To appear in *The Journal of Cryptology*, Springer-Verlag.

Selecting Cryptographic Key Sizes

Arjen K. Lenstra[1], Eric R. Verheul[2]

[1] Citibank, N.A., and Technische Universiteit Eindhoven,
1 North Gate Road, Mendham, NJ 07945-3104, U.S.A.,
arjen.lenstra@citicorp.com
[2] PricewaterhouseCoopers, GRMS Crypto Group,
Goudsbloemstraat 14, 5644 KE Eindhoven, The Netherlands,
eric.verheul@[nl.pwcglobal.com, pobox.com]

Abstract. In this article we offer guidelines for the determination of key sizes for symmetric cryptosystems, RSA, and discrete logarithm based cryptosystems both over finite fields and over groups of elliptic curves over prime fields. Our recommendations are based on a set of explicitly formulated parameter settings, combined with existing data points about the cryptosystems.

1 Introduction

1.1 The Purpose of this Paper

Cryptography is one of the most important tools that enable e-commerce because cryptography makes it possible to protect electronic information. The effectiveness of this protection depends on a variety of mostly unrelated issues such as cryptographic key size, protocol design, and password selection. Each of these issues is equally important: if a key is too small, or if a protocol is badly designed or incorrectly used, or if a password is poorly selected or protected, then the protection fails and improper access can be gained.

In this article we give some guidelines for the determination of cryptographic key sizes. For each of a number of cryptosystems we describe the effort and cost required for a successful attack, where the cost may be measured in several different ways. Other protocol- or password-related issues are not discussed. We do not aim to predict the future, but if current trends persist, then following our guidelines will result in acceptable security for commercial applications of cryptography.

Key size recommendations are scattered throughout the cryptographic literature or may, for a particular cryptosystem, be found in vendor documentation. Unfortunately it is often hard to tell on what premises (other than marketability) the recommendations are based. As far as we know this article is the first uniform, clearly defined, and properly documented treatment of this subject for the most important generally accepted cryptosystems. We formulate a set of explicit parameter settings and apply these uniformly to existing data about the cryptosystems. The resulting key size recommendations are thus obtained in a uniform mechanical way, depending only on our default settings, but independent of further assumptions or non-scientific considerations. The resulting key size recommendations are intended for designers who want a 'conservative' estimate for the key sizes for various schemes over the next 20 to 30 years.

Our key size recommendations are not intended as 'best estimates' based on arguments arguing for or against certain implementation-related difficulties. Even though some of these arguments may be not without merit, they are avoided. Basing a security argument on something that currently happens to be perceived as a problem as opposed to basing it on the more intrinsic bigger picture is, in our opinion, wishful thinking.

Despite our attempt to be objective we do not expect that our defaults are to everyone's taste. They can, however, easily be changed without affecting the overall approach, thereby making this article useful also for those who object to our choices or the resulting key size recommendations. Other papers containing key size recommendations are [3, 5] (symmetric key cryptosystems), [29] (RSA), [16] (RSA and elliptic curve cryptosystems), and [38] (symmetric and asymmetric key cryptosystems). An extended abstract of this article appeared in [22].

Although the choice of key sizes usually gets the most attention, nearly all failures are, in our experience, not due to inadequate key sizes but to protocol or password deficiencies. To illustrate this, the cryptographic key sizes used by the popular email encryption program "Pretty Good Privacy" (PGP) offer an acceptable level of security for current applications. But the user-password that protects the private PGP keys stored on an Internet-accessible PC does not necessarily offer the same security. Even if the user is relatively security-conscious and selects a password consisting of 9 characters randomly chosen from 62 alphanumeric choices, the resulting security is comparable to the security offered by the recently broken "Data Encryption Standard" and thereby unacceptable by today's standards.

An even more disturbing example can be found in many network configurations. In one example each user may select a password that consists of 14 characters, which should, in principle, offer enough security. Before transmission over the network the passwords are encrypted, with the interesting feature however that each password is split into two parts of at most 7 characters each, and that each of the two resulting parts is treated separately, i.e., encrypted and transmitted over the network. This effectively reduces the password length of 14 to 7, which is not sufficiently secure. For more examples we refer to [1]. Thus, application of the guidelines given here makes sense only after one is convinced of the overall security of the design, of its implementation, and of end-to-end system engineering.

Our suggestions are based on reasonable extrapolations of developments that have taken place during the last few decades. This approach may fail: a single bright idea may prove that all currently popular cryptographic protocols are considerably less effective than expected. It may even render them completely ineffective, as shown by the following two examples. In the 1980s the then popular knapsack-based cryptosystems were suddenly wiped out by a new type of attack. More recently, three independent groups of researchers showed that elliptic curve cryptosystems based on the use of curves of trace one are easily breakable.

In this article we discuss only cryptosystems for which it is believed to be unlikely that such catastrophes will ever occur. Nevertheless, for some of these systems non-trivial, but non-catastrophic, new cryptanalytic insights are obtained on a fairly regular basis. So far, a gradual increase in key sizes has been an effective countermeasure against these new insights. From an application point of view it is to be hoped that this will not change anytime soon. It is the purpose of this article to give an idea by how much key sizes have to be increased to maintain a comfortable margin of security.

If sufficiently large quantum computers can be built, then all asymmetric key cryptosystems discussed in this article are insecure [34]. It is unclear if quantum computers are feasible at all. Our suggestions do not take quantum computers into account. Neither do we incorporate the potential effects of molecular computing [28].

1.1.1 Remark. Many of the considerations discussed in this article, and the default choices we make, concern parameters and issues that are at best of secondary importance. They are included for the non-specialized reader who may not immediately be able to recognize the relative importance or potential impact of the various issues related to key size selection.

1.2 Run Time Convention

All run time estimates in this article are based on actual run times or reliable estimates of run times on a 450MHz Pentium II processor, at the time of writing of this paper one of the most popular commonly available processors. A 'PC' always refers to this processor.

In the literature, computing power is often measured in Mips-Years, where a Mips-Year is defined as the amount of computation that can be performed in one year by a single DEC VAX 11/780. This measure has often been criticized because it is unclear how it can be used in a consistent manner for processors with instruction sets different from the VAX. We fully agree with the concerns expressed in [37]. Nevertheless, because of its popularity and the wide acceptance it has gained, we use this measure here as well. We use the convention that one year of computing on a PC is equivalent to 450 Mips-Years, where it should be kept in mind that ultimately all our estimates are based on run times on a PC and not on the literal definition or our definition of Mips-Years. As shown in 2.2.4 the two definitions are, however, sufficiently close. Our Mips-Year figures should therefore be compatible with Mips-Year figures found elsewhere. We write MMY for one million Mips-Years.

1.3 Lower Bounds

The guidelines in this article are meant as lower bounds in the sense that keys of sizes equal to or larger than the recommended sizes attain at least a certain specified level of security. From a security point of view it is acceptable to err on the conservative side by recommending keys that may be slightly larger than actually required. Most key size guidelines in this article are therefore obtained by systematically underestimating the computational effort required for a successful attack. Thus, keys are estimated to be weaker than they are in reality, which is acceptable for our purpose of finding lower bounds. In some cases slight overestimates of the attack effort are used instead, but in those cases there are other factors that ensure that the desired level of security is achieved.

1.4 Equivalence of Attack Efforts

We present key size recommendations for several different cryptosystems. For a certain specified level of security these recommendations may be expected to be equivalent in the sense that the computational effort or number of Mips-Years (Subsection 1.2) for a successful attack is more or less the same for all cryptosystems under consideration. So, from a computational point of view the different cryptosystems offer more or less equivalent security when the recommended key sizes are used.

This computationally equivalent security should not be confused with, and is not necessarily the same as, security with equivalent cost of equipment, or cost-equivalent security for short. Here we say that two systems offer cost-equivalent security if accessing or acquiring the hardware that allows a successful attack in a certain fixed amount of time costs the same amount of dollars for both systems. Note that although the price is the same, the hardware required may be quite different for the two different attacks; some attacks may use PCs, for other attacks it may be possible to get the required Mips-Years relatively cheaply by using special-purpose hardware. Following our guidelines does not necessarily result in cost-equivalent security. In 3.2.5 and Subsection 4.5 we indicate how our guidelines may be changed to obtain cost equivalence, thereby possibly giving up computational equivalence.

There are at least two reasons why we use computationally equivalent security as opposed to cost-equivalent security. Most importantly, we found that computational equivalence allows rigorous analysis, mostly independent of our own judgment or preferences. Analysis of cost equivalence, on the other hand, depends on subjective choices that change over time, and that have a considerable effect on the outcome. Thus, for cost equivalence there is a whole spectrum of 'reasonable' outcomes, depending on one's perception of what is reasonable. In Subsection 4.5 we present three points of the spectrum.

Another reason why we restricted ourselves to computational equivalence is that, in the model we have adopted, we need a workable notion of equivalence to achieve our goal of determining acceptable key size recommendations – achieving any type of equivalence in itself has never been our goal. Whether or not the resulting recommendations are indeed acceptable depends on how acceptable our model is found to be.

1.4.1 Remark on published versus unpublished attacks. The analyses in this paper are often based on recently published cryptanalytic results. But, as can be seen below (in particular in 3.1.2), we never use these published results to assess the security of cryptographic systems, only to derive data about the computational effort involved in a successful attack. Thus, arguments such as 'a 512-bit RSA key was broken only in 1999 (cf. 2.4.6), so 1024-bit RSA keys must be safe for quite a while' are not used in this article [38]. Does anyone seriously believe that published attacks represent the state of the art? It may safely be assumed that unpublished work is many years ahead of what the public at large gets to see: a public announcement that a system is broken provides at best a rather trivial upper bound – and a very simple-minded one, in our opinion – for the date that the system became vulnerable. This is illustrated in Remark 3.1.8. See also 2.4.5 and 3.1.3.

1.5 Organization of this Paper

In Section 2 we describe the cryptographic primitives for which we derive key size recommendations, namely the cryptographic primitives that are mentioned in the Wassenaar Arrangement (Subsection 2.1). In Section 3 we present the model underlying our key size recommendations. The model is based on a number of variables that parametrize environmental factors affecting the security or perceived security of key size choices. The role of the parameters is described and conservative default settings are suggested.

In Section 4 we apply the model from Section 3 to the cryptographic primitives from Section 2. This results in a number of formulas from which, for instance with the default settings, key size recommendations can be derived. In Section 5 we discuss some of the implications of our key size recommendations.

2 Cryptographic Primitives

2.1 The Wassenaar Arrangement

The Coordinating Committee for Multilateral Export Controls (COCOM) was an international organization regulating the mutual control of the export of strategic products, including cryptographic products, from member countries to countries that jeopardize their national security. Member countries, e.g. European countries and the US, implemented the COCOM regulations in national legislation (e.g. the ITAR in the US).

The Wassenaar Arrangement is a follow-up of the COCOM regulations. The current restrictions in the Wassenaar Arrangement (December 1998) with respect to cryptography are rather detailed [42]. For five types of cryptographic primitives a maximum key size is given for which export does not require a license. Due to the nature of the Wassenaar Arrangement, it is not surprising that it turns out that these key sizes do not provide for adequate protection of the majority of commercial applications.

In this article we limit ourselves to these cryptographic primitives. In the remainder of this section we review for each of these cryptographic primitives some facts and data that are relevant for our purposes:

– A brief description.
– The key size recommendation from the Wassenaar Arrangement.
– The most important known (i.e., published) attacks.
– The effectiveness of those attacks using generic software implementations.
– The effectiveness of those attacks using special-purpose hardware.
– The effectiveness of guessing (Remark 1.1.1).
– The effectiveness of incomplete attacks (Remark 1.1.1).
– Past cryptanalytic progress.

We distinguish the cryptographic primitives into symmetric key (or secret key) and asymmetric key (or public key) cryptosystems. Such systems are instrumental to build e-commerce enabling solutions and, more specifically, can be used to achieve confidentiality, integrity, authenticity, and non-repudiation of electronic information. For simplicity we assume two communicating parties, a sender S and a receiver R, who want to maintain confidentiality of the communication from S to R. At the end of the section we briefly mention cryptographic hash functions as well.

2.2 Symmetric Key Cryptosystems

2.2.1 Description. In symmetric key cryptosystems S and R share a key. To maintain confidentiality the key should be kept secret. The size of the key, i.e., its number of bits, depends on the symmetric key cryptosystem. Often both the message and its encryption consist of a whole number of blocks, where a block consists of a fixed number of bits that depends on the symmetric key cryptosystem. The best-known symmetric key cryptosystem is the Data Encryption Standard (DES), introduced in 1977, with key size 56 bits and block size 64 bits. Other examples of symmetric key cryptosystems are:

– Two Key Triple DES (key size 112, block size 64);
– IDEA (key size 128, block size 64);
– RC5 (variable key and block sizes);
– The forthcoming Advanced Encryption Standard (AES), with key sizes of 128, 192, or 256 bits and block size 128.

2.2.2 Wassenaar Arrangement. The maximum symmetric key size allowed by the Wassenaar Arrangement is 56 bits for 'niche market' applications and 64 bits for 'mass market'.

2.2.3 Attacks. Despite many years of research, no method has been published that breaks a DES-encrypted message substantially faster than exhaustive key search, i.e., trying all 2^{56} different keys. The expected number of trials of exhaustive key search is 2^{55}.

2.2.4 Software data points. Nowadays the DES is not considered to be sufficiently secure. In 1997 a DES key was successfully retrieved after an Internet search of approximately 4 months ([31] and Remark 3.1.3). The expected computing power required for such a software exhaustive key search is underestimated as 0.5 MMY (Subsection 1.2). This estimate is based on the Pentium based figures that a single DES block encryption with a fixed key requires 360 Pentium clock cycles [8] or 500 Pentium clock cycles with a variable key [2]. Furthermore, our estimate lies between two DEC VAX 11/780 estimates that can be found in [9, 29]. It follows that our Mips-Years convention is sufficiently accurate.

Half a million Mips-Years is roughly 13,500 months on a PC. This is equivalent to 4 months on 3,500 PCs, because an exhaustive key search can be evenly divided over any number of processors. For a proper security analysis one therefore has to evaluate and keep track of the total computational power of the Internet.

