. 9

[17] D. Bernstein and T. Lange. Safecurves: choosing safe curves for elliptic-curve cryptography. http://safecurves.cr.yp.to/index.html, Jan 2014. 14

[18] Blackberry. Package net.rim.device.api.crypto 3.6. http://www.blackberry.com/developers/docs/3.6api/net/rim/device/api/crypto/package-summary.html. 18

[19] Blackberry. Package net.rim.device.api.crypto 6.0. http://www.blackberry.com/developers/docs/6.0.0api/net/rim/device/api/crypto/package-summary.html. 19

[20] Blackberry. How to encrypt internal and external file systems on blackberry smartphones. http://www.blackberry.com/btsc/KB16088, November 2011. 20

[21] S. Blake-Wilson, V. Gupta, C. Hawk, and B. Moeller. Rfc 4492 - elliptic curve cryptography (ecc) cipher suites for transport layer security (tls). http://www.ietf.org/rfc/rfc4492.txt, May 2006. 18, 21

[22] C. Bouvier. The filtering step of discrete logarithm and integer factorization algorithms. http://hal.inria.fr/docs/00/80/88/40/PDF/article.pdf, April 2013. 9

[23] B. Carlstrom. Cipher suite as defined in tls 1.0 spec. https://android.googlesource.com/platform/libcore/+/4ae3fd787741bfe1b808f447dcb0785250024119/luni/src/main/java/org/apache/harmony/xnet/provider/jsse/CipherSuite.java, December 2011. 18

[24] L. R. Celemín. Complexity and cryptography. mceliece. cryptosystem. http://www.cs.uni-paderborn.de/fileadmin/Informatik/AG-Bloemer/lehre/2011/ss/seminar/Lara_Rojo_-_McEliece_Cryptosystem.pdf, July 2011. 16

[25] Certicom. How ecc can improve internet communications. http://www.certicom.com/index.php/component/content/article/45-cc-volume-1-no-4/520-how-ecc-can-improve-internet-communications, May 2005. 20

[26] Q. Cheng, D. Wan, and J. Zhaung. Traps to the bgjt-algorithm for discrete logarithms. http://eprint.iacr.org/2013/673.pdf, October 2013. 6, 10

[27] Comodo. Ecc certificate addendum to the comodo ev certification practice statement v.1.03. http://www.comodo.com/repository/ECC_addendum_to_the_EV_Certification_Practice_Statement.pdf, March 2008. 21

[28] R. E. Crandall and D. P. Mitchell. Small memory footprint fast elliptic encryption. http://www.google.com/patents/US7650507, January 2010. 17

[29] T. Dierks and C. Allen. The tls protocol version 1.0. http://tools.ietf.org/html/rfc2246, January 1999. 20

[30] T. Dierks and E. Rescorla. The tls protocol version 1.1. http://tools.ietf.org/html/rfc4346, April 2006. 20

[31] T. Dierks and E. Rescorla. The tls protocol version 1.2. http://tools.ietf.org/html/rfc5246, August 2008. 20, 21

[32] N. Elenkov. Upgraded bouncycastle libs. http://nelenkov.blogspot.com/2011/12/using-ecdh-on-android.html, December 2011. 18

[33] C. A. C. Engineering. Public-key authenticated encryption crypto_box. http://nacl.cr.yp.to/box.html, August 2010. 21

[34] Entrust. Entrust ecc certificates. http://www.entrust.net/ecc-certs/index.htm. 21

[35] Filezilla. Filezilla server 0.9.38. https://filezilla-project.org/versions.php?type=server, June 2011. 20

[36] Filezilla. Filezilla client 3.6.0-beta1. https://filezilla-project.org/versions.php?type=client, December 2012. 20

[37] Filezilla. Sftp using ssh2: Key based authentication. https://wiki.filezilla-project.org/Howto, May 2013. 20

[38] A. O. Freier, P. Karlton, and P. Kocher. The ssl protocol version 3.0. http://tools.ietf.org/html/draft-ietf-tls-ssl-version3-00, November 1996. 20

[39] M. Friedl, N. Provos, T. de Raadt, K. Steves, D. Miller, D. Tucker, J. McIntyre, T. Rice, and B. Lindstrom. Openssh 5.7. http://openbsd.das.ufsc.br/openssh/txt/release-5.7. 20

[40] P. Gaudry, L. Sanselme, and E. Thomé. Mpfq : Fast finite fields library. http://mpfq.gforge.inria.fr/doc/doc.html, June 2007. 11

[41] P. Gaudry, L. Sanselme, and E. Thomé. Mpfq : Fast finite fields library. http://eprint.iacr.org/2013/071.pdf, June 2007. 11

[42] Y. Gluck, N. Harris, and A. Prado. Breach: Reviving the crime attack. http://breachattack.com/resources/BREACH%20-%20SSL,%20gone%20in%2030%20seconds.pdf, August 2013. 1

[43] GnuTLS. Gnutls 3.2.12. http://www.gnutls.org/manual/gnutls.html#ciphersuite. 19

