Selecting Cryptographic Key Sizes

Extended Abstract

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

[1] Citibank, N.A., 1 North Gate Road, Mendham, NJ 07945-3104, U.S.A,
arjen.lenstra@citicorp.com
[2] PricewaterhouseCoopers, GRMS Crypto Group, P.O. Box 85096, 3508 AB Utrecht,
The Netherlands, Eric.Verheul@[nl.pwcglobal.com, pobox.com]

Abstract. In this article we give guidelines for the determination of cryptographic key sizes. Our recommendations are based on a set of explicitly formulated hypotheses, combined with existing data points about the cryptosystems. This article is an abbreviated version of [15].

1 Introduction

1.1. Introduction. 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. Key size recommendations are scattered throughout the cryptographic literature or may 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 [15], of which this is an extended abstract, 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 hypotheses about future developments and apply these uniformly to existing data about the cryptosystems. The resulting key size recommendations are thus obtained in a uniform mechanical way independent of further assumptions or non-scientific considerations. Despite our attempt to be objective we do not expect that our model is to everyone's taste. The underlying model can, however, easily be changed without affecting the overall approach, thereby making this article useful also for those who object to our results.

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 any of the currently popular cryptographic protocols is

---

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 suggested in this article. The authors do not have any financial or other material interests in the conclusions attained in this paper, 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.

considerably less secure than expected. It may even render them completely insecure, as illustrated by the sudden demise of the once popular knapsack-based cryptosystems. In this article we discuss only cryptosystems for which it is believed to be unlikely that such catastrophes will ever occur. 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. 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 (cf. [21]). It is unclear if quantum computers are feasible at all, and our suggestions do not take them into account. Neither do we consider the potential effects of molecular-computing (cf. [19]).

1.2. Run time convention. All our run time estimates are based on actual run times or reliable estimates of run times on a 450MHz Pentium II processor. A 'PC' always refers to this processor. Computing power is often measured in Mips Years (MY), 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 and we agree with the concerns expressed in [24]. Nevertheless we use MY here as well. We use the convention that one year of computing on a PC is equivalent to 450 MY, but ultimately all our estimates are based on run times on a PC and not on the actual or our definition of MY. The two definitions are, however, sufficiently close (cf. [15]). Our MY figures are therefore compatible with MY figures found elsewhere. We write MMY for one million MY.

1.3. Lower bounds. Our guidelines are 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 guidelines are therefore obtained by systematically underestimating the 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 MY 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, *equipment cost equivalent* security, or *cost equivalent* security for short. 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 two different attacks may require different hardware.

Following our guidelines does **not** necessarily result in cost equivalent security. In (4.5) we indicate how our guidelines may be changed to obtain cost equivalence, thereby possibly giving up computational equivalence.

The most important reason why we opted for computationally equivalent security as opposed to cost equivalent security is that 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 (4.5) we present three points of the spectrum.

2 The 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 from member countries to countries that jeopardize their national security. The Wassenaar Arrangement (WA) is a follow-up of the COCOM regulations. In this article we limit ourselves to the 5 types of cryptographic primitives for which a maximum key size that does not require an export license is given in the WA (December 1998, cf. www.wassenaar.org).

We distinguish the cryptographic primitives into symmetric-key (or secret-key) and asymmetric-key (or public-key) cryptosystems and briefly mention cryptographic hash functions as well.

2.2. Symmetric key cryptosystems.
Description. In symmetric key cryptosystems the parties share a secret key. The size of the key is its number of bits and depends on the symmetric key cryptosystem.
Wassenaar Arrangement. The maximum symmetric key size allowed by the WA is 56 bits for 'niche market' applications and 64 bits for 'mass market'.
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}.
Software data points. In 1997 a DES key was successfully retrieved after an Internet search of approximately 4 months (cf. www.rsa.com/des). The expected computing power required for such a software exhaustive key search is underestimated as 0.5 MMY (cf. (1.3)). This estimate is based on the Pentium based figures that a single DES block encryption with a fixed key requires 360 clock cycles or 500 clock cycles with a variable key (cf. [6], [1]). Our estimate lies between two DEC VAX 11/780 estimates that can be found in [7] and [20]. Half a million MY is roughly 13500 months on a PC or 4 months on 3500 PCs, because an exhaustive key search can be evenly divided over any number of processors. For a proper security analysis one therefore has to keep track of the total computational power of the Internet.
Special-purpose hardware data points. At the cost of a one-time investment a hardware attack is substantially faster than a software attack. In 1980 a $50 million parallel DES key searching machine was proposed with an expected search time of 2

days (cf. [9], [8]), followed in 1993 by a $1 million, 3½ hour design (cf. [26]). In 1998 a $130,000, 112 hour machine was built (cf. [13], [11]).

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 the correct key is guessed is small: even for a 50-bit key there is a total probability of one in a million that it is found if 10^9 people each make a different guess. With the same effort, the probability of success halves for each additional key bit. Exhaustive key search is nothing more than systematic guessing.

Incomplete attacks. The success probability of exhaustive key search is proportional to the fraction of the key space searched.