2.2.5 Special-purpose hardware data points. At the cost of a one-time investment a hardware attack is substantially faster than a software attack. In 1977 a 20 million dollar parallel DES key searching machine was proposed with an expected search time of 12 hours [11]. We write '[$20 million, 12 hours, 1977]-hardware' for this design. In [10] it was corrected to [$50 million, 2 days, 1980]-hardware. Mike Wiener published a detailed [$1 million, 3.5 hours, 1993]-hardware design [43], and special purpose [$130,000, 112 hours, 1998]-hardware was actually built [19]; see also [13].

2.2.6 Effectiveness of guessing. There is always the possibility that someone may find a key simply by guessing it. For reasonable key sizes the probability that this happens is small: even for a 50-bit key there is a total probability of one in a million that it is found if one billion people each make a different guess. With the same effort, the probability of success halves for each additional key bit: for a 60-bit key it becomes only one in a billion. Note that exhaustive key search is nothing more than systematic guessing.

2.2.7 Incomplete attacks. The success probability of exhaustive key search is proportional to the fraction of the key space searched; i.e., for any x, $0 \leq x \leq 1$, the chance is x that the key is found after searching a fraction x of the key space.

2.2.8 Cryptanalytic progress. We assume no major changes, i.e., that future symmetric key cryptosystem designs do not allow faster attacks than exhaustive key search. Also, we assume that a design that turns out to allow a faster attack will no longer be used. Below we assume the existence of a generic symmetric key cryptosystem of arbitrary key size for which exhaustive key search is the best attack. It follows that for a b-bit key a successful attack can be expected to require on the order of 2^{b-1} invocations of the underlying function.

2.3 Asymmetric Key Cryptosystems Overview

In asymmetric key cryptosystems the receiver R has a private key (which R keeps secret) and a corresponding public key that anyone, including S, has access to. The sender S uses R's public key to encrypt information intended for R, and R

uses its private key to decrypt the encrypted message. If the private key can be derived from the public key, then the system can be broken. What the private and public keys consist of, and how hard it is to break the system, depends on the type of asymmetric key cryptosystem. For cryptanalytic and historic reasons we distinguish the following three types:

1. Classical asymmetric systems;
2. Subgroup discrete logarithm systems;
3. Elliptic curve systems.

These three types of systems are discussed in more detail in the next three subsections.

2.4 Classical Asymmetric Systems

Classical Asymmetric Systems refer to RSA, due to Rivest, Shamir, and Adleman, and traditional discrete logarithm systems, such as the Diffie-Hellman and ElGamal schemes.

2.4.1 RSA description. In RSA the public key contains a large non-prime number, the so-called RSA modulus. It is chosen as the product of two large primes. If these primes can be found then the private key can be found, thereby breaking the system. Thus, the security of RSA is based on the difficulty of the integer factorization problem (cf. 2.4.10). The size of an RSA key refers to the bit-length of the RSA modulus. This should not be confused with the actual number of bits required to store an RSA public key, which may be slightly more.

2.4.2 TDL description. In a traditional discrete logarithm (TDL) system the public key consists of a finite field $\mathrm{GF}(p)$ of size p, a generator g of the multiplicative group $\mathrm{GF}(p)^*$ of $\mathrm{GF}(p)$, and an element y of $\mathrm{GF}(p)^*$ that is not equal to 1. We assume that the field size p is such that $p-1$ has a prime factor of roughly the same order of magnitude as p. The private key is the smallest positive integer m such that $g^m = y$. This m is referred to as the discrete logarithm of y with respect to g. The private key m is at least 1 and at most $p-2$. If m can be found, the system can be broken. Thus, the security of TDL systems is based on the difficulty of computing discrete logarithms in the multiplicative group of a finite field. The size of a TDL key refers to the bit-length of the field size p. The actual number of bits required to store a TDL public key is larger, since the public key contains g and y as well.

2.4.3 Wassenaar Arrangement. Both the maximal RSA modulus size and the maximal field size allowed by the Wassenaar Arrangement are 512 bits, i.e., RSA moduli and p as above should be less than 2^{512}.

2.4.4 Attacks. Factoring an RSA-modulus n by exhaustive search amounts to trying all primes up to $\sqrt{n}$. Finding a discrete logarithm by exhaustive search requires on the order of p operations in $\mathrm{GF}(p)$. Thus, if exhaustive search were the best attack on these systems, then 112-bit RSA moduli or 56-bit p's would give security comparable to the DES. However, there are much more efficient attacks than exhaustive search and much larger keys are required. Surprisingly, the methods to attack these two entirely different problems are similar, and for this reason we treat RSA and TDL systems as the same category.

The fastest factoring algorithm published today is the Number Field Sieve, invented in 1988 by John Pollard. Originally it could be used only to factor numbers of a special form, such as the ninth Fermat number $2^{512}+1$ (factored in 1990). This original version is currently referred to as the Special Number Field Sieve (SNFS) as opposed to the General Number Field Sieve (NFS), which can handle numbers of arbitrary form, including RSA moduli. On heuristic grounds the NFS can be expected to require time proportional to

$$e^{(1.9229+o(1))\ln(n)^{1/3} ln(ln(n))^{2/3}} \tag{1}$$

to factor an RSA modulus n, where the $o(1)$ term goes to zero as n goes to infinity. For notational convenience we refer to (1) as $L[n]$, which is an abbreviated version of the more common definition

$$L[n,u,v]=e^{(v+o(1))\ln(n)^{u} ln(ln(n))^{1-u}}. \tag{2}$$

The run time $L[n]$ is called subexponential in the input size n because as n goes to infinity it is less than n^c for any constant $c>0$. The storage requirements of the NFS are proportional to $\sqrt{L[n]}$. The expected run time of the SNFS is $L[n,1/3,1.5262]$; thus, the SNFS is much faster than the NFS, but it cannot be used to attack RSA moduli. If p is a prime number then a discrete logarithm variation of the NFS, which we refer to as 'DLNFS', finds a discrete logarithm in $\mathrm{GF}(p)$ in expected time proportional to $L[p]$.

These run time estimates – with omission of the $o(1)$ as is customary – cannot be used directly to estimate the number of operations required to factor a certain n or to compute discrete logarithms in a certain $\mathrm{GF}(p)$. For instance, the discrete logarithm problem in $\mathrm{GF}(p)$ is considerably more difficult than factoring an n of about the same size as p, but $L[p]$ and $L[n]$ are approximately equal if the $o(1)$'s are omitted. However, as shown by extensive experiments, the estimates can be used for limited range extrapolation. If one knows, by experimentation, that factoring an RSA modulus n using the NFS takes time t, then factoring some other RSA modulus $m>n$ will take time close to $t\frac{L[m]}{L[n]}$ (omitting the $o(1)$'s), if the sizes of n and m do not differ by too much, say by not more than 100 bits. If, however, m is much bigger than n, then the effect of the $o(1)$ going to zero can no longer be ignored, and $t\frac{L[m]}{L[n]}$ will be an overestimate of the time to factor m [36]. The same run time extrapolation method applies to the DLNFS.

2.4.5 NFS background. For a better appreciation of the security offered by classical asymmetric systems when comparing them to other asymmetric systems, we describe a few more details of the NFS. It consists of two major steps, a sieving step and a matrix step, which in theory both take an equal amount of computing time as n goes to infinity. For numbers in our current range of interest (say, up to 700 bits), however, the matrix step takes only a fraction of the computing time of the sieving step. The sieving step can be evenly distributed over any number of processors, with hardly any need for communication, resulting in a linear speedup. The computing power required for the sieving step of large scale factorizations can in principle quite easily be obtained on any loosely coupled network of computers such as the Internet. The matrix step on the other hand does not allow such a straightforward parallelization.

The situation is worse for the DLNFS. Although, as in the NFS, the DLNFS sieving and matrix steps are in theory equally hard, the DLNFS matrix step is several orders of magnitude more time- and memory-consuming than the NFS matrix step. Currently the matrix step is considered to be the major bottleneck obstructing substantially larger factorizations or even mildly interesting discrete logarithm computations. Efforts are underway to implement it on a fast and high-bandwidth network of PCs. Even though the effectiveness of that approach is still uncertain, early experiments look encouraging [24, 39] and there is no reason to believe that parallelization of the matrix step will not be successful.

It is tempting to use the perceived difficulty and apparent 'unparallelizability' of the matrix step as an argument in favor of RSA keys smaller than solely based on the estimated computational cost of breaking them. It is unclear to us, however, how this perceived difficulty should be factored in, and, more importantly, we find it imprudent to do so because it is unlikely that it will last. Indeed, there are strong and consistent indications that very fast networks of rather large PCs have been designed, and may even have been built, that would be able to tackle matrices that are very far out of reach for generally accessible computer systems. In this context we repeat (Remark 1.4.1) that it is naïve to believe that the published factorization of a 512-bit RSA modulus referred to in 2.4.6 below is the best one can do at this point (see also 3.1.2).

2.4.6 Software data points. The largest published factorization using the NFS is that of the 512-bit number RSA155, which is an RSA modulus of 155 decimal digits, in August of 1999 [6]. This factoring effort was estimated to cost at most 20 years on a PC with at least 64Mbytes of memory (or a single day on 7500 PCs). This time was spent almost entirely on the sieving step. It is less than 10^4 Mips-Years and corresponds to fewer than $3 * 10^{17}$ operations, whereas $L[10^{155}] = 2 * 10^{19}$ (omitting the $o(1)$). This shows that $L[n]$ overestimates the number of operations to be carried out for the factorization of n. The run time given here is the actual run time of the RSA155 factoring effort and should not be confused with the estimates given in [37], which appeared around the same time; these estimates are 100 times too high [26]. The largest number factored using the SNFS is the 233-digit (and 773-bit) number $2^{773} + 1$, in November of 2000, in less than $17,000$ Mips-Years. These run times are only a fraction of the cost of a software DES key search, but the amount of memory needed by the NFS is several orders of magnitude larger.

Practical experience with the DLNFS is still limited. It is generally accepted that, for any b up to about 500, factoring b-bit integers takes about the same amount of time as computing discrete logarithms in $(b-x)$-bit fields, where x is a small constant around 20. For b going to infinity there is no distinction between the hardness of b-bit factoring and b-bit discrete logarithms. Below we do not present separate key size suggestions for TDL systems and we recommend using the RSA key size suggestions for TDL systems as well.

2.4.7 Special-purpose hardware data points. Special-purpose hardware devices are occasionally proposed for the most time consuming step of factoring algorithms such as the sieving step of the NFS, but no useful data points have been published. Recently, Adi Shamir proposed the TWINKLE opto-electronic sieving device [33, 21]. This device, if feasible at all, does not affect the asymptotic run time of the NFS, nor does it affect the matrix step.

Due to the complexity of the underlying factorization algorithms and the corresponding hardware design for any special-purpose hardware factoring device, it would be difficult to achieve parallelization at a reasonable cost and at a scale comparable to hardware attacks on the DES, but it may not be impossible. Also, by the time a special-purpose design could be operational it is conceivable that it would no longer be competitive due to new algorithmic insights and faster general-purpose processors. Given the current state of the art we consider it to be unlikely that special-purpose hardware will have a noticeable impact on the security of RSA moduli.

But we find it imprudent to ignore the possibility altogether, and warn against too strong a reliance on the belief that special-purpose attacks on RSA are impossible. To illustrate this, the quadratic sieve factoring method was implemented successfully on a Single-Instruction-Multiple-Data architecture [12]. A SIMD machine is by no means special-purpose hardware, but it could be relatively cheap compared to ordinary PCs.

2.4.8 Effectiveness of guessing. Obviously, key sizes for classical asymmetric systems have to be larger than 512 to obtain any security at all (where 512 is the size of the 'broken' RSA modulus RSA155; cf. 2.4.6). It may safely be assumed that breaking the system by guesswork is out of the question: it would require at least 254 correctly guessed bits for RSA or 512 bits for TDL. So, from this point of view, classical asymmetric systems seem to be more secure than symmetric key cryptosystems. For RSA there is more to this story, as shown in 2.4.9 below.

2.4.9 Incomplete attacks. Both the NFS and the DLNFS are effective only if run to completion. There is no chance that any results will be obtained early. RSA, however, can also be attacked by the Elliptic Curve Method (ECM). After a relatively small amount of work this method produces a factor with substantially higher probability than mere guesswork. To give an example, if one billion people were to attack a 512-bit RSA modulus, each by running the ECM for just one hour on their PC, then the probability that one of them would factor the modulus is more than 10%. For a 768-bit RSA modulus the probability of success of the same computational effort is about one in a million. Admittedly, this is a very low success probability for a tremendous effort – but the success probability is orders of magnitude larger than guessing, while the amount of work is of the same order of magnitude. No discrete logarithm equivalent of the ECM has been published. The details of our ECM run time predictions are beyond the scope of this article. See also Subsection 5.9.

2.4.10 Cryptanalytic progress. Classical asymmetric systems are the prime example of systems for which the effectiveness of cryptanalysis is steadily improving. Roughly speaking the effect of algorithmic improvements over the last 25 years turned out to be comparable to the effect of faster hardware; cf. Remark 4.3.1.2.

The current state of the art of factoring (and discrete logarithm) algorithms should not be interpreted as the culmination of many years of research but is just a snapshot of work in progress. It may be due to the relative complexity of the methods used that so many more or less independent improvements and refinements have been made and – without any doubt – will be made. We illustrate this point with a list of some of the developments since the early seventies, each of which had a substantial effect on the difficulty of factoring or computing discrete logarithms: continued fraction method, linear sieve, quadratic sieve, multiple polynomial variation, Gaussian integers, loosely coupled parallelization, multiple large primes, special number field sieve, structured Gaussian elimination, number field sieve, singular integers, lattice sieving, block Lanczos or conjugate gradient, sieving-based polynomial selection for the NFS, and, most recently, parallelized block Lanczos. We find it reasonable to assume that this trend of continuous algorithmic developments will continue in the years to come.

