History of Cryptographic Key Sizes

Nigel P Smart, Emmanuel Thomé

History of Cryptographic Key Sizes*

Nigel P. Smart[1] and Emmanuel Thomé[2]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
[2] Université de Lorraine, CNRS, INRIA, Nancy, France

1 Introduction

Picking a cryptographic algorithm to use is often the simplest part in designing a system which uses cryptography. The difficult part comes in how one combines different algorithms together to meet some security objective and how the related parameters to the underlying cryptographic algorithms are selected. Probably the most important parameter is the key size, and one needs to select this for each algorithm separately, so that the combined system meets the desired security level.

Different algorithms have different metrics for measuring key strength; thus breaking a 128-bit key length for AES is not the same difficulty as breaking a 128-bit key length for RSA (the former is infeasible even with a quantum computer, whilst the latter is an undergraduate student project). To ease this issue Lenstra and Verheul published in 2000 an influential paper entitled *Selecting Cryptographic Key Sizes* [59], with a longer version in [61]. This was the first systematic attempt to measure the relative difficulty of different problems and provide some form of scientific guidance.

Determining key sizes is not an exact science, but it is an combination of four related factors:

1. A stable understanding of the best algorithm(s) to solve the computational problem.
2. Good asymptotic estimates of running time and storage requirements of such algorithm(s).
3. Some data points from large-scale computations, eg of cryptographic challenges.
4. The 'tightness' of any security proof which the cryptographic scheme relies upon.

In this chapter we will ignore the later point as in practice often security proofs are used to validate designs rather than used to set key sizes. Thus 'tightness' in security proofs, i.e. the gap between the difficulty of breaking the scheme versus the difficulty of breaking the underlying hard mathematical problem is often ignored in practice.

At the time of publication the default key size for many symmetric algorithms was 80 bits of security, although no standardized algorithm had this level of security or higher (AES did not become a standard until 2002), whereas RSA keys were suggested to be around 1024 bits in length, with ECC keys of 160 bits in length. Although in practice 512-bit RSA keys were widespread, despite a 512-bit RSA modulus being factored in the previous year [20].

* This material will be published in revised form in Computational Cryptography edited by Joppe W. Bos and Martijn Stam and published by Cambridge University Press. See www.cambridge.org/9781108795937.

Following Lenstra's paper a number of official and community recommendations followed. The most important of the official recommendations being those of NIST [72], ANSSI [4], and BSI [19]. The most influential of the community led recommendations being the series of reports published by the ECRYPT projects in various years spanning from 2004 to 2018, with ENISA sponsoring the reports in 2013 and 2014[3]. The ECRYPT reports not only covered key size, but also other aspects of how a cryptographic primitive should be used, such as recommendations for modes of operation etc.

To derive recommendations for cryptographic key size one needs to take into account all the possible algorithms, and ways of implementing them, that can be used to recover keys. In doing so, one needs to also take into account that ones attacker might not be just some 'script kiddy' working in their bedroom, but could be a well funded nation state actor with access to large software and hardware resources.

2 Attacking Symmetric Algorithms with Software and Hardware

Attacking symmetric algorithms (block ciphers and hash functions) using hardware (and in some cases software) has a long history. Some of this work has been motivated by the desire to show that existing algorithms are insecure and need replacing (such as in the case of DES and MD5), some has been motivated to show just how inefficient such approaches will be (such as in the case of AES), whilst some has been to enable new applications (such as in the case of bitcoin mining of SHA-256).

2.1 Attacking DES

The most famous, and impactful, of the research in this area has been the long history of breaking the DES algorithm. Recall that DES is one of the oldest cryptographic primitives to be standardized for public use, and when proposed the key length was set to be 56 bits. Even at the time Martin Hellman and Whit Diffie estimated that DES would be insecure against a determined attacker, by estimating that one could build a parallel computer to evaluate one million DES operations per second at a cost of $20 million [25]. Accounting for inflation this would amount to $86 million in todays money.

This idea was left, in the open community at least, as an intellectual curiosity for twenty years until in 1998 the Electronic Frontier Foundation (EFF) built the *Deep Crack* machine [27]. Deep Crack could find a single DES key in 22 hours, with a cost of building the machine of $\$200,000$. The reduction in cost, and the time needed to perform the attack, coming from the relentless march of improvements in computer hardware.

With the advent of FPGAs the design and build costs for such machines came within the range of University researchers. Fast forward another decade to 2006, and the COPACOBANA FPGA based cracker [51] was built, which cost about $\$10,000$ and could break a single DES key in 6.4 days. At roughly the same time it was estimated [79] that,

[3] The various ECRYPT reports can be found at `https://www.ecrypt.eu.org/`.

after a pre-computation of a week on a $\$12,000$ device, one could break any DES keys in half an hour using just a $\$12$ FPGA.

Whilst with special purpose hardware one has a high upfront cost and then the cost of attacking each key is minimal, and only really costs in terms of time, pure software attacks on DES keys are possible using standard computers. In [50] Lenstra and his co-authors estimated that the dollar cost of finding a key in as short a time as possible on the Amazon EC2 cloud service was $\$14,700$ in 2012. This reduced to $\$4,500$ by 2018, purely due to the reduction in prices by Amazon for its cloud service. This hardness measure, in terms of dollar cost per key, is a more telling cost for weak algorithms such as DES. For a secure algorithm one expects the dollar cost per key to be so high, that it dwarfs the size of the world economy.

2.2 Attacking AES

The attacks mentioned above on the venerable DES cipher led NIST to look for a replacement. In 1997 a competition was announced to find a replacement, and in 2000 the winner, a cipher called Rijndael from Belgium, was selected. The Rijndael algorithm, was then selected as the Advanced Encryption Standard (AES). AES comes in three variants, each with block size 128-bits, the variants are a cipher with key size 128-bits, one with 192-bits and one with 256-bits. Whilst in theory the same approaches that were applied to DES can also be applied to AES, the effects are noticeably different. For example [50] estimates that in 2018 it would cost around $\$10^{19}$ million to break AES-128 using Amazon's EC2 cloud service.

2.3 Attacking Hash Functions

Not all breaking of mainstream symmetric algorithms has been done via raw computer power, some have been broken by advances in cryptanalysis, combined with raw computing power. A class example of this is the MD5 hash algorithm. This was proposed in 1991 by Ron Rivest and it saw widespread deployment in many applications. Soon after deployment (in 1996) Dobbertin found a collision in the compression function used within the algorithm, and cryptographers began recommending that MD5 not be used. However, this did not stop the widespread deployment of MD5 in the subsequent decades.

However in 2004 a collision for the full MD5 was announced by a group of cryptographers from China led by Wang, effectively breaking the hash function at the CRYPTO Rump Session (see [39] for a contempory discussion and [99] for the paper). However, this was a random collision and thus was not a *direct* threat to computer systems. Soon after Lenstra and co-authors constructed two X.509 certificates with the same hash value [96].

Rapid progress followed, using off-the-shelf computers one can now find collisions in specific situations in a matter of hours. That MD5 was broken led to other attacks on computer systems; for example the *Flame* malware from 2012 attacked Microsoft operating systems by exploiting problems with MD5.

When flaws in MD5 were discovered the recommendation was to switch to the SHA-1 algorithm. This was a very similar algorithm, but cryptographers thought it

would provide longer term security compared to MD5. However, whilst stronger it was not that much stronger. A year after finding a collision in MD5, Wang's team gave another announcement at the CRYPTO Rump Session, this time giving a theoretical attack against SHA-1. Further improvements to this theoretical attack were announced over the coming decade. Then in 2015 using GPU time on Amazon's EC2 Cloud Service Stevens et al announced the first collision in SHA-1 requiring $2000 cost per new collision on EC2[4]. In 2020 Leurent and Peyrin announced the first practical chosen prefix collision on SHA-1. This method requires an expenditure of less than $\$100,000$ per collision, and builds upon the work in [66].

The flaws in SHA-1 announced in 2005 led NIST to release a modified, more secure version, called SHA-2 (which comes in various output size, the most popular being SHA-256). In addition NIST announced a competition for a new hash algorithm, to be called SHA-3. The competition was started in 2006, and in 2012 was won by a Belgium team with their Keccak algorithm. Lenstra et al's estimates of the cost of breaking SHA-256 via running EC2 instances but the cost at $\$10^{58}$ million dollars. Currently, no estimates exist for the cost of finding a collision in SHA-3.