Cryptanalytic progress. We assume the existence of a generic symmetric key cryptosystem of arbitrary key size that is about as fast as the DES and for which exhaustive key search is the best attack. Thus, 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. If the private key of an asymmetric key cryptosystem can be derived from the public key, then the system can be broken. What the keys consist of, and how hard it is to break the system, depends on the type of asymmetric key cryptosystem. We distinguish the following three types:

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

2.3.1. Classical asymmetric systems. These refer to RSA and traditional discrete logarithm (TDL) systems, such as the Diffie-Hellman scheme and ElGamal systems.

RSA description. In RSA the public key contains a large non-prime number, the RSA modulus, which is chosen as the product of two large primes. The security of RSA is based on the difficulty of the integer factorization problem. The size of an RSA key refers to the bit-length of the RSA modulus. This should not be confused with the number of bits required to store an RSA public key, which is usually slightly more.

TDL description. In a TDL system the public key consists of a finite field F_p of size p, a generator g of the multiplicative group $(F_p)^*$ of F_p, and an element y of $(F_p)^*$ 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 discrete logarithm of y with respect to g, i.e., the smallest positive integer m such that $g^m = y$. The private key m is at least 1 and at most $p-2$. 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 number of bits required to store a TDL public key is larger, since it contains g and y as well.

Wassenaar Arrangement. Both the maximal RSA modulus size and the maximal field size allowed by the WA are 512 bits.

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 F_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 and much larger keys are

required. The methods to attack these two entirely different problems are similar, which is why we treat RSA and TDL systems as the same category.

The fastest factoring algorithm published today is the Number Field Sieve (NFS), which is based on an idea by John Pollard. On heuristic grounds NFS can be expected to require time proportional to

$$L[n] = e^{(1.9229 + o(1)) * \ln(n)^{1/3} * \ln(\ln(n))^{2/3}}$$

to factor an RSA modulus n, where the o(1) term goes to zero as n goes to infinity. This run time is called *subexponential* in n because as n goes to infinity it is less than n^c for any $c > 0$. The storage requirements of the NFS are proportional to $\sqrt{L[n]}$. If p is a prime number then a discrete logarithm variation of the NFS (DLNFS) finds a discrete logarithm in F_p in expected time proportional to $L[p]$.

These run time estimates cannot be used directly to estimate the number of operations required to factor a certain n or to compute discrete logarithms in a certain F_p. For n and p of about the same size, $L[n]$ and $L[p]$ are approximately equal if the o(1)'s are omitted, but the discrete logarithm problem in F_p is considerably more difficult than factoring n. 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 NFS takes time t, then factoring some other RSA modulus $m > n$ will take time close to $t*L[m]/L[n]$ (omitting the o(1)'s), if the sizes of n and m do not differ by too much. If, however, m is much bigger than n, then the effect of the o(1) going to zero can no longer be ignored (cf. [23]), and $t*L[m]/L[n]$ will be an overestimate of the time to factor m. The same run time extrapolation method applies to the DLNFS.

Software data points. The largest published factorization using the NFS is that of the 512-bit number RSA155, an RSA modulus of 155 decimal digits (cf. [5]). This 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). It is less than 10^4 MY 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 [24] which appeared around the same time and which are 100 times too high (cf. [17]). This run time is only a fraction of the cost of a software DES key search, but the NFS requires much more memory.

Practical experience with the DLNFS is still limited. It is generally accepted that, for any b in the current range of interest, 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. Below we do not present key size suggestions for TDL systems and recommend using the RSA key size suggestions for TDL systems as well.

Special-purpose hardware data points. Special-purpose hardware devices are occasionally proposed for factoring algorithms, but no useful data points have been published. Due to the complexity of the underlying factorization algorithms and the corresponding hardware design it is for any special-purpose hardware factoring device 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. 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 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 (cf. [10]). A SIMD machine is by no means special-purpose hardware, but it could be relatively cheap compared to ordinary PCs.

Effectiveness of guessing. Key sizes for classical asymmetric systems have to be larger than 512 to obtain any security at all. Breaking the system by guesswork is thus out of the question. 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 below.

Incomplete attacks. Both the NFS and the DLNFS are effective only if run to completion. RSA, however, can be attacked also by the Elliptic Curve Method (ECM). After a relatively small amount of work this method produces a factor with much higher probability than mere guesswork: 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. See also (5.9).

Cryptanalytic progress. Classical asymmetric systems are the prime example of systems for which the effectiveness of cryptanalysis is steadily improving. 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. 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, and sieving-based polynomial selection for NFS. We 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 public exponent is small. We therefore explicitly assume that breaking RSA is equivalent to factoring the RSA modulus. In particular, we assume that the public exponent for RSA is sufficiently large. 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. There is strong evidence that the latter problem is equivalent to computing discrete logarithms

2.3.2. Subgroup discrete logarithm systems.

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 $(F_p)^*$. 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.

Wassenaar Arrangement. The maximum SDL field size allowed by the WA is 512 bits – there is no maximum allowed key size. A popular subgroup size is 160 bits.