It has never been proved that breaking RSA is equivalent to factoring the RSA modulus. Indeed, for RSA there is evidence that the equivalence does not hold if the so-called public exponent (another part of the RSA public key) is small. We therefore introduce the explicit assumption that breaking RSA is equivalent to factoring the RSA modulus. Based on recent results in this area the public exponent for RSA must be sufficiently large. Values such as 3 and 17 can no longer be recommended, but commonly used values such as $2^{16}+1=65537$ still seem to be fine. If one prefers to stay on the safe side one may select an odd 32-bit or 64-bit public exponent at random.

Furthermore we restrict ourselves to TDL based protocols for which attacks are provably equivalent to either computing discrete logarithms or solving the Diffie-Hellman problem – the problem of finding g^{ab} given g^a and g^b for known g (but unknown a and b). There is strong evidence that the latter problem is equivalent to computing discrete logarithms. We explicitly exclude, however, TDL based protocols that rely on the so-called Decision Diffie-Hellman problem – the problem of distinguishing g^{ab} from g^c when g, g^a, g^b, g^{ab}, and g^c for a random c are given [18].

2.5 Subgroup Discrete Logarithm Systems

2.5.1 Description. Subgroup discrete logarithm (SDL) systems are like traditional discrete logarithm systems, except that g generates a relatively small, but sufficiently large, subgroup of the multiplicative group $\mathrm{GF}(p)^*$, an idea due to Schnorr. The size of the subgroup is prime and is indicated by q. The private key m is at least 1 and at most $q-1$. The security of SDL is based on the difficulty of computing discrete logarithms in a subgroup of the multiplicative group of a finite field. These can be computed if discrete logarithms in the full multiplicative group can be computed. Therefore, the security of an SDL system relies on the sizes of both q and p. Nevertheless, the size of an SDL key simply refers to the bit-length of the subgroup size q, where the field size p is given by the context. The actual number of bits required to store an SDL public key is substantially larger than the SDL key size q, since the public key contains p, g and y as well.

2.5.2 Wassenaar Arrangement. The maximum SDL field size allowed by the Wassenaar Arrangement is 512 bits – there is no maximum allowed key size. A popular subgroup size is 160 bits. That choice is used in the US Digital Signature Algorithm, with field sizes varying from 512 to 1024 bits.

2.5.3 Attacks. Methods that can be used to attack TDL systems also can be used to attack SDL systems. The field size p should therefore satisfy the same security requirements as in TDL systems. But the subgroup discrete logarithm problem can also be attacked directly by Pollard's rho method, which dates from 1978, and by Shanks' even older baby-step-giant-step method. These methods can be applied to any group, as long as the group elements allow a unique representation and the group law can be applied efficiently – unlike the DLNFS it does not rely on any special properties that group element representations may have. The expected run time of Pollard's rho method is exponential in the size of q, namely $1.25\sqrt{q}$ group operations, i.e., multiplications in $\mathrm{GF}(p)$. Its storage requirements are very small. Shanks' method needs about the same number of operations but needs storage for about $\sqrt{q}$ group elements.

Pollard's rho method can easily be parallelized over any number of processors, with very limited communication, resulting in a linear speedup [40]. This is another illustration of the power of parallelization and another reason to keep track of the computational power of the Internet. Furthermore, there is no post-processing involved in Pollard's rho (unlike the (DL)NFS, where after completion of the sieving step the cumbersome matrix step has to be carried out), although for the parallelized version substantial amounts of storage space should be available at a central location.

2.5.4 Data points. We have not been able to find any useful data about the effectiveness of an attack on SDL systems using the parallelized version of Pollard's rho method. Our figures below are based on an adaptation of data points for elliptic curve systems. This is described in detail in 4.2.5.

2.5.5 Effectiveness of guessing. As long as SDL keys are not shorter than the 112 bits permitted by the Wassenaar Arrangement for EC systems (cf. 2.6.2), guessing the private key requires guessing at least 112 bits, which may safely be assumed to be infeasible.

2.5.6 Incomplete attacks. The success probability of Pollard's rho method is, roughly speaking, proportional to the square of the fraction of the work performed, i.e., for any x, $0 \leq x \leq 1$, the chance is x^2 that the key is found after performing a fraction x of the expected $1.25\sqrt{q}$ group operations. So, doing ten percent of the work yields a one percent success rate.

2.5.7 Cryptanalytic progress. Since the invention of Pollard's rho method in 1978 no new results have been obtained that threaten SDL systems, with the exception of the efficient parallelization of Pollard's rho method in 1996. The only reasonable extrapolation of this rate of algorithmic progress is to assume that no substantial progress will be made. Progress would almost necessarily imply an entirely new approach and may instantaneously wipe out all practical SDL systems. The results in [27, 35] that, in a certain generic model of computation, Pollard's rho method is essentially the best one can do may be comforting in this context. It should be kept in mind, however, that the generic model does not apply to any practical situation that we are aware of, and that the possibility of a subexponential attack against SDL systems cannot be ruled out.

2.6 Elliptic Curve Systems

2.6.1 Description. Elliptic curve (EC) systems are like SDL systems, except that g generates a subgroup of the group of points on an elliptic curve E over a finite field $\mathrm{GF}(p)$, an idea independently due to Koblitz and Miller. The size q of the subgroup generated by g is prime and the private key m is in the range $[1, q-1]$.

The security of EC systems is based on the difficulty of computing discrete logarithms in the subgroup generated by g. These can be computed if discrete logarithms in the full group of points on an elliptic curve over a finite field can be computed. This problem is known as the ECDL problem. No better method to solve the ECDL problem is known than by solving the problem in all cyclic subgroups and by combining the results. The difficulty of the ECDL problem therefore depends on the size of the largest prime divisor of the order of the group of points of the curve (which is close to p). For that reason, p, E, and q are usually chosen such that the sizes of p and q are close. Thus, the security of EC systems relies on the size of q, and the size of an EC key refers to the bit-length of the subgroup size q. The actual number of bits required to store an EC public key may be substantially larger than the EC key size q, since the public key contains p, E, g, and y as well.

A description of the group of points on an elliptic curve over a finite field and how such points are represented or operated upon is beyond the scope of this article. Neither do we discuss how appropriate elliptic curves and finite fields can or should be selected.

2.6.2 Wassenaar Arrangement. The maximum EC key size allowed by the Wassenaar Arrangement is 112 bits, with unspecified field size. For prime fields a popular size is 160 bits both for the field size and the subgroup size. For non-prime fields an example of a commercially available choice is $p = 2^{163}$ with a 161-bit q.

2.6.3 Attacks. A DLNFS equivalent or other subexponential method to attack EC systems has never been published. The most efficient method published to attack EC systems is Pollard's parallelizable rho method, with an expected run time of $0.88\sqrt{q}$ group operations. This run time is exponential in the size of q. The expected number of iterations is a factor $\sqrt{2}$ smaller than for SDL systems, due to the result independently described in [15, 46]. If field inversions are properly handled, the average number of field multiplications per group operation is approximately 12 [14].

2.6.4 Software data points. Because p and q are assumed to be of the same order of magnitude the cost of the group operation is proportional to $(\log_2(q))^2$. Data about the effectiveness of an attack using Pollard's rho method can be found in [7]. From the estimates given there we derive that a 109-bit EC system with $p = 2^{109}$ should take about 18,000 years on a PC (or, equivalently, one year on 18,000 PCs) which is about 8 MMY. This computation is feasible on a large network of computers. It also follows from [7] that an attack on a 109-bit EC system with a prime p of about 109 bits should take about 2.2 MMY. This is an underestimate because it is based on primes of a special form and thus overly optimistic for general primes [14]. Nevertheless, it is used as the basis for extrapolations to estimate the effort required for software attacks on larger EC systems over prime fields (Subsection 1.3).

2.6.5 Special-purpose hardware data points. In 1996 an attack against a 120-bit EC system with $p = 2^{155}$ was sketched (and published 3 years later, cf. [40]) based on a special-purpose hardware design that achieves a 25 million fold parallelism, i.e., 330,000 special-purpose processor chips each running 75 independent Pollard rho processes. Building this machine would cost 10 million dollar and its run time would be about 32 days. The designers claim that an attacker can do better by using current silicon technology and that further optimization may be obtained from pipelining. On the other hand, in [7] it is mentioned that 131-bit EC systems 'are expected to be infeasible against realistic software and hardware attacks', where 131-bit systems over 131-bit fields are about 32 times harder to break than 120-bit systems over 155-bit fields. This shows that there is no clear distinction between which computations are considered to be feasible and which are not, and that drawing a conclusion from a cost evaluation is mostly a matter of personal taste and preferences (cf. 3.1.2). The pipelined design is further considered in Subsection 3.2.

2.6.6 Effectiveness of guessing. As long as EC keys are not shorter than the 112 bits permitted by the Wassenaar Arrangement, guessing the private key requires guessing at least 112 bits, which may safely be assumed to be infeasible.

2.6.7 Incomplete attacks. As with Pollard's rho attack against SDL systems, the chance is x^2 that the key is found after performing a fraction x of the expected $0.88\sqrt{q}$ group operations.

2.6.8 Cryptanalytic progress. With the exception of the result from [15, 46], no progress threatening the general ECDL problem has been made since the invention of Pollard's rho method in 1978 and its parallelization in 1996 (cf. 2.5.7). The key word here is 'general', because EC related cryptanalytic results are obtained quite regularly. So far these results mostly affect, or rather 'wipe out', special cases, e.g. curves for which the order of the group of points or the underlying finite field have special properties. For the non-specialized user this is hardly comforting: EC systems are relatively complicated and designers often apply special cases to avoid nasty implementation problems.

We make the explicit assumption that curves are picked at random, i.e., that special cases are not used, and that only curves over prime fields are used. Based on this assumption and the lack of cryptanalytic progress affecting such curves it is not unreasonable to assume that there will be no substantial progress in the years to come. It is, however, not hard to find researchers who find that EC systems have not been around long enough to fully trust them and that the rich mathematical structure of elliptic curves may still have some surprises in store. Others argue that the ECDL problem has been studied extensively, and that the lack of progress affecting well-chosen EC systems indicates that they are sufficiently secure. We do not want to take a position in this argument but note that some recent developments [17, 41] seem to support the former standpoint.

For the purposes of the present paper, we simply suggest two key sizes for EC systems: one based on 'no cryptanalytic progress' and one based on 'cryptanalytic progress at a rate comparable to RSA and TDL systems', the latter despite our fear or conviction that any new cryptanalytic insight against EC systems, such as a subexponential method, may prove to be fatal. The reader may then interpolate between the two types of extrapolations according to her own taste.

2.7 Cryptographic Hash Functions

2.7.1 Description. A cryptographic hash function is a function that maps an arbitrary length message to a fixed length 'hash' of the message, satisfying various properties that are beyond the scope of this article. The size of the hash function is the length in bits of the resulting hash. Examples of cryptographic hash function are MD4, MD5 (both of size 128), SHA-1, RIPEMD-160 (both of size 160), and, most recently, SHA-256 (of size 256).

2.7.2 Attacks. We assume that a successful attack against a cryptographic hash function consists of finding s and t with $s \neq t$ such that the hashes of s and t are the same. If such s and t cannot be found the hash function is called 'collision-resistant'. For hash functions that are only required to be 'target collision-resistant' (i.e., it is supposed to be infeasible to find an s that hashes to a given target hash value), the sizes may be halved assuming the hash function is properly used. Cryptographic hash functions can be attacked by the so-called birthday paradox attack. The number of hash function applications required by a successful attack is expected to be proportional to $2^{x/2}$, where x is the size of the hash function.

2.7.3 Software data points. In [4] 241, 345, 837, and 1016 Pentium cycles are reported for MD4, MD5, SHA-1, and RIPEMD-160, respectively. This compares to 360 to 500 cycles for the DES depending on fixed or variable keys, as reported in [2, 8] (cf. 2.2.4). Thus, the software speed of a hash function application as used by a birthday paradox attack is comparable to the software speed of a single DES block encryption.

2.7.4 Special-purpose hardware data points. Special-purpose hardware has been designed for several hash functions. We may assume that their speed is comparable to the speed of special-purpose exhaustive key search hardware.

2.7.5 Cryptanalytic progress. We assume the existence of a generic cryptographic hash function for which the birthday paradox attack is the best attack. If a proposed design allows a faster attack, we assume that it will no longer be used. We assume that an exhaustive key search attack on our generic symmetric key cryptosystem of key size b can be expected to take about the same time as a birthday paradox attack on our generic cryptographic hash function of size $2b$. Thus, a lower bound for the size of cryptographic hash functions follows by doubling the lower bound for the size of symmetric key cryptosystems. Because of this simple 'rule of thumb', sizes of cryptographic hash functions are not discussed in the sequel. If speeds differ, adjust accordingly.

3 The Model

3.1 Key points

In this subsection we present the four points on which the choice of cryptographic key sizes depends primarily:

1. Life span: the expected time the information needs to be protected.
2. Security margin: an acceptable degree of infeasibility of a successful attack.
3. Computing environment: the expected change in computational resources available to attackers.
4. Cryptanalysis: the expected developments in cryptanalysis.

Efficiency and storage considerations concerning the cryptographic keys may also influence the choice of key sizes, but since they are not directly security-related they are not discussed here.

3.1.1 Life span. In the table in the next section key sizes are suggested for the cryptosystems discussed in Section 2, depending on the expected life span of the cryptographic application. It is the user's responsibility to decide until what year the protection should be effective, or how the expected life span corresponds to popular security measures such as 'short-term', 'medium-term', or 'long-term' security. The user's decision may depend on the value of the data to be encrypted.

3.1.2 Security margin. A cryptosystem can be assumed to be secure only if it is considered to be sufficiently infeasible to mount a successful attack. Unfortunately, it is hard to quantify what precisely is meant by 'sufficiently infeasible' (cf. 2.6.5). One could, for instance, decide that a key size for a certain cryptosystem is secure for current applications if breaking it would be, say, 10^6 times harder than the largest key size that can currently be broken for that cryptosystem. There are several problems with this approach.