SHA-256 is used in the Bitcoin system as the hard problem on which miners operate to authenticate a block. Each miner needs to evaluate two evalations of SHA-256 and then see if the resulting output has a special form before they can claim their reward. The profitability of this mining operation led to first FPGA and then ASIC mining hardware. The total Bitcoin system (as of 2020) is hashing at a rate of roughly 2^{66} hashes per second. To find a collision in SHA-256 is expected to take 2^{128} such hashes, thus even with the entire Bitcoin network we can only expect to find a hash in 2^{36} years!

3 Software Attacks on Factoring and Discrete Logarithms

The integer factorization problem and the discrete logarithm problem in finite fields were the two hard problems underpinning the security of most of public-key cryptography in its first decades of existence. While a variety of primitives appeared since that rely on the assumed hardness of other mathematical problems, the prevalence of the primitives that rely on integer factorization and discrete logarithm, such as RSA and the Diffie–Hellman key exchange using multiplicative groups of finite fields, is clear.

One may encounter at times the argument that the hardness of factoring is a well-studied problem because it has been actively studied for a long time (decades, centuries, or millenia for the most bodacious statements of this kind). In fact, the first important algorithmic improvements over naive methods date back to the early 1970s, with work by Shanks [92], Pollard [75], and Morrison and Brillhart [71]. The motivations for these early endeavours towards smarter and smarter factoring methods were, for example, the factorization of discriminants of number fields, or the factorization of numbers of the form $b^n \pm 1$, which are potentially useful in a variety of settings ranging from combinatorics to group theory.

The advent of the RSA cryptosystem in the late 1970s, and its dependence on the hardness of factoring, led to a considerable surge in academic interest (and certainly

[4] `https://sites.google.com/site/itstheshappening/`

non-academic interest as well). A good measure of the perceived (in)feasibility limit by 1978 is given in the RSA paper itself [80]: factoring a 100-decimal digit integer was estimated to require about a century of computation time. We will see shortly that this key size withstood factoring attempts for hardly a decade. This provides two immediate lessons. First, in hindsight, four decades of algorithmic progress have evidently gone a long way. Second, estimating hardness is a tricky matter.

The Morrison-Brillhart CFRAC algorithm can be regarded as the ancestor of a series of algorithms that includes in particular the quadratic sieve and the number field sieve, both of which have numerous variants. For four decades, it has been a cryptographically important task to regularly provide good estimates of how hard factoring actually is in practice, using either commodity hardware or various kinds of more special-purpose equipment, ranging from academic supercomputers to specially designed hardware accelerators. Such estimates include the development of algorithmic ideas that are very effective in improving the computation time, often based on the constraints of the chosen computation platform.

3.1 Factoring by Email

The early 1980s saw several improvements of the state of the art of integer factorization. In their 1988 paper *Factoring by electronic mail* [57], Lentra and Manasse adressed the variety of existing factoring records at the time, and provided a pragmatic answer to the question: *how big are the integers we can factor within one month of elapsed time, if we only want to use computing time that we can get for free?* This article was influential in the definition of the line of work of academic records for the following decades. It firmly installs, however, a distinction between academic records, done with this strategy, and what can be done by governements or possibly criminal organizations, who are ready to spend money to meet their goals.

Lenstra and Manasse used the MPQS algorithm, which is the multiple polynomial variant of the quadratic sieve. This algorithm is described in [94]. In MPQS (and also in CFRAC, which MPQS superseded), one can separate the computation in two main steps. First, the *relation collection* step searches for multiplicative relations involving integers modulo N. In a later *matrix* step, these relations are combined to form an congruence of squares. Finding this combination is equivalent to finding an element in the nullspace of a sparse matrix over the binary field GF(2). As a matter of fact, these two steps have also been the two main steps of all large factoring and discrete logarithm computations since.

The relation collection step in MPQS can be distributed *massively*. By 1988, standard hardware that was capable to contribute significantly to the relation collection towards the factorization of numbers of, say, 100 decimal digits, was relatively common in universities and research labs. Furthermore, the amount of output that was produced by each machine participating in the computation was relatively low. This made it possible to gather the results using any commodity means, and electronic mail was a perfect fit for that goal, usefully leveraging existing infrastructure.

The largest computation done with MPQS, following this approach of gathering contributions from hundreds of enthusiasts, was the factorization of RSA-129. This

computation solves a challenge that was concocted by the RSA authors and presented in Martin Gardner's column in the August 1977 issue of *Scientifc American*. Atkins, Graff, Lenstra and Leyland factored the public modulus in 1994 and were able to unveil the secret RSA-encrypted message that was "The Magic Words are Squeamish Ossifrage" [6]. A striking fact about this computation is the number of contributors (more than 600) and the variety of machines used, ranging from 16MB 80386sx PCs to Cray C90s (the code was even ported to ... fax machines).

The RSA-129 challenge was the most famous of a number of challenge factorization problems. The RSA company itself published a list of factorization challenges at various key lengths to encourage people to look into the security of cryptography based on factoring. This became known as the RSA Challenge list.

3.2 The development of the number field sieve

The RSA-129 paper concluded, quite naturally, with extrapolations on the hardness of reaching "most wanted" targets such as the factorization of 512-bit RSA moduli. Such keys were in widespread use at the time. With MPQS, it was estimated this this goal was almost feasible using massive power, but also qualified as "the last gasp of an elderly workhorse". The more recent Number Field Sieve, with better asymptotical behaviour, appeared probably more fit for that goal.

The Number Field Sieve originated in an astute way, designed by Pollard in 1988, to factor the seventh Fermat number $2^{128}+1$. Pollard used *cubic integers*. More mathematics were entering the scene. Pollard's method was understood by many as a curiosity. Its reliance an almost coincidental match between cubic integers and the special form of $2^{128}+1$ raised doubts as to the possibility of extending the approach to other numbers.

Generalizations did come. By the end of 1989, Lenstra, Lenstra, Manasse and Pollard had extended Pollard's original approach to further numbers, and were able in 1990 to factor the 9-th Fermat number $2^{512}+1$, which was a bit of a holy grail to factoring enthusiasts. Several challenges were still to overcome in order to factor *general* numbers. Not the least of which was the fact that the matrix step, it seemed, had to be solved with coefficient ring $\mathbb{Z}$ —it later appeared that solving over $\mathbb{Z}/2\mathbb{Z}$ was enough. Several people contributed to loosening the constraints. The 1993 book *The Development of the Number Field Sieve* [56] includes further work by Bernstein, Buhler, Couveignes, Pomerance, leading to a *General* Number Field Sieve (GNFS) algorithm which was convincingly able to factor large numbers with good performance.

While early successes with GNFS were obtained while the algorithm was still being developed, the RSA-129 MPQS mark was improved in 1996, with the factorization of RSA-130 by GNFS. Because the Number Field Sieve had the peculiarity of being able to take advantage of the *special* form of integers to be factored in some cases, a good benchmark could no longer be the good old way of picking challenge numbers from the Cunningham tables, or otherwise defined numbers that had "some mathematical interest". This good benchmark was defined in 1991 by the RSA challenge list, formed of good RSA moduli, with two secretly chosen prime factors of similar size.

Following RSA-130, several records followed, most often picked from the RSA challenge list. All were factored with GNFS. This provided a nice view of the progress

of computational power on the one hand, and also progress on implementation and algorithmic refinements. The landmark factorization of a 512-bit RSA modulus was reached in 1999, the culmination of a significant effort from many contributors, coordinated by CWI in Amsterdam. As of today the list of factored RSA-challenge numbers is given in Table 1[5].

Challenge Name	Digits	Bits	Date Factored	Factored by
RSA-100	100	330	Apr 1, 1991	A. K. Lenstra
RSA-110	110	364	Apr 14, 1992	A. K. Lenstra and M.S. Manasse
RSA-120	120	397	Jul 9, 1993	T. Denny et al.
RSA-130	130	430	Apr 10, 1996	A. K. Lenstra et al.
RSA-140	140	463	Feb 2, 1999	H. te Riele et al.
RSA-150	150	496	Apr 16, 2004	K. Aoki et al.
RSA-155	155	512	Aug 22, 1999	H. te Riele et al.
RSA-160	160	530	Apr 1, 2003	J. Franke et al.
RSA-170	170	563	Dec 29, 2009	D. Bonenberger and M. Krone
RSA-576	174	576	Dec 3, 2003	J. Franke et al.
RSA-180	180	596	May 8, 2010	S. A. Danilov and I. A. Popovyan
RSA-190	190	629	Nov 8, 2010	A. Timofeev and I. A. Popovyan
RSA-640	193	640	Nov 2, 2005	J. Franke et al.
RSA-200	200	663	May 9, 2005	J. Franke et al.
RSA-210	210	696	Sep 26, 2013	R. Propper
RSA-704	212	704	Jul 2, 2012	S. Bai, E. Thomé and P. Zimmermann
RSA-220	220	729	May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
RSA-230	230	762	Aug 15, 2018	S. S. Gross
RSA-768	232	768	Dec 12, 2009	T. Kleinjung et al.
RSA-240	240	795	Nov 24, 2019	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann
RSA-250	250	829	Feb 28, 2020	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P. Zimmermann

Table 1. The solved RSA Challenges

3.3 Discrete Logarithms in Finite Fields

The development of algorithms to solve discrete logarithms in large prime characteristic finite fields $\mathbb{F}_p$ follows roughly the same development of the case of factoring[6]. The

[5] Source `https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`.
[6] We leave the case of non-prime fields to later

basic algorithm is to perform a sieving stage, followed by a large matrix step. The key difference is in the matrix step. For factoring the matrix equation one is trying to solve is modulo 2, whereas for discrete logarithms it is modulo q, where q is the large prime factor of $p - 1$.