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 SDL 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 if 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 q, namely $1.25\sqrt{q}$ group operations, i.e., multiplications in F_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 resulting in a linear speedup (cf. [25]). Furthermore, there is no post-processing involved in Pollard's rho (unlike the (DL)NFS, where after completion of the first step a cumbersome matrix step has to be carried out), although for the parallelized version substantial amounts of storage space should be available.

Data points. We have not been able to find any useful data about the effectiveness of the parallelized Pollard rho attack on SDL systems. Our figures below are based on an adaptation of data points for elliptic curve systems, cf. (4.1).

Effectiveness of guessing. As long as SDL keys are not shorter than 112 bits (permitted by the WA for EC systems, see below), guessing the private key requires guessing at least 112 bits. This may safely be assumed to be infeasible.

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, $0 \le x \le 1$, the chance is x^2 that the key is found after performing a fraction x of the expected $1.25\sqrt{q}$ group operations.

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 progress is to assume that no substantial progress will be made. The results in [18, 22] that, in a certain generic model of computation, Pollard's rho 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.

2.3.3. Elliptic curve systems.

Description. Elliptic curve (EC) systems are like SDL systems, except that g generates a subgroup of the group H of points on an elliptic curve E over a finite field F_p. 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 a subgroup of H. These can be computed if discrete logarithms in H 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 H (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.

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

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. The number of field multiplications per group operation is about 12.

Software data points. The cost of the group operation is proportional to $(\log_2(q))^2$. From the estimates given on www.certicom.com/chal 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 www.certicom.com/chal 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 (cf. [12]). Nevertheless, it is used as the basis for extrapolations to estimate the effort required for software attacks on larger EC systems over prime fields (cf. (1.3)).

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. [25]). Building this design would cost $10 million and it would take 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. This is further discussed in (3.6).

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

Incomplete attacks. As with Pollard's rho attack against SDL systems its success probability is proportional to the square of the fraction of the work performed.

Cryptanalytic progress. The remarks made above on SDL systems apply here as well. It is therefore not unreasonable to base our figures below on the assumption that there will be no substantial progress in the years to come. For EC systems this is not something we feel comfortable with, because EC related cryptanalytic results are obtained quite regularly. So far, most of these results affected only special cases. We

therefore make the explicit assumption that curves are picked at random and that only curves over prime fields are used. Even then, it is not hard to find researchers who believe 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 EC systems are sufficiently secure. We do not want to take a position in this argument and we simply suggest two key sizes for EC systems: one based on 'no cryptanalytic progress' and one based on 'cryptanalytic progress at the same rate as for RSA and TDL systems'. The reader may then interpolate between the two types of extrapolations according to her own taste.

2.4. Cryptographic hash functions.
Description. A cryptographic hash function is a function that maps an arbitrary length message to a fixed length 'hash', satisfying various properties that are beyond the scope of this article. The size of the hash function is the length in bits of the hash.
Attacks. Cryptographic hash functions can be attacked by the 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. We assume that cryptographic hash functions have to be 'any collision-resistant'. For 'target collision-resistant' hashes the sizes may be halved assuming the hash function is properly used.
Software data points. In [3] 241, 345, 837, and 1016 Pentium cycles are reported for MD4, MD5, SHA-1, and RIPEMD-160, respectively. 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 (cf. (2.2)).
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.
Cryptanalytic progress. We assume the existence of a generic cryptographic hash function of speed comparable to the existing functions mentioned above and for which the birthday paradox attack is the best attack. It follows that an attack on our generic symmetric key cryptosystem of key size b can be expected to take about the same time as an attack on our generic cryptographic hash function of size $2b$. Thus, a lower bound for the size of the latter 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.

3 The model

3.1. Key points. The choice of cryptographic key sizes depends primarily on the following four points:

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

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

3.2. Life span. In the table in Section 4 key sizes are suggested, 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.

3.3. Security margin. A cryptosystem can be assumed to be secure only if it is considered to be sufficiently infeasible to mount a successful attack. It is hard to quantify what this means precisely. One could, for instance, decide that a key size for a certain cryptosystem is secure if breaking it would be, say, 10^6 times harder than the largest key size that can currently be broken. There are several problems with this approach. First of all, the choice 10^6 is rather arbitrary. Secondly, there is no reason to believe that the 'largest key broken so far' accurately represents the best that can currently be done. In the third place, for some of the cryptographic primitives considered here data may not be available or they may be outdated (SDL, TDL), thereby ruling out uniform application of this approach. We opt for a different approach.

Hypothesis I. As the basis for our extrapolations we assume that the DES was at least sufficiently secure for commercial applications until 1982 because it was introduced in 1977 and stipulated to be reviewed every five years. We therefore hypothesize that in 1982 a computational effort of 0.5 MMY was believed to provide an adequate security margin for commercial DES applications against software attacks (cf. (2.2)). As far as hardware attacks are concerned, we assume that the "$50 million, 2 days" DES key searching machine (cf. (2.2)) from 1980 was not considered to be a serious threat for commercial applications of the DES at least until 1982. We stress 'commercial applications' because, even for 1980 budgets, $50 million and 2 days are not an insurmountable obstacle for certain organizations. Our hypothesis is further discussed below (cf. (3.8)). We note that quite different assumptions allow an approach similar to ours, though the resulting guidelines will be different (cf. (4.4)).