First of all, the choice 10^6 is rather arbitrary. Secondly, it is naïve to believe that the largest published key broken so far accurately represents the best that can currently be done (Remark 1.4.1). In the third place, for some of the cryptographic primitives considered here data may not be available (TDL, cf. 2.4.6, and SDL, cf. 2.5.4), or they may be outdated, thereby ruling out uniform application of this approach. Finally, the problem of any fixed security margin is that there are always users who prefer a different choice. We opt for a different approach by offering a flexible choice of security margin.

Definition I. The security margin s is defined as the year until which a user was willing to trust the DES.

The rationale for this definition of security margin is that the security offered by the DES is something most users can relate to, for instance because their company used the DES until a certain year. Furthermore, different choices of s allow us to satisfy different security needs. Another advantage of our choice of security margin is discussed in Remark 3.1.4.

The DES was introduced in 1977 and stipulated to be reviewed every five years. We therefore assume that the DES was at least sufficiently secure for commercial applications until 1982.

Default Setting I. Our default setting for s is $s = 1982$.

Our default setting for s assumes that in 1982 a computational effort of 0.5 MMY provided an adequate security margin for commercial DES applications against software attacks (cf. 2.2.4). As far as hardware attacks are concerned, the DES key searching [$50 million, 2 days, 1980]-hardware (cf. 2.2.5) was not a serious threat for commercial applications of the DES at least until 1982. We stress 'commercial applications' because, even for 1980 budgets, 50 million dollar and 2 days were by no means an insurmountable obstacle for certain organizations. Our default setting for s is further discussed below (Remark 3.1.8). Although all our results are based on the default setting $s = 1982$, they can easily be adapted to produce key size recommendations for any other reasonable value of s. In Subsection 4.4 it is indicated how this can be done.

The maximal value of a 'commercial application' of the DES, either back in 1982 or right now, is the value of the company encrypting the data. Thus, over time there is no intrinsic difference between the possible value of commercial applications of the DES. As businesses move online the number of commercial applications is increasing, but volume is not a security factor. A concept of 'value' is therefore not directly incorporated in our model, but value can be compared to the cost of an attack using our notion of 'cost equivalence' (Subsection 4.5 and the second to last column of Table 1). See also 3.1.1. We are grateful to an anonymous referee for suggesting us to clarify this point.

3.1.3 Remark on security margin. A particular choice for s does not imply that the DES is thought to be vulnerable from year s on [38]; it means that the user who picked s is willing to trust the DES until the year s. Of course, any responsible user maintains a comfortable margin between the moment until which she is willing to use a system and the moment when she believes the system to be vulnerable. It is baffling that anyone would seriously believe [38] that the DES was not actually broken until 1997, the year that it was publicly demonstrated (Remark 1.4.1 and 3.1.8).

3.1.4 Remark on security margin and incomplete attacks. It should be understood that our definition of security margin (Definition I) also takes into account the probability of success of incomplete attacks. Indeed, trusting the DES implies that one finds it to be sufficiently resistant to all types of potential attackers. That is, the whole spectrum between, on the one hand, attackers that search a fraction close to 1 of the key space and, on the other hand, attackers that search a fraction close to 0 of the key space. The former is assumed to be too expensive to carry out, and for the latter it is assumed that the probability of success is too low.

Note that the success probability of exhaustive key search is proportional to the fraction of the key space searched (cf. 2.2.7), but that for Pollard's rho method the probability of success is only proportional to the square of the fraction of the work performed (cf. 2.5.6 and 2.6.7). Therefore, an incomplete attack against EC or SDL systems has a smaller probability of success than a similarly incomplete attack against the DES. Thus, if EC or SDL key sizes may be expected to satisfy a certain security margin, they also offer resistance against incomplete attacks that is at least equivalent to the resistance offered by the DES. Because furthermore incomplete attacks against RSA and TDL systems cannot be expected to be successful at all (cf. 2.4.9 and Subsection 5.9), we conclude that the effect of incomplete attacks has effectively been taken care of in our model. See also Subsection 5.8.

3.1.5 Computing environment. To estimate how the computing power available to attackers may change over time we use Moore's law. Moore's law states that the density of components per integrated circuit doubles every 18 months. A widely accepted interpretation of this law is that the computing power per chip doubles every 18 months. There is some skepticism whether this law will, or even can, hold much longer because new technologies will eventually have to be developed to keep up with it. Therefore we allow the user to define the following slight variation of Moore's law that is less technology dependent.

Definition II. The variable $m > 0$ is defined as the number of months it takes on average for an expected two-fold processor speed-up and memory size increase.

Default Setting II. Our default setting for m is $m = 18$.

Definition III. The 0,1-valued variable t defines how m must be interpreted:
- If $t = 1$ the amount of computing power and random access memory (RAM) one gets for a dollar is expected to double every m months.
- If $t = 0$ the amount of computing power and RAM is expected to double every m months, irrespective of the price.

Default Setting III. Our default setting for t is $t = 1$.

Default Setting II corresponds to a popular interpretation of Moore's law. Combined with Default Setting III it leads to a less technology dependent version of Moore's law that may hold even if Moore's traditional law no longer holds because of technological limitations.

So far Default Settings II and III seem to be sufficiently accurate: every 18 months the amount of computing power and RAM one gets for a dollar doubles. With these default settings it follows that for the same cost one expects to get a factor of $2^{10*12/18} \approx 100$ more computing power and fast memory every 10 years, either in software on multipurpose chips (PCs) or using special-purpose hardware.

To illustrate this, it is not unreasonable to assume that a cheaper and slower version of the DES key searching [\$50 million, 2 days, 1980]-hardware (cf. 2.2.5) would be [\$1 million, 100 days, 1980]-hardware, i.e., 50 times less hardware and therefore 50 times slower. With Default Settings II and III the latter hardware may be expected to be $2^{8.7}$ times faster in 1993, since there are $12*13 = 18*8.66$ months between 1980 and 1993. Since $2^{8.7} \approx 406$ and 100/406 days is about 6 hours, this would result in [\$1 million, 6 hours, 1993]-hardware which is indeed close to Wiener's [\$1 million, 3.5 hours, 1993]-hardware design (cf. 2.2.5).

On the other hand, further extrapolation suggests [\$1 million, 0.6 hours, 1998]-hardware for DES key searching. That is approximately equivalent to [\$130,000, 4.6 hours, 1998]-hardware, and thereby about 24 times faster than the [\$130,000, 112 hours, 1998]-hardware that was actually built in 1998 [19]. According to [20] this anomaly is due to the fact that building the 130,000 dollar machine was, relatively speaking, a small scale enterprise where every doubling of the budget would have quadrupled the performance. Obviously this non-linear improvement applies only as long as the device is relatively small.

If $t = 0$ it is assumed that the computational resources available to attackers double every m months, so their budgets are not immediately relevant. If $t = 1$ the effect of budget increases and inflation have to be taken into account. This leads to the following definition.

Definition IV. The variable $b > 0$ is defined as the number of years it takes on average for an expected two-fold increase of budget.

Default Setting IV. Our default setting for b is $b = 10$.

The US Gross National Product shows a trend of doubling every ten years: \$1630 billion in 1975, \$4180 billion in 1985, and \$7269 billion in 1995, where each figure is given in contemporary dollars. Default Setting IV leads to the assumption that the budgets of organizations – including the ones breaking cryptographic keys – double every ten years, measured in contemporary dollars.

Note that with Default Setting IV the effect of budget increases is very small; cf. Remark 1.1.1.

3.1.6 Combination of Defaults Settings I, II, III and IV. If in 1982 an amount of computing power of 0.5 MMY is assumed to be infeasible to invest in an attack on a commercial cryptographic application, then $\sim 100 * 2 * 0.5 = 100$ MMY is infeasible in 1992. Furthermore, $\sim 200 * 100 = 2 * 10^4$ MMY is infeasible in 2002, and $4 * 10^6$ MMY is infeasible in 2012. These figures agree with Odlyzko's estimates based on computing power that may be available on the Internet [29]. Our estimates are, however, obtained in an entirely different fashion.

3.1.7 Cryptanalysis. It is impossible to say what cryptanalytic developments will take place, or have already taken place surreptitiously. We find it reasonable to assume that the pace of (published) future cryptanalytic findings and their impact are not going to vary dramatically compared to what we have seen from 1970 until 1999, as described in 2.2.8, 2.4.10, 2.5.7, 2.6.8, and 2.7.5. Nevertheless, we allow some flexibility in the choice of expected cryptanalytic progress.

As indicated in 2.2.8 and 2.7.5 we assume that there will be no cryptanalytic developments affecting symmetric key cryptosystems or hash functions: if there is progress we assume that the affected system or function is replaced by a system or function that is not affected.

It follows from 2.4.10 and 2.6.8 that we have to take a more flexible approach to asymmetric cryptosystems.

Definition V. The number $r > 0$ is defined as the number of months it is expected to take on average for cryptanalytic developments affecting classical asymmetric systems to become twice as effective, i.e., r months from now we may expect that attacking the same classical asymmetric system costs half the computational effort it costs today.

Default Setting V. Our default setting for r is $r = 18$.

Default Setting V corresponds closely to cryptanalytic progress affecting classical asymmetric systems during the past 25 years, as mentioned in 2.4.10; cf. Remark 4.3.1.2.

Definition VI. The number $c \geq 0$ is defined as the number of months it is expected to take on average for cryptanalytic developments affecting EC systems (chosen as indicated in 2.6.8) to become twice as effective, unless $c = 0$ in which case no EC cryptanalytic progress is expected.

Default Setting VI. Our default setting for c is $c = 0$.

Default Setting VI corresponds with the fact that there has not been substantial cryptanalytic progress affecting EC systems, assuming the system has been properly chosen as indicated in 2.6.8.

Since there has been no cryptanalytic progress affecting SDL systems since the invention of Pollard's rho method (and its parallelization) other than progress affecting the full multiplicative group (cf. 2.5.7), we assume no cryptanalytic progress affecting SDL systems. Although for EC systems the situation is similar (i.e., for properly chosen parameters no progress to speak of over the last 10 or so years) we chose to allow progress for EC cryptanalysis (with Default Setting VI 'no progress') because, unlike SDL systems, it is not hard to find researchers who find it not unlikely that there will be EC cryptanalytic progress. We do not find it realistic to exclude the possibility of cryptanalytic progress affecting classical asymmetric systems, so r is assumed to be strictly positive.

3.1.8 Remark on default settings. We do not expect that everyone agrees with our default settings. In particular Default Setting I is debatable. Note, however, that it does not assume that the DES was unbreakable in 1977 or 1982. It assumes that the DES offered enough security for commercial applications, not that well-funded government agencies were unable to break it back in 1977. In this context it may be entertaining to mention that Mike Wiener, after presenting his [$1 million, 3.5 hours, 1993]-hardware design at a cryptography conference, was told that he had done a nice piece of work and he was offered a similar machine at only 85% of the cost – with the catch that it was 5 years old [45]. In any case, anyone who feels that our default 1982 infeasibility assumption is too weak or too strong can still use the key size recommendations that result from Default Setting I, i.e., $s = 1982$. In Subsection 4.4 it is explained how this may be done.

Neither do we expect that everyone agrees with Default Settings II, III, and IV. Some argue that Moore's law cannot hold much longer, others argue that it is well understood that Moore's law is very likely to die around 2012 or so, and still others [20] find that for big machines Moore's law is too pessimistic. Default Settings II, III, and IV thus represent a reasonable compromise, in particular because they allow a technology independent interpretation of Moore's law – even if technology gets worse, if that were possible, acquiring computing power may become cheaper.

3.2 Software Versus Special-Purpose Hardware Attacks

The proposed key sizes in the next section are obtained by combining Default Settings I-VI with the software based Mips-Years data points from Section 2. This implies that all extrapolations are based on 'software only' attacks and result in computationally equivalent key sizes (Subsection 1.4). One may object that this does not take special-purpose hardware attacks into account. In this subsection we discuss to what extent this is a reasonable decision, and how our results should be interpreted to take special-purpose hardware attacks into account as well.

3.2.1 Symmetric key systems. In 1980 the DES could either be broken at the cost of 0.5 MMY (cf. 2.2.4), or using [$50 million, 2 days, 1980]-hardware (cf. 2.2.5). In 3.1.5 we have shown that this is consistent with Default Setting II and Wiener's 1993 design. It follows from this consistency that the 1982 relation between software and special-purpose hardware attacks on the DES has not changed. In particular, if one assumes that the DES was sufficiently resistant against a special-purpose hardware attack in 1982, the same holds for the symmetric key sizes suggested for the future, even though they are based on extrapolations of 'software only' attacks. We note that our estimates and the resulting cost of special hardware designs for exhaustive key search are consistent with the estimates given in [3, 5].

Furthermore, it seems reasonable to assume that a DES attack of one MMY is comparable to an attack by [$10 million, 20 days, 1980]-hardware or, using Default Setting II, [$\$200 * 10^6/2^{10.66}$ =$125,000, 1 day, 1996]-hardware.

3.2.2 EC systems. The cost of a software attack on a 109-bit EC system with $p = 2^{109}$ was estimated as 8 MMY (cf. 2.6.4), so that attacking a 120-bit EC system with $p = 2^{155}$ should take about $(2^{(120-109)/2}) * (155/109)^2 \approx 91$ times as many Mips-Years, i.e., about 730 MMY. The [$10 million, 32 days, 1996]-hardware design attacking a 120-bit EC system with $p = 2^{155}$ (cf. 2.6.5) should thus be more or less comparable to 730 MMY. However, the designers of the hardware device remark that their design was based on 1992 (or even older) technology which can be improved by using 1996 technology. So, by Default Setting II, the 'upgraded' [$10 million, 32 days, 1996]-hardware design could be more or less comparable with $730 * 2^{(1996-1992)/1.5} \approx 4600$ MMY. It follows that an EC attack of one MMY is comparable to [$70,000, 1 day, 1996]-hardware.