The methods to collect relations are also almost identical. Indeed in 2000 Schirokauer [85] showed how to use the Number Field Sieve to solve discrete logarithms in the field $\mathbb{F}_p$, and that is now the method of choice. Just as with factoring there is the basic Number Field Sieve, and a more efficient variant called the Special Number Field Sieve which can utilize special properties of the representation of the number p if they exist.

At the time of writing (Autumn 2019) a record for discrete logarithms has just been announced for a general 240-decimal digit, 795-bit prime field [18]. The record for primes of special form are slightly larger, with a discrete logarithm computation in 2016 for a a 308-decimal digit, or 1024-bit, prime [29]. Note that discrete logarithms are slightly harder than factoring, judging by the publication time of records for general 768-bit factoring or prime discrete logarithms (2009 and 2016, respectively) or the of records for special 1024-bit factoring or prime discrete logarithms (2007 [5] and 2016). However, the largest known records were done at the same time, and indicate that the difference gap between the two problems is not as large as usually considered [18].

4 Hardware for Factoring

At the heart of modern factoring algorithms, and to some extent almost all algorithms for solving discrete logarithms, is the so-called sieving step. This sieving step is essentially trying to solve a more complex version of the classical problem of finding simulataneous solutions to modular diophantine equations

$$x = a_i \pmod{m_i} \quad \text{for} \quad i = 1, \ldots, t.$$

Classically 'machines' to solve this problem date back to as far back as 1896, when Lawrence presented a mechanical design for such a machine. In 1912 the first actual design was implemented by four French mathematicians Gérardi, Kraitchik, and Pierre and Eugène Carissan. The most famous of the mechanical devices where those built by the Lehmers in the 1930's some of which can now be found in the Computer History Museum in Silicon Valley. For a discussion of the history of such early machines see [89] and [53, 55, 54].

The advent of modern factoring algorithms led to a renewed interest in such 'sieving machines'. In the early 1980s a special purpose computer was designed to implement the CFRAC factoring algorithm by Smith and Wagstaff [78], called the 'Georgia Cracker'. This was accompanied by another design by Rudd, Buell and Chiarull who designed a machine in 1984 which could factor numbers ten times faster than existing general purpose computers [81].

In the early 2000's there was renewed interest in special purpose hardware for the sieving step. This was initiated by a EuroCrypt 1999 Rump Session talk in which Adi Shamir introduced his idea for the TWINKLE machine. In classical sieving we sieve a region by assigning each region some space in a computer memory, then we use time

to go through all the primes in the factorbase so as to encode the contribution of each prime to each location in the space. TWINKLE switches this around, and uses light based computation (much like one of Lehmer's devices [54]). The idea now is that space is used up by lights which flash according to a timing signal dependent on the prime, thus each light corresponds to a prime; whilst we use time to represent the sieving region. At a time signal corresponding to when more lights flash we have an element in the sieving region which needs to be investigated further. The initial TWINKLE design was presented in [90] and then analysed in [58] by Lenstra and Shamir. It was deemed too costly to build in order to factor numbers of interest.

TWINKLE was soon followed up in 2003 by another idea of Shamir and Tromer called TWIRL [91]. This dispensed with the optical computing methodology, but kept the time/space reversal, of TWINKLE. The design was based on novel routing networks to represent the sieving operation. The design was never built, but was considered feasible for factoring 1024-bit integers in a paper by Lenstra and co-authors [64]; one estimate gives a cost of $1.1 million dollars to factor a 1024-bit integer in one year. A related design, based on idea of Bernstein (see below), was presented by Geiselmann and Steinwandt in [32, 33]. It too was considered practical for 1024-bit moduli, but slightly more expensive to build than TWIRL.

Related to TWIRL is a design from 2005 by Franke and others called SHARK [28]. This used a different form of routing network, but could utilize more off-the-shelf components. The authors estimated that one could factor 1024-bit numbers in a year at a cost of $200 million dollars. This is higher than TWIRL, but was considered easier to build with current technology; thus the design risk was less.

It is not only the sieving step which has been the subject of hardware design proposals. Modern factoring also involves a large matrix step. In 2001 Bernstein presented a proposal for the matrix step, which in fact influenced the methods of Geiselmann and Steinwandt mentioned above. Bernstein's idea was to reduce the amount of idle storage in the standard matrix algorithm by using a form of mesh sorting. The design was analysed, again by Lenstra and co-authors, in [63], where they concluded that the method was not cost-effective in factoring 1024-bit integers.

Special purpose, but commodity hardware, has also been used to factor integers. The Graphical Processing Units (GPUs) in modern gaming consoles provide a cheap form of computing. Often the hardware is sold at a discount as the manufacturer aims to get their investment back in sales of games. Combined with the amazing parallel processing capability of modern GPUs, games consoles form an attractive way of obtaining supercomputer performance at a fraction of the price. In 2011 it was shown how to utilize such hardware to factor integers, using a variant of the Elliptic Curve Factoring Method (ECM) [16, 15].

5 Attacking Cryptosystems Based on Elliptic Curves

For an elliptic curve $E(\mathbb{F}_q)$ over the finite field $\mathbb{F}_q$ to be used in cryptography, one *usually* selects a curve which has a subgroup of prime order p, with $p \approx q$. By Hasse's Theorem we cannot have $p \gg q$, and for efficiency we want to keep q as small as possible, but the complexity of the best algorithm for attacking an elliptic curve has complexity $O(\sqrt{p})$.

Thus to obtain 80-bit security, one selects a 160-bit curve, to obtain 128-bit security one selects a 256-bit curve and so on.

Inspired by the RSA Challenges, Certicom (a Canadian company which specialised in ECC based cryptography) in 1997 launched the Certicom ECC Challenge. This comprised a set of toy challenges (for 79-bit, 89-bit and 97-bit curves), Level I challenges (for 109-bit and 131-bit), plus Level II challenges (for 163-bit, 191-bit, 239-bit and 359-bit). The challenges were also sub-divided into those over finite fields of characteristic two and those over large prime fields.

By the end of 1999 all of the toy challenges had been solved in both categories of fields. These attacks were performed via the method of van Oorschot and Wiener [97], which are themselves extensions of the ideas of Pollard [76, 77]. Clients were distributed across the internet on peoples idle machines, with communication back to the central control point being performed either by e-mail or via socket connections. The 109-bit challenges were then all solved by 2004. Some teams have attempted to solve the 131-bit challenge, but so far no reports of success are available.

The application of gaming consoles mentioned earlier has also been applied to attacking elliptic curve based systems. For example [17] discusses breaking a 112-bit elliptic curve instance using gaming consoles.

In [50] Lenstra and his co-authors consider the cost of breaking ECC keys using Amazon EC2 instances alone. Their estimates give that determining a 256-bit ECC key will require one to spend $\$10^{20}$ million on EC2. With the estimated costs for the Level II Certicom challenges being $\$10^{6}, \$10^{11}, \$10^{18}, \$10^{36}$ million respectively. Current key sizes are recommended to be 256-bits for ECC, to correspond to the 128-bit security of AES-128, or 512-bits, to correspond to the 256-bit security of AES-256. Thus, baring the advent of a quantum computer, current recommended key sizes for ECC can be considered secure.

However, one needs to avoid 'special curves' which have attacks which perform better than $O(\sqrt{p})$. One such class of curves are those defined over finite fields with $q = \ell^n$. An example of such attacks is the Weil descent attack when $q = 2^n$ and n is composite, as explained in [31], the so-called GHS-attack. This attack exploits special properties of elliptic curves over such fields for composite n. The attack works much like factoring methods; in the first stage one collects relations and then in the second one uses linear algebra to find specific discrete logarithms. Thus whilst this attack has no effect on key *sizes* it does have an effect on the choice of specific key *types*.