[44] F. Gologlu, R. Granger, G. McGuire, and J. Zumbragel. On the function field sieve and the impact of higher splitting probabilities. http://eprint.iacr.org/2013/074.pdf, February 2013. 5, 8

[45] Google. Signing your applications. http://developer.android.com/tools/publishing/app-signing.html#cert. 20

[46] Google. Upgraded bouncycastle libs. https://code.google.com/p/android/issues/detail?id=3280, July 2009. 18

[47] Google. Android 3.2.4 tls. https://android.googlesource.com/platform/libcore/+/android-3.2.4_r1, August 2011. 18

[48] Google. Chromium and the nss shared db. https://code.google.com/p/chromium/wiki/LinuxCertManagement, July 2013. 20

[49] R. Granger. New record for the computation of discrete logarithms in finite fields. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;fe9605d9.1304, April 2013. 6

[50] R. Granger, Kleinjung, and J. Zumbrägel. How to solve discrete logarithms in $f_{2^{4\cdot1223}}$ and $f_{2^{12\cdot367}}$. http://arxiv.org/pdf/1402.3668v2.pdf, February 2014. 6

[51] C. A. Group. Creation functions. http://magma.maths.usyd.edu.au/magma/handbook/text/1403, December 2013. 10

[52] C. A. Group. Magma computational algebraic system. http://magma.maths.usyd.edu.au/magma/, February 2014. 11

[53] S. Innovation. Ntru pkcs tutorial. https://www.securityinnovation.com/uploads/Crypto/NTRU%20PKCS%20Tutorial.pdf, January 2014. 16

[54] S. Innovation. Open source ntru public key cryptography algorithm and reference code. https://github.com/NTRUOpenSourceProject/ntru-crypto, March 2014. 16

[55] Jacob. Support for ecdsa keys in putty and puttygen. http://www.chiark.greenend.org.uk/ sgtatham/putty/wishlist/ecdsa.html, May 2013. 20

[56] Jasonp_sf. C library implementing a suite of algorithms to factor large integers. http://sourceforge.net/projects/msieve/, February 2014. 4

[57] A. Joux. Computation of discrete logarithms in $gf(2^{6168}) = gf((2^{257})^{24})$. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;49bb494e.1305, February 2013. 7

[58] A. Joux. Discrete logarithms in the finite field $gf(2^{6168}) = gf((2^{257})^{24})$. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;49bb494e.1305, April 2013. 6

[59] A. Joux. A new index calculus algorithm with complexity $l(\frac{1}{4})$ in small characteristic fields. https://eprint.iacr.org/2013/095.pdf, February 2013. 3, 5, 6, 9, 13

[60] A. Joux. New record for the computation of discrete logarithms in finite fields. https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1303&L=NMBRTHRY&D=0&P=13682, March 2013. 6

[61] A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields a comparison with the gaussian integer method. http://www.ams.org/journals/mcom/2003-72-242/S0025-5718-02-01482-5/S0025-5718-02-01482-5.pdf, November 2002. 10

[62] A. Joux and R. Lercier. The function field sieve in the medium prime case. http://www.iacr.org/cryptodb/archive/2006/EUROCRYPT/2435/2435.pdf, June 2006. 8

[63] W. Koch. Elliptic curves in gnupg status?(ecc support). http://www.mail-archive.com/gnupg-usersgnupg.org/msg20573.html, December 2012. 20

[64] T. Legion. Bouncy castle crypto apis for java. http://www.bouncycastle.org/java.html. 19

[65] T. Legion. Bouncy castle specifications. http://www.bouncycastle.org/specifications.html, December 2013. 18, 19

[66] B. Lynn. The pairing-based cryptography library. http://crypto.stanford.edu/pbc/, June 2013. 5

[67] N. Mathewson. Migrate server identity keys to ed25519. https://gitweb.torproject.org/torspec.git/blob/5380544e8e30408c30c057a3f4b8157815b0a059:/proposals/220-ecc-id-keys.txt, August 2013. 20

[68] N. Mavrogiannopoulos. Experimental support for elliptic curves. http://lists.gnu.org/archive/html/info-gnu/2011-05/msg00015.html, May 2011. 19

[69] R. J. McEliece. A public key cryptosystem based on algebraic coding theory. http://www.cs.colorado.edu/ jrblack/class/csci7000/f03/papers/mceliece.pdf, January 1978. 16

[70] D. Micciancio and O. Regev. Lattice-based cryptography. http://www.cims.nyu.edu/ẽgev/papers/pqc.pdf, July 2008. 23

[71] Microsoft. Cipher suites in schannel. http://msdn.microsoft.com/en-us/library/windows/desktop/aa374757(v=vs.85).aspx, June 2011. 18

[72] Microsoft. Cryptography next generation. http://technet.microsoft.com/en-us/library/cc730763(v=ws.10).aspx, August 2011. 18, 20