With 3.2.1 we find that one MMY is equivalent to [$70,000 to $125,000, 1 day, 1996]-hardware depending on an EC or a DES attack. Because of the consistency of these conversions it is tempting to suggest that one MMY is approximately equivalent to [$100,000, 1 day, 1996]-hardware; more generally, that one MMY would be equivalent to [$\$10^5/2^{(y-1996)/1.5}$, 1 day, y]-hardware in year y. That is, one MMY is equivalent to [$25,000, 1 day, 1999]-hardware. This conversion formula would allow us to go back and forth between software and special-purpose hardware attacks, and make our entire model applicable to hardware attacks as well.

In our opinion the consistency between the two conversions is a mere coincidence without much practical merit. In the first place, the estimate holds only for relatively simple minded DES or EC cracking devices for elliptic curves over non-prime fields (i.e., those with $p = 2^k$), not for elliptic curves over prime fields and certainly not for full-blown PCs. For prime fields the hardware would be considerably slower, whereas in software EC systems over prime fields can be attacked faster than those over non-prime fields (cf. 2.6.4). Thus, for special-purpose hardware attacks on EC systems over prime fields the above consistency no longer holds.

In the second place, according to [44], the pipelined version of the EC-attacking special-purpose hardware referred to above would be about 7 times faster, which means that also for special-purpose hardware attacks on EC systems over non-prime fields the consistency between DES and EC attacks is lost. Also according to [44], the prime field version of the pipelined device would be about 2^4 to 2^5 times slower than the non-prime field version. It should be noted that the details of the pipelined device have never been published (and most likely will never be published [45]).

As mentioned in 2.6.8, we consider only EC systems that use randomly selected curves over prime fields. Therefore we may base our recommendations on 'software only' attacks, if we use the software based data point that a 109-bit EC system can be attacked in 2.2 MMY (cf. 2.6.4). This can be seen as follows. The 2.2 MMY underestimates the true cost, and is lower than the 8 MMY cost to attack the non-prime field of equivalent size. The latter can be done using non-pipelined special-purpose hardware in a way that is more or less consistent with our DES infeasibility assumption, as argued above. For special-purpose hardware a non-prime field can be attacked faster than a prime field of equivalent size, so if we use the naïve DES-consistent hardware conversion, then the hypothetical special-purpose hardware that follows from extrapolation of the 2.2 MMY figure to larger prime fields substantially underestimates the true hardware cost. That means that the resulting key sizes are going to be too large, which is acceptable since we are deriving lower bounds for key sizes (Subsection 1.3).

The more realistic prime field equivalent of the non-DES-consistent pipelined device for non-prime fields is, based on the figures given above, at least $2^4 * 8/(2.2 * 7) > 8$ times slower than our hypothetical hardware. This implies that the more realistic hardware would lead to lower key sizes than the hypothetical hardware. Thus, it is acceptable to stick to the latter (Subsection 1.3). It follows that, if one assumes that the DES was sufficiently resistant against a special-purpose hardware attack in the year indicated by the security margin s as in Definition I, the same holds for the EC key sizes suggested for the future, even though they are based on extrapolations of 'software only' attacks.

3.2.3 SDL systems. The same holds for SDL systems because our analysis of SDL key sizes is based on the EC analysis as described in 4.2.5 below.

3.2.4 Classical asymmetric systems. For classical asymmetric systems we do not consider special-purpose hardware attacks, as argued in 2.4.7. The issue of software attacks on classical asymmetric systems versus special-purpose hardware attacks on other cryptosystems is discussed in 3.2.5 below.

3.2.5 Cost comparison of software and special-purpose hardware attacks. Our key size recommendations below are computationally equivalent (Subsection 1.4) and, as argued in 3.2.2, they all offer security at least equivalent to the 1982 security of the DES (based on Default Setting I), both against software and special-purpose hardware attacks. That does not necessarily imply that the key sizes for the various cryptosystems are also cost equivalent (Subsection 1.4), because the equipment costs of the 1982 software and special-purpose hardware attacks on the DES are not necessarily equal either.

One point of view is that accessing the hardware required for software attacks is, or ultimately will be, essentially for free. This is supported by all Internet based cryptosystem attacks so far and other large computational Internet projects such as SETI. Adoption of this simple-minded rule would make computational and cost equivalence identical, which is certainly not generally acceptable [44]. Unfortunately, a precise equipment cost comparison defies exact analysis, primarily because no precise 'cost of a PC' can be pinpointed, but also because a truly complete analysis has never been carried out for the pipelined EC attacking design from [44, 45]. As pointed out in Subsection 1.4 this is one of the reasons that we decided to use computational equivalence as the basis for our results. Nevertheless, we sketch how an analysis based on cost equivalence could be carried out.

Definition VII. The number $P > 0$ is defined as the price in US dollars of a stripped down PC with at least 64 Megabytes of RAM. By a stripped down PC we mean a 450 MHz Pentium II processor, a mother-board, and communications hardware.

Default Setting VII. Our default setting for P is $P = 100$.

According to newspaper advertisements fully equipped PCs can be bought for prices varying from 0 to 450 dollars. The 'free' machines support the point of view that software attacks are for free. Default Setting VII assumes that one does not want to deal with the strings attached to the free machines and is based on wholesale extrapolation of current prices. Our choice disregards the possibility of a much larger quantity discount one should be able to negotiate for a very large order.

Assuming Default Setting VII, one million software Mips-Years is equivalent to [\$$365 * 10^6 * 100/450$ =\$81 million, 1 day, 1999]-hardware. Compared to the exhaustive DES key search [\$125,000, 1 day, 1996] $\approx$ [\$31,250, 1 day, 1999]-hardware from 3.2.1, a software Mips-Year is thus about

$$\frac{365 * 10^6 * \$100}{450 * \$31,250} \approx 100 * 26 = 26 * P$$

times more expensive. Compared to the pipelined [\$70,000/7, 1 day, 1996] $\approx$ [\$2600, 1 day, 1999]-hardware to attack EC systems over non-prime fields referred to in 3.2.2, a software Mips-Year is more than $3 * 10^4 = 300 * P$ times more expensive, but at most about $2 * 10^3 = 20 * P$ times more expensive than the prime field version of the pipelined design.

It follows that for our purposes software Mips-Years are at most $26 * P$ times more expensive than Mips-Years produced by special-purpose hardware. In Subsection 4.5 it is shown how this factor $26 * P$ can be used to derive cost-equivalent key sizes from the computationally equivalent ones.

Note that the factor $26 * P$ should be taken with a large grain of salt. Its scientific merit is in our opinion questionable because it is based on the presumed infeasibility of special-purpose hardware attacks on RSA (cf. 2.4.7 and the pipelined design in [12]).

3.3 Memory considerations

The processors contributing to a parallelized exhaustive key search do not require a substantial amount of memory. This is also the case for the processors involved in a parallelized attack using Pollard's rho method against SDL or EC systems. Although for the parallelized version of Pollard's rho method substantial storage space has to be available at a central location, we assume that storage requirements do not have to be taken into account to estimate SDL and EC system key sizes.

For parallelized NFS attacks against classical asymmetric systems, however, each of the contributing processors needs a relatively large amount of RAM of speed compatible with the processor speed. Until recently memory access times and not processor speeds determined the effective run times of the standard type of sieving used: a clock rate twice as fast would often result in only marginally faster sieving. This is because standard sieving requires very little computation and consists almost exclusively of constant updates of more or less random locations in a large chunk of memory, and thus does not allow efficient caching. Straightforward extrapolation of run times to faster processors was therefore impossible.

Newer generations of processors with larger memories allow efficient implementation of NFS lattice sieving, which is, compared to standard sieving, a relatively compute-intensive method. Its efficiency depends mostly on the processor speed, and memory access time hardly matters. To illustrate this, we observed that the speed of NFS lattice sieving on Pentium processors grows strictly linearly with the processor speed, with an interesting larger-than-expected speed-up when moving from Pentium I to Pentium II processors: an average sieving step operation for the result presented in [6] takes 15.8 seconds on a 133MHz Pentium I, 12.7 seconds on a 166MHz Pentium I, 5.34 seconds on a 300MHz Pentium II, and 3.61 seconds on a 450 MHz Pentium II. Here all processors execute the same binary that uses about 48MB of their about 200MB RAMs.

As a consequence, there does not seem to be any reason not to extrapolate NFS run times in the standard fashion. At worst the extrapolated sieving times are lower than the actual ones, making factoring look easier than it actually is, and thereby making the RSA key size recommendations somewhat larger (Subsection 1.3).

The amount of memory required by the NFS grows with the squareroot of the run time. Since m (Definition II) is assumed to be strictly positive, available RAM grows linearly with the processor speed. Thus, since current processors have in general enough memory for problems that are currently solved using the NFS, we may assume that future processors have more than enough memory to tackle future problems. Combining these observations we conclude that the

NFS memory requirements do not explicitly have to be taken into account when extrapolating NFS run times to future processors and larger RSA moduli or field sizes.

4 Lower Bound Estimates for Cryptographic Key Sizes

4.1 Introduction

In this section we present formulas that can be used to derive lower bounds for cryptographic key sizes. In Subsections 4.2 to 4.5 we concentrate on key size recommendations that can be expected to offer an acceptable security margin until a year specified by the user. In Subsection 4.6 we describe how key size recommendations can be derived that can be expected to offer a level of security that is currently (i.e., at the time of writing of this article) at least equivalent to a symmetric key size specified by the user.

The recommendations in this paper are based on the default settings. To use other settings, refer to Subsection 4.4, or use the Java applet provided by Kai Puolamäki [30].

4.1.1 Remark on precision. Our 'progress' parameters m, r, and c (from Definitions II, V, and VI, respectively) are measured in months, because that corresponds to the way Moore's law is often formulated. Below, however, time is measured in whole years, as is the security margin s (from Definition I). In principle we could adopt a much finer granularity and, for instance, use the more precise data point that a 511.7-bit RSA modulus was broken in 1999.64. In our opinion that would give a misleading sense of precision that would be inappropriate for an article of this sort.

One may object – and we would not disagree – that key size recommendations should not be given on a year-by-year basis, as we do below. In our experience, however, the uncertainties inherent in this type of 'back-of-the-envelope' engineering are not appreciated by all intended users: if a year is not specified in our tables, they may end up using an interpolated value, instead of simply using the next year up. If calculated properly, there is nothing wrong with interpolated values (the curves are convex, and we are only interested in lower bounds), but it is more convenient, and safer, to simply provide values for all years.

Another point of criticism is that we do not round the values resulting from our formulas, thereby failing to reflect that they are crude estimates at best. Thus, if according to some formula, a key size of 1537 bits is believed to be adequate for a certain year (and certain parameter settings), then we print the value 1537 in our tables, and not 1500, 1536, 1568, or 1600. We wholeheartedly agree that something like 1537 gives a misleading sense of precision, and of course we considered rounding values, but we decided not to do so for a couple of reasons. First of all, we would always have to round up, but we would have to use different granularities for RSA and TDL recommendations compared to those for symmetric key, SDL, or ECC systems. Without any doubt the resulting relatively long RSA keys would be interpreted as the authors' bias against RSA and in favor of ECC, something we want to avoid at all cost. Secondly, and this may seem strange to many readers, there is an amazingly common belief, or misunderstanding, that RSA keys must have a length that is divisible by a non-trivial power of 2 such as 32, 64, or 128. We do not want to fuel this misconception by recommending RSA key sizes that are all 0 modulo 32 or even 10, or that show any other pattern that can (and will) be misunderstood. Thus, rounding is fine, but the user will have to do it – we just provide the bare, unbiased numbers.

We are grateful to an anonymous referee for bringing up this subject once again. We hope these paragraphs clarify our opinions and decisions.

4.2 Key size formulas for a given year

4.2.1 Infeasible number of Mips-Years IMY. Suppose that key sizes have to be determined that achieve at least a specified security margin until year y. Breaking the DES takes $5 * 10^5$ Mips-Years (cf. 2.2.4). This amount of computation offered an acceptable level of security in the year s (Definition I in 3.1.2). Based on Definitions I-IV in 3.1.2 and 3.1.5 it follows that in year y, i.e., $y - s$ years later, an amount of computation of

$$IMY(y) = 5 * 10^5 * 2^{12(y-s)/m} * 2^{t(y-s)/b} \text{ Mips-Years}$$

offers an acceptable level of security. Here $IMY(y)$ stands for 'Infeasible number of Mips-Years for year y'. The factor $2^{12(y-s)/m}$ is due to the expected processor speed-up in the period from year s to year y (Definitions I and II in 3.1.2 and 3.1.5), and the factor $2^{t(y-s)/b}$ reflects the expected increase in the budget available to an attacker (Definitions I, III, and IV in 3.1.2 and 3.1.5). The resulting value $IMY(y)$ is used to derive key sizes that offer an acceptable level of security until year y, for all cryptographic primitives considered in Section 2.

4.2.2 Symmetric key systems. For symmetric key cryptosystems we introduce the possibility that the block-encryption speed of the symmetric key system to be used is different from the block-encryption speed of the DES.

Definition VIII. The variable $v > 0$ is defined as the ratio of the number of cycles required for a single block encryption using the DES and the symmetric key system the user wishes to use.

Default Setting VIII. Our default setting for v is $v = 1$.

Because the symmetric key system to be used is v times slower than the DES, attacking it goes v times slower as well. It follows that if the symmetric key system is used with a key d of at least

$$56 + \log_2(IMY(y)/(5 * 10^5 * v)) = 56 + (y - s)(12/m + t/b) - \log_2(v) \text{ bits},$$

with $IMY(y)$ as in 4.2.1, then the security offered by the symmetric key system until year y is at least computationally and cost equivalent (cf. 3.2.1) to the security offered by the DES in year s. Here we use that the DES has a 56-bit key (cf. 2.2.1), that it can be attacked in $5*10^5$ Mips-Years (cf. 2.2.4), and that there is no faster attack method than exhaustive search (cf. 2.2.8).