In recent years the method of Weil descent has been extended to elliptic curves over general fields of the form $\mathbb{F}_{\ell^n}$. This study was initiated by Semaev [87], who introduced the Semaev summation polynomials into the Weil restriction methodology. This idea was then developed by Gaudry [30] and Diem [24]. A nice succinct statement of these results is that if ℓ is a prime power and n is such that

$$a \cdot \sqrt{\log(\ell)} \leq n \leq b \cdot \sqrt{\log(\ell)}$$

for some fixed positive real numbers $a < b$ then asymptotically one can solve the discrete logarithm problem on an elliptic curve $E(\mathbb{F}_{\ell^n})$ in time

$$e^{O(\log(\ell^n)^{2/3})}.$$

Another class of weak curves are the so-called anomalous curves, which are the curves for which $p = q$. These were shown to be very weak in three papers which appeared at roughly the same time [84, 88, 95].

5.1 Pairing Based Cryptography

The most interesting class of curves for which there is a possible weakness are those which possess a pairing to a finite field. In particular there are classes of elliptic curves $E(\mathbb{F}_q)$ for which there are numbers $k \leq n$ for which there is an efficiently computable map

$$\hat{t} : E(\mathbb{F}_q) \times E(\mathbb{F}_{q^k}) \longmapsto \mathbb{F}_{q^n}.$$

In fact such pairs (k, n) exist for all elliptic curves, but curves for which (k, n) are small are very rare indeed.

In the early days of ECC it was common to select curves, for efficiency reasons, which had $k = 1$ and $n \leq 6$. However, this means one can map the discrete logarithm problem in $E(\mathbb{F}_q)$ into the discrete logarithm problem in $\mathbb{F}_{q^n}$ [69]. Suppose $E(\mathbb{F}_q)$ is chosen to have 80-bit symmetric security, i.e. 160-bit ECC security, then this means the finite field $\mathbb{F}_{q^n}$ is of size less than 960. In the early 1990's this killed off such curves in the case when $n \neq 6$.

When $n = 6$ one seemed to have a nice balance in the security, as in those days 1000 bit finite field discrete logarithms were considered as secure as 160-bit elliptic curve discrete logarithms. This 'magic' property of the number six, i.e. $6 \cdot 160 \approx 1000$, was utilized by Lenstra and Verheul to construct the XTR cryptosystem in [60] in 2000.

However, the move to 128-bit symmetric security, and the equivalent 256-bit elliptic curve security and the supposed equivalence of 3072 bit finite field discrete logarithm security, led to the number six loosing its magic properties and being replaced by the magic number 12, since $12 \cdot 256 = 3072$. We shall return to the number 12 below. Thus the existence of such pairings seemed to imply that such curves should be avoided at all costs in cryptographic applications.

However, the interest in pairings on elliptic curves was ignited when Joux [42] found a constructive application of such mappings. This was quickly followed by other applications, such as Identity Based Encryption [14, 82]. Soon a whole zoo of applications of pairings was created. But such a zoo needed secure curves to work on.

The main security requirement of pairing based systems is that the curve $E(\mathbb{F}_q)$ needs a large subgroup of prime order p with $\log_2 p$ being twice the desired symmetric security level, so $p = 2^{256}$ say. We also would like q to be small to enable good efficiency. But we need a pairing to exist for which n is such that the finite field $\mathbb{F}_{q^n}$ gives hard discrete logarithms.

There turned out to be three popular choices.

1. Pick $q^n = \ell^m$, for $\ell = 2$ or $\ell = 3$. The advantage here is one obtains very good implementation efficiency. The problem is the discrete logarithm in low characteristic fields turned out to be not as hard as originally thought (see below).

2. Pick $p \approx q$ and select n to make the discrete logarithm hard. Thus we return to picking n to be (say) the magic number 12. Such curves are usually called "pairing-friendly". A typical choice here was the so-called Barreto–Naehrig curves [11], called BN curves. The problem here is that such curves are subject to a potential 'medium prime' attack (again see below).
3. Pick $n = 2$, and make q huge, e.g. $q \approx 2^{1500}$ but keep p of size 2^{256} (say). Here one loses efficiency, but we are pretty certain of security.

We now discuss these last three cases in turn:

When $q^n = \ell^m$ for small ℓ: Algorithms to solve discrete logarithms in small characteristic field have had a very interesting history. In the case of $\ell = 2$, the first algorithms can be dated back to 1982 [40]. But the real advance came in 1984 when Coppersmith [22], building upon earlier ideas of Blake et al [13], gave an algorithm with heuristic complexity $L_{q^n}(1/3, c + o(1))$. This was the first algorithm to have such a complexity, and this was a decade before the NFS algorithm was known for the factoring problem with a similar complexity.

The idea behind Coppersmith's algorithm was shown, after the invention of the Number Field Sieve, to be closely related to another algorithm called the Function Field Sieve which was invented by Adleman in 1994 [2]. This algorithm was improved slightly by Joux and Lercier in 2002 [44], but there seemed to be little further improvements that could be made.

Then in 2013 a series of papers and email announcements started coming out. In looking at fields of the form $q = 2^m$, an announcement by Joux in Febrary of 2013 gave a $L_{q^n}(1/4, c + o(1))$ complexity algorithm and a new record of $m = 1778$, which was way above existing records [43]. Then in the same month Granger et al solved discrete logarithms in a field with $m = 1971$, using a variant of the Function Field Sieve [35]. In March Joux responded with another record of $m = 4080$, with Granger et al responding with $m = 6120$ in April. The back-and-forth continued, with Joux posting a record of $m = 6168$ in May. In June 2013 Barbulescu et al presented a quasi-polynomial time algorithm for fields of the form $q^n = 2^m$ [8]. Finally in January 2014 Granger et al posted the solution to a problem with $m = 9234$. Thus in one year the entire area of pairing based cryptography over finite fields of characteristic two had been destroyed. The current record is a huge value of $m = 30750$ set by Granger et al in 2019 [37].

The case of characteristic three finite fields, which held a lot of promise for pairing based cryptography, faired not much better. In the period from 2010 to 2016 the record advanced from a 676-bit field, up to 4841 [1].

Medium Prime Finite Field Discrete Logarithms The 'medium prime' case of the finite field discrete logarithm problem is for the case $q^n = \ell^m$, where ℓ is not small and m is not one. It was not really until the advent of pairing based cryptography and XTR in the early 2000's that this problem became of interest to cryptographers. The first major work on this case of the problem was by Granger and Vercauteren [36], and Joux and Lercier [45], which showed that a variant of the Function Field Sieve could be applied in this case to obtain a complexity of $L_{q^n}(1/3, c + o(1))$. In [46] the analysis was then carried out for the Number Field Sieve algorithm as well. This enabled a

$L_{q^n}(1/3, c + o(1))$ algorithm to be applied in all finite fields; with the precise algorithm to be used depending on the sizes of ℓ and m.

In 2015 Barbulescu et al [10] initiated the renewed study of an earlier algorithm of Schirokauer called the Tower Number Field Sieve (TNFS), which had appeared in [85]. There followed a rapid sequence of papers looking again at the TNFS as a means to attack the third type of pairing parameters mentioned above, e.g. [9, 47, 48, 83].

No efficient implementation of the TNFS algorithm exists as of 2019. Yet, hardness estimates can be done using rough extrapolations. This was done for example by Barbulescu and Duquesne in [7] or Guillevic in [38], who provided comprehensive analyses of key sizes for such pairing based systems. In particular their recommendation is that for 128-bits of symmetric security a BN curve needs to be selected with at least $q \approx 2^{446}$, which makes the implementation advantages of BN curves disappear, since q is no longer of the order of 2^{256}. Thus curves would need to be selected, to obtain the same balance between security and performance, with a higher embedding degree n. This is a particular problem as such BN curves have recently been standardized for use in systems such a the Trusted Computing Module.

When $n = 2$: In this case the theoretical algorithm of choice is that of Joux et al [46]. Using an adaption of this method, Barbulescu in 2014 announced a discrete logarithm record in a finite field of the form $\mathbb{F}_{q^2}$, for a prime q of 264-bits, where the interesting subgroup had order $p \approx 2^{261}$. To date this is the largest problem solved, and hence it appears that picking q of approximately 512 to 1500 bits should result still result in a secure pairing system. At the 128-bit security level their estimates correspond to using an embedding degree of $n = 16$ or $n = 18$, whereas at the 256-bit security level an embedding degree of $m = 24$ is to be preferred, and even that is not optimal.

6 Post Quantum Cryptography

The problem with the above analysis of the difficulty of the factoring and discrete logarithm problems, is that it becomes totally redundant if a quantum computer is ever built. In 1994 Shor [93] came up with a polynomial time algorithm for both the factoring problem, and the discrete logarithm problem in any finite abelian group. The only problem with Shor's algorithm is that one requires a quantum computer to run it.