3.4. Computing environment.

Hypothesis II. To estimate how the computing power available to attackers may change over time we use a variation of 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. Therefore we hypothesize a less technology dependent variation that so far seems to be sufficiently accurate: every 18 months the amount of computing power and random access memory one gets for a dollar doubles. Thus, for the same cost one gets 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 1980 "$50 million, 2 days" DES key searching machine would be a "$1 million, 100 days" machine, i.e., 50 times less hardware and therefore 50 times slower. According to our version of Moore's law the $1 million machine 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 $2^{8.7} \approx 406$ the 1993 version would need about 100/406 days, i.e., about 6 hours, which is indeed close to the 3½ hours required by the $1 million design from [26]. On the other hand, further extrapolation suggests that in

1998 a $1 million machine may be expected to take 0.6 hours, or that a $130,000 machine would take 4.6 hours, i.e., about 24 times faster than the machine that was actually built in 1998 (cf. [13]). This anomaly is due to the fact that building the $130,000 machine was, relatively speaking, a small scale enterprise where every doubling of the budget would have quadrupled the performance (cf. [14]).

Hypothesis III. Our version of Moore's law implies that we have to consider how budgets may change over time. The US Gross National Product shows a trend of doubling every ten years: $1630 billion in 1975 measured in 1975 dollars, $4180 billion in 1985 measured in 1985 dollars, and $7269 billion in 1995 in 1995 $'s. This leads to the hypothesis that the budgets of organizations doubles every ten years.

Combination of Hypotheses I, II, and III. If in 1982 an amount of computing power of 0.5 MMY is assumed to be infeasible to invest in an attack, then 100 ($\approx 2*100*0.5$) MMY is infeasible in 1992. Furthermore, $2*10^4$ ($\approx 200*100$) MMY is infeasible in 2002, and $4*10^6$ MMY is infeasible in 2012.

3.5. Cryptanalysis.

Hypothesis IV. 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. For classical asymmetric systems the effect of cryptanalytic developments illustrated in (2.3) is similar to Moore's law, i.e., 18 months from now we may expect that attacking the same classical asymmetric system costs half the computational effort it costs today, cf. (4.2). For all other systems we assume that no substantial cryptanalytic developments will take place, with the exception of elliptic curve systems for which we use two types of extrapolations: no progress and progress à la Moore.

3.6. Software versus special-purpose hardware attacks. The proposed key sizes in the next section are obtained by combining Hypotheses I-IV with the software based MY data points. This implies that all extrapolations are based on 'software only' attacks and result in computationally equivalent key sizes (cf. (1.4)). One may object that this does not take special-purpose hardware attacks into account. Here 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.

Symmetric key systems. In 1980 the DES could either be broken at the cost of 0.5 MMY, or using a "$50 million, 2 days" machine. This is consistent with our version of Moore's law and the 1993 design from [26]. Thus, it seems reasonable to assume that a DES attack of one MMY is comparable to an attack by [$10 million, 20 days, 1980]-hardware or, using Moore's law, by [$\$200/2^{10.66}$ million = $125,000, 1 day, 1996]-hardware. It also follows that the 1982 relation between software and special-purpose hardware attacks on the DES has not changed. Thus, 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. Our estimates and the resulting cost of special hardware designs are consistent with the estimates given in [2] and [4].

EC systems. The cost of a software attack on a 109-bit EC system with $p = 2^{109}$ was estimated as 8 MMY, 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 longer, i.e., about 730 MMY. The [$10 million, 32 days, 1996]-hardware design attacking a 120-bit EC system with $p = 2^{155}$ (cf. (2.3.3)) should thus be comparable to 730 MMY. However, that design was based on 1992 technology which can be improved by using 1996 technology. So, by Moore's law, the 'upgraded' [$10 million, 32 days, 1996]-hardware design could be comparable with $730*6.35 \approx 4600$ MMY. It follows that an EC attack of one MMY is comparable to [$70,000, 1 day, 1996]-hardware.

We find that one MMY is equivalent to [$70,000 to $125,000, 1 day, 1996]-hardware. Thus, it is tempting to suggest that one MMY is approximately equivalent to [10^5, 1 day, 1996]-hardware; more generally, that one MMY would be equivalent to [$10^5/2^{2*(y-1996)/3}$, 1 day, *y*]-hardware in year *y*. 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. In the first place, the estimate holds only for relatively simple minded DES or EC cracking devices for EC systems over non-prime fields (i.e., those with $p = 2^k$), not for EC systems over prime fields or full-blown PCs. For prime fields the hardware would be slower, whereas in software EC systems can be attacked faster over prime fields than over non-prime fields (cf. (2.3.3)). Thus, for special-purpose hardware attacks on EC systems over prime fields the consistency no longer holds. Secondly, the pipelined version of the EC-attacking special-purpose hardware from [25] would be about 7 times faster (cf. [27]), so that also for special-purpose hardware attacks on EC systems over non-prime fields the consistency between DES and EC attacks is lost. The prime field version of the pipelined device would be 2^4 to 2^5 times slower than the non-prime field version (cf. [27]). The details of the pipelined device have not been published, cf. [28].