4.2.3 Classical asymmetric systems. For classical asymmetric systems we use the asymptotic run time $L[n]$ of the NFS (omitting the $o(1)$) as defined in 2.4.4 combined with the data point that a 512-bit key was broken in 1999 at the cost of less than 10^4 Mips-Years (cf. 2.4.6). Furthermore, we expect cryptanalytic progress by a factor $2^{12(y-1999)/r}$ compared to the state of the art in 1999, the year of the data point (cf. 2.4.10 and Definition V in 3.1.7). It follows that if the classical asymmetric key size k is chosen such that

$$\frac{L[2^k]}{IMY(y)*2^{12(y-1999)/r}} \geq \frac{L[2^{512}]}{10^4},$$

then the security offered by classical asymmetric systems until year y is at least computationally equivalent to the security offered by the DES in year s. If, on the other hand, the classical asymmetric key size k' is chosen such that

$$\frac{L[2^{k'}]}{IMY(y)*2^{12(y-1999)/r}} \geq \frac{L[2^{512}]}{10^4*26*P},$$

then the security offered by classical asymmetric systems until year y is at least cost equivalent to the security offered by the DES in year s (Definition VII in 3.2.5). The factor $26*P$ is explained in 3.2.5.

Because the data point used slightly overestimates the cost of factoring a 512-bit key and because we omit the $o(1)$, the difficulty of breaking classical asymmetric systems is overestimated (cf. 2.4.4), i.e., the classical asymmetric key sizes should be slightly larger than given in Table 1. We did not attempt to correct this, because the effect is minor and may disappear if the RSA key sizes given in Table 1 are rounded in a reasonable way (Remark 4.1.1).

4.2.4 EC systems. For EC systems we use the expected growth rate of the number of group operations required by Pollard's rho method (cf. 2.6.3), the expected growth of the cost of the group operations (cf. 2.6.4), and the optimistic estimate that a 109-bit EC system can be broken in 2.2 MMY (cf. 2.6.4). Furthermore, if $c > 0$ (Definition VI in 3.1.7), we expect cryptanalytic progress by a factor $2^{12(y-1999)/c}$ compared to the state of the art in 1999 (the year of the data point). We set $C = 1$ if $c = 0$ and $C = 2^{12(y-1999)/c}$ otherwise. It follows that if the EC key size u is chosen such that

$$\frac{2^{u/2}*u^2}{IMY(y)*C} \geq \frac{2^{109/2}*109^2}{2.2*10^6},$$

then the security offered by EC systems until year y is at least computationally and cost equivalent (cf. 3.2.2) to the security offered by the DES in year s. The factors u^2 and 109^2 account for the relative speed of the arithmetic operations to be performed by Pollard's rho method.

4.2.5 SDL systems. For SDL systems we use finite field size $\bar{k}$ with $\bar{k}$ either equal to k or k' as in 4.2.3 (cf. 2.4.6, 2.5.1, and 2.5.3). Because no suitable SDL data points are available (cf. 2.5.4) we estimate that arithmetic operations in a $\bar{k}$-bit finite field are $\bar{k}^2/(109^2 * 9)$ times more expensive than arithmetic operations in an elliptic curve group over a 109-bit finite field (where the '9' underestimates the number of field multiplications required for an EC operation, estimated as 12 in 2.6.3). Since for SDL $\sqrt{2}$ more iterations in Pollard's rho method may be expected than for EC systems, it follows that if the subgroup size z satisfies

$$\frac{2^{z/2} * \bar{k}^2}{IMY(y)} \geq \frac{2^{109/2} * 109^2 * 9}{\sqrt{2} * 2.2 * 10^6}$$

and the finite field size is at least $\bar{k}$, then the security offered by SDL systems until year y is at least equivalent to the security offered by the DES in year s: computationally equivalent if $\bar{k} = k$ and cost equivalent if $\bar{k} = k'$ with k and k' as in 4.2.3. Note that the above expression for z is equivalent to

$$z \geq 109 + 2\log_2\left(\frac{IMY(y) * 109^2 * 9}{\bar{k}^2 * \sqrt{2} * 2.2 * 10^6}\right).$$

The resulting sizes are too large because the 2.2 MMY estimate is on the low side. This optimism is to a small extent corrected by the optimistic choice of 9 field multiplications (where 12 or 13 would be more accurate [14]). It follows from a straightforward analysis that the subgroup size resulting from the above formula is of the required difficulty, independent of the EC data point, if a multiplication in a field of size $\bar{k}$ takes about $\bar{k}^2/69$ Pentium clock cycles. According to our own experiments with reasonably fast but non-optimized software a field multiplication can be done in $\bar{k}^2/24$ Pentium clock cycles, so that the subgroup sizes resulting from the EC-based data point are at most two bits too large (Subsection 1.3).

4.3 Lower bounds for computationally equivalent key sizes

For years ranging from 1982 to 2050 and for Default Settings I-VIII the computationally equivalent key size recommendations resulting (Remark 4.1.1) from the formulas given in Subsection 4.2 are given in Table 1. Furthermore, Table 1 contains key size recommendations for $c = 18$, i.e., cryptanalytic progress affecting EC systems comparable to Default Setting V for the cryptanalytic progress affecting classical asymmetric systems. For cost-equivalent key size recommendations see Subsection 4.5.

4.3.1 Remarks on the computation of Table 1.

1. Strictly speaking the data for years before 1999 do not make sense for the 'EC with $c = 18$' column, because we already know that for random curves over prime fields such progress did not occur before 1999. Nevertheless, these data can be found in Table 1 as well, in italics. It is described in Subsection 4.4 in what circumstances these data, and the other data in italics, may be used.

2. The data in Table 1 do not change significantly if the '512-bit, 10^4 Mips-Years, 1999' data point is replaced by, for instance, '333-bit, 30 Mips-Years, 1988' (the first 100-digit factorization) or '429-bit, 5000 Mips-Years, 1994' (the factorization of the RSA-Challenge; 5000 Mips-Years overestimates the time it took to break the RSA-Challenge despite the remarks made in [37]). This validates our default setting for r for cryptanalytic progress affecting classical asymmetric systems, cf. 2.4.10 and 3.1.7.

4.3.2 Using Table 1. Assuming one agrees with Default Settings I-VII, Table 1 can be used as follows. Suppose one is developing a commercial application in the year 2000 in which the confidentiality or integrity of the electronic information has to be guaranteed for 20 years, i.e., until the year 2020. Looking at the row for the year 2020 in Table 1, one finds that an amount of computing of $2.94 * 10^{14}$ Mips-Years in the year 2020 may be considered to be as infeasible as $5 * 10^5$ Mips-Years was in 1982 (cf. 2.2.4). Security computationally equivalent (Subsection 1.4) to that offered by the DES in 1982 is obtained by using in the year 2020 (while keeping Remark 4.1.1 in mind):

- Symmetric keys of at least 86 bits, and hash functions of at least 172 bits;
- RSA moduli of at least 1881 bits; the meaning of the '1472' given in the second entry of the same column is explained in Subsection 4.5.
- Subgroup discrete logarithm systems with subgroups of at least 151 bits with finite fields of at least 1881 bits. Thus, for an SDL system such as XTR it follows that $\log_2(q) \approx 151$ and $6 * \log_2(p) \approx 1881$ [23].
- Elliptic curve systems over prime fields of at least 161 bits if one is confident that no cryptanalytic progress will take place, and at least 188 bits if one prefers to be more careful.

If finite fields are used in SDL or EC systems that allow significantly faster arithmetic operations than suggested by our estimates, the data in Table 1 can still be used: if the field arithmetic goes x times faster, keys should be roughly $2 * \log_2(x)$ bits larger than indicated in Table 1. As noted above, however, the field arithmetic is already assumed to be quite fast. Similarly, if one does not agree that the data point used for EC systems underestimates the actual cost and that we overestimated the cost by a factor x, i.e., that the 2.2 MMY to attack 109-bit EC systems (cf. 2.6.4) should be only $2.2/x$ MMY, add roughly $2 * \log_2(x)$ bits to the suggested EC key sizes.

Note that it does not follow from Table 1 or the default settings that 1024-bit RSA keys will be safe only until 2002 [38]. It follows from Table 1 that until the year 2002, RSA keys of 1024 bits can be expected to offer security computationally equivalent to the DES in 1982. In this context, see also Remarks 1.4.1 and 3.1.3.

Table 1. Lower bounds for computationally equivalent key sizes, assuming $s = 1982$, $m = 18$, $t = 1$, $b = 10$, $r = 18$, $c = 0$ and $c = 18$, $v = 1$.

Year	Symmetric Key Size	Classical Asymmetric Key Size and SDL Field Size		SDL Key Size	Elliptic Curve Key Size $c = 0$	Elliptic Curve Key Size $c = 18$	Infeasible number of Mips-Years	Lower bound for hardware cost in US$ for a 1 day attack (cf. 4.5)	Corresponding number of years on a 450MHz Pentium II PC
1982	56	417	288	102	105	*85*	$5.00 * 10^{5}$	$3.98 * 10^{7}$	$1.11 * 10^{3}$
1984	58	463	320	105	108	*89*	$1.45 * 10^{6}$	$4.57 * 10^{7}$	$3.22 * 10^{3}$
1986	60	513	352	107	111	*96*	$4.19 * 10^{6}$	$5.25 * 10^{7}$	$9.31 * 10^{3}$
1988	61	566	384	109	114	*101*	$1.21 * 10^{7}$	$6.04 * 10^{7}$	$2.69 * 10^{4}$
1990	63	622	448	112	117	*106*	$3.51 * 10^{7}$	$6.93 * 10^{7}$	$7.80 * 10^{4}$
1991	63	652	448	113	119	*109*	$5.97 * 10^{7}$	$7.43 * 10^{7}$	$1.33 * 10^{5}$
1992	64	682	480	114	120	*112*	$1.02 * 10^{8}$	$7.96 * 10^{7}$	$2.26 * 10^{5}$
1993	65	713	512	116	121	*114*	$1.73 * 10^{8}$	$8.54 * 10^{7}$	$3.84 * 10^{5}$
1994	66	744	544	117	123	*117*	$2.94 * 10^{8}$	$9.15 * 10^{7}$	$6.53 * 10^{5}$
1995	66	777	544	118	124	*121*	$5.00 * 10^{8}$	$9.81 * 10^{7}$	$1.11 * 10^{6}$
1996	67	810	576	120	126	*122*	$8.51 * 10^{8}$	$1.05 * 10^{8}$	$1.89 * 10^{6}$
1997	68	844	608	121	127	*125*	$1.45 * 10^{9}$	$1.13 * 10^{8}$	$3.22 * 10^{6}$
1998	69	879	640	122	129	*129*	$2.46 * 10^{9}$	$1.21 * 10^{8}$	$5.48 * 10^{6}$
1999	70	915	672	123	130	130	$4.19 * 10^{9}$	$1.29 * 10^{8}$	$9.31 * 10^{6}$
2000	70	952	704	125	132	132	$7.13 * 10^{9}$	$1.39 * 10^{8}$	$1.58 * 10^{7}$
2001	71	990	736	126	133	135	$1.21 * 10^{10}$	$1.49 * 10^{8}$	$2.70 * 10^{7}$
2002	72	1028	768	127	135	139	$2.06 * 10^{10}$	$1.59 * 10^{8}$	$4.59 * 10^{7}$
2003	73	1068	800	129	136	140	$3.51 * 10^{10}$	$1.71 * 10^{8}$	$7.80 * 10^{7}$
2004	73	1108	832	130	138	143	$5.98 * 10^{10}$	$1.83 * 10^{8}$	$1.33 * 10^{8}$
2005	74	1149	864	131	139	147	$1.02 * 10^{11}$	$1.96 * 10^{8}$	$2.26 * 10^{8}$
2006	75	1191	896	133	141	148	$1.73 * 10^{11}$	$2.10 * 10^{8}$	$3.84 * 10^{8}$
2007	76	1235	928	134	142	152	$2.94 * 10^{11}$	$2.25 * 10^{8}$	$6.54 * 10^{8}$
2008	76	1279	960	135	144	155	$5.01 * 10^{11}$	$2.41 * 10^{8}$	$1.11 * 10^{9}$
2009	77	1323	1024	137	145	157	$8.52 * 10^{11}$	$2.59 * 10^{8}$	$1.89 * 10^{9}$
2010	78	1369	1056	138	146	160	$1.45 * 10^{12}$	$2.77 * 10^{8}$	$3.22 * 10^{9}$
2011	79	1416	1088	139	148	163	$2.47 * 10^{12}$	$2.97 * 10^{8}$	$5.48 * 10^{9}$
2012	80	1464	1120	141	149	165	$4.19 * 10^{12}$	$3.19 * 10^{8}$	$9.32 * 10^{9}$
2013	80	1513	1184	142	151	168	$7.14 * 10^{12}$	$3.41 * 10^{8}$	$1.59 * 10^{10}$
2014	81	1562	1216	143	152	172	$1.21 * 10^{13}$	$3.66 * 10^{8}$	$2.70 * 10^{10}$
2015	82	1613	1248	145	154	173	$2.07 * 10^{13}$	$3.92 * 10^{8}$	$4.59 * 10^{10}$
2016	83	1664	1312	146	155	177	$3.51 * 10^{13}$	$4.20 * 10^{8}$	$7.81 * 10^{10}$
2017	83	1717	1344	147	157	180	$5.98 * 10^{13}$	$4.51 * 10^{8}$	$1.33 * 10^{11}$
2018	84	1771	1376	149	158	181	$1.02 * 10^{14}$	$4.83 * 10^{8}$	$2.26 * 10^{11}$
2019	85	1825	1440	150	160	185	$1.73 * 10^{14}$	$5.18 * 10^{8}$	$3.85 * 10^{11}$
2020	86	1881	1472	151	161	188	$2.94 * 10^{14}$	$5.55 * 10^{8}$	$6.54 * 10^{11}$
2021	86	1937	1536	153	163	190	$5.01 * 10^{14}$	$5.94 * 10^{8}$	$1.11 * 10^{12}$
2022	87	1995	1568	154	164	193	$8.52 * 10^{14}$	$6.37 * 10^{8}$	$1.89 * 10^{12}$
2023	88	2054	1632	156	166	197	$1.45 * 10^{15}$	$6.83 * 10^{8}$	$3.22 * 10^{12}$
2024	89	2113	1696	157	167	198	$2.47 * 10^{15}$	$7.32 * 10^{8}$	$5.48 * 10^{12}$
2025	89	2174	1728	158	169	202	$4.20 * 10^{15}$	$7.84 * 10^{8}$	$9.33 * 10^{12}$
2026	90	2236	1792	160	170	205	$7.14 * 10^{15}$	$8.41 * 10^{8}$	$1.59 * 10^{13}$
2027	91	2299	1856	161	172	207	$1.21 * 10^{16}$	$9.01 * 10^{8}$	$2.70 * 10^{13}$
2028	92	2362	1888	162	173	210	$2.07 * 10^{16}$	$9.66 * 10^{8}$	$4.59 * 10^{13}$
2029	93	2427	1952	164	175	213	$3.52 * 10^{16}$	$1.04 * 10^{9}$	$7.81 * 10^{13}$
2030	93	2493	2016	165	176	215	$5.98 * 10^{16}$	$1.11 * 10^{9}$	$1.33 * 10^{14}$
2032	95	2629	2144	168	179	222	$1.73 * 10^{17}$	$1.27 * 10^{9}$	$3.85 * 10^{14}$
2034	96	2768	2272	171	182	227	$5.01 * 10^{17}$	$1.46 * 10^{9}$	$1.11 * 10^{15}$
2036	98	2912	2400	173	185	232	$1.45 * 10^{18}$	$1.68 * 10^{9}$	$3.22 * 10^{15}$
2038	99	3061	2528	176	188	239	$4.20 * 10^{18}$	$1.93 * 10^{9}$	$9.33 * 10^{15}$
2040	101	3214	2656	179	191	244	$1.22 * 10^{19}$	$2.22 * 10^{9}$	$2.70 * 10^{16}$
2042	*103*	*3371*	*2784*	*182*	*194*	*248*	$3.52 * 10^{19}$	$2.55 * 10^{9}$	$7.82 * 10^{16}$
2044	*104*	*3533*	*2944*	*185*	*197*	*255*	$1.02 * 10^{20}$	$2.93 * 10^{9}$	$2.26 * 10^{17}$
2046	*106*	*3700*	*3072*	*187*	*200*	*260*	$2.95 * 10^{20}$	$3.36 * 10^{9}$	$6.55 * 10^{17}$
2048	*107*	*3871*	*3232*	*190*	*203*	*265*	$8.53 * 10^{20}$	$3.86 * 10^{9}$	$1.90 * 10^{18}$
2050	*109*	*4047*	*3392*	*193*	*206*	*272*	$2.47 * 10^{21}$	$4.44 * 10^{9}$	$5.49 * 10^{18}$