A quantum computer uses quantum bits (so called qubits) as opposed to classical bits. Currently very small numbers of quantum bits can be processed for a short period of time without falling prey to noise. Thus, the issue is whether one can build a quantum computer, with enough qubits, which can run noise-free for long enough to factor an interesting number, or break an interesting discrete logarithm.

The current quantum factoring records are much less than that which can be done with classical computers. In 2001 a team from IBM factored the number 15 (into 3×5 surprisingly enough) using a quantum computer with seven qubits [98]. At the time of writing (Autumn 2019) the current record is to factor $4088459 = 2017 \times 2027$, which was done on a 5 qubit processor [23]. However, despite these numbers being small the *threat* of a quantum computer means that people have turned to examining so called post-quantum systems.

These post-quantum public key cryptographic systems are necessarily based on different hard problems (lattice based problems, coding theory based problems, problems in isogeny theory and so on). However, these problems need to go through the same analysis of difficulty to determine key sizes as has previously happened for the pre-quantum algorithms. At the time of writing NIST is running a 'competition' to determine what type of post-quantum will be most suitable in the coming decades.

Lattice Based PQC: One of the most promising areas of post-quantum cryptography is those systems based on lattices. The most basic lattice problem is the shortest-vector problem, which is the problem of determining the smallest non-zero vector in the set

$$\{ \mathbf{x} = A \cdot \mathbf{k} : \mathbf{k} \in \mathbb{Z}^n \}$$

when A is a random square integer matrix of dimension n. The most famous classical algorithm to solve this problem is the LLL algorithm [62], and its modern improvements such as the BKZ algorithm, which was introduced in [86] and then improved in other works such as [21]. However, none of the classical algorithms to solve this problem can be significantly improved using a quantum computer. Thus, the closest vector problem seems a very good candidate to use to build post-quantum cryptographic systems.

Just like with factoring and elliptic curve based systems before them, lattice based cryptanalysis has been spurred on by a series of challenge problems. The most famous of these are the Darmstadt Lattice Challenge `https://www.latticechallenge.org/`. This presents a number of challenges for different dimensions n, and for different variants of lattice problems.

The most important variant of the challenges is that given by the Learning-With-Errors (LWE) problem. This is the problem to recover the vector $\mathbf{s} \in \mathbb{Z}^n$ given only a random matrix A and the vector $\mathbf{b}$ which satisfies

$$\mathbf{b} = A \cdot \mathbf{s} + \mathbf{e} \pmod{q}$$

for some vector $\mathbf{e}$ of small euclidean norm. The LWE problem is *related* to the classical closest vector problem in lattices, but it can be used to more easily construct cryptographic systems.

The problem with lattices problems, and LWE problems in particular, is that the potential attack algorithms have many different parameters which can be tweaked, and many different sub-procedures may have (small) potential quantum speed-ups. Thus determining a precise 'key size' estimate is hard. To help this with the LWE problem there is software which performs this calculation for one called the *lwe-estimator*: `https://bitbucket.org/malb/lwe-estimator/src/master/`.

Coding Based PQC: Basing public key cryptography on the difficult problem of decoding random linear codes is a long standing construction. The earliest designs go back to McEliece [68] in 1978. What is quite remarkable about code based cryptography is that the state-of-the-art in terms of cryptanalysis has remained relatively constant over the last forty years. This is in part due to the problem of decoding random linear codes being known to be an NP-hard.

Again to spur the community into studying these problems, especially due to the NIST 'competition', a set of challenge problems have been created `http://decodingchallenge.org/`. The problems, much like the Darmstadt Lattice Challenge problems are divided into generic decoding problems, and problems related more closely to proposed cryptographic systems.

MQ Based PQC: It is well known that the SAT problem is NP-Complete, thus it is clearly an attractive proposition to try to base cryptographic schemes on problems related to SAT. One such attempt is to use so-called MQ based problems. In these problems one is given a large set of multi-variate quadratic equations over a finite field (usually one of characteristic two) and one is asked to find a solution. The first such important proposal in this space was by Matsumoto and Imai in 1988 [67], who presented a proposal for public key signatures and public key encryption. This was, however, broken in 1995 by Patarin [73].

The basic idea remains attractive. Whilst almost all work on public key encryption based on MQ systems has not led to a good system, the situation for public key signatures is very different. Patarin and his co-authors introduced the 'Balance Oil and Vinegar' signature scheme in 1997 [74] and then the 'Unbalanced Oil and Vinegar' scheme in 1999 [49]. These ideas have kept being worked on, and a number of submissions to the NIST 'competition' are based on the Oil and Vinegar construction.

Again to help research in cryptanalysis of MQ based cryptography a web site with a set of challenges has been set up `https://www.mqchallenge.org/`.

Symmetric Key Based PQC: Another way to construct post-quantum signatures is to dispense with complex constructions based on hard mathematical problems, and simply base public key cryptography upon symmetric key cryptography. This is because quantum computers are not expected to pose much of a threat to standard symmetric key cryptography. Two of the main post-quantum signature schemes to precisely this.

The first construction based on hash-functions goes back to an old idea from 1979, namely the one-time signature scheme of Lamport [52]. This when combined with Merkle-trees can be used to form a statefull many-time signature scheme, [70]. To remove the statefull nature of the signature scheme, an algorithm called SPHINCS was proposed in 2015 [12].

The second construction makes use of the difficulty of inverting one-way functions, and is based on a technique from theoretical cryptography called MPC-in-the-Head [41]. This methodology was extended in a series of works [34, 3]. When instantiated with low-complexity block ciphers such signature schemes are rather efficient.

7 Key Size Recommendation

The asymptotic complexity of GNFS to factor an integer N is of the following form, as $N \to \infty$:

$$L(N)^{1+o(1)}, \text{ with } L(N) = \exp\left((64/9)^{1/3}(\log N)^{1/3}(\log\log N)^{2/3}\right).$$

Back almost 30 years ago, the complexity of MPQS was written with a similar shaped equation, with a larger exponent $1/2$ instead of $1/3$ in the exponent. But the $(1 + o(1))$

part was present there too. In that MPQS context, the RSA-129 paper contained the following cautious words, to guard against the temptation to ignore the $o(1)$ and use simply the analogue of $L(N)$ as an operation count: *since* $o(1)$ *is neither 0 nor constant, this practice hardly makes sense*. Even knowing the computation time from one number N_1 gives only shallow ground for an extrapolation to another size N_2 based on the ratio $L(N_2)/L(N_1)$: this cannot account for the growth of what is hidden by $o(1)$. Yet, with due care, extrapolations based on such ratios were proposed over the years, based on known GNFS records, in order to answer the much-needed question: what key size will match the appropriate security level by year X ?

Table 2 is given in [26] as a summary of recommendations from various sources. It summarizes the key size recommendations for the 'standard' algorithms currently in use, i.e. it ignores the current zoo of post-quantum algorithm proposals. The column 'Legacy' refers to key sizes which are currently in use and which should be retired as soon as possible (but not urgently); whereas near term refers to a time period until 2026 (ten years ahead for a report published in 2016), whereas long term refers to projecting security beyond that horizon.

			Future System Use	
	Parameter	Legacy	Near Term	Long Term
Symmetric Key Size	k	80	128	256
Hash Function Output Size	m	160	256	512
MAC Output Size	m	80	128	256
RSA Problem	$\ell(n) \geq$	1024	3072	15360
Finite Field DLP	$\ell(p^n) \geq$	1024	3072	15360
	$\ell(p), \ell(q) \geq$	160	256	512
ECDLP	$\ell(q) \geq$	160	256	512
Pairing	$\ell(q^n) \geq$	1024	3072	15360
	$\ell(p), \ell(q) \geq$	160	256	512

Table 2. Key Size Analysis, where $\ell(\cdot)$ refers to the bit-length of the parameter.

Another way of measuring security was also introduced by Lenstra and co-authors in [65]. In this paper the authors suggest measuring the energy required to break a cryptosystem. This idea stemmed from a remark made about the factorization of the RSA-768 challenge, in that the authors estimated that it used as much energy as would have sufficed to bring two Olympic size swimming pools to the boil from a starting temperature of 20 degrees Celsius.

The traditional 80-bit security level is equivalent to boiling the average daily rain fall of the Netherlands (which is apparently according to [65] the healthy equivalent to 2^{15} swimming pools per day). To obtain 128-bit security one needs to expend the energy to boil off all the water on the planet. Thus with this metric one can rest assured that ciphers that require 2^{128} operations to break them will remain secure for a very long time indeed.