As mentioned in (2.3.3), we consider only EC systems that use randomly selected curves over prime fields. We show that 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. (2.3.3)). 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 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 naive DES-consistent hardware conversion, then the hypothetical special-purpose hardware that follows from extrapolation of the 2.2 MMY figure to larger prime fields 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 (cf. (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 (cf. (1.3)). It follows that, if one assumes that the DES was sufficiently resistant against a special-purpose hardware attack in 1982, the same holds for the EC key sizes suggested for the future, even though they are based on extrapolations of 'software only' attacks.

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

Classical asymmetric systems. For classical asymmetric systems we do not consider special-purpose hardware attacks, as argued in (2.3.1). The issue of software attacks on classical asymmetric systems versus special-purpose hardware attacks on other cryptosystems is discussed below.

Equipment cost comparison of software and special-purpose hardware attacks. Our recommendations below are computationally equivalent and, as argued above, they all offer security at least equivalent to the 1982 security of the DES, 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, 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 for free, as in all Internet based cryptosystem attacks so far and other large computational Internet projects. Adoption of this rule would make computational and cost equivalence identical, which is not generally acceptable (cf. [27]). A precise equipment cost defies exact analysis, primarily because no precise 'cost of a PC' can be pinpointed. Nevertheless, we sketch how an analysis based on cost equivalence could be carried out.

According to newspaper advertisements fully equipped PCs (cf. (1.2)) can be bought for prices varying from $0 to $450. The 'free' machines support the point of view that software attacks are for free. Assume that one does not want to deal with the strings attached to the free machines and that a stripped down PC (i.e., a 450 MHz Pentium II processor, a mother-board, and communications hardware) costs $100. It follows that [$81 million, 1 day, 1999]-hardware is equivalent to at least one million software MY, disregarding the possibly much larger quantum discount one should be able to negotiate for an order of this size. Compared to the above exhaustive key search [$125,000, 1 day, 1996] ≈ [$31,000, 1 day, 1999]-hardware, software MY are thus about 2500 times more expensive. Compared to the pipelined [$70,000/7, 1 day, 1996] ≈ [$2500, 1 day, 1999]-hardware to attack EC systems over non-prime fields, software MY are more than $3*10^4$ times more expensive, but at most about $2*10^3$ times more expensive than the prime field version of the pipelined design.

It follows that for our purposes software MY are at most 2500 times more expensive than MY produced by special-purpose hardware. In (4.5) it is shown how this factor 2500 can be used to derive equipment cost equivalent key sizes from the computationally equivalent ones. The factor 2500 should be taken with a grain of salt. Its scientific merit is in our opinion questionable because it is based on a guess for the price of stripped down PCs and the presumed infeasibility of special-purpose hardware attacks on RSA (cf. (2.3.1) and the pipelined design in [10]).

3.7. Memory considerations. In [15] we explain why (NFS) memory requirements do not explicitly have to be taken into account when extrapolating run times.

3.8. Remark. We do not expect that everyone agrees with our hypotheses. In particular Hypothesis I is debatable. Note that we did not assume anything about the (un)breakability of the DES in any year. We assumed that it 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, 3½ 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 (cf. [28]). Anyone who prefers a stronger or weaker infeasibility assumption can still use our approach, as shown in (4.4). Also our Hypothesis II is certainly not to everyone's taste. Some argue that Moore's law cannot hold much longer, others (cf. [14]) find it too pessimistic. Hypothesis II thus represents a reasonable compromise.

4 Lower bound estimates for cryptographic key sizes

4.1. Method of computation. For year *y* we first compute IMY(*y*), the number of MY considered to be infeasible for that year, based on Hypotheses I-III:

$$\mathrm{IMY}(y) = 0.5 * 10^6 * 2^{2(y-1982)/3} * 2^{(y-1982)/10}.$$

The resulting value is used to derive key sizes that should be sufficiently secure until year *y*, for all cryptographic primitives considered in Section 2. For symmetric key cryptosystems the key size is computed as the smallest integer that is at least

$$56 + \log_2(\mathrm{IMY}(y) / (0.5 * 10^6)) = 23y / 30 - 1463.533.$$

For classical asymmetric systems we use the asymptotic run time L[*n*] of the NFS (omitting the o(1)), the data point that a 512-bit key was broken in 1999 at the cost of less than 10^4 MY (cf. (2.3.1)) and Hypothesis IV that cryptanalytic progress à la Moore is expected, and we determine the smallest size *s* such that

$$L[2^s] * 10^4 \geq L[2^{512}] * 2^{2(y-1999)/3} * \mathrm{IMY}(y).$$

Because the data point used overestimates the cost of factoring a 512-bit key and because we omit the o(1) the difficulty of breaking classical asymmetric systems with keys of size *s* is overestimated (cf. (2.3.1)), i.e., the RSA and TDL key sizes should be even larger than given in Table 1.