4.4 Alternative Security Margin

Default Setting I (cf. 3.1.2) assumes that the DES offered enough security for commercial applications until the year 1982, but not beyond 1982. For corporations that have used the DES beyond 1982 or even until the late 1990s the resulting default infeasibility assumption of 0.5 MMY in 1982 (cf. 2.2.4) may be too strong. For others it may be too weak. Here we explain how to use Table 1 to look up key sizes for year y, for example $y = 2005$, if $s = 1982 + x$, i.e., if one trusts the DES until the year $1982 + x$. Here x is negative if our infeasibility assumption is considered to be too weak and positive otherwise. We assume the default settings for the other parameters. So, for example, $x = 13$ if one trusts the DES until 1995. Of course Remark 4.1.1 applies again.

- Symmetric keys: take the entry for year $y - x$, i.e., $2005 - 13 = 1992$ in our example. The resulting symmetric key size suggestion is 64 bits.
- Classical asymmetric keys: take the entry for year $y - 23 * x/43$, i.e., $2005 - 23 * 13/43 \approx 1998$ in our example. So 879-bit RSA and TDL keys should be used.
- SDL keys: let k' be the classical asymmetric key size for year $y - 23 * x/43$, let z be the SDL size for year $y - x$, and let k be the classical asymmetric key size for year $y - x$, then use a subgroup of size $z + 4 * \log_2(k) - 4 * \log_2(k')$ over a field of size k'. In our example $k' = 879$, $z = 114$, and $k = 682$, so that a subgroup of size $114 + 4 * \log_2(682) - 4 * \log_2(879) \approx 113$ bits should be used with an 879-bit field.
- EC systems with $c = 0$: take the '$c = 0$' entry for year $y - x$, i.e., $2005 - 13 = 1992$ in the example. The resulting EC key size suggestion is 120 bits.
- EC systems with $c = 18$: take the '$c = 18$' entry for year $y - 23 * x/43$, i.e., $2005 - 23 * 13/43 \approx 1998$ in our example. The resulting EC key size suggestion is 129 bits.

The Table 1 entries in italics for years before 1999 may be used in the last application; the other italics entries may be used if $x < 0$.

The correctness of these methods can be seen as follows. Let $k(y, s)$ denote the classical asymmetric key size recommendation k for a certain year y and security margin s. We want to find the year $\bar{y}$ for which $k(\bar{y}, s) = k(y, s + x)$, where $s = 1982$ by Default Setting I. From the definition of $IMY(y)$ in 4.2.1 and the way k is chosen in 4.2.3 it follows that

$$(\bar{y} - s)(12/m + t/b) + 12(\bar{y} - 1999)/r = (y - s - x)(12/m + t/b) + 12(y - 1999)/r,$$

from which we find that $\bar{y} = y - 23 * x/43$ if the default settings are used. The other results follow in the same way.

4.5 Cost-Equivalent Key Sizes

Table 1 can be used to derive cost-equivalent key sizes in the following manner, if the default settings are used. A lower bound for the equipment cost for a successful one day attack is given in the second to last column of Table 1, in year y in dollars of year y.

4.5.1 Symmetric key and EC systems. The symmetric key sizes are derived based on the definition of the security margin s which imply sufficient resistance against either software or special-purpose hardware attacks. The EC key sizes are based on estimates that are cost consistent with the symmetric key sizes (cf. 3.2.2). So for symmetric key and EC systems no corrections are necessary.

4.5.2 Classical aymmetric systems. For classical asymmetric systems, Mips-Years are supposedly $26 * P$ times as expensive, cf. 3.2.5. For our computational purposes only this is equivalent to assuming that the DES offers acceptable security until about 1997, since $12/m + t/b = 23/30$, $2^{15*23/30}$ is close to $26 * P$ for $P = 100$ (Default Setting VII, 3.2.5), and $1982 + 15 = 1997$. Thus, using Subsection 4.4, classical asymmetric key sizes that are equipment cost equivalent to symmetric and EC key sizes for year y can be found in Table 1 in the classical asymmetric key size column for year $y - (23 * 15)/43 = y - 8$. The resulting key sizes, rounded up to the nearest multiple of 32, are given as the second entry in the classical asymmetric key sizes column of Table 1. Breaking such keys requires a substantially smaller number of Mips-Years than the infeasible number of Mips-Years for year y, but acquiring the required Mips-Years is supposed to be prohibitively expensive.

Note that this value is rounded up to the next multiple of 32, despite Remark 4.1.1, reflecting the inherently inaccurate choice $P = 100$ in Default Setting VII (cf. 3.2.5).

4.5.3 SDL systems. For subgroup discrete logarithm systems in year y, let z and k be the subgroup and finite field size, respectively, for year y, and let k' be the finite field size for year $y - 8$. For cost equivalence with symmetric and EC key sizes in year y use subgroups of size $z + 4 * \log_2(k) - 4 * \log_2(k')$ over finite fields of size k'. As a rule of thumb, subgroups of size $z + 2$ over finite fields of size k' will do.

As an example, in the year 2000 the following key sizes are more or less equipment cost equivalent: 70-bit symmetric keys, 682-bit classical asymmetric keys, 127-bit subgroups with 682-bit finite fields, and 132-bit EC keys.

A similar straightforward analysis can be carried out for any other setting for the parameter P. For instance, for $P = 10$ or $P = 1000$ the $y - 8$ should be changed into $y - 6$ or $y - 10$, respectively.

4.6 Key Sizes Currently Equivalent to Given Symmetric Key Size.

4.6.1 Formulas for key sizes equivalent to symmetric key size. Suppose that key sizes have to be determined that are currently at least equivalent to a symmetric key size d. Note that the resulting formulas must be independent of our assumptions on security margin, hardware advances, or cryptanalytic progress. The only settings used here are P (cf. 3.2.5) and v (cf. 4.2.2), because they are the only settings relevant for the current circumstances.

Compared to breaking a 56-bit DES key at an expected cost of $5 * 10^5$ Mips-Years, breaking a key of size d used in conjunction with a symmetric key system that is v times slower than the DES can be expected to take

$$EMY(d) = 2^{d-56} * 5 * 10^5 * v \text{ Mips-Years},$$

where EMY stands for 'Equivalent number of Mips-Years'.

If the classical asymmetric key size k is chosen such that

$$\frac{L[2^k]}{EMY(d)} \geq \frac{L[2^{512}]}{10^4}$$

(cf. 4.2.3), then the security offered by classical asymmetric systems is currently at least computationally equivalent to the security offered by a symmetric key of size d. But if the classical asymmetric key size k' is chosen such that

$$\frac{L[2^{k'}]}{EMY(d)} \geq \frac{L[2^{512}]}{10^4 * 26 * P}$$

(cf. 4.2.3), then the security offered by classical asymmetric systems is currently at least cost equivalent to the security offered by a symmetric key of size d.

If the EC key size u is chosen such that

$$\frac{2^{u/2} * u^2}{EMY(d)} \geq \frac{2^{109/2} * 109^2}{2.2 * 10^6}$$

(cf. 4.2.4), then the security offered by EC systems is currently at least computationally and cost equivalent to the security offered by a symmetric key of size d.

If the SDL subgroup size z satisfies

$$z \geq 109 + 2 * \log_2 \left(\frac{EMY(d) * 109^2 * 9}{\bar{k}^2 * \sqrt{2} * 2.2 * 10^6} \right)$$

(cf. 4.2.5), where $\bar{k}$ is the finite field size, then the security offered by SDL systems is currently at least equivalent to the security offered by a symmetric key of size d: computationally equivalent if $\bar{k} = k$ and cost equivalent if $\bar{k} = k'$, with k and k' as above.

From the formulas given here and in Subsection 4.2 it is obvious how formulas should be obtained for key sizes equivalent to a given symmetric key size in a given year: use the formulas from Subsection 4.2 with $IMY(y)$ replaced by $EMY(d)$.

4.6.2 Looking up currently computationally equivalent key sizes. Assuming the default settings, Table 1 can also be used to look up the key sizes that follow from the formulas in 4.6.1. Given a symmetric key size d, asymmetric key sizes that are currently computationally equivalent to it can be looked up as follows. For classical asymmetric systems look up the classical asymmetric key size for year $y' = 30 * d/43 + 1950.8$. This formula follows by solving the equation

$$EMY(d) = IMY(y') * 2^{12(y'-1999)/r}$$

for y' (cf. 4.2.1). For the other systems let y be the year in Table 1 in which d occurs in the symmetric key size column. For SDL look up the SDL key size z for year y, the classical asymmetric key size k' for year y', and the classical asymmetric key size k for year y; then subgroups of size $z + 4 * \log_2(k) - 4 * \log_2(k')$ over a field of size k' offer security that is currently computationally equivalent, in the year 1999, to symmetric keys of size d. For EC simply look up the EC key size for year y and '$c = 0$'.

Given a classical asymmetric key size k, the currently computationally equivalent symmetric key size can be found by looking up the year y in which k occurs, and by using symmetric key size $43 * y/30 - 2796.2$; this follows immediately from $y' = 30 * d/43 + 1950.8$.

As an example, for a symmetric key of size $d = 85$ we find that $y = 2019$ and $y' = 30 * 85/43 + 1950.8 = 2010.1$. Currently computationally equivalent key sizes are: about 1375 bits for classical asymmetric keys, subgroups of size $150 + 2 = 152$ over 1375 bits fields, and EC systems of 160-bits. Similarly, for a classical asymmetric key of size $k' = 1024$ we find that $y = 2002$ and that a currently computationally equivalent symmetric key size is given by $43 * 2002/30 - 2796.2 \approx 74$. The latter corresponds to a currently computationally equivalent EC key size of 139 bits.

4.6.3 Looking up currently cost-equivalent key sizes. Given a symmetric key size d, asymmetric key sizes that are currently cost equivalent to it can be looked up in a very similar way: just replace 1950.8 and 2796.2 from 4.6.2 by 1942.9 and 2784.9, respectively. This formula follows by solving the equation

$$\frac{EMY(d)}{26 * P} = IMY(y') * 2^{12(y'-1999)/r}$$

for y'. Here we use Default Setting VII (i.e., $P = 100$, cf. 3.2.5) as in Subsection 4.5. As an example, for a symmetric key of size $d = 85$ we find that $y = 2019$ and $y' = 30 * 85/43 + 1942.9 = 2002.2$. Currently cost-equivalent key sizes are: about 1036 bits for classical asymmetric keys, subgroups of size $150 + 2 = 152$ over 1036 bits fields, and EC systems of 160 bits.

Similarly, for a classical asymmetric key of size $k = 1024$ we find that $y = 2002$ and that a currently cost-equivalent symmetric key size is given by $43 * 2002/30 - 2784.9 \approx 85$.

5 Practical Consequences

5.1 DSS

The US Digital Signature Standard (DSS) uses 160-bit subgroups with field sizes ranging from 512 to 1024 bits, and a 160-bit hash function. According to Table 1 only the largest field size (1024) can be recommended for commercial applications and then only until the year 2002. The other sizes can be recommended until 2013 for the hash function, and until 2026 for the subgroup size. Assuming the default settings, the security offered by the DSS may become inadequate very soon, unless the DSS is used in combination with a 1513-bit finite field until 2013. A change in the field size does not affect the size of the DSS signatures. Beyond 2013 the 160-bit size of SHA-1, the cryptographic hash function used in conjunction with the DSS, may no longer be adequate. Note, however, that the hash size may have to match the subgroup size, so that changing the hash size may force a change in the subgroup size that would otherwise not have been necessary until 2026.