Bibliography

[1] Adj, Gora, Canales-Martínez, Isaac, Cruz-Cortés, Nareli, Menezes, Alfred, Oliveira, Thomaz, Rivera-Zamarripa, Luis, and Rodríguez-Henríquez, Francisco. 2016. *Computing discrete logarithms in cryptographically-interesting characteristic-three finite fields*. Cryptology ePrint Archive, Report 2016/914. `http://eprint.iacr.org/2016/914`.

[2] Adleman, Leonard M. 1994. The function field sieve. Pages 108–121 of: Adleman, Leonard M., and Huang, Ming-Deh A. (eds), *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*. Lecture Notes in Computer Science, vol. 877. Springer.

[3] Albrecht, Martin R., Rechberger, Christian, Schneider, Thomas, Tiessen, Tyge, and Zohner, Michael. 2015. Ciphers for MPC and FHE. Pages 430–454 of: Oswald, Elisabeth, and Fischlin, Marc (eds), *EUROCRYPT 2015, Part I*. LNCS, vol. 9056. Sofia, Bulgaria: Springer, Heidelberg, Germany.

[4] ANSSI. 2010. *Référentiel Général de Sécurité, Annexe B1 Mécanismes cryptographiques : Régles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques*. Version 1.20.

[5] Aoki, Kazumaro, Franke, Jens, Kleinjung, Thorsten, Lenstra, Arjen K., and Osvik, Dag Arne. 2007. A Kilobit Special Number Field Sieve Factorization. Pages 1–12 of: Kurosawa, Kaoru (ed), *ASIACRYPT 2007*. LNCS, vol. 4833. Kuching, Malaysia: Springer, Heidelberg, Germany.

[6] Atkins, Derek, Graff, Michael, Lenstra, Arjen K., and Leyland, Paul C. 1995. The Magic Words are Squeamish Ossifrage. Pages 263–277 of: Pieprzyk, Josef, and Safavi-Naini, Reihaneh (eds), *ASIACRYPT'94*. LNCS, vol. 917. Wollongong, Australia: Springer, Heidelberg, Germany.

[7] Barbulescu, Razvan, and Duquesne, Sylvain. 2019. Updating Key Size Estimations for Pairings. *Journal of Cryptology*, **32**(4), 1298–1336.

[8] Barbulescu, Razvan, Gaudry, Pierrick, Joux, Antoine, and Thomé, Emmanuel. 2014. A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. Pages 1–16 of: Nguyen, Phong Q., and Oswald, Elisabeth (eds), *EUROCRYPT 2014*. LNCS, vol. 8441. Copenhagen, Denmark: Springer, Heidelberg, Germany.

[9] Barbulescu, Razvan, Gaudry, Pierrick, Guillevic, Aurore, and Morain, François. 2015a. Improving NFS for the Discrete Logarithm Problem in Non-prime Finite Fields. Pages 129–155 of: Oswald, Elisabeth, and Fischlin, Marc (eds), *EUROCRYPT 2015, Part I*. LNCS, vol. 9056. Sofia, Bulgaria: Springer, Heidelberg, Germany.

[10] Barbulescu, Razvan, Gaudry, Pierrick, and Kleinjung, Thorsten. 2015b. The Tower Number Field Sieve. Pages 31–55 of: Iwata, Tetsu, and Cheon, Jung Hee (eds), *ASIACRYPT 2015, Part II*. LNCS, vol. 9453. Auckland, New Zealand: Springer, Heidelberg, Germany.

[11] Barreto, Paulo S. L. M., and Naehrig, Michael. 2006. Pairing-Friendly Elliptic Curves of Prime Order. Pages 319–331 of: Preneel, Bart, and Tavares, Stafford

(eds), *SAC 2005*. LNCS, vol. 3897. Kingston, Ontario, Canada: Springer, Heidelberg, Germany.

[12] Bernstein, Daniel J., Hopwood, Daira, Hülsing, Andreas, Lange, Tanja, Niederhagen, Ruben, Papachristodoulou, Louiza, Schneider, Michael, Schwabe, Peter, and Wilcox-O'Hearn, Zooko. 2015. SPHINCS: Practical Stateless Hash-Based Signatures. Pages 368–397 of: Oswald, Elisabeth, and Fischlin, Marc (eds), *EUROCRYPT 2015, Part I*. LNCS, vol. 9056. Sofia, Bulgaria: Springer, Heidelberg, Germany.

[13] Blake, Ian F., Mullin, Ronald C., and Vanstone, Scott A. 1984. Computing Logarithms in GF(2^n). Pages 73–82 of: Blakley, G. R., and Chaum, David (eds), *CRYPTO'84*. LNCS, vol. 196. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[14] Boneh, Dan, and Franklin, Matthew K. 2001. Identity-Based Encryption from the Weil Pairing. Pages 213–229 of: Kilian, Joe (ed), *CRYPTO 2001*. LNCS, vol. 2139. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[15] Bos, Joppe W., and Kleinjung, Thorsten. 2012. ECM at Work. Pages 467–484 of: Wang, Xiaoyun, and Sako, Kazue (eds), *ASIACRYPT 2012*. LNCS, vol. 7658. Beijing, China: Springer, Heidelberg, Germany.

[16] Bos, Joppe W., Kleinjung, Thorsten, Lenstra, Arjen K., and Montgomery, Peter L. 2011. Efficient SIMD Arithmetic Modulo a Mersenne Number. Pages 213–221 of: *20th IEEE Symposium on Computer Arithmetic, ARITH 2011, Tübingen, Germany, 25-27 July 2011*.

[17] Bos, Joppe W., Kaihara, Marcelo E., Kleinjung, Thorsten, Lenstra, Arjen K., and Montgomery, Peter L. 2012. Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction. *IJACT*, **2**(3), 212–228.

[18] Boudot, Fabrice, Gaudry, Pierrick, Guillevic, Aurore, Heninger, Nadia, Thomé, Emmanuel, and Zimmermann, Paul. 2020. *Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment*. Cryptology ePrint Archive, Report 2020/697. `https://eprint.iacr.org/2020/697`.

[19] BSI. 2013. *Kryptographische Verfahren: Empfehlungen und Schlüssellängen. BSI TR- 02102*. version 2013.2.

[20] Cavallar, Stefania, Dodson, Bruce, Lenstra, Arjen K., Lioen, Walter M., Montgomery, Peter L., Murphy, Brian, te Riele, Herman, Aardal, Karen, Gilchrist, Jeff, Guillerm, Gérard, Leyland, Paul C., Marchand, Joël, Morain, François, Muffett, Alec, Putnam, Chris, Putnam, Craig, and Zimmermann, Paul. 2000. Factorization of a 512-Bit RSA Modulus. Pages 1–18 of: Preneel, Bart (ed), *EUROCRYPT 2000*. LNCS, vol. 1807. Bruges, Belgium: Springer, Heidelberg, Germany.

[21] Chen, Yuanmi, and Nguyen, Phong Q. 2011. BKZ 2.0: Better Lattice Security Estimates. Pages 1–20 of: Lee, Dong Hoon, and Wang, Xiaoyun (eds), *ASIACRYPT 2011*. LNCS, vol. 7073. Seoul, South Korea: Springer, Heidelberg, Germany.

[22] Coppersmith, Don. 1984. Fast evaluation of logarithms in fields of characteristic two. *IEEE Trans. Information Theory*, **30**(4), 587–593.

[23] Dash, Avinash, Sarmah, Deepankar, Behera, Bikash K., and Panigrahi, Prasanta K. 2018. *Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer*.