For SDL systems we use the just determined *s* as field size for year *y*. Because no suitable data points are available, we use the optimistic estimate that an EC system over a prime field of 109 bits can be broken in 2.2 MMY (cf. (2.3.3)) and that an elliptic curve operation takes on average 9 field multiplications. Combined with the relative speed of Pollard's rho method, and the expected growth of the cost of the field operations, we find that the size of the subgroup size q can be taken as

$$109 + 2 * \log_2(109^2 * \mathrm{IMY}(y) * 9 / (s^2 * \sqrt{2} * 2.2 * 10^6)).$$

The resulting sizes are at most two bits too large (cf. (1.3) and [15]).

For EC systems we use the same optimistic estimate that a 109-bit system can be broken in 2.2 MMY combined with the expected growth rate of the number of group operations required by Pollard's rho method and the expected growth of the cost of the group operations, to determine the smallest size *t* such that

$$t^2 * 2^{(t-109)/2} \geq 109^2 * \text{IMY}(y) / (2.2 * 10^6).$$

The resulting t represents the recommended EC key size lower bound if no cryptanalytic progress is assumed, where it should be noted that t is on the large side because the 2.2 MMY estimate is rather optimistic (cf. (1.3)). An alternative key size that assumes cryptanalytic progress à la Moore is found by taking the smallest u with

$$u^2 * 2^{(u-109)/2} \geq 2^{2(y-1999)/3} * 109^2 * \text{IMY}(y) / (2.2*10^6).$$

4.2. Remarks on the computation of Table 1.

1. All estimates may be computed for years before 1999. Some of the resulting data can be found in Table 1. Strictly speaking this does not make sense for the "EC with progress" column. It is described in (4.4) how to use the data in italics.
2. The results do not change significantly if the RSA data point is replaced by other (older) factoring data points, which validates our decision to adopt a Moore-like law for cryptanalytic progress affecting classical asymmetric systems.

4.3. Using Table 1. Assuming one agrees with our hypotheses, Table 1 can be used as follows. Suppose electronic information has to be guaranteed until the year 2020. Looking at the row for the year 2020 in Table 1, one finds that an amount of computing of $2.9*10^{14}$ MY in the year 2020 may be considered to be as infeasible as $0.5*10^6$ MY was in 1982. Security comparable to the security offered by the DES in 1982 is therefore obtained by using, in the year 2020:

- 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 same column is explained in 4.5
- SDL systems with subgroups of **at least** 151 bits over fields of **at least** 1881 bits.
- EC systems over prime fields of **at least** 161 bits if one trusts that no cryptanalytic progress will occur, **at least** 188 bits if one wants to be more careful.

If finite fields are used that allow faster operations than suggested by our estimates, the SDL and EC data in Table 1 can still be used: if the arithmetic goes x times faster, keys should be roughly $2*\log_2(x)$ bits larger than indicated. As noted above the field arithmetic is already assumed to be quite fast. Similarly, if one finds that the data point used for EC systems overestimates the cost by a factor x, i.e., that the 2.2 MMY to attack 109-bit EC systems should be only $2.2/x$ MMY, add roughly $2*\log_2(x)$ bits to the suggested EC key sizes.

4.4. Alternative security margin. For corporations that have used the DES beyond 1982 our infeasibility assumption of 0.5 MMY in 1982 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 one trusts the DES until the year $1982 + x$, where x is negative if our infeasibility assumption is considered to be too weak and positive otherwise. So, for example, $x = 13$ if one trusts the DES until 1995.

- 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: take the classical asymmetric key size s' for year $y - 23*x/43$, the SDL size t for year $y - x$, and the classical asymmetric key size s for year $y - x$ and use a subgroup of size $t + 4*\log_2(s) - 4*\log_2(s')$ over a field of size s'. In our example $s' = 879$, $t = 114$, and $s = 682$ so that a subgroup of size $114 + 4*\log_2(682) - 4*\log_2(879) \approx 113$ bits should be used with a 879-bit field.
- EC systems without progress: take the 'without progress' 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 progress à la Moore: take the 'with progress' 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 $x < 0$.

Table 1

Lower bounds for computationally equivalent key sizes,
assuming cryptanalytic progress à la Moore affecting classical asymmetric systems