According to [25], NIST is working on a revision for the DSS, with key sizes as reported in Table 2 (and hash size matching the size of q).

Table 2. Proposed key sizes for the revised DSS

size q	160	256	384	512
size p	1024	3072	7680	15360

These values are in close agreement with the values that follow from our current cost equivalence model as in Subsection 4.5 (i.e., with Default Setting VII $P = 100$, cf. 3.2.5). However, it follows from Table 1 that the p sizes have to grow much faster than proposed in Table 2 if current cryptanalytic trends persist and if equivalence between the sizes of p and q has to be maintained in the future.

5.2 Effect on cryptosystem speed

RSA keys that are supposed to be secure until 2040 are about three times larger than the popular 1024-bit RSA keys that are currently secure. That makes those large keys 9 to 27 times slower to use: 9 for signature verification or encryption assuming a fixed length public exponent, 27 for the corresponding signature generation or decryption. TDL systems will slow down by a factor 27 compared to those that are currently secure. SDL systems slow down by about a factor 11 compared to currently secure SDL systems, because of the growth of the underlying finite field combined with the growth of subgroup size. The speed of EC systems, however, is hardly affected: a slowdown by a factor of at most 4, assuming cryptanalytic progress with $c = 18$. Within a few years, however, faster processors will have solved these performance problems if our default setting for m turns out to be reasonable. Note, however, that this may not be the case in more restricted environments such as smartcards, where bandwidth and power consumption constraints also have a more limiting effect on key sizes.

5.3 512-bit RSA keys

Despite the fact that they were already considered to be suspicious in 1990, 512-bit RSA keys are still widely used all over the Web. For instance, 512-bit RSA moduli are used in the international version of Secure Socket Layer (SSL) secured webservers to exchange session keys. An attacker who breaks an SSL RSA modulus will be able to access all session keys used by the SSL server, and hence all information protected by those keys. According to Table 1, 512-bit RSA keys should not have been used beyond 1986.

It should be noted that, apart from the security risk of using 512-bit RSA keys, there are also considerable publicity risks in using them: organizations using them may get bad media-coverage when it is found out, because a 512-bit RSA key was factored in August 1999. Although this result is the first published factorization of a 512-bit RSA modulus, it would be naïve to believe that it is the first time such a factorization has been obtained (Remark 1.4.1 and 2.4.5).

5.4 768-bit RSA keys

According to Table 1 usage of 768-bit RSA keys can no longer be recommended. Even in the cost-equivalent model 768-bit RSA keys will soon no longer offer security comparable to the security of the DES in 1982.

5.5 RSA and EC

If one evaluates $L[2^{1024}]$ (cf. 2.4.4) omitting the $o(1)$ the result is close to the number of 32-bit operations to be performed by an attack using Pollard's rho method on a 160-bit EC system. It was shown in 2.4.6, however, that $L[n]$ substantially overestimates the actual number of operations to be performed by the NFS factorization of n. Nevertheless, in the (commercial) cryptographic literature 1024-bit RSA and 160-bit EC systems are often advertised as offering more or less the same level of security.

If one is interested in currently computationally equivalent security then 1024-bit RSA and 139-bit EC systems or 1375-bit RSA and 160-bit EC systems may be considered to be comparable, as follows from the example in 4.6.2. For currently cost-equivalent security the example in 4.6.3 suggests that 1024-bit to 1035-bit RSA and 160-bit EC systems may be comparable. This last comparison depends strongly on the setting one deems reasonable for the parameter P, as explained in 3.2.5 and Subsection 4.5.

5.6 SDL and EC

The gap between the suggested SDL and EC key sizes widens slowly. This is due to the rapidly growing size of the underlying finite fields in SDL, which makes the finite field operations required for an attack using Pollard's rho method relatively slow. Note that the field size for SDL systems can be found in the classical asymmetric key size column of Table 1.

5.7 Effectiveness of guessing

The sizes suggested in Table 1 for the year 2000 or later give keys that are in practice infeasible to guess.

5.8 Effectiveness of Incomplete Attacks

Spending only a fraction $IMY(y)/x$ of the full effort $IMY(y)$ (cf. 4.2.1) required to break a system using the key sizes suggested for year y leads to success probability $1/x$ for exhaustive search (symmetric systems; cf. 2.2.7), 0 for the (DL)NFS (classical asymmetric systems, cf. 2.4.9; for the ECM see Subsection 5.9), or $1/x^2$ for Pollard's rho method (SDL and EC; cf. 2.5.6 and 2.6.7). This implies that on average incomplete attacks cannot be expected to pay off. Despite the lack of appreciable economic incentive an attacker may nonetheless try to harness a small fraction of the required run time and get a non-negligible chance that his efforts bear fruit. As noted in Remark 3.1.4, however, if our definition of security margin (cf. 3.1.2) is acceptable, then this risk is acceptable as well.

5.9 Effectiveness of Elliptic Curve Method

The Elliptic Curve Method (ECM) finds a 167-bit factor of a 768-bit number with probability 0.63 after spending 6200 Mips-Years, under the assumption that such a factor exists [47]. Based on this data point, we have computed the probability that the ECM successfully factors RSA moduli of the sizes specified in Table 1, assuming we invest the corresponding $IMY(y)$ Mips-Years (cf. 4.2.1) in each factoring attempt: for a 952-bit RSA modulus the probability of success is $2.6 * 10^{-7}$ after spending $7.1 * 10^{9}$ Mips-Years (for $y = 2000$), deteriorating to probability $1.9 * 10^{-9}$ for a 1149-bit modulus in 2005, and $1.2 * 10^{-11}$ for 1369 bits in 2010. It follows that, despite the impossibly large investment, the ECM cannot be expected to break keys of the suggested sizes. The ECM success probability vanishes with the years, consistent with the fact that the NFS is asymptotically superior to the ECM. Note that these probabilities apply only to regular RSA where the modulus has two prime factors of about equal size. If the primes have different sizes [32] or if there are more primes dividing the modulus ([31]: the 'Multiprime' variation of RSA), the success probability of the ECM is considerably higher.

5.10 Wassenaar Arrangement for mass market applications

Currently the Wassenaar Arrangement allows 64-bit symmetric keys and 512-bit classical asymmetric keys for mass market applications. According to Table 1 and publicly available data on successful attacks it would be advisable (in 2001) to increase the 512-bit bound for classical asymmetric keys to a more reasonable bound such as 736 or 832 bits.

Disclaimer. The contents of this article are the sole responsibility of its authors and not of their employers. The authors or their employers do not accept any responsibility for the use of the cryptographic key sizes recommended in this article. The authors do not have any financial or other material interests in the conclusions attained in this article, nor were they inspired or sponsored by any party with commercial interests in cryptographic key size selection. The data presented in this article were obtained in a two stage approach that was strictly adhered to: formulation of the model and collection of the data points, followed by computation of the lower bounds. No attempt has been made to alter the resulting data so as to better match the authors' (and possibly others') expectations or taste. The authors made every attempt to be unbiased as to their choice of favorite cryptosystem, if any. Although the analysis and the resulting guidelines seem to be quite robust, this will no longer be the case if there is some 'off-the-chart' cryptanalytic or computational progress affecting any of the cryptosystems considered here. Indeed, according to at least one of the present authors, strong long-term reliance on any current cryptosystem without very strong physical protection of all keys involved – including public ones – is irresponsible. This does not necessarily imply lack of trust in public key cryptosystems – it reflects mixed feelings about the way they are implemented or embedded in applications.

Acknowledgment. The authors want to thank Stuart Haber for his extensive comments on several versions of this article, Kai Puolamäki for providing the Java applet that allows easy computation of the key size formulas, Joe Buhler, Bruce Dodson, Don Johnson, Paul Leyland, Alfred Menezes, Andrew Odlyzko, Michael Wiener, and Paul Zimmermann for their helpful remarks, and two anonymous referees for their much appreciated comments (cf. 3.1.2 and Remark 4.1.1).

References

1. Ross Anderson, *Why cryptosystems fail*, Comm. of the ACM, v. 37, n. 11, Nov. 1994, 32-40.
2. E. Biham, *A fast new DES implementation in software*, Proceedings of Fast Software Encryption, LNCS 1267, Springer 1994.
3. M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, *Minimal key lengths for symmetric ciphers to provide adequate commercial security*, www.bsa.org/policy/encryption/cryptographers_c.html, January 1996.
4. A. Bosselaers, *Even faster hashing on the Pentium*, rump session presentation at Eurocrypt'97, May 13, 1997; www.esat.kuleuven.ac.be/~cosicart/pdf/AB-9701.[pdf, ps.gz].
5. J.R.T. Brazier, *Possible NSA decryption capabilities*, jya.com/nsa-study.htm.
6. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al., *Factorization of a 512-bit RSA modulus*, Proceedings Eurocrypt 2000, LNCS 1807, 1-17, Springer 2000.
7. www.certicom.com, 1997.
8. www.counterpane.com/speed.html.
9. M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, J.J. Quisquater, *Efficient hardware and software implementations of the DES*, Proceedings Crypto'84, Springer 1984.

10. W. Diffie, BNR Inc. report, 1980.
11. W. Diffie, E. Hellman, *Exhaustive cryptanalysis of the NBS Data Encryption Standard*, Computer, v. 10 (1977), 74-84.
12. B. Dixon, A.K. Lenstra, *Factoring integers using SIMD sieves*, Proceedings Eurocrypt'93, LNCS 765, 28-39, Springer 1993.
13. Electronic Frontier Foundation, *Cracking DES*, O'Reilly, July 1998.
14. R. Gallant, personal communication, August 1999.
15. R. Gallant, R. Lambert, S. Vanstone, *Improving the parallelized Pollard lambda search on binary anomalous curves*; available from www.certicom.com/chal/download/paper.ps, 1998.
16. D.B. Johnson, *ECC, Future Resiliency and High Security Systems*, March 30, 1999, available from www.certicom.com.
17. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, Proceedings ANTS IV, LNCS 1838, 358-394, Springer 2000.
18. A. Joux, K. Nguyen, *Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups*, available from http://eprint.iacr.org, 2000.
19. P.C. Kocher, *Breaking DES*, RSA Laboratories' Cryptobytes, v. 4, no 2 (1999), 1-5; also at www.rsasecurity.com/rsalabs/pubs/cryptobytes.
20. P.C. Kocher, personal communication, September 1999.
21. A.K. Lenstra, A. Shamir, *Analysis and optimization of the TWINKLE factoring device*, Proceedings Eurocrypt 2000, LNCS 1807, 35-52, Springer 2000.
22. A.K. Lenstra, E.R.Verheul, *Selecting Cryptographic Key Sizes*, Proceedings PKC 2000, LNCS 1751, 446-465, Springer 2000; full version available from www.cryptosavvy.com.
23. A.K. Lenstra, E.R.Verheul, *The XTR public key system*, Proceedings Crypto 2000, LNCS 1880, 1-19, Springer 2000; available from www.ecstr.com.
24. P. Leyland, personal communication, September 1999 - February 2001.
25. A.J. Menezes, personal communication, September 1999.
26. P.L. Montgomery, letter to the editor of IEEE Computer, August 1999.
27. V.I. Nechaev, *Complexity of a determinate algorithm for the discrete logarithm*, Mathematical Notes, 55 (2) 1994, 155-172. Translated from Matematicheskie Zametki, 55(2), 91-101, 1994. This result dates from 1968.
28. *Tiniest circuits hold prospect of explosive computer speeds*, The New York Times, July 16, 1999; *Chip designers look for life after silicon*, The New York Times, July 19, 1999.
29. A.M. Odlyzko, *The future of integer factorization*, RSA Laboratories' Cryptobytes, v. 1, no. 2 (1995), 5-12; also at www.research.att.com/amo/doc/crypto.html or www.rsasecurity.com/rsalabs/pubs/cryptobytes.
30. K. Puolamäki, Java applet on www.cryptosavvy.com.
31. www.rsa.com and www.rsasecurity.com.
32. A. Shamir, *RSA for paranoids*, RSA Laboratories' Cryptobytes, v. 1, no. 3 (1995), 1-4.
33. A. Shamir, *Factoring integers using the TWINKLE device*, Proceedings CHES '99, LNCS 1717, Springer 1999.
34. P.W. Shor, *Algorithms for quantum computing: discrete logarithms and factoring*, Proceedings of the IEEE 35th Annual Symposium on Foundations of Computer Science, 124-134, 1994.
35. V. Shoup, *Lower bounds for discrete logarithms and related problems*, Proceedings Eurocrypt'97, LNCS 1233, 256-266, Springer 1997.
36. R.D. Silverman, rump session presentation at Crypto'97.

37. R.D. Silverman, *Exposing the Mythical Mips-Year*, IEEE Computer, August 1999, 22-26.
38. R.D. Silverman, *A cost-based security analysis of symmetric and asymmetric key lengths*, RSA Laboratories Bulletin, number 13, April 2000.
39. Simon Singh's cipher challenge, www.simonsingh.com/cipher.htm.
40. P.C. van Oorschot, M.J. Wiener, *Parallel collision search with cryptanalytic applications*, Journal of Cryptology, v. 12 (1999), 1-28.
41. E.R. Verheul, *Evidence that XTR is more secure than supersingular elliptic curves*, Proceedings Eurocrypt 2001, to appear, Springer 2001.
42. www.wassenaar.org.
43. M.J. Wiener, *Efficient DES key search*, manuscript, Bell-Northern Research, August 20, 1993.
44. M.J. Wiener, *Performance Comparison of Public-Key Cryptosystems*, RSA Laboratories' Cryptobytes, v. 4, no. 1 (1998), 1-5; also at www.rsasecurity.com/rsalabs/pubs/cryptobytes.
45. M.J. Wiener, personal communication, 1999.
46. M.J. Wiener, R.J. Zuccherato, *Faster attacks on elliptic curve cryptosystems*, in S. Tavares and H. Meijer, eds., Selected areas in Cryptography'98, LNCS 1556, 1999, Springer 1999.
47. P. Zimmermann, personal communication, 1999.

This article was processed using the LaTeX macro package with LLNCS style