[24] Diem, Claus. 2011. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, **147**, 75–104.
[25] Diffie, Whitfield, and Hellman, Martin E. 1977. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, **10**(6), 74–84.
[26] ECRYPT. 2018. *Algorithms, Key Size and Protocols Report*. `https://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf`.
[27] (Ed.)., J. Gilmore. 1998. *Cracking DES: Secrets of Encryption Research, Wiretap Politics and Chip Design*. Electronic Frontier Foundation, O'Reilly & Associates.
[28] Franke, Jens, Kleinjung, Thorsten, Paar, Christof, Pelzl, Jan, Priplata, Christine, and Stahlke, Colin. 2005. SHARK: A Realizable Special Hardware Sieving Device for Factoring 1024-Bit Integers. Pages 119–130 of: Rao, Josyula R., and Sunar, Berk (eds), *CHES 2005*. LNCS, vol. 3659. Edinburgh, UK: Springer, Heidelberg, Germany.
[29] Fried, Joshua, Gaudry, Pierrick, Heninger, Nadia, and Thomé, Emmanuel. 2017. A Kilobit Hidden SNFS Discrete Logarithm Computation. Pages 202–231 of: Coron, Jean-Sébastien, and Nielsen, Jesper Buus (eds), *EUROCRYPT 2017, Part I*. LNCS, vol. 10210. Paris, France: Springer, Heidelberg, Germany.
[30] Gaudry, Pierrick. 2009. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, **44**, 1690–1702.
[31] Gaudry, Pierrick, Hess, Florian, and Smart, Nigel P. 2002. Constructive and Destructive Facets of Weil Descent on Elliptic Curves. *Journal of Cryptology*, **15**(1), 19–46.
[32] Geiselmann, Willi, and Steinwandt, Rainer. 2003. A Dedicated Sieving Hardware. Pages 254–266 of: Desmedt, Yvo (ed), *PKC 2003*. LNCS, vol. 2567. Miami, FL, USA: Springer, Heidelberg, Germany.
[33] Geiselmann, Willi, and Steinwandt, Rainer. 2004. Yet Another Sieving Device. Pages 278–291 of: Okamoto, Tatsuaki (ed), *CT-RSA 2004*. LNCS, vol. 2964. San Francisco, CA, USA: Springer, Heidelberg, Germany.
[34] Giacomelli, Irene, Madsen, Jesper, and Orlandi, Claudio. 2016. ZKBoo: Faster Zero-Knowledge for Boolean Circuits. Pages 1069–1083 of: Holz, Thorsten, and Savage, Stefan (eds), *USENIX Security 2016*. Austin, TX, USA: USENIX Association.
[35] Gölogluu, Faruk, Granger, Robert, McGuire, Gary, and Zumbrägel, Jens. 2013. On the Function Field Sieve and the Impact of Higher Splitting Probabilities — Application to Discrete Logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. Pages 109–128 of: Canetti, Ran, and Garay, Juan A. (eds), *CRYPTO 2013, Part II*. LNCS, vol. 8043. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.
[36] Granger, Robert, and Vercauteren, Frederik. 2005. On the Discrete Logarithm Problem on Algebraic Tori. Pages 66–85 of: Shoup, Victor (ed), *CRYPTO 2005*. LNCS, vol. 3621. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.
[37] Granger, Robert, Kleinjung, Thorsten, and Zumbrägel, Jens. 2018. On the discrete logarithm problem in finite fields of fixed characteristic. *Trans. Amer. Math. Soc.*, **370**, 3129–3145.
[38] Guillevic, Aurore. 2020. A Short-List of Pairing-Friendly Curves Resistant to Special TNFS at the 128-Bit Security Level. Pages 535–564 of: Kiayias, Aggelos,

Kohlweiss, Markulf, Wallden, Petros, and Zikas, Vassilis (eds), *PKC 2020, Part II*. LNCS, vol. 12111. Edinburgh, UK: Springer, Heidelberg, Germany.

[39] Hawkes, Philip, Paddon, Michael, and Rose, Gregory G. 2004. *Musings on the Wang et al. MD5 Collision*. Cryptology ePrint Archive, Report 2004/264. `http://eprint.iacr.org/2004/264`.

[40] Hellman, Martin E., and Reyneri, Justin M. 1982. Fast Computation of Discrete Logarithms in GF(q). Pages 3–13 of: Chaum, David, Rivest, Ronald L., and Sherman, Alan T. (eds), *CRYPTO'82*. Santa Barbara, CA, USA: Plenum Press, New York, USA.

[41] Ishai, Yuval, Kushilevitz, Eyal, Ostrovsky, Rafail, and Sahai, Amit. 2007. Zero-knowledge from secure multiparty computation. Pages 21–30 of: Johnson, David S., and Feige, Uriel (eds), *39th ACM STOC*. San Diego, CA, USA: ACM Press.

[42] Joux, Antoine. 2000. A One Round Protocol for Tripartite Diffie-Hellman. Pages 385–394 of: Bosma, Wieb (ed), *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*. Lecture Notes in Computer Science, vol. 1838. Springer.

[43] Joux, Antoine. 2014. A New Index Calculus Algorithm with Complexity $L(1/4 + o(1))$ in Small Characteristic. Pages 355–379 of: Lange, Tanja, Lauter, Kristin, and Lisonek, Petr (eds), *SAC 2013*. LNCS, vol. 8282. Burnaby, BC, Canada: Springer, Heidelberg, Germany.

[44] Joux, Antoine, and Lercier, Reynald. 2002. The Function Field Sieve Is Quite Special. Pages 431–445 of: Fieker, Claus, and Kohel, David R. (eds), *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*. Lecture Notes in Computer Science, vol. 2369. Springer.

[45] Joux, Antoine, and Lercier, Reynald. 2006. The Function Field Sieve in the Medium Prime Case. Pages 254–270 of: Vaudenay, Serge (ed), *EUROCRYPT 2006*. LNCS, vol. 4004. St. Petersburg, Russia: Springer, Heidelberg, Germany.

[46] Joux, Antoine, Lercier, Reynald, Smart, Nigel, and Vercauteren, Frederik. 2006. The Number Field Sieve in the Medium Prime Case. Pages 326–344 of: Dwork, Cynthia (ed), *CRYPTO 2006*. LNCS, vol. 4117. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[47] Kim, Taechan, and Barbulescu, Razvan. 2016. Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case. Pages 543–571 of: Robshaw, Matthew, and Katz, Jonathan (eds), *CRYPTO 2016, Part I*. LNCS, vol. 9814. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[48] Kim, Taechan, and Jeong, Jinhyuck. 2017. Extended Tower Number Field Sieve with Application to Finite Fields of Arbitrary Composite Extension Degree. Pages 388–408 of: Fehr, Serge (ed), *PKC 2017, Part I*. LNCS, vol. 10174. Amsterdam, The Netherlands: Springer, Heidelberg, Germany.

[49] Kipnis, Aviad, Patarin, Jacques, and Goubin, Louis. 1999. Unbalanced Oil and Vinegar Signature Schemes. Pages 206–222 of: Stern, Jacques (ed), *EUROCRYPT'99*. LNCS, vol. 1592. Prague, Czech Republic: Springer, Heidelberg, Germany.

[50] Kleinjung, Thorsten, Lenstra, Arjen K., Page, Dan, and Smart, Nigel P. 2012. Using the Cloud to Determine Key Strengths. Pages 17–39 of: Galbraith, Steven D., and Nandi, Mridul (eds), *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*. Lecture Notes in Computer Science, vol. 7668. Springer.

[51] Kumar, Sandeep, Paar, Christof, Pelzl, Jan, Pfeiffer, Gerd, and Schimmler, Manfred. 2006. Breaking Ciphers with COPACOBANA - A Cost-Optimized Parallel Code Breaker. Pages 101–118 of: Goubin, Louis, and Matsui, Mitsuru (eds), *CHES 2006*. LNCS, vol. 4249. Yokohama, Japan: Springer, Heidelberg, Germany.

[52] Lamport, Leslie. 1979 (Oct.). *Constructing Digital Signatures from a One-way Function*. Technical Report SRI-CSL-98. SRI International Computer Science Laboratory.

[53] Lehmer, Derrick H. 1928. The mechanical combination of linear forms. *American Mathematical Monthly*, **35**, 114–121.

[54] Lehmer, Derrick H. 1933. A photo-electric number sieve. *American Mathematical Monthly*, **40**, 401–406.

[55] Lehmer, Derrick N. 1932. Hunting big game in the theory of numbers. *Scripta Mathematica I*, 229–235.

[56] Lenstra, Arjen K., and Lenstra Jr., Hendrik W. (eds). 1993. *The development of the number field sieve*. Lecture Notes in Math., vol. 1554. Springer.

[57] Lenstra, Arjen K., and Manasse, Mark S. 1990. Factoring by Electronic Mail. Pages 355–371 of: Quisquater, Jean-Jacques, and Vandewalle, Joos (eds), *EUROCRYPT'89*. LNCS, vol. 434. Houthalen, Belgium: Springer, Heidelberg, Germany.

[58] Lenstra, Arjen K., and Shamir, Adi. 2000. Analysis and Optimization of the TWINKLE Factoring Device. Pages 35–52 of: Preneel, Bart (ed), *EUROCRYPT 2000*. LNCS, vol. 1807. Bruges, Belgium: Springer, Heidelberg, Germany.

[59] Lenstra, Arjen K., and Verheul, Eric R. 2000a. Selecting Cryptographic Key Sizes. Pages 446–465 of: Imai, Hideki, and Zheng, Yuliang (eds), *PKC 2000*. LNCS, vol. 1751. Melbourne, Victoria, Australia: Springer, Heidelberg, Germany.