Year	Symmetric Key Size	Classical Asymmetric Key Size (and SDL Field Size)	Subgroup Discrete Logarithm Key Size	Elliptic Curve Key Size		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 450MHz PentiumII PC
				progress				
				no	yes			
1982	56	417 $_{288}$	102	105	*85*	$5.00 * 10^5$	$3.98 * 10^7$	$1.11 * 10^3$
1983	57	440 $_{288}$	103	107	*88*	$8.51 * 10^5$	$4.27 * 10^7$	$1.89 * 10^3$
1984	58	463 $_{320}$	105	108	*89*	$1.45 * 10^6$	$4.57 * 10^7$	$3.22 * 10^3$
1985	59	488 $_{320}$	106	110	*93*	$2.46 * 10^6$	$4.90 * 10^7$	$5.47 * 10^3$
1986	60	513 $_{352}$	107	111	*96*	$4.19 * 10^6$	$5.25 * 10^7$	$9.31 * 10^3$
1987	60	539 $_{384}$	108	113	*98*	$7.13 * 10^6$	$5.63 * 10^7$	$1.58 * 10^4$
1988	61	566 $_{384}$	109	114	*101*	$1.21 * 10^7$	$6.04 * 10^7$	$2.69 * 10^4$
1989	62	594 $_{416}$	111	116	*104*	$2.06 * 10^7$	$6.47 * 10^7$	$4.58 * 10^4$
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	$5.98 * 10^{16}$	$1.11 * 10^{9}$	$1.33 * 10^{14}$
2031	94	2560 2080	167	178	218	$1.02 * 10^{17}$	$1.19 * 10^{9}$	$2.26 * 10^{14}$
2032	95	2629 2144	168	179	222	$1.73 * 10^{17}$	$1.27 * 10^{9}$	$3.85 * 10^{14}$
2033	96	2698 2208	169	181	223	$2.95 * 10^{17}$	$1.37 * 10^{9}$	$6.55 * 10^{14}$
2034	96	2768 2272	171	182	227	$5.01 * 10^{17}$	$1.46 * 10^{9}$	$1.11 * 10^{15}$
2035	97	2840 2336	172	184	230	$8.53 * 10^{17}$	$1.57 * 10^{9}$	$1.90 * 10^{15}$
2036	98	2912 2400	173	185	232	$1.45 * 10^{18}$	$1.68 * 10^{9}$	$3.22 * 10^{15}$
2037	99	2986 2464	175	186	235	$2.47 * 10^{18}$	$1.80 * 10^{9}$	$5.49 * 10^{15}$
2038	99	3061 2528	176	188	239	$4.20 * 10^{18}$	$1.93 * 10^{9}$	$9.33 * 10^{15}$
2039	100	3137 2592	178	189	240	$7.14 * 10^{18}$	$2.07 * 10^{9}$	$1.59 * 10^{16}$
2040	101	3214 2656	179	191	244	$1.22 * 10^{19}$	$2.22 * 10^{9}$	$2.70 * 10^{16}$
2041	*102*	*3292 2720*	*180*	*192*	*247*	$2.07 * 10^{19}$	$2.38 * 10^{9}$	$4.60 * 10^{16}$
2042	*103*	*3371 2784*	*182*	*194*	*248*	$3.52 * 10^{19}$	$2.55 * 10^{9}$	$7.82 * 10^{16}$
2043	*103*	*3451 2880*	*183*	*195*	*252*	$5.99 * 10^{19}$	$2.73 * 10^{9}$	$1.33 * 10^{17}$
2044	*104*	*3533 2944*	*185*	*197*	*255*	$1.02 * 10^{20}$	$2.93 * 10^{9}$	$2.26 * 10^{17}$
2045	*105*	*3616 3008*	*186*	*198*	*257*	$1.73 * 10^{20}$	$3.14 * 10^{9}$	$3.85 * 10^{17}$
2046	*106*	*3700 3072*	*187*	*200*	*260*	$2.95 * 10^{20}$	$3.36 * 10^{9}$	$6.55 * 10^{17}$
2047	*106*	*3785 3168*	*189*	*201*	*264*	$5.02 * 10^{20}$	$3.60 * 10^{9}$	$1.11 * 10^{18}$
2048	*107*	*3871 3232*	*190*	*203*	*265*	$8.53 * 10^{20}$	$3.86 * 10^{9}$	$1.90 * 10^{18}$
2049	*108*	*3959 3328*	*192*	*204*	*269*	$1.45 * 10^{21}$	$4.14 * 10^{9}$	$3.23 * 10^{18}$
2050	*109*	*4047 3392*	*193*	*206*	*272*	$2.47 * 10^{21}$	$4.44 * 10^{9}$	$5.49 * 10^{18}$

4.5. Equipment cost equivalent key sizes. Assuming the $100 price for a stripped down PC (cf. (3.6)) and the resulting factor of 2500 are acceptable, Table 1 can be used to derive equipment cost equivalent key sizes in the following manner. 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*.

The symmetric key sizes are derived based on Hypothesis 1, and the EC key sizes are based on estimates that are cost consistent with the symmetric key sizes (cf. (3.6)), so for those systems no corrections are necessary.

For classical asymmetric systems, MY are supposedly 2500 times as expensive, which is, for our computational purposes only, equivalent to assuming that the DES offers acceptable security until about 1997, since 1997 − 1982 = 15 and $2^{(15*23/30)}$ (cf. (4.1)) is close to 2500. Thus, using (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 MY than the infeasible number of MY for year y, but acquiring the required MY is supposed to be prohibitively expensive.

For subgroup discrete logarithm systems in year y, let t and s be the subgroup and finite field size, respectively, for year y, and let s' 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 $t + 4*\log_2(s) - 4*\log_2(s')$ over finite fields of size s'. As a rule of thumb, subgroups of size $t + 2$ over finite fields of size s' 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 PC price one may prefer. For instance, for $10 or $1000 per PC the $y - 8$ should be changed into $y - 6$ or $y - 10$, respectively.