[73] Microsoft. Description of the support for suite b cryptographic algorithms that was added to ipsec in windows vista service pack 1, in windows server 2008, and in windows 7. http://support.microsoft.com/kb/949856, October 2012. 18

[74] Microsoft. Cng algorithm identifiers. http://msdn.microsoft.com/en-us/library/windows/desktop/aa375534(v=vs.85).aspx, May 2013. 18

[75] C. Monico. Gpl'd implementation of the general number field sieve (gnfs) for factoring integers. http://www.math.ttu.edu/ cmonico/software/ggnfs/, May 2005. 4

[76] Mozilla. Encryption technologies available in nss 3.11. http://www-archive.mozilla.org/projects/security/pki/nss/nss-3.11/nss-3.11-algorithms.html, February 2007. 18, 19

[77] Mozilla. Nss 3.11.10 release notes. https://developer.mozilla.org/en-US/docs/NSS/NSS_3.11.10_release_notes.html, December 2008. 18, 19

[78] Mozilla. Security in firefox 2. https://developer.mozilla.org/en-US/docs/Security_in_Firefox_2, November 2009. 20

[79] NIST. Fips pub 186-3. http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf, June 2009. 16

[80] NIST. Fips pub 186-4. http://nvlpubs.nist.gov/nistpubs/FIPS/NISTIPS1864.pdf, July 2013. 16

[81] OpenSSL. Openssl ecc ciphers. http://www.openssl.org/docs/apps/ciphers.html#Elliptic_curve_cipher_suites_. 19, 20

[82] OpenSSL. Changes between 0.9.8n and 1.0.0. http://www.openssl.org/news/changelog.html, March 2010. 19, 20

[83] Oracle. Ciphersuites supported by sunjsse. http://docs.oracle.com/javase/6/docs/technotes/guides/security/SunProviders.html#SupportedCipherSuites. 19

[84] Oracle. Java™ 6 security enhancements. http://docs.oracle.com/javase/6/docs/technotes/guides/security/enhancements.html. 19

[85] Oracle. Java cryptography architecture for java platform standard edition 7. http://docs.oracle.com/-javase/7/docs/technotes/guides/security/SunProviders.html. 19

[86] K. G. Paterson. On the security of rc4 in tls and wpa. http://www.isg.rhul.ac.uk/tls/, August 2013. 1

[87] B. Poettering. Seccure elliptic curve crypto utility for reliable encryption. http://point-at-infinity.org/seccure/, August 2006. 19

[88] T. Ritter. Details on crime attack. https://www.isecpartners.com/blog/2012/september/details-on-the-crime-attack.aspx, September 2011. 1

[89] Ruby-Doc. Openssl::pkey::ec. http://www.ruby-doc.org/stdlib-1.8.7/libdoc/openssl/rdoc/OpenSSL/P-Key/EC.html. 19

[90] P. Sarkar and S. Fitzgerald. Attacks on ssl. https://www.isecpartners.com/media/106031/ssl_attacks_survey.pdf, August 2013. 1

[91] P. Sepehrdad, S. Vaudenay, and M. Vuagnoux. Discovery and exploitation of new biases in rc4. http://infoscience.epfl.ch/record/152526/files/RC4_1.pdf, December 2010. 1

[92] B. Sturmfels. What is a grobner basis? http://math.berkeley.edu/ bernd/what-is.pdf, November 2005. 9

[93] Symantec. Root 5 verisign class 3 public primary ca - g4. http://www.symantec.com/page.jsp?id=roots, November 2007. 21

[94] Symantec. Faq: Ecc and dsa certificates website security solutions. http://www.symantec.com/content/en/us/enterprise/fact_sheets/b-ecc_dsa_faq_DS.en-us.pdf, July 2013. 21

[95] Symantec. Symantec ssl certificates with the dsa algorithm. https://www.symantec.com/content/en/us/enterprise/fact_sheets/b-symantec_ssl_certificates_with_the_dsa_algorithm_DS.en-us.pdf, February 2013. 21

[96] Symantec. Symantec ssl certification with the ecc algorithm. http://www.symantec.com/en/au/content/en/us/about/presskits/b-symantec_ssl_certification_with_ecc_algorithm.en-us.pdf, March 2013. 21

[97] Thaidn. Overview of beast. http://vnhacker.blogspot.com/2011/09/beast.html, September 2011. 1

[98] Thawte. Thawte primary root ca – g2 (ecc). https://www.thawte.com/roots/index.html, November 2007. 21

[99] R. Tyley. Spongy castle -repackage of bouncy castle for android. http://rtyley.github.io/spongycastle/, January 2014. 18

[100] WolfSSL. Cyassl user manual. http://www.yassl.com/documentation/CyaSSL-Manual.pdf, September 2013. 18

[101] WolfSSL. Cyassl release 2.9.0. http://www.yassl.com/yaSSL/Docs-cyassl-changelog.html, February 2014. 18

[102] Zimmerma. Elliptic curve method records. http://www.loria.fr/ zimmerma/records/ecmnet.html, February 2014. 14