[60] Lenstra, Arjen K., and Verheul, Eric R. 2000b. The XTR Public Key System. Pages 1–19 of: Bellare, Mihir (ed), *CRYPTO 2000*. LNCS, vol. 1880. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[61] Lenstra, Arjen K., and Verheul, Eric R. 2001. Selecting Cryptographic Key Sizes. *Journal of Cryptology*, **14**(4), 255–293.

[62] Lenstra, Arjen K., Lenstra, Hendrik W., and Lovasz, László. 1982. Factoring polynomials with rational coefficients. *Math. Ann.*, **261**, 515–534.

[63] Lenstra, Arjen K., Shamir, Adi, Tomlinson, Jim, and Tromer, Eran. 2002. Analysis of Bernstein's Factorization Circuit. Pages 1–26 of: Zheng, Yuliang (ed), *ASIACRYPT 2002*. LNCS, vol. 2501. Queenstown, New Zealand: Springer, Heidelberg, Germany.

[64] Lenstra, Arjen K., Tromer, Eran, Shamir, Adi, Kortsmit, Wil, Dodson, Bruce, Hughes, James P., and Leyland, Paul C. 2003. Factoring Estimates for a 1024-Bit RSA Modulus. Pages 55–74 of: Laih, Chi-Sung (ed), *ASIACRYPT 2003*. LNCS, vol. 2894. Taipei, Taiwan: Springer, Heidelberg, Germany.

[65] Lenstra, Arjen K., Kleinjung, Thorsten, and Thomé, Emmanuel. 2013. Universal Security - From Bits and Mips to Pools, Lakes - and Beyond. Pages 121–124 of: Fischlin, Marc, and Katzenbeisser, Stefan (eds), *Number Theory and Cryptography - Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*. Lecture Notes in Computer Science, vol. 8260. Springer.

[66] Leurent, Gaëtan, and Peyrin, Thomas. 2019. From Collisions to Chosen-Prefix Collisions Application to Full SHA-1. Pages 527–555 of: Ishai, Yuval, and Rijmen, Vincent (eds), *EUROCRYPT 2019, Part III*. LNCS, vol. 11478. Darmstadt, Germany: Springer, Heidelberg, Germany.

[67] Matsumoto, Tsutomu, and Imai, Hideki. 1988. Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. Pages 419–453 of: Günther, C. G. (ed), *EUROCRYPT'88*. LNCS, vol. 330. Davos, Switzerland: Springer, Heidelberg, Germany.

[68] McEliece, Robert J. 1978. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *DSN Progress Report*, **44**, 114–116.

[69] Menezes, Alfred, Vanstone, Scott A., and Okamoto, Tatsuaki. 1991. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. Pages 80–89 of: *23rd ACM STOC*. New Orleans, LA, USA: ACM Press.

[70] Merkle, Ralph C. 1990. A Certified Digital Signature. Pages 218–238 of: Brassard, Gilles (ed), *CRYPTO'89*. LNCS, vol. 435. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[71] Morrison, M. A., and Brillhart, J. 1975. A method of factoring and the factorization of F_7. *Math. Comp.*, **29**(129), 183–205.

[72] NIST. 2012. *Special Publication 800-57. Recommendation for key management Part 1: General (Revision 3)*.

[73] Patarin, Jacques. 1995. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. Pages 248–261 of: Coppersmith, Don (ed), *CRYPTO'95*. LNCS, vol. 963. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[74] Patarin, Jacques. 1997. *The Oil and Vinegar Algorithm for Signatures*. Presented at the Dagstuhl Workshop on Cryptography.

[75] Pollard, John M. 1974. Theorems on factorization and primality testing. *Proc. Cambr. Philos. Soc.*, **76**, 521–528.

[76] Pollard, John M. 1978. Monte Carlo algorithms for index computation (mod p). *Math. Comp.*, **32**, 918–924.

[77] Pollard, John. M. 2000. Kangaroos, Monopoly and Discrete Logarithms. *Journal of Cryptology*, **13**, 437–447.

[78] Pomerance, Carl, Smith, J. W., and Wagstaff, S. S. 1983. New Ideas for Factoring Large Integers. Pages 81–85 of: Chaum, David (ed), *CRYPTO'83*. Santa Barbara, CA, USA: Plenum Press, New York, USA.

[79] Quisquater, J.-J, and Standaert, F. 2005. Exhaustive key search of the DES: Updates and refinements. *SHARCS 2005*.

[80] Rivest, Ronald L., Shamir, Adi, and Adleman, Leonard M. 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the Association for Computing Machinery*, **21**(2), 120–126.

[81] Rudd, Walter G., Buell, Duncan A., and Chiarulli, Donald M. 1984. A High Performance Factoring Machine. Pages 297–300 of: Agrawal, Dharma P. (ed),

Proceedings of the 11th Annual Symposium on Computer Architecture, Ann Arbor, USA, June 1984. ACM.

[82] Sakai, Ryuichi, Ohgishi, Kiyoshi, and Kasahara, Masao. 2000 (Jan.). Cryptosystems based on Pairing. In: *SCIS 2000*.

[83] Sarkar, Palash, and Singh, Shashank. 2016. A General Polynomial Selection Method and New Asymptotic Complexities for the Tower Number Field Sieve Algorithm. Pages 37–62 of: Cheon, Jung Hee, and Takagi, Tsuyoshi (eds), *ASIACRYPT 2016, Part I*. LNCS, vol. 10031. Hanoi, Vietnam: Springer, Heidelberg, Germany.

[84] Satoh, T., and Araki, K. 1998. Fermat quotients and the polynomial time discrete log algorithm for anomalouselliptic curves. *Comm. Math. Univ. Sancti. Pauli*, **47**, 81–92.

[85] Schirokauer, Oliver. 2000. Using number fields to compute logarithms in finite fields. *Math. Comput.*, **69**(231), 1267–1283.

[86] Schnorr, Claus-Peter. 1987. A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms. *Theor. Comput. Sci.*, **53**, 201–224.

[87] Semaev, Igor. 2004. *Summation polynomials and the discrete logarithm problem on elliptic curves*. Cryptology ePrint Archive, Report 2004/031. `http://eprint.iacr.org/2004/031`.

[88] Semaev, Igor A. 1998. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Math. Comput.*, **67**(221), 353–356.

[89] Shallit, Jeffrey, Williams, Hugh C., and Morain, Francois. 1995. Discovery of a lost factoring machine. *The Mathematical Intelligencer*, 41–47.

[90] Shamir, Adi. 1999. Factoring Large Numbers with the Twinkle Device (Extended Abstract). Pages 2–12 of: Koç, Çetin Kaya, and Paar, Christof (eds), *CHES'99*. LNCS, vol. 1717. Worcester, Massachusetts, USA: Springer, Heidelberg, Germany.

[91] Shamir, Adi, and Tromer, Eran. 2003. Factoring Large Number with the TWIRL Device. Pages 1–26 of: Boneh, Dan (ed), *CRYPTO 2003*. LNCS, vol. 2729. Santa Barbara, CA, USA: Springer, Heidelberg, Germany.

[92] Shanks, Daniel. 1971. Class number, a theory of factorization, and genera. Pages 415–440 of: Lewis, D. J. (ed), *1969 Number theory institute*. Proc. Sympos. Pure Math., vol. 20. Amer. Math. Soc.

[93] Shor, Peter W. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. Pages 124–134 of: *35th FOCS*. Santa Fe, NM, USA: IEEE Computer Society Press.

[94] Silverman, Robert D. 1987. The multiple polynomial quadratic sieve. *Math. Comp.*, **48**, 329–339.

[95] Smart, Nigel P. 1999. The Discrete Logarithm Problem on Elliptic Curves of Trace One. *Journal of Cryptology*, **12**(3), 193–196.

[96] Stevens, Marc, Lenstra, Arjen K., and de Weger, Benne. 2007. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. Pages 1–22 of: Naor, Moni (ed), *EUROCRYPT 2007*. LNCS, vol. 4515. Barcelona, Spain: Springer, Heidelberg, Germany.

[97] van Oorschot, Paul C., and Wiener, Michael J. 1999. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, **12**(1), 1–28.

[98] Vandersypen, Leiven M.K., Steffen, Matthias, Breyta, Gregory, Yannoni, Costantino S., Sherwood, Mark H., and Chuang, Isaac L. 2001. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, **414**(6866), 883–887.

[99] Wang, Xiaoyun, and Yu, Hongbo. 2005. How to Break MD5 and Other Hash Functions. Pages 19–35 of: Cramer, Ronald (ed), *EUROCRYPT 2005*. LNCS, vol. 3494. Aarhus, Denmark: Springer, Heidelberg, Germany.