5 Practical consequences of Table 1

5.1. DSS. DSS key sizes can be recommended for commercial applications only until the year 2002 for the field size, until 2013 for the hash function, and until 2026 for the subgroup size. For security until 2013, it is advisable to use the DSS with a 1513-bit finite field. Beyond 2013 the 160-bit size of SHA-1 may no longer be adequate. Changing it may force a change in the subgroup size that would otherwise not have been necessary until 2026. According to [16], 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

5.2. Effect on cryptosystem speed. RSA keys that are supposed to be secure until 2040 are about 3 times larger than popular 1024-bit RSA keys, making those large keys 9 to 27 times slower to use: 9 for signature verification or encryption with a fixed length public exponent, 27 for signature generation or decryption. TDL systems will slowdown by a factor 27 compared to 1024-bit versions. SDL systems slowdown by about a factor 11 compared to currently secure SDL systems. The speed of EC

systems is hardly affected. Within a few years faster processors will have solved these performance problems by our Moore assumption. 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. RSA keys of 512 bits are widely used all over the Web, e.g. in SSL protected communications. According to Table 1, such keys should not have been used beyond 1986. A 512-bit RSA key was factored in August 1999 (cf. [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 equipment 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. It is often informally argued that 1024-bit RSA and 160-bit EC systems offer more or less the same level of security. This is not what one would conclude from Table 1. We discuss several aspects of this contentious issue in [15].

5.6. SDL and EC. The gap between the suggested SDL and EC key sizes widens slowly due to the rapidly growing size of the finite fields in SDL.

5.7. Effectiveness of guessing. The sizes suggested in Table 1 for the year 2000 or later are in practice infeasible to guess.

5.8. Effectiveness of incomplete attacks. As shown in [15], incomplete attacks can on average not be expected to pay off.

5.9. Effectiveness of Elliptic Curve Method. As shown in [15], the Elliptic Curve Method cannot be expected to break keys of the suggested sizes.

5.10. Wassenaar Arrangement for mass market applications. The WA allows 64-bit symmetric keys and 512-bit classical asymmetric keys for mass market applications. According to Table 1 it is advisable to increase the 512-bit bound for classical asymmetric keys to 672 or 768 bits.

Acknowledgements. The authors want to thank Joe Buhler, Bruce Dodson, Stuart Haber, Paul Leyland, Alfred Menezes, Andrew Odlyzko, Michael Wiener, and Paul Zimmermann for their helpful remarks.

References

1. Eli Biham, A fast new DES implementation in software.
2. 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.
3. A. Bosselaers, Even faster hashing on the Pentium, manuscript, Katholieke Universiteit Leuven, May 13, 1997.

4. J.R.T. Brazier, Possible NSA decryption capabilities, http://jya.com/nsa-study.htm.
5. S. Cavallar, B. Dodson, A.K. Lenstra, B. Murphy, P.L. Montgomery, H.J.J. te Riele, et al., Factorization of a 512-bit RSA modulus, manuscript, October 1999.
6. www.counterpane.com/speed.html.
7. M. Davio, Y. Desmedt, J. Goubert, F. Hoornaert, J.J. Quisquater, Efficient hardware and software implementations of the DES, Proceedings Crypto'84.
8. W. Diffie, BNR Inc. report, 1980.
9. W. Diffie, E. Hellman, Exhaustive cryptanalysis of the NBS Data Encryption Standard, Computer, v. 10 (1977), 74-84.
10. B. Dixon, A.K. Lenstra, Factoring integers using SIMD sieves, Proceedings Eurocrypt'93, LNCS 765, 28-39.
11. Electronic Frontier Foundation, Cracking DES, O'Reilly, July 1998.
12. Rob Gallant, personal communication, August 1999.
13. P.C. Kocher, Breaking DES, RSA Laboratories' Cryptobytes, v. 5, no 2 (1999); also at www.rsa.com/rsalabs/pubs/cryptobytes.
14. P.C. Kocher, personal communication, September 1999.
15. A.K. Lenstra, E.R. Verheul, Selecting Cryptographic Key Sizes, submitted for publication, September 1999; available at www.cryptosavvy.nl.
16. A.J. Menezes, personal communication, September 1999.
17. P.L. Montgomery, letter to the editor of IEEE Computer, August 1999.
18. 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 back from 1968.
19. 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.
20. 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.rsa.com/rsalabs/pubs/cryptobytes.
21. 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.
22. V. Shoup, Lower bounds for discrete logarithms and related problems, Proceedings Eurocrypt'97, LNCS 1233, 256-266.
23. R.D. Silverman, rump session presentation at Crypto'97.
24. R.D. Silverman. Exposing the Mythical MIPS Year, IEEE Computer, August 1999, 22-26.
25. P.C. van Oorschot, M.J. Wiener, Parallel collision search with cryptanalytic applications, Journal of Cryptology, v. 12 (1999), 1-28.
26. M.J. Wiener, Efficient DES key search, manuscript, Bell-Northern Research, August 20, 1993.
27. M.J. Wiener, Performance Comparison of Public-Key Cryptosystems, RSA Laboratories' Cryptobytes, v. 4, no. 1 (1998), 1-5; also at www.rsa.com/rsalabs/pubs/ cryptobytes.
28. M.J. Wiener, personal communication, 